МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНЫХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

"Інформаційна технологія керування проєктами візуального програмування для платформи Dynamo Studio"

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Ободяк В.К.**

Студент гр. ІН.м-92 **Красковський Р.О.**

СУМИ 2020

РЕФЕРАТ

Записка: 136 стор., 36 рис., 3 табл., 1 додаток, 19 джерел.

Об’єкт дослідження — процес керування проєктами візуального програмування для платформи Dynamo Studio.

Мета роботи — розробити засоби створення параметричної ітераційної моделі дизайну і дослідити створену модель.

Методи дослідження — в процесі дослідження були використані такі технології як JavaScript, Bootstrap, JQuery, ASP.NET, Dynamo Studio API, Autodesk Froge, BIM 360, WIX Toolset.

Результати — розроблено засоби створення параметричної ітераційної моделі дизайну і досліджено створену модель. Веб-сервіс розроблений і готовий для використання та інтегрування в інші системи. Додаток розроблений і готовий для інсталяції. Розроблено алгоритм індексації буферу вершин для тріангульованої геометрії, який оптимізує розмів файлів.

DYNAMO, AUTODESK FORGE, DESIGN EXPLORER, BIM 360,
WIX, ASP.NET, C#, MVVM

Факультет _____*ЕлІТ*_____ Кафедра ____*Комп'ютерних наук*_____

Спеціальність *Інформатика*

Затверджую:

Зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТОВІ

Красковському Роману Олександровичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія керування проєктами візуального програмування для платформи Dynamo Studio*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Інформаційний огляд. 2) Постановка задачі 3) Вибір методів вирішення 4) Проєктування інформаційної технології 5) Розробка інформаційної технології*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної магістерської роботи	Термін виконання проекту (роботи)	Примітка
1.			
2.			
3.			
4.			

Студент – дипломник _____

(підпис)

Керівник проекту _____

(підпис)

ЗМІСТ

ВСТУП

Вивчення ітераційної моделі дизайну дає краще розуміння варіацій майбутнього проекту в залежності від його параметрів. У різних випадках це можуть бути різні параметри: від стандартних довжин до кількості і витрат. Все це можна дослідити і вибрати правильне рішення завдяки багатовимірної параметричної моделі.

Оскільки є потреба вивчення моделі, то потрібен і інструмент її створення. Тому розробка програмного рішення для створення дизайнів такої моделі є актуальним питанням.

Основною метою даної роботи є швидке створення дизайну, при якому не потрібно буде створення багатьох ітерацій, а лише однієї, в якій, змінюючи її параметри, буде автоматично створена нова ітерація без ручної зміни її структури.

Можливим рішенням цієї задачі є розробка додатку CAD системи Autodesk Dynamo Studio для створення і відповідно веб-системи для вивчення ітераційної моделі. Завдяки NODE структурі Dynamo Studio, вирішується питання з незручністю створення окремих ітерацій. Суть додатку у конвертації кожної ітерації після зміни параметрів у кастомний формат та відправки її на сервер веб-системи, на якій можна буде досліджувати створений дизайн.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Поняття веб-системи

Веб-система являє собою веб-сайт, що надає користувачеві можливість використовувати різні сервіси, які співпрацюють в рамках одного веб-додатку. Для таких систем є обов'язковою наявність підсистеми керування правами доступу, забезпечуючи безпеку захищенної інформації [2].

Веб-система містить у собі серверну і клієнтську частини. Клієнтська частина веб-системи реалізує користувацький UI, створює запити до віддаленого серверу і чекає відповіді від нього. Серверна частина, коли отримує запит, обробляє його, та робить різноманітні обичилення, а потім формує з використанням веб-сторінку і надсилає її до клієнта (рис. 1.1).

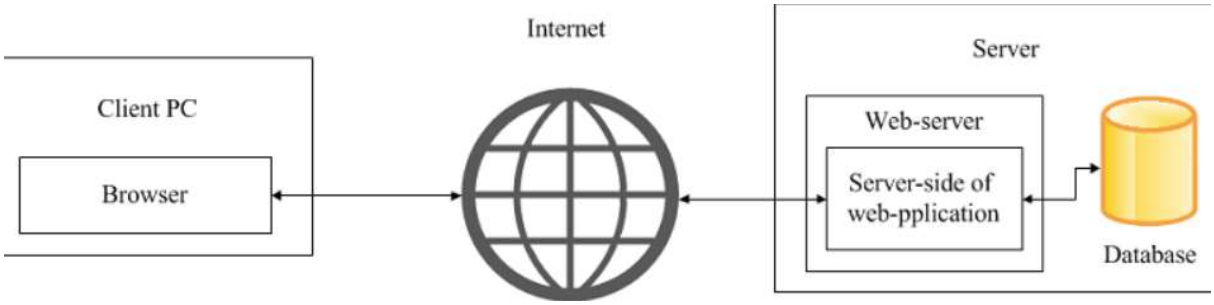

Рисунок 1.1 — Структура моделі веб-додатків [11]

Сам веб-додаток може виступати в якості клієнта інших служб, наприклад, бази даних або інших веб-додатків, розташованих на іншому сервері [12].

1.2 Технології, які використовуються у веб-додатках

Для реалізації клієнтської частини веб-додатків в наш час найчастіше використовуються наступні технології:

- HTML;
- CSS.
- ActiveX;
- JavaScript;
- Silverlight.

Зараз є дуже пополярним підхід до розробки веб-сайтів, який називається Ajax. Використовуючи Ajax, сторінки сайтів не перезавантажуються, а лише підвантажують нові дані з серверної частини, що додає їм інтерактивності та продуктивності [9].

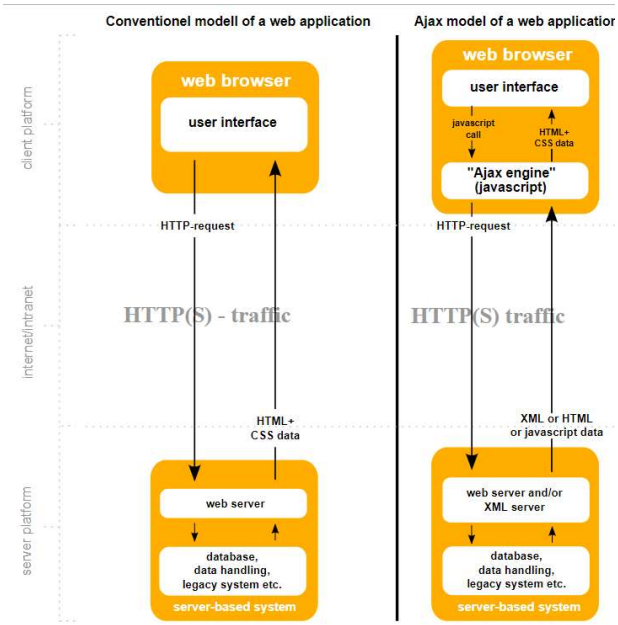

Рисунок 1.2 — Звичайна модель для веб-додатків у порівнянні з програмою, що використовує Ajax [11]

JSON являє собою альтернативу формату XML, в порівнянні з яким є біль зручним, простим і не потребує великої кількості форматування.

Об'єкт JSON це формат даних з ключовим значенням, який зазвичай рендерится в фігурних дужках.

Приклад об'єкту JSON:

```
{
    "key_one": "value_one",
    "key_two": "value_two"
}
```

Хоч це і короткий приклад, і JSON міг би бути набагато довший, це показує те, що цей формат в основному встановлюється двома фігурними дужками, які виглядають так { }, а дані з ключовими значеннями знаходяться між ними.

Для створення веб-додатків на стороні сервера використовуються різноманітні технології та будь-які мови програмування , здатні здійснювати вивід в стандартну консоль [12].

Таблиця 1.1 — Технології та мови програмування, які використовуються на сервері

Назва	**Ліцензування**	**Веб-сервер**
ASP	власницька	спеціальний
ASP.NET	власницька	спеціальний
C/C++	вільна	майже будь-який
Java	вільна	купа, серед них і вільні
Perl	вільна	майже будь-який
PHP	вільна	майже будь-який
Python	вільна	майже будь-який
Ruby	вільна	майже будь-який
Nodejs	MIT	пропріетарний

1.3 Додаток (Add-in) для Dynamo Studio

Додаток – це клас відносно невеликих програм, що доповнюють і розширяють можливості основного застосунку (в даному випадку Dynamo Studio).

Dynamo Studio - це візуальне середовище програмування, яке дозволяє дизайнерам досліджувати параметричні концептуальні проекти та автоматизувати завдання(рис. 1.3). Вона дозволяє:

- Інтегрувати автоматизовані процеси в систему інформаційного моделювання будівель (BIM).

- Розширювати свої проекти до сумісних робочих процесів для документування, координації та аналізу.
- Писати код за допомогою простого та потужного інтерфейсу сценаріїв.

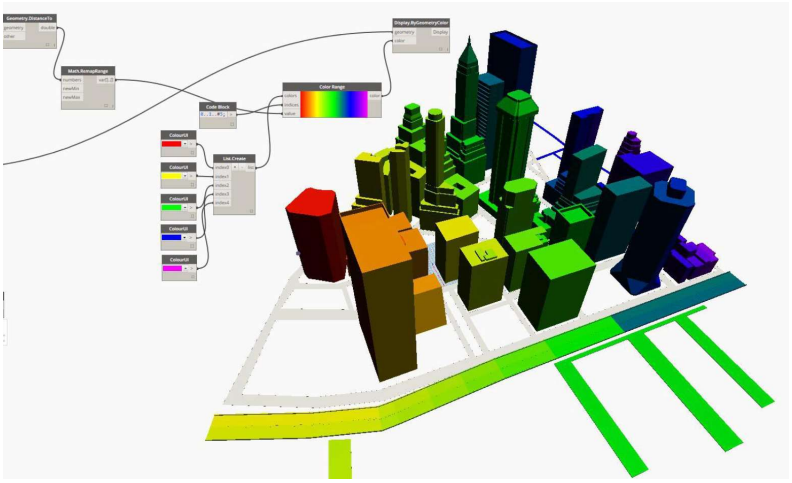

Рисунок 1.3 – Дизайн створений у Dynamo Studio

Для простого пояснення роботи Dynamo Studio створимо сферу(рис. 1.4)

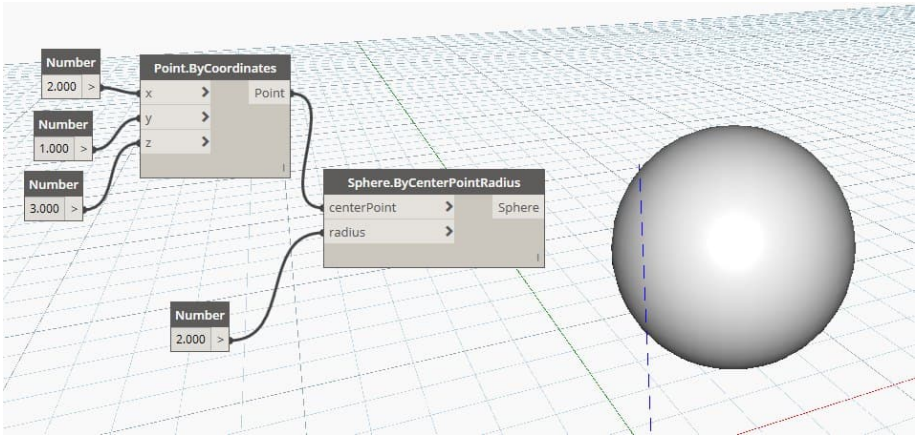

Рисунок 1.4 – Сфера у Dynamo Studio

На рисунку 1.4 можна побачити деякі користувацькі форми (окрім сфери). Ці формі називають нодами, або вузлами. Ці вузли можна сприймати, як методи (функціїї) у програмуванні. Вони приймають деякі параметри та повертають результат. Наприклад вузол Point.ByCoordinates приймає три числа та повертає тривимірну точку. А в свою чергу Sphere.BeCenterPointRadius приймає цю ж точку і число для радіусу та створю з цих параметрів сферу.

Слід додати, що Dynamo Studio може працювати у автоматичному і ручному режимах. В автоматичному режимі вузли спрацьовують на будь яку зміну параметрів, а в ручному користувач повинен натиснути кнпоку, щоб запустити роботу вузлів.

Суть додату для Dynamo Studio – це створення своїх власних вузлів, які розширять функціонал.

1.4 Використання Autodesk Forge

Autodesk Forge – це хмарна платформа для розробників від Autodesk. Во на дозволяє отримати доступ до дизайнерських та інженерних даних. Якщо розробник хочете автоматизувати процеси, зв’язати команди та робочі процеси або візуалізувати свої дані – то він може створювати ці програми та багато іншого, використовуючи API Forge. Платформа Forge пропонує API та сервіси, які допоможуть отримати доступ та використовувати конструкторські та технічні дані через хмару

Forge дозволяє не лише отримувати, а й додавати нові дизайнерські та інженерні дані до хмари.

Також ця хмарна платформа для розробників надає змогу створити соціальну авторизацію на базі Autodesk [13].

Повний перелік можливостей Forge:

- **Authentication** - створення токенів на основі стандарту OAuth 2.0 для автентифікації запитів, що надходять до API-інтерфейсів Forge та SDK.
- **BIM 360** - інтегрування з платформою Autodesk BIM 360, щоб розширити можливості та охопити сегменти будівельної екосистеми, які не мають прямого доступу до даних BIM.
- **Data management** - отримання доступу до даних через команду BIM 360, Fusion Team, BIM 360 Docs та Службу зберігання об’єктів, щоб створювати програми для відображення та розширення даних, що створює нові можливості для користувачів.

- **Design Automation** - автоматизування повторюванних завдань, використовуючи масштаби платформи Forge і запускаючи сценарії для дизайнерських файлів у хмарі.
- **Model Derivative** - виведення вихідних даних, доступних для перегляду у Forge Viewer у більш ніж 60 форматів CAD, та витягування метаданих моделі, а також окремих об'єктів в моделі.
- **Token Flex** - доступ до платформи даних використовуючи Autodesk Token Flex для створення звітів про споживання, використання та деталі контракту.
- **Viewer** - візуалізація данних 3D та 2D моделі в браузері. Моделі можуть надходити з широкого кола програм, таких як AutoCAD, Fusion 360, Revit та багатьох інших.

1.5 Тріангуляція та індексація буферу вершин

Тріангуляція - це перетворення(конвертація) геометрії на тріангули (або симплекси). Тріангуляція 3D об'єкта являє собою розбиття на тетраедри, що притикаються один до одного. Навіть в самій геометрії використовують різні визначення тріангуляції в залежності від розділу. Тріангуляція T простору $\mathbb{R}^{n+1}$ - це підрозбиття $\mathbb{R}^{n+1}$ точок на (n + 1)-вимірні симплекси такі що:

1. будь-які два симплекси в T перетинаються в спільній грані ребру чи вершині, або взагалі не перетинаються;
2. будь-яка обмежена множина в $\mathbb{R}^{n+1}$ перетинає скінченну кількість симплексів з T.

Тріангуляція множини точок, або, як її ще називають, тріангуляція дискретної множини точок $P \subset \mathbb{R}^{n+1}$ — це поділ опуклої оболонки точок на тріангули так, що виконується перша умова з попереднього означення та множина точок, що є вершинами симплексів розбиття збігається з P. Тріангуляція Делоне є найвідомішим видом тріангуляції множини точок.

Тріангуляція многокутника — це розбиття многокутника на симлекси, що мають спільні ребра з умовою, що множина вершин трикутників збігається з

множиною вершин многокутника. Тріангуляція многокутників є основою для великої кількості алгоритмів в геометрії, для прикладу, у розв’язку задачі «галерея мистецтв» . Гранична тріангуляція Делоне — це адаптація тріангуляції Делоне від множин точок до многокутників, у загальнішому — до планарних графів.

Тріангуляція поверхні — це сітка трикутників, яка частково або повністю покриває поверхню заданої геометрії.

У методі кінцевих елементів тріангуляція використовується як сітка, що є основою для подальших розрахунків. У цьому випадку трикутники повинні утворювати набір у області детермінації функції. Для того, щоб бути придатною для розрахунку, тріангуляція повинна мати в кожному випадку різні типи трикутників, залежно від критеріїв моделювання звичайних елементів. Наприклад, деякі методи вимагають гострих або прямокутних трикутників, щоб сформувати сітку без тупих кутів. Відомо багато методів, що використовують гратки, що містять уточнення Делоне, такі як другий алгоритм Чу та алгоритм Руперта [15].

В більш загальних топологічних просторах, тріангуляція — це розбиття на простіші комплекси, що гомеоморфні простору.

Буфер індексів дозволяє зберігати більше даних моделі в меншому просторі.

Замість того, щоб зберігати по три вершини на кожен трикутник(дві вершини для кожного трикутна майже завжди будуть спільними для іншого трикутника, а одна для двох інших), можна використовувати спільні вершини між кількома трикутниками, оскільки в більшості сіток, кожен симплекс ділиться на три і більше трикутників, що в підсумку стає досить великим виграшем і робить модель більш ефективною, оскільки вершини мають більше шансів опинитися в кеші вершин. Для прикладу розглянемо чотирикутник, який наведено на рисунку 1.5.

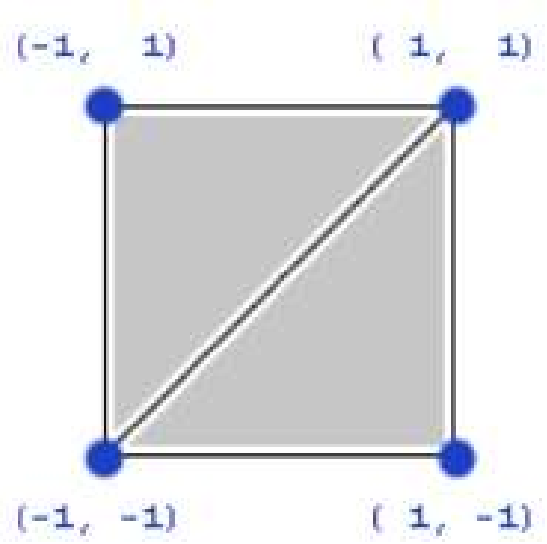

Рисунок 1.5 – Тріангульований чотирикутник

Якщо використовувати примітивний тип «Список трикутників» для візуалізації двох трикутників, кожен трикутник буде зберігатися як 3 окремі вершини, що призведе до подібного буфера вершин (рис. 1.6).

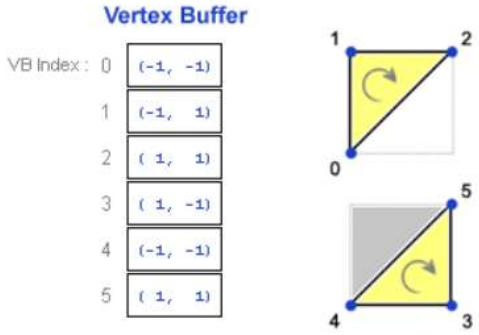

Рисунок 1.6 – Буфер симплексів

Як можна помітити, буфер вершин містить повторювані дані в місцях 0 і 4, 2 і 5. Це має сенс, оскільки два трикутники мають спільні дві вершини. Ці повторювані дані марнотратні, і буфер вершин можна стиснути, використовуючи індексний буфер. Менший буфер індексів зменшує кількість даних вершин, які повинні надсилатися графічному адаптеру. Ще важливіше те, що використання буфера індексу дозволяє адаптеру зберігати вершини у кеші вершин. Якщо вимальований примітив містить нещодавно використовувану вершину, цю вершину можна отримати з кешу, замість того, щоб читати з буфера індексів, що призводить до значного збільшення продуктивності.

Буфер індексу "індексується" у буфер вершин, тому кожна унікальна вершина повинна зберігатися лише один раз у буфері вершин. На схемі наведено індексований підхід до попереднього сценарію малювання (рис. 1.7).

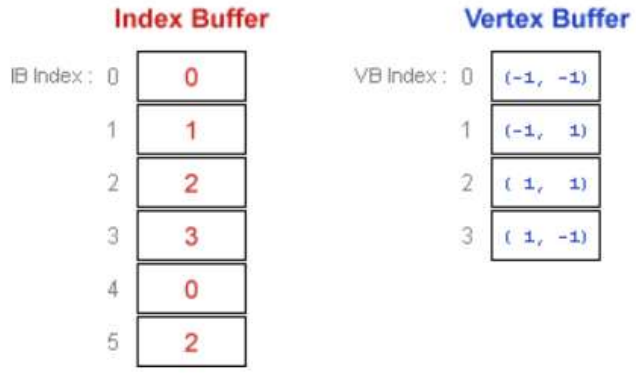

Рисунок 1.7 – Індексований буфер симплексів

Буфер індексів зберігає значення індексу буфера симплексів, які посилаються на певну вершину всередині буфера індексів. Буфер індексів можна розглядати як масив вершин, тому індекс VB - це просто індекс буфера вершин для цільової вершини. Подібним чином, IB Index - це індекс у буфері індексу.

1.6 Design Explorer

Студія CORE розробила Design Explorer, інструмент з відкритим вихідним кодом для дослідження просторових дизайнів в Інтернеті. Design Explorer - це інтерфейс, який дозволяє візуалізувати і фільтрувати групи ітерацій - набори проєктних рішень, які тісно пов'язані між собою і потенційно розкидані по величезному багатовимірному просторі можливостей (рис. 1.8).

Користувачі експортують свої моделі дизайну з додатків параметричної розробки (Grasshopper, Catia і т.п.) у вигляді файлу data.csv і серії зображень і моделей очок. Дані простору проектування генеруються шляхом автоматичного обходу параметричної моделі , або за допомогою алгоритму оптимізації, такого як Galapagos або Octopus.

Design Explorer зчитує файл data.csv і генерує двомірну візуалізацію простору дизайну, яку називають графіком паралельних координат (з сіткою мініатюр та іншими фішками для кінцевого користувача інтерфейсом) [5].

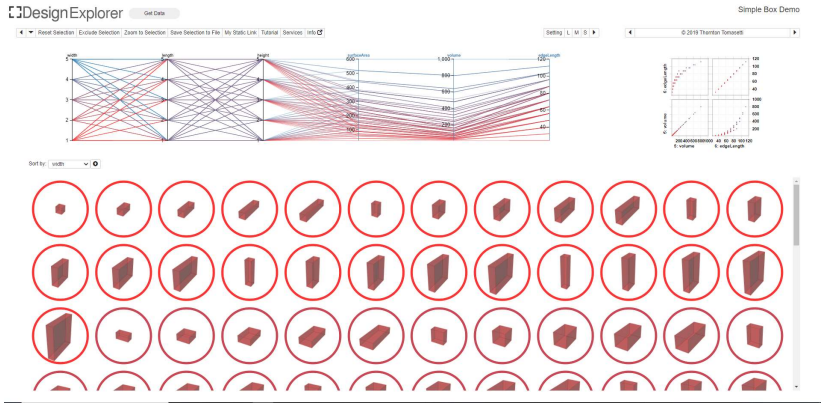

Рисунок 1.8 – Design Explorer

1.7 Маркери доступу та маркери оновлення

Маркер доступу (або токен) – це маркер, який надає сервер користувачеві після аутентифікації, для отримання доступу до даних. Найбільш популярним маркером аутентифікації є JSON Web Token.

JSON Web Token (JWT) - це JSON об'єкт, який визначений у відкритому стандарті RFC 7519. Він вважається одним з безпечних способів передачі інформації між двома учасниками. Для його створення необхідно визначити заголовок (header) із загальною інформацією по маркеру, корисні дані (payload), такі як id користувача, його роль і т.п. і підписи (signature) .

Простими словами, JWT - це лише рядок у наступному форматі header.payload.signature.

Припустимо, що ми хочемо зареєструватися на сайті. У нашому випадку є три учасники - користувач user, сервер додатки application server і сервер аутентифікації authentication server. Сервер аутентифікації буде забезпечувати користувача маркером, за допомогою якого він пізніше зможе взаємодіяти з додатком.

Додаток використовує JWT для перевірки аутентифікації користувача в такий спосіб:

- Спершу користувач заходить на сервер аутентифікації за допомогою аутентификаційного ключа (це може бути пара логін / пароль, або Facebook ключ, або Google ключ і т.д.).

- Потім сервер аутентифікації створює JWT і відправляє його користувачеві.
- Коли користувач робить запит до API додатка, він додає до нього отриманий раніше JWT.
- Коли користувач робить API запит, додаток може перевірити за поданою із запитом JWT є користувач тим, за кого себе видає. У цій схемі сервер додатки налаштований так, що зможе перевірити, чи є вхідний JWT саме тим, що був створений сервером аутентифікації.

Зазвичай маркер надається з коротким терміном дії (наприклад 15 хвилин). Та щоб не авторизовуватись кожні 15 хвилин, є поняття маркеру оновлення. Його користувач отримує разом із маркером доступу. За допомогою нього можна оновлювати маркер доступу, відправляючи його на сервер. Термін дії такого токену може бути дуже довгим (тиждень, місяць, півроку).

1.8 Постановка задачі

Проведений аналітичний огляд показав, що для досягення поставленої мети необхідно виконати такі завдання:

1. Вибрати методи вирішення поставленої задачі
2. Спроєктувати функціональні процеси
3. Виконати програмну реалізацію

Необхідно розробити веб-систему, який матиме наступні можливості:

- аутентфікація через Autodesk;
- створення/Редагування/Видалення дизайну;
- створення/Зміна/Видалення ітерації;
- шаринг дизайну за посиланням;
- зв’язок з базою данних;
- зв’язок з хмарою;
- перегляд усіх дизайнів;
- пергляд усіх ітерацій дизайну;
- сортування і фльтрування ітерацій в залежності від параметрів дизайну;
- прегляд статистичної інформації по дизайну;
- перегляд окремої ітерації дизайну в 2D і 3D.

Розроблювана веб-система повинна мати веб-інтерфейс, за допомогою якого будуть відображена таблиця дизайнів. Після вибору дизайну, повинна відкритися сторінка ітерацій обраного дизайну. При натисканні на ітерацію повинен з’явитися засіб для перегляду обраної ітерації. Також потрібно додати сторінку, де користувач зможе налаштувати свою хмару для подальшої роботи.

Необхідно розробити додаток для Dynamo Studio, який матиме наступні можливості:

- Аутентфікація через Autodesk.
- Створення/Вибір дизайну.
- Створення ітерації.
    1. Створення тз аповнення параметрів.
    2. Тріангуляція геометрії.
    3. Створення 2D зображення для 3D геометрії.
    4. Створення JSON файлу з тріангульованою геометрією.
- Відправка ітерації на сервер.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Вибір клієнтських технологій

Оскільки одним із результатом роботи є веб-додаток, було обрано використовувати такі веб-технології як:

- HTML — стандартна мова розмітки веб-сторінок. Документ HTML обробляється та відображається у вікні браузера у прописаній інтерпритації [4].
- Каскадні таблиці стилів (CSS) — мова, яка використовується для стилізації сторінок, створених мовами розмітки[6].
- JavaScript — динамічна, об'єктно-орієнтована мова програмування. Вона створює можливість на клієнті взаємодіяти з користувачем, обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки [7].

Ще було обрано фреймворк Bootstrap, у якого досить важко знайти конкурентів і на сьогоднішній день є найпопулярнішим UI фреймворком [3]. Він містить CSS і HTML паттерни для багатьох елементів інтерфесу, а також додаткові плагіни JavaScript [12].

2.2 Вибір серверних технологій

Для серверної частини вебб-додатку було обрано використовувати такі веб-технології як:

- **ASP.NET –** це платформа розробки веб-систем. Вона дозволяє створювати backend-логіку веб-систем за допомогою .NET і мови програмування C#. У собі, цей фреймворк містить багато автоматизованих моментів роботи з базою даних, аутентіфікацією, створенням WEB-API [8].
- **Entity Framework** - структура об'єктно-реляційного відображення (ORM) з відкритим кодом для ADO.NET. Раніше був однією з частин .NET Framwork, але після виходу EF6 відокремився [10].
- **MsSQL (Microsoft SQL Server)** - система управління базами даних, розроблена Microsoft. Оскільки MsSQL є сервером даних, то ця система виконує свю найважиливішу функцію, а саме зберігає та надає дані у відвовідь на запити

додатків. Ці додатки можуть бути і на самому сервері, або на інших серверах у мережі [17].

Серед інших платформ розробки можна виділити:

- Java EE або Java Enterprise Edition являє платформу для створення корпоративних додатків на мові Java. Перш за все це сфера веб-додатків і веб-сервісів. Java EE складається з набору API і середовища виконання. Java EE розширює стандартну платформу Java.
- PHP – це скриптова мова, яка зазвичай використовується у веб-розробці на стороні сервера. Оскільки мова скриптова – це означає, що PHP використовується для автоматизації процесів, які в іншому випадку повинні були б виконуватися поетапно в коді сайту кожного разу, коли виникають ці самі процеси.

Використання Java EE має велику перевагу в тому, що розробник може покластися на стандарт, який дозволяє розробляти спираючись на API - в основному просто інтерфейси і анотації - при тому, що дійсна реалізація фреймворка не включається в додаток. Власне залежність Java EE «поставляється» сервером додатків, що означає, що вона потрібна тільки для компіляції, а не включається в пакет установки. У цього є побічний ефект який полягає в тому, що war-файл в основному залишається порожнім - він містить тільки class файли вашої програми. А все інше повинно бути надано Java EE реалізацією [20].

PHP - частково об'єктно-орієнтована мова. Це дає можливість повторно використовувати код, що економить час і сили в процесі розробки. Існує безліч фреймворків PHP: Symfony, CodeIgniter, Laravel, Joomla, WordPress та ін. Кожен з них має свій функціонал і заточений під певні завдання, але в той же час PHP в сучасних проектах заново створює весь контекст на кожен запит (всі об'єкти, підключення до бази і т.п.). ASP.NET як і JavaEE зберігає контекст, що сильно економить час на великих проектах [19].

Був обраний ASP.NET, оскільки він має ряд переваг перед іншими платформами для створення WEB-додатків. Мабуть, найважливішим з них є

інтеграція із серверами Windows та засобами програмування. Використання ASP.NET робить розробку, конфігурацію та публікацію WEB-додатків простими, тому що усі перелічені задачі можно зробити в одному середовищі під назваою Visual Studio[1].

Основні переваги, які дає розробникам WEB-додатків використання ASP.NET:

- ASP.NET - це суто серверна технологія, тому код обробляється на сервері Windows перед тим, як відображатись у веб-браузері. Тому програми ASP.NET виконуються швидше, ніж інтерпретовані сценарії.
- Такі функції ASP.NET, як раннє зв'язування, компіляція JIT, кешування та підтримка власної оптимізації, забезпечують високий рівень продуктивності. За допомогою .NET розробник не обмежується JIT, але й має можливість використовувати AOT, якщо він хоче усунути затримки запуску.
- Фреймворк ASP.NET не залежить від мови, тобто розробник може вибрати будь-яку мову програмування (C#, J#, VB тощо), яка найкраще підходить для необхідної програми.
- Типи даних Common Language Specification у всіх програмах .NET однакові, тому перетворення типів не потрібне, коли викликаються методи C++, C# з Visual Basic або з Vice Versa.
- ASP.NET надає повну підтримку XML, CSS та інших нових, а також усталених веб-стандартів.
- Впровадження стану перегляду допомагає автоматично підтримувати стан елементів керування між зворотніми подіями подіями.
- Завдяки вбудованій конфігурації ASP.NET легко розгорнути. Немає необхідності реєструвати компоненти, оскільки інформація про конфігурацію вбудована.
- Нарешті, ASP.NET зменшує кількість коду, необхідного для розробки великих додатків.

2.3 Технології для додатку Dynamo Studio

Єдина технологія за допомогою якої можна створити додаток для Dynamo Studio – це **Dynamo Studio API**. Даний API містить такі пакети:

- ZeroTouchLibrary - пакет для побудови zero touch вузлів Dynamo, який містить наступні бібліотеки: DynamoUnits.dll, ProtoGeometry.dll.
- WpfUILibrary – пакет для створеня UI для вузлів: DynamoCoreWpf.dll, CoreNodeModels.dll, CoreNodeModelWpf.dll.
- DynamoServices - бібліотека з стандартними сервісам Dynamo.
- Core – пакет який надає доступ до інфраструктури Dynamo: DSIronPython.dll, DynamoApplications.dll, DynamoCore.dll, DynamoInstallDetective.dll, DynamoShapeManager.dll, DynamoUtilities.dll, ProtoCore.dll, VMDataBridge.dll.
- Tests - пакет який надає доступ до тестування інфраструктури Dynamo: DynamoCoreTests.dll, SystemTestServices.dll, TestServices.dll.
- DynamoCoreNodes – пакет для отримання доступу до вузлів стандартної інфрастуктури Dynamo: Analysis.dll, GeometryColor.dll, DSCoreNodes.dll. [16]

Dynamo Studio API потребує .NET платформи, а отже можна використати такі мови програмування як:

- C# - це об'єктно-орієнтована мова програмування, що використовується з веб-службами на основі XML на платформі .NET і призначена для підвищення продуктивності у розробці. C # може похвалитися безпекою типу, збиранням сміття, спрощеними деклараціями типів, підтримкою версій та масштабованістю та іншими функціями, які роблять розробку рішень швидшою та простішою, особливо для COM та веб-служб.
- VB.NET - найсучасніша мова програмування на базі BASIC. VB.NET є першим у сімействі повністю об'єктно-орієнтованих мов, він підтримує всі основні принципи ООП, окрім множинного успадкування.

Хоча існують відмінності між Visual Basic .NET та Visual C # .NET, обидві ці мови - це першокласні мови програмування, які базуються на Microsoft .NET Framework, і вони однаково потужні. Visual Basic .NET - справжня об'єктно-орієнтована мова програмування, що включає нові та вдосконалені функції, такі як успадкування, поліморфізм, інтерфейси та перевантаження. І Visual Basic .NET, і Visual C # .NET використовують загальномовне середовища виконання (CLR). Проблем із продуктивністю між Visual Basic .NET та Visual C # .NET майже не виникає. Visual C # .NET може мати ще кілька "потужних" функцій, таких як обробка некерованого коду, а Visual Basic .NET може бути трохи простіший, надаючи такі функції, як пізнє прив'язування. Єдині принципові відмінності між мовами можна знайти у реалізації інтерфейсів та у декларуванні, піднесенні та обробці подій.

Але все ж таки, мова програмування C# має більш сучасний і зрозумілий синтаксис, що добре для подальшої підтримки додатку. І найважливіше те, що VB.NET не має множинного успадкування, що дуже ускладнить подальшу розробку.

Для створення 2D зображень геометрії було обрано OpenGL. **OpenGL** - це кросмовна , кросплатформна програма інтерфейсу програмування (API) для візуалізації 2D та 3D векторної графіки . API зазвичай використовується для взаємодії з графічним процесором (GPU) для досягнення апаратно прискореного візуалізації.

OpenGL - це бібліотека візуалізації. Що OpenGL не робить, це не зберігає інформацію про "об'єкт". Все, що бачить OpenGL, - це набір трикутників і сітку їх розміщення, за допомогою якої трикутники і можна зобразити. Він не пам’ятає, що розробник намалював лінію в одному місці, а кулю в іншому. Через це загальним способом використання OpenGL є намалювати все, що потрібно для малювання, а потім показати це зображення за допомогою команди заміни буфера, що залежить від платформи. Якщо розробнику потрібно оновити зображення, він малює все заново, навіть якщо потрібно оновити лише частину зображення. Якщо розробник хоче анімувати об’єкти, що рухаються на екрані,

йому потрібен цикл, який постійно очищає та перемальовує екран. Існують методи лише оновлення частини екрана. І розробник може використовувати OpenGL із цими методами. Але сам OpenGL не робить цього у своїй внутрнішній логгіці, розробник повинен пам’ятати, де і що він намалював. Потрібно дуже гарно розуміти , що потребує оновлення, і очищати лише ту частину екрана, яку треба.

Оскільки геометрія, яку потрібно обробляти тріангульована, то ця бібліотека ідельний вибір.

2.4 Тріангуляція та індексація буферу вершин

Створення алгоритму тріангуляції займе багато часу, тому є сенс використати уже готові бібліотеки для цього. Існують такі технології для C#, які можуть допомогти у цьому питанні:

- **Dynamo Mesh Toolkit**. Набір інструментів Dynamo Mesh Toolkit надає інструменти для імпорту симплексів з зовнішніх форматів файлів, створення симплексів з геометричних об'єктів Dynamo і ручної побудови симплексів по їх вершинам і індексам. Бібліотека також надає інструменти для зміни симплексів, відновлення симплексів або вилучення горизонтальних зрізів. Набір інструментів Mesh Toolkit зазвичай використовує і створює елементи, які відрізняються від стандартних елементів Geometry, які поставляються «з коробки» з Dynamo.
- **Triangle.NET –**генерує двовимірні (обмежені) триангуляції Делоне та високоякісні сітки точкових наборів або плоских прямолінійних графіків. Це C # порт Jonatha.

На віддмінну від Mesh Toolkit, Triangle.NET підтримує лише 2D геометрію, що зовсім не підходить для поставленної задачі. І до того ж, оскільки, Triangle.NET не підтримує геометрії Dynamo, потрібно буде витрачати час на її конвертацію.

Тому було обрано саме Dynamo Mesh Toolkit, який працює як і з 2D так і з 3D геометрією, не займає багато місця, а також є бібліотекою, що створена за допомогою Dynamo Studio API, а отже є повнсітю сумісною з Dynamo Studio.

Для індексування буфера вершин буде створено власний алгоритм на основі інформаційного огляду.

2.5 Інсталятор

Для .NET додатків є багато інстяляторів, але найпопулярнішим з них є :

- **WIX Toolset**
- **InstallerShield**

InstallShield - це середовище для налаштування проєкту, сценаріїв та остаточного випуску дистрибутива - як у MSI, так і в різних віртуальних форматах. Це одне з найбільш відомих рішень у галузі створення інсталяторів для платформи Windows. Проєкти можна імпортувати з Microsoft Visual Studio або створювати з існуючих шаблонів. Створюючи проєкт з нуля, можна вибрати один із декількох типів:

- Базовий MSI-проект - використовується технологія Windows Installer
- Проект InstallScript - InstallScript використовується для контролю установки.
- InstallScript MSI-проект - спільне використання Windows Installer і InstallScript для розгортання установки.

WiX toolset - набір інструментів, що дозволяють створювати інсталяційні пакети Windows Installer (.MSI і .MSM) на основі XML -опису.

Якщо потрібно управляти установниками в складному середовищі ідеальний вибір WIX, тому що:

- Визначення параметрів установки зберігаються в форматі XML це дає повний контроль над базовою технологією установки windows; схема XML зазвичай близько слідує схемі бази даних Windows.
- Його легко інтегрувати в автоматизовану збірку
- Частини налаштування можуть бути згенеровані автоматично
- Він дозволяє визначати невеликі багаторазові модулі і управляти складними залежностями між ними.
- Ніяких витрат або проблем з ліцензуванням (до WIX усі повинні були використовувати один "Installshield PC")

Чому формат XML є перевагою? Це дозволяє повністю використовувати системи управління версіями коду. Огляд змін, вивчення історії або навіть злиття змін між гілками - це легкий вітерець. Порівнюючи це з проектами installshield, які є непрозорими двійковими блоками, бачимо гарний плюс.

Що мається на увазі під керуванням складними залежностями: з WIX у нас є ComponentGroup для кожної бібліотеки, організованої в пару wixlibs . Кожна група компонентів посилається на інші групи компонентів, від яких вона залежить, за допомогою символу ComponentGroupRef . Розробникам програми установки додатків потрібно тільки посилатися на групи компонентів прямих залежностей, і wix робитиме rest, слідуючи посиланнях. В результаті введення нової залежності вимагає тільки одного локального зміни. Наші автоматизовані збірки і wix роблять rest, щоб відновити всі налаштування.

Завдяки своїй простоті та автоматизованості було вирішено обрати сам WIX Toolset.

3. ПРОЄКТУВАННЯ ФУНКЦІОНАЛЬНИХ ПРОЦЕСІВ

3.1 Бізнес-логіка технології

Основний випадок використання виглядає наступним чином:

1. користувач створює обліковий запис на веб-сайті за допомогою свого облікового запису Autodesk;
2. користувач використовує свій обліковий запис Autodesk для надання дозволу на використання свого персонального хмарного сховища BIM360 (Обліковий запис Autodesk необхідний для використання Dynamo Studio);
3. користувач створює модель з параметрами в студії Dynamo;
4. користувач додає до моделі вузли (рис. 3.1):
   a. користувач додає Select design node, вводить свій логін та пароль з кроку 1 та використовує вузол для створення дизайну або вибору існуючого;
   b. користувач додає вузол Parameters, з'єднує його з Select design node та з'єднує входи, що визначають параметри моделі;
   c. користувач додає вузол Converter node і з'єднує його з моделлю;
   d. користувач додає вузол Send node і з'єднує його з вузлами з кроку a-c;
5. send node вузол надсилає дані на сервер за допомогою API REST;
6. користувач використовує свій браузер для перегляду дизайну в Інтернеті.

Кроки 1-2 необхідно виконати лише один раз. Після реєстрації користувач може їх опустити і розпочати безпосередньо з кроку 3.

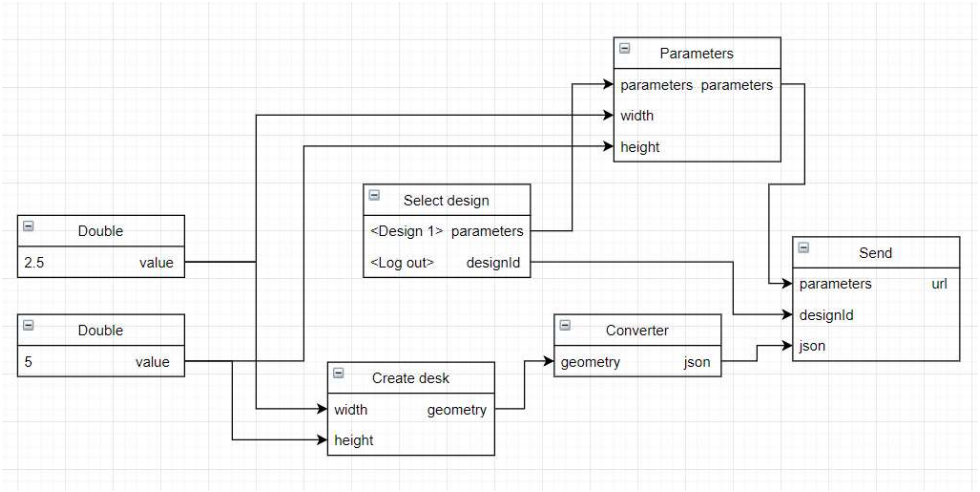

Рисунок 3.1 – Схема вузлів для Dynamo Studio

3.2 Загальна структура

Загалом додаток Dynamo Studio буде відправляти дизайни та його її ітерації до WEB-API сервери, де частину даних будуть зберігатись на сервері, а інша частина (3D моделі та зображення) на хмарі BIM360. WEB-сайт отримуватиме дані від WEB-API серверу, який буде брати частину даних у себе, а іншу на хмарі (рис. 3.2).

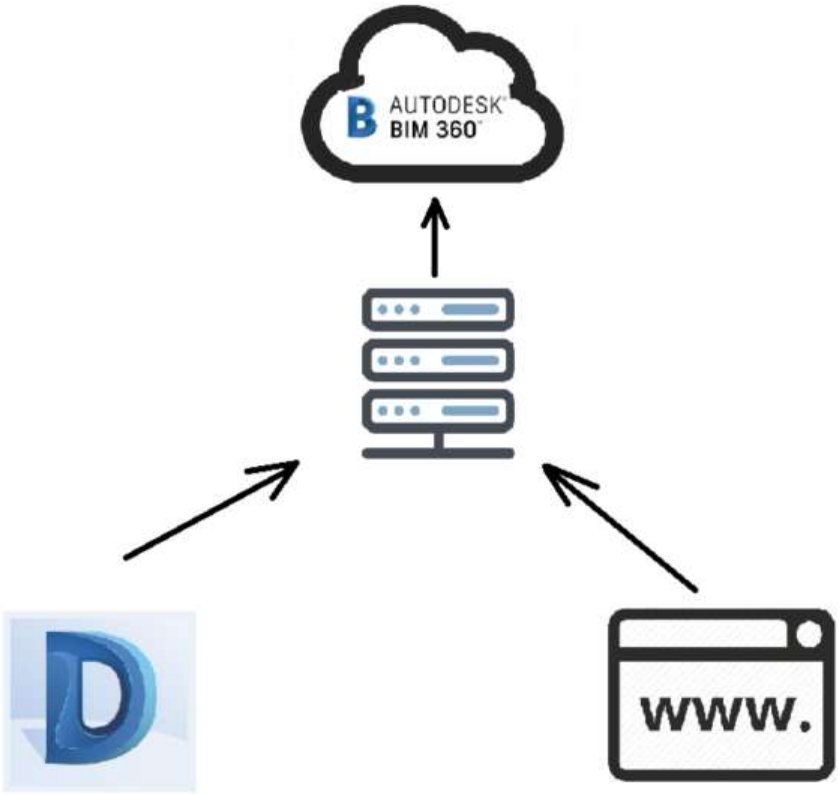

Рисунок 3.2 – Загальна структура

Частина даних, яка зберігається на сервері – це CSV-файл дизайну, зі списком ітерацій, параметрами дизайну та ідентифікаторами зображення та 3D моделі для хмари BIM 360. Це зроблено з метою зручного отримання даних ітерацій для WEB-сайту. Оскільки їх буде легше впровадити для коду Design Explorer, який пряцює з CSV-файлами.

3.3 Проєктування бази даних

WEB-api сервер буде мати базу даних. Було вирішено створити наступні сутності:

1. Users – користувачі;
    a. Id – ідентифікатор користувача ;
    b. FirstName –ім’я користувача;
    c. LastName – прізвище користувача;
    d. Emai – електронна пошта користувача;
    e. RefreshToken – маркер оновлення Autodesk;
    f. AccessToken – маркер доступу Autodesk;
    g. Expires – термін дії маркеру доуступу;
    h. Folder – ідентифікатор дерикторії хмари BIM360;
    i. Project - ідентифікатор проєкту хмари BIM360;
    j. UserName – юзернейм користувача.
2. Roles – ролі;
    a. ID – ідентифікатор ролі;
    b. Name – назва ролі.
3. UserRoles – привязка користувача до ролі;
    a. UserId – ідентифікатор користувача;
    b. RoleId – ідентифікатор ролі.
4. Designs – дизайни;
    a. Id – ідентифікатор дизайну;
    b. UserId – ідентифікатор користувача;
    c. Path – шлях для збереження CSV файлів зі списком ітерацій на сервері;
    d. Folder – ідентифікатор дерикторії на хмарі BIM360;
    e. StartDate – дата створення;
    f. ChangeDate – дата останньої зміни.
5. ShareLinks – посилання на дизайн, яким поділився користувач;

a. GUID – унікальний строковий ідентифікатор, який служить для створення посилання, ідентифікатор посилання;
b. DesignId – ідентифікатор дизайну.

6. RefreshTokens – маркери оновлення авторизаційних маркерів доступу;
   a. Id – ідентифікатор токену;
   b. Subject – опис, додаткова інформація;
   c. ClientId – ідентифікатор клієнта(браузер, додаток);
   d. IssuedUtc – дата створення;
   e. ExpiresUtc – дата закінчення терміну дії;
   f. ProtectedTicket – маркер.

Взаємозв'язок сутностей можна побачити на ER-діаграмі(рис. 3.3).

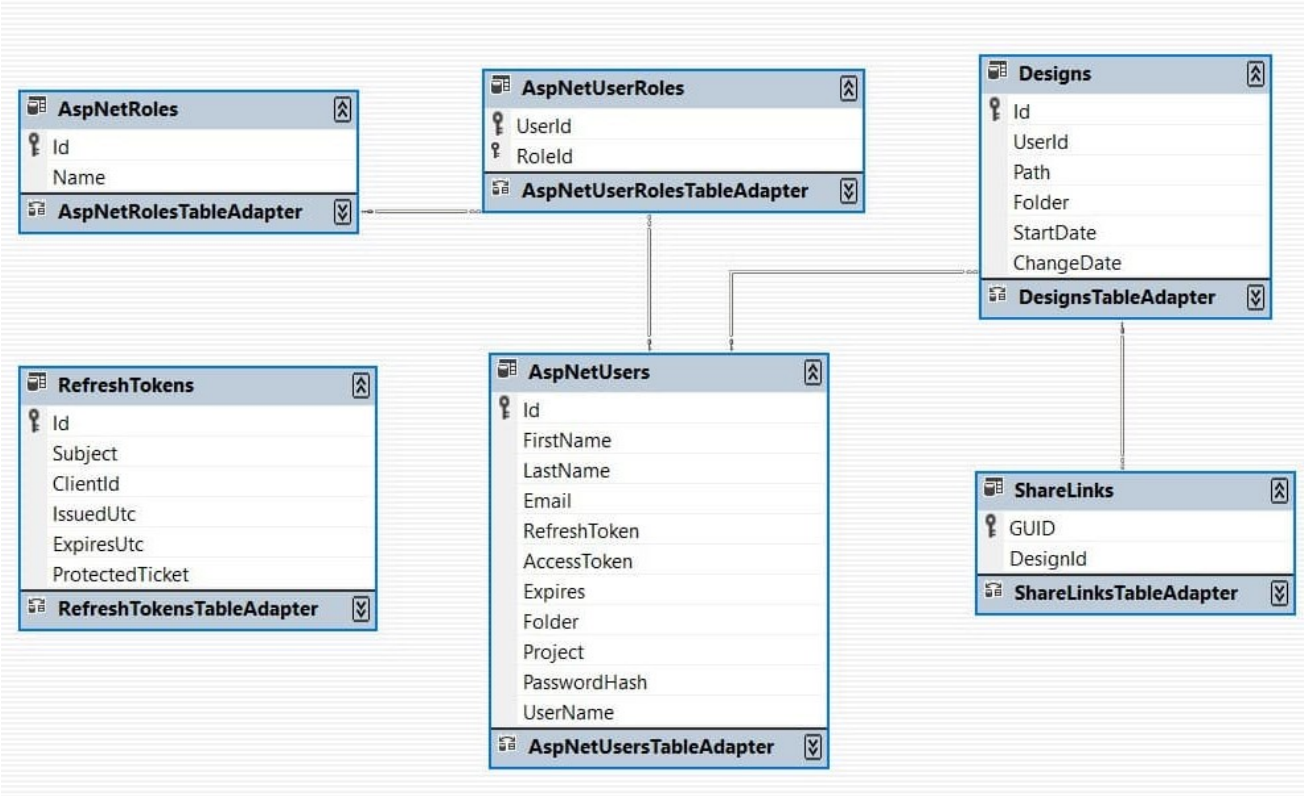

Рисунок 3.3 – ER-діаграма

3.4 Проєктування веб-додатку

WEB-API сервер

Для WEB-api серверу потрібні наступні маршрути, які повинен використовути WEB-сайт для реалізації постановки задачі інформаційної технології

Таблиця 3.1 — Маршрути API

Шлях	**Тип**	**Опис**	**Авторизація**
/api/v1/authentication/gettoken	POST	Повертає маркер доступу	Forge client id
/api/v1/authentication/refreshtoken	POST	Оновлює маркер доступу	Refresh token
/api/v1/authentication/revoketoken	POST	Видяляє маркер оновлення	Refresh token
/api/v1/users	POST	Створює користувача	Forge client id
/api/v1/users	GET	Повертає список усіх користувачів	Admin
/api/v1/users/<user_id>	DELETE	Видаляє користувача	Admin
/api/v1/users/<user_id>/designs	GET	Повертає список дизайнів конкретного користувача	Access token
/api/v1/users/<user_id>/designs	POST	Створює дизайн	Access token
/api/v1/users/<user_id>/designs /<design_id>	GET	Повертає дизайн користувача за ідентифікатором	Access token
/api/v1/users/<user_id>/designs /<design_id>	DELETE	Видаляє дизайн за ідентифікатором	Access token
/api/v1/users/<user_id>/designs /<design_id>/iterations	POST	Створює ітерацію дизайну	Access token
/api/v1/users/<user_id>/designs /<design_id>/iterations/<iteration_id>	DELETE	Видаляє ітерацію дизайну	Access token
/api/v1/users/<user_id>/designs /<design_id>/share	POST	Створює посилання на дизайн	Access token
/api/v1/users/<user_id>/designs /<design_id>/share	DELETE	Видаляє посилання на дизайн	Access token

WEB-сайт

Для WEB-сайту було вирішено створити наступну модель контролерів, яку наведено на рисунку 3.4.

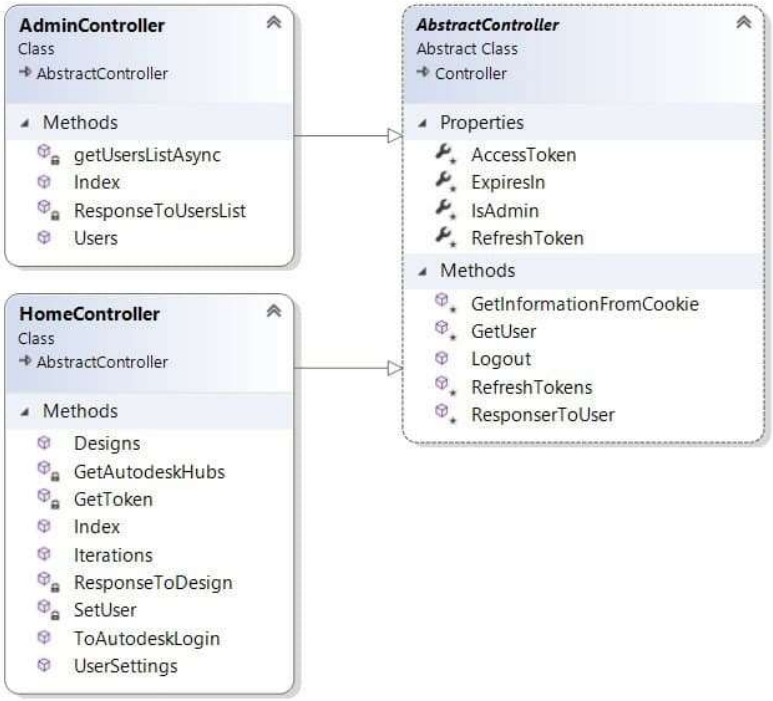

Рисунок 3.4 – Діаграма класів WEB-сайту

Використані такі контролери:

- AbstractController – спільний контролер для звичайних користувачів та адміністраторів, містить у собі дані для авторизації та методи їх керуванням.
- HomeController – контролер для усіх користувачів, який відповідає за усі користувацькі сторінки та їх керування.
- AdminController – допоміжний контролер для адміністраторів, який розширює можливості звичайного користувача.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Реалізація веб-додатку

WEB-API сервер

API обробляє всі операції в додатку. Як веб-сайт, так і надбудова використовують API для створення або зчитування даних. Він створений за допомгою технології ASP.NET WEB API.

Для авторизації використовуються веб-маркери JSON. API генерує маркери доступу з обмеженим проміжком часу та оновлює маркери з довгостроковим проміжком часу. У кожного користувача може бути кілька токенів оновлення. Кожен маркер оновлення представляє сеанс у програмі (тому веб-сайт та надбудова Dynamo Studio використовують різні маркери оновлення).

Для валідації маркерів було створено кастомний OAuth провайдер, який наслідується від OAuthAuthorizationServerProvider, який надає OWIN. Для того, щоб він обробляв маркери потрібно навісити атрибут авторизації([Authorize]) для методів, які цього потребують.

Створюються маркери за допомогою кастомного JWTFormat провайдеру, що наслідується від ISecureDataFormat<AuthenticationTicket>, який також надає OWIN. В маркер заносяться дані користувача та термін дії маркеру. Усі ці дані кодуються у Base64Url за допомогою семитричного ключа. Маркер також має підпис, щоб його не можна було підмінити зловмисникам.

Маркери оновлення також мають кастомний провайдер, який наслідується від IAuthenticationTokenProvider, що надає OWIN. Він служить лише для оновлення маркеру доступу.

Усі дані зберігаються в локальній базі даних (окрім маркеру доступу). 3D-моделі зберігаються в особистому хмарному сховищі BIM360 користувача.

Доступ до функцій API обмежений ролями. Реалізація ролей була досягнена за допомогою OWIN та стандартних сутностей аутентифікації ASP.NET для баз даних. Ролі також прописуються у маркер доступу. Для

розділення методів по ролям використовується параметер ролей для атрибуту авторизації ([Authorize(Roles = "Admin")]).

Також для керування користувачами та ролями були створені кастомні менеджери на основі менеджерів OWIN Identity.

Керування базою даних вібувається за допомогою Entity Framewok. Для того, щоб працювати з нею були створені та додані моделі до контексту данних. Щоб використатовувати стандартні сутностості (Users, Roles і т.п.) для аутентифікації можна наслідувати створений контекст даних (DataContext) від IdentityDbContext. Для того щоб зв'язати контекст даних з базою даних, а також додати зв'язок між базою даних і додатокм потрібно прописати строку в файлі конфігурації. Таким чином додаючи, змінюючи, видаляючи дані з контексту будуть змінюватись дані і в прописаній у конфігураціях базі даних.

```
<add name="DataContext" connectionString="Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='DynamoDesignExplorerStore.mdf';Integrated
Security=True" providerName="System.Data.SqlClient" />
```

Для тестування використано локальну базу даних, дані якої збергаються в спеціальному файлі mdf. Entity Framework також створює потрібні таблиці при їх відсутності.

Більш детальна роль WEB-api серверу буде розглянута далі у наступних пунктах програмної реалізації.

WEB-сайт

Він створений за допомгою технології ASP.NET MVC. Для веб-додатку було створено такі ролі:

- Гість - може переглянути дизайн за посиланням.
- Користувач - може переглядати / створювати / змінювати / видаляти власні дизайни.
- Адміністратор - може переглядати / видаляти користувачів, може переглядати / створювати / змінювати / видаляти всі дизайни.

Також були створені такі сторінки

1. Домашня сторінка - цільова сторінка з описом програми (рис. 4.1)

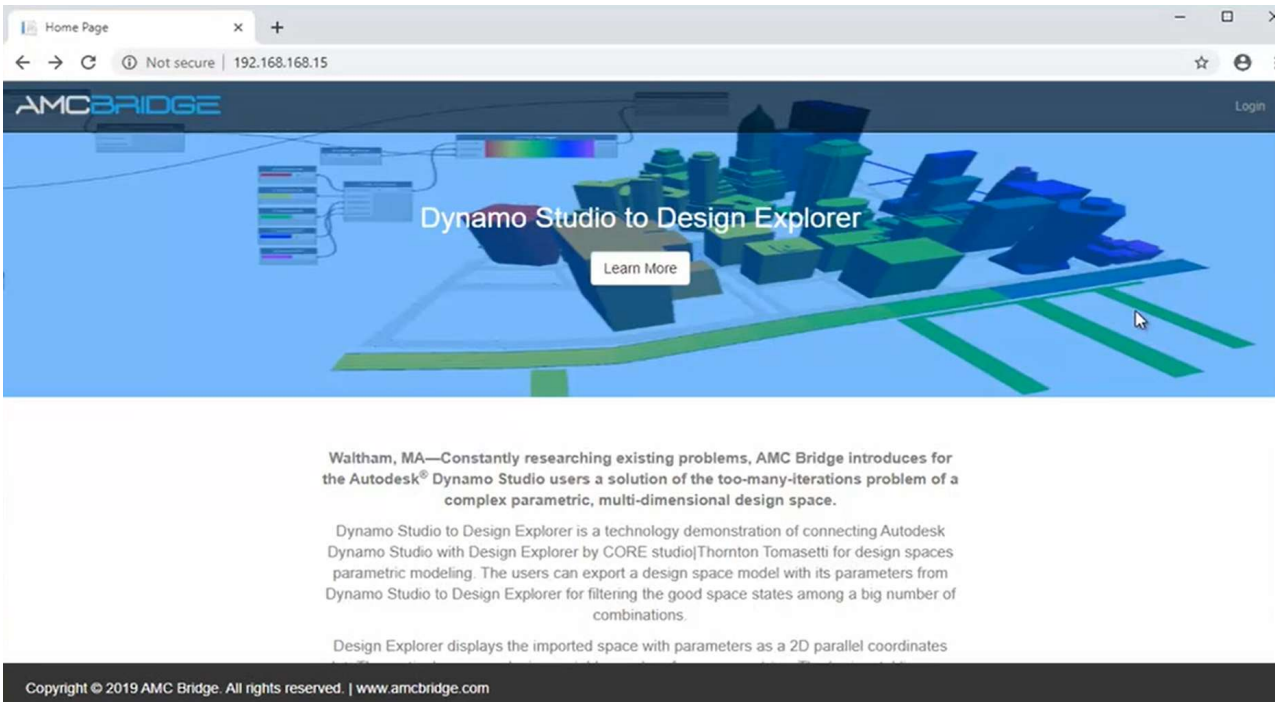

Рисунок 4.1 – Домашня сторінка

2. Сторінка «Вхід» (перенаправлення на сторінку авторизації Autodesk) (рис. 4.2)

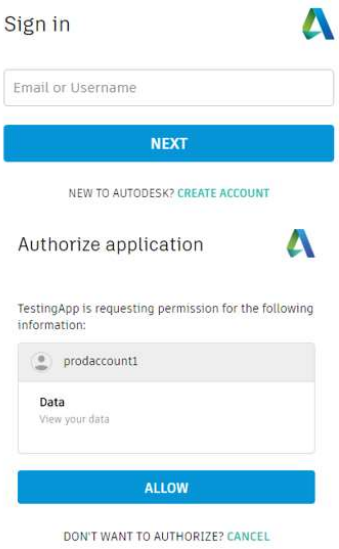

Рисунок 4.2 – Сторінка для входу Autodesk

Для створення соціальної авторизації Autodesk, потрібно створити програму на Autodesk Forge. Саме ця програма буде обробляти запити та повертати маркери доступу та оновлення, які будут використовуватись для

доступу до BIM360. Для початку потрібно мати Autodesk акаунт, і після входу на сайті Autodesk Forge в панелі навігації потрібно обрати «Мої програми» (рис. 4.3)

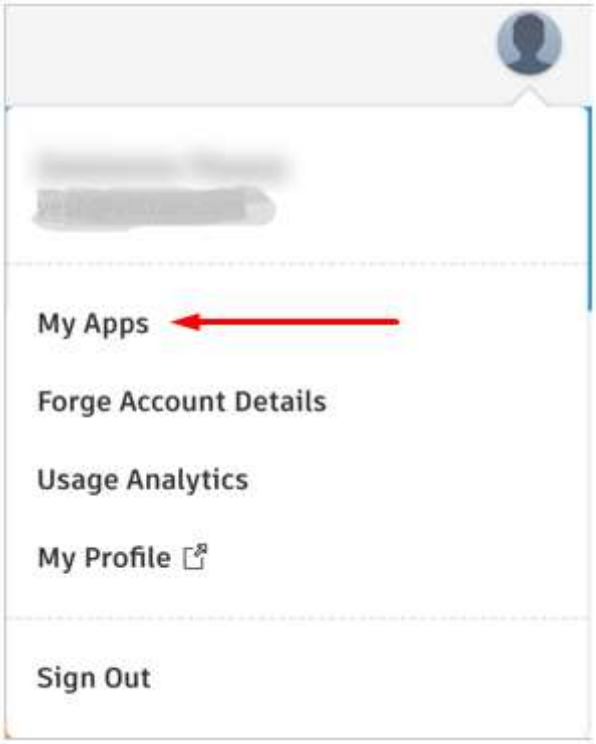

Рисунок 4.3 – Панель навігації Autodesk Forge

Далі на сторінці «Мої програми» потрібно натиснути кнопку «Створити програму» (рис. 4.4)

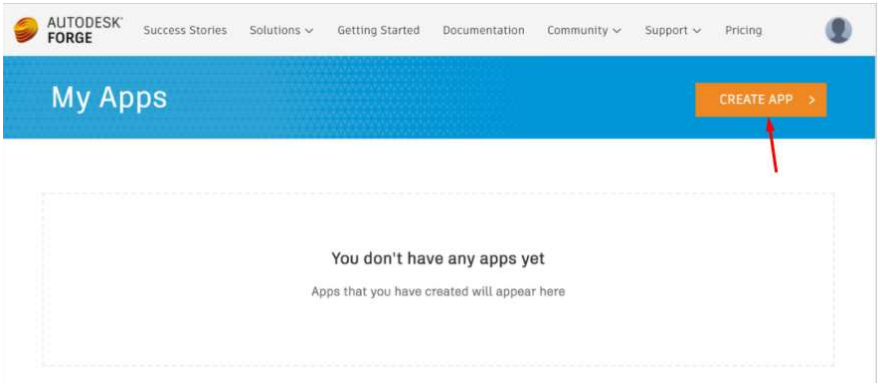

Рисунок 4.4 – Сторінка «Мої програми» Autodesk Forge

Після цього розробнику запропонується обрати, які API потрібно підтримувати, в нашому випадку обираємо лише BIM360 API та Data Management API (рис. 4.5).

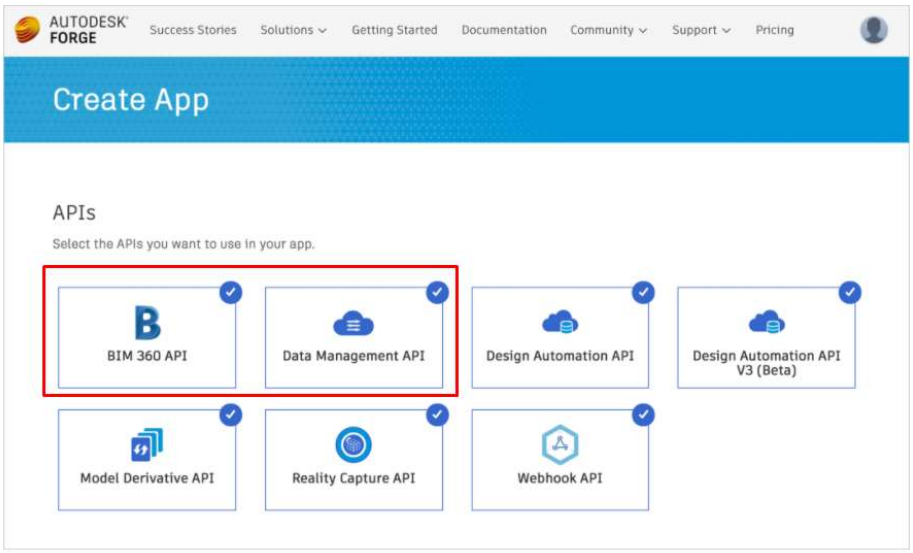

Рисунок 4.5 – Вибір API Autodesk Forge

І залишається лише створити назву програми та прописати зворотнє посилання (рис. 4.6). На зворотнє посилання будуть надходити маркери доступу та оновлення у випадку 2-етапної аутентифікації, та код для отримання маркерів у впиадку 3-етапної аутентифікаціїї. Було вирішено використовувати 3-етапну, адже вона безпечніша. І звісно зворотнє посиллання буде посиланням на розробляємий сайт, який у свою чергу передасть код для отримання маркерів на WEB-api сервер. Після отримання маркерів для Autodesk Forge сервер створить маркери для доступу сайту до серверу. Адже «спілкуватись» далі з Autodesk буде лише WEB-api сервер.

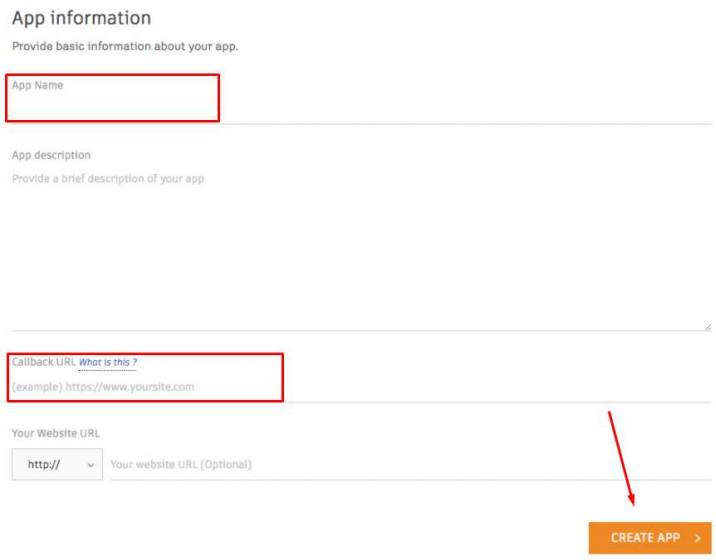

Рисунок 4.6 – Створення програми Autodesk Forge

Для того щоб Forge розумів для якої саме програми користувач буде використовуввати авторизацію, він надає клієнтський ідентифікатор та секрет (рис. 4.7). Клієнтський ідентифікатор буде прописані у запиті на авторизацію, а секрет потрібен як додатковий елемент на запит на меркери.

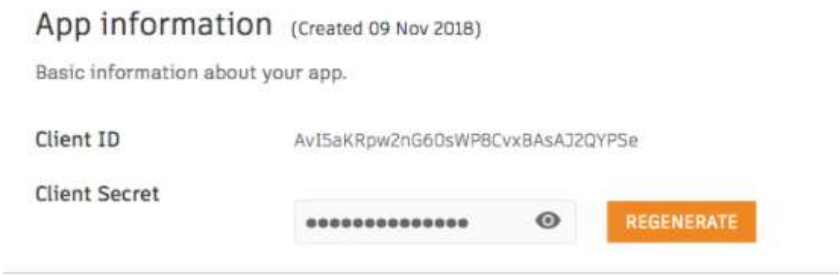

Рисунок 4.7 –Інформація про програму Autodesk Forge

Ось приклад посилання для авторзації після якої повертається код для отримання маркерів:

```
curl -v
'https://developer.api.autodesk.com/authentication/v1/authorize'
-X 'GET'
-H 'Content-Type: application/x-www-form-urlencoded'
-d '
  client_id=rvTqccvG9PPDPdE4P6tfajoAtiX2pb1L&
  response_type=code&
  redirect_uri=http%3a%2f%2flocalhost%3a61210&
  scope=data%3aread+data%3awrite+data%3acreate
'
```

І використовуючт отриманий код, створюється запит на WEB-api сервер, який робить запит на Forge на отримання маркерів доступу та маркерів оновлення:

```
curl -v 'https://developer.api.autodesk.com/authentication/v1/gettoken'
 -X 'POST'
 -H 'Content-Type: application/x-www-form-urlencoded'
 -d '
   client_id= rvTqccvG9PPDPdE4P6tfajoAtiX2pb1L&
   client_secret=eUruM8HRyc7BAQ1e&
   grant_type=authorization_code&
   code=wroM1vFA4E-Aj241-quh_LVjm7UldawnNgYEHQ8I&
   redirect_uri= http%3a%2f%2flocalhost%3a61210 '
```

Далі WEB-api сервер створює власні маркери оновлення та доступу для того щоб користувач мав можливіть посилати аторизовані запити на нього. Це робить за допомогою OAuth провайдкра та технології OWIN, що належать до технолгій ASP.NET Identity. Саме створений кастомний OAuth провайдер потім валідує аутентифікацію користувача.

Після авторизації користувач буде перенаправлений на сторінку «Дизайни». Якщо це була перша атворизація, користувачеві запропонується налаштувати хмару BIM 360 (рис. 4.8)

Рисунок 4.8 – Сторінка “Дизайнів” (Вибір хабу BIM360)

Спочатку користувач повинен обрати хаб хмари а потім, із запропонованих в хабі, проєкт (рис. 4.9)

Рисунок 4.9 – Сторінка “Дизайнів” (Вибір проєкту BIM360)

Сама структура BIM360 складаєтсья з хабів, які у свою чергу містять проєкти, в яких існують файли або дерикторії. Усі запити сайт посилає на WEB-api сервер за допомогою AJAX, а свою чергу сервер робить запити на Forge і повертає відповіді.

Для отримання списку хабів виористовується наступний запит

```
curl -v 'https://developer.api.autodesk.com/project/v1/hubs'
  -X 'GET'
```

```
 -H ' Authorization : Bearer [autodesk_access_token]'
```

А щоб отримати список проєктів в хабів потрібно створити такий запит

```
curl -v 'https://developer.api.autodesk.com/project/v1/hubs/[hub_id]/projects'
 -X 'GET'
 -H ' Authorization : Bearer [autodesk_access_token]'
```

Після налаштування, сторінка «Дизайнів» містить перелік дизайнів, створених користувачем (рис. 4.10)

- Сторінка є не доступною для несанкціонованих користувачів
- Користувач може перейти на сторінку дизайну, видалити дизайн

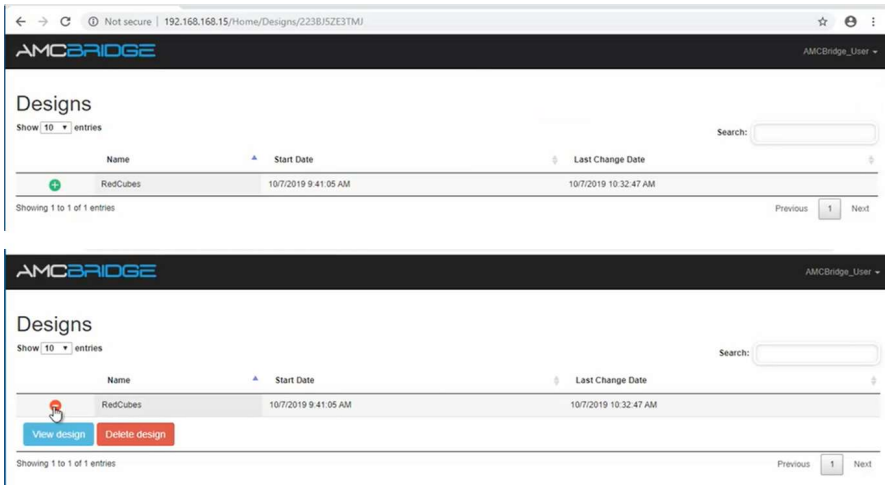

Рисунок 4.10 – Сторінка “Дизайнів”

Список дизайнів отримується наступним чином, запитом на WEB-api сервер

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs'
 -X 'GET'
 -H ' Authorization : Bearer [host_access_token]'
```

Видалення дизайну

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs/[design_id]'
 -X 'DELETE'
 -H ' Authorization : Bearer [host_access_token]'
```

Оскільки дизайн містить ітерції, то їх потрібно також видалити, більше про видалення ітерацій буде далі. Але також після їх видалення потрібно також позбутися і дерикторії дизайну

```
curl -v 'https://developer.api.autodesk.com/data/v1/projects/[project_id]/folders/[folder_id]’
 -X 'DELETE'
 -H ' Authorization : Bearer [autodesk_access_token]
   Content-type: application/vnd.api+json'
-d '[Body]’
```

Тіло запиту задане у JSON форматі і має на ступний паттерн

```
{
 "jsonapi": {
  "version": "1.0"
 },
 "data": {
  "type": "folders",
  "id": "[folder_id]",
  "attributes": {
   "hidden": true
  }
 }
}
```

Сторінка «Перегляд дизайну» було вирішено взяти уже готову з фуннкіоналом, який відповідає вимогам поставленої задачі, а саме Design Explorer (рис. 4.11)

- Доступна для всіх, хто має посилання
- Базується на вихідному коді Design Explorer [14]
- Дає можливість дослідити, а також видалити дизайн

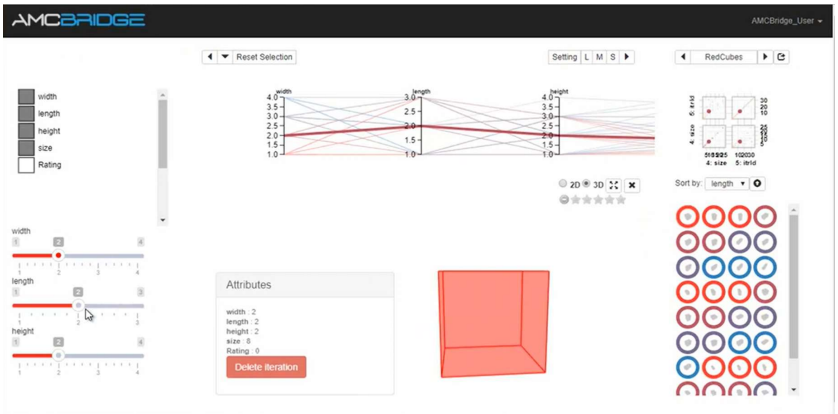

Рисунок 4.11– Сторінка “Ітарацій дизайну”

3D- моедль відображається за допомогою three.js (WebGL) , в даному випадку він приймає геометрію задану у JSON-файлі. Про формат цього файлу можна почитати далі.

Для отримання списку ітерацій виконується запит

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs/[design_id]'
  -X 'GET'
  -H ' Authorization : Bearer [host_access_token]'
```

Цей запит повертає CSV-файл з ітераціям. Його структура була описана раніше.

За допомогую інформації у файлі сайт робить запити для отримання кожного зображення (preview) ітерації на WEB-api, який робить запит на Forge (для отримання файлів BIM 360)

```
curl -X GET -H "Authorization: Bearer [autodesk_access_token]"
"https://developer.api.autodesk.com/oss/v2/buckets/[butcket]/objects/[img]"
```

Після вибору ітерації також робиться запит на отримання 3D-моделі записаної у форматі JSON

```
curl -X GET -H "Authorization: Bearer [autodesk_access_token]"
"https://developer.api.autodesk.com/oss/v2/buckets/[butcket]/objects/[model]"
```

Запит майже анологічний попередньому і також робиться на WEB-api сервер, який уже далі взаємодіє з BIM360

Для видалення ітерації виконується такий запит

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs/[design_id]/terations/[iteration_id]'
 -X 'DELETE'
 -H ' Authorization : Bearer [host_access_token]'
```

Окрім того, що потрібно видалити ітерацію з файлу потрібно видалити файли на BIM360 тому йдуть ще два запити (на видалення зображення там моделі) від WEB-api серверу.

```
curl -v ''https://developer.api.autodesk.com/data/v1/versions'
 -X 'DELETE'
 -H ' Authorization : Bearer [autodesk_access_token]
   Content-type: application/vnd.api+json'
-d '[Body]’
```

Тіло запиту задане у JSON форматі і має на ступний паттерн

```
{
 "jsonapi": {
  "version": "1.0"
 },
 "data": {
  "type": "versions",
  "attributes": {
   "extension": {
    "type": "versions:autodesk.core:Deleted",
    "version": "1.0"
   }
  },
  "relationships": {
   "item": {
```

```
      "data": {
        "type": "items",
        "id": "[ID_FILE]"
      }
    }
  }
 }
}
```

І вже після цього можна вважати видалення ітерації успішним.

Звісно ця сторінка має функціонал, щоб поділитися дизайном (рис. 4.12):

Рисунок 4.12 – Шаринг дизайну

Створюється такий запит

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs/[design_id]/share'
  -X 'GET'
  -H ' Authorization : Bearer [host_access_token]'
```

На сервері створюється посилання типу

```
https://[host] /Home/Iterations/[user_owner_id]/[design_id]/[guid]
```

Сторінки мають навігацію (рис. 4.13):

- Мої дизайни.
- Профіль.
- Вихід.

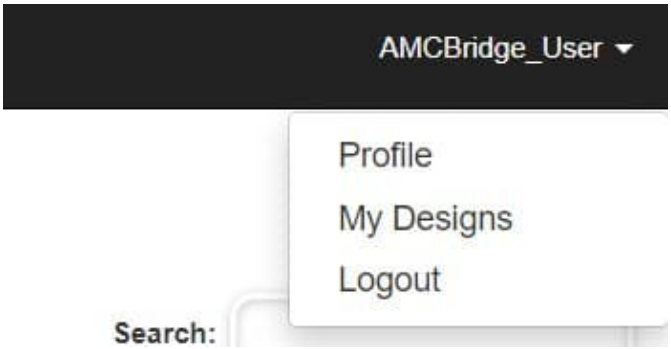

Рисунок 4.13 – Навігація

Сторінка адміністратора містить перелік користувачів (рис. 4.14):

- Сторінка доступна лише для адміністраторів
- Адміністратор може продивитися та має усі права доступу до сторінки «Дизайнів» користувача, а також видалити самого користувача

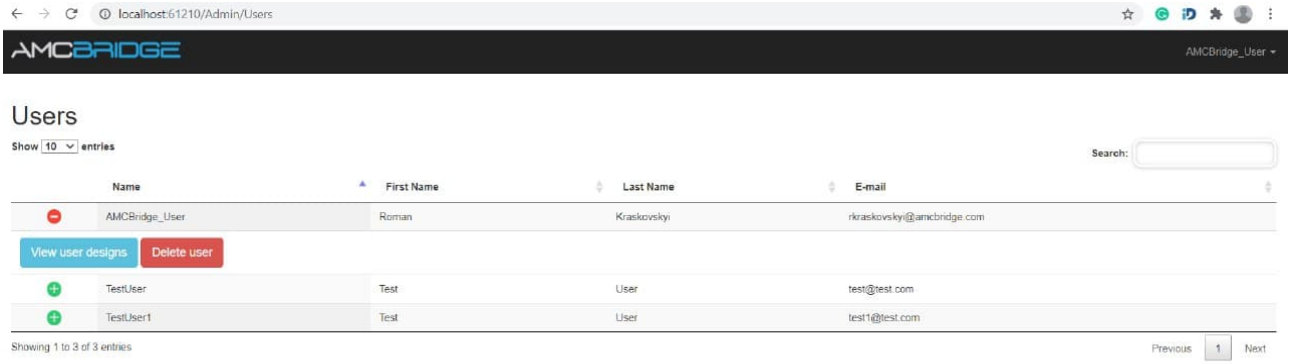

Рисунок 4.14 – Сторінка адміністратора

Список користувачів отримується наступним чином

```
curl -v 'https://[host] /api/v1/users'
  -X 'GET'
  -H ' Authorization : Bearer [host_access_token]'
```

Для видалення користувача виконується наступний запит

```
curl -v 'https://[host] /api/v1/users/[user_id]'
  -X 'DELETE'
  -H ' Authorization : Bearer [host_access_token]'
```

Також була створена сторінка налаштування, де можна подивитись дані користувача, а також змінити налаштування хабу (рис. 4.15). Сторінка доступна лише авторизованим користувачам.

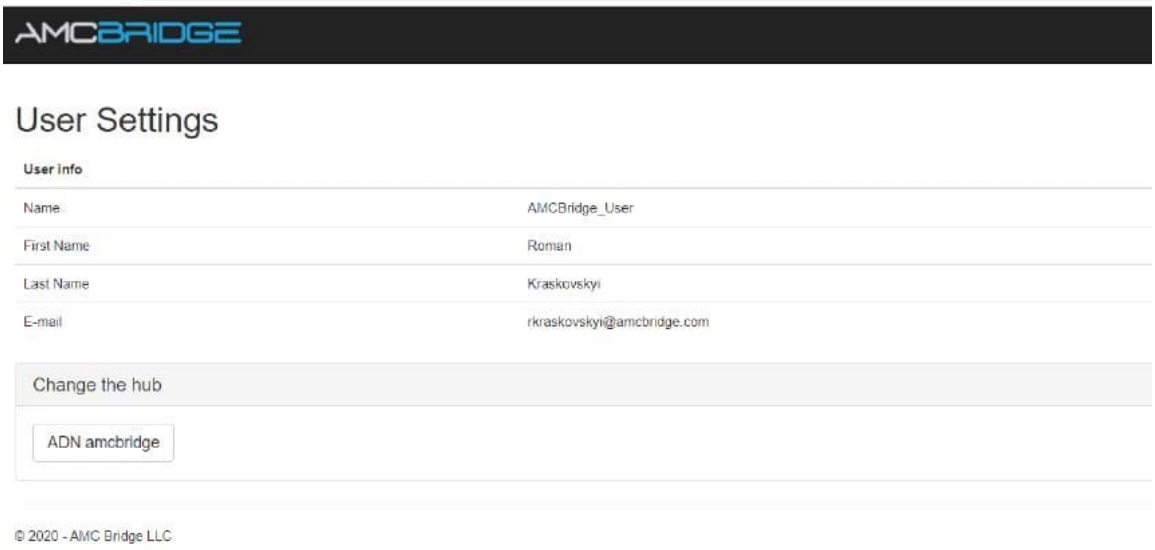

Рисунок 4.15– Сторінка налаштування

На цій сторінці можна побачити особисті дані користувача, їх змінити не можна адже вони прив`язані до данних Autodesk акаунту, якщо користувач хоче їх змінити, то робити це потріюно на сайті Autodesk.

Інформація про користувача отримується так

```
curl -v 'https://[host] /api/v1/users/[user_id]'
  -X 'GET'
  -H ' Authorization : Bearer [host_access_token]'
```

Налаштування хмари аналогічне, налаштуваню при першій авторизації.

4.2 Реалізація додатку Dynamo Studio

Додаток дозволяє створювати дизайни в середовищі Dynamo Studio. Він складається з декількох спеціальних вузлів, описаних нижче.

Таблиця 4.1 — Вузли Dynamo

Вузол	Вхідні дані	Вихідні дані	Опис
Select Design	null	projectId parameter names	Вузол дозволяє увійти та вибрати проект Вузол має додатковий інтерфейс користувача, який стає видимим / невидимим залежно від контексту: кнопка входу, комбінація дизайнів, кнопка виходу Інформацію про вхід (ім'я користувача, ідентифікатор користувача, маркер доступу, оновити маркер) слід додатково зберігати в зашифрованому файлі в папці Temp Створення вузла або натискання кнопки входу повинно викликати процес входу. Він перевіряє інформацію у файлі та використовує її, якщо вона правильна. Інакше відображається форма модальної авторизації Форма авторизації - це елемент управління WebBrowser, який перенаправляє користувача на сторінку авторизації Autodesk Користувач, який увійшов, може вийти (ця дія видаляє файл авторизації) Користувач, який увійшов, може побачити комбобокс зі своїми проектами. У ньому завжди є спеціальний варіант <Створити новий дизайн> Вибір <Створити новий дизайн> показує форму створення модального дизайну Форма створення дизайну дозволяє ввести назву конструкції та вказати її параметри Вузол виводить ідентифікатор обраної конструкції та її параметри Усі параметри - це числа з плаваючою комою, редагувати параметри існуючої конструкції неможливо
Converter	geometry colors	json	Вузол дозволяє перетворити геометрію в формат Design Explorer json
Parameters	parameter names parameter 1 … parameter n	parameter names and values	Вузол дозволяє вводити значення параметрів За замовчуванням у нього є лише один вхід Після підключення входу для вибору вузла проектування для кожного параметра генерується вхід із власним іменем Вихід - це список параметрів ключових значень. Вихідний сигнал дійсний, лише якщо всі входи підключені
Send Design	parameters designId json	url	Проектує ітерацію із входів та надсилають її на сервер (необов'язково) Не надсилається ітерація дизайну в основний потік. Замість цього він додає його до черги. Якщо черга не порожня, вона обробляється у фоновому потоці

Узі вузли створені задопомогою WPF та паттерну MVVM. Model, View, ViewModel (MVVM паттерн) - це шаблон, який складається трьох основних елементвів. Model - просто містить дані і не має нічого спільного з жодною бізнес-логікою. ViewModel - працює як зв’язок між Model та View та робить елементі інтерфейсу інтерактивними. View - просто зберігає відформатовані дані і по суті делегує все до Model..

Select design вузол має наступний вигляд при авторизації (рис. 4.16)

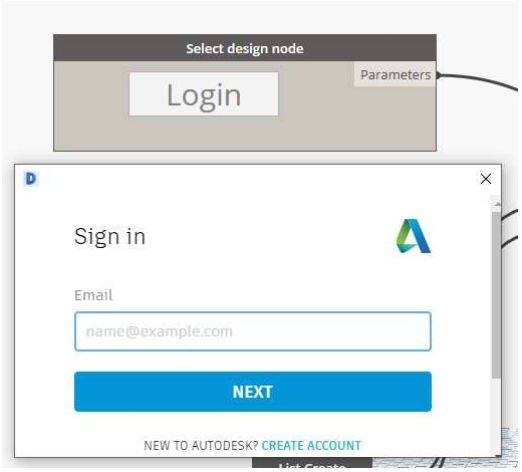

Рисунок 4.16 – Select design вузол, авторизація

Форма логіну являє собою міні-браузер, тому аутентифікація проходить аналогічно WEB-сайту. Дані аутентифікації зберігаються на персональному комп’ютері користувача у зашифрованому вигляді.

Після входу у нього з’являються такі елементи (рис. 4.17):

- Refresh – оновити список існуючих дизайнів
- New design – створити новий дизайн
- Confirm select – пітвердження вибору існуючого дизайну або створення нового
- Search – пошук дизайну по імені
- Logout – вихід

Рисунок 4.17 – Select design вузол

Якщо обрати створення нового дизайну вузол прийме наступний вигляд (рис. 4.18):

- Add – додати параметр;
- Remove – видалити параметр.

Параметри дизайну(не плутати з параметрами вузлів) можуть бути трьох видів:

- in:[параметр] - вхідні (наприклад: ширина);
- out: [параметр] - вихідні (наприклад: площа);
- [параметр] – інформаційні (наприклад: опис).

Вхідні і вихідні параметри дизайну дозволяють створювати графіки та статистичні дані для досліджень на веб-сторінці описаній вище.

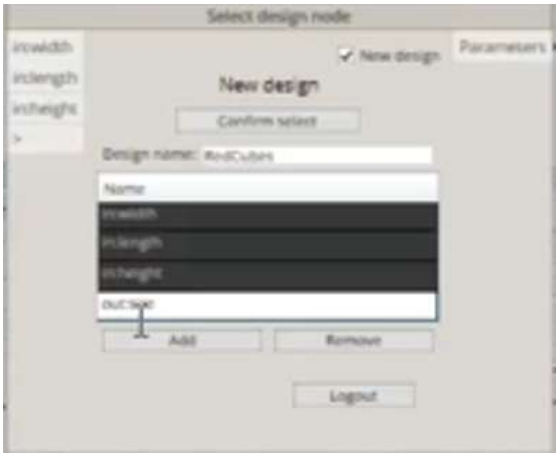

Рисунок 4.18– Select design вузол: створення нового дизайну

Після додавання параметрів, вони з’являються у вхідних параметрах самого вузла. Цей вузол повертає список параметрів з їх значеннями, а також дані про сам дизайн і дані авторизації.

Якщо користувач уже має дизайни, select design вузол матиме наступний вигляд (рис. 4.19):

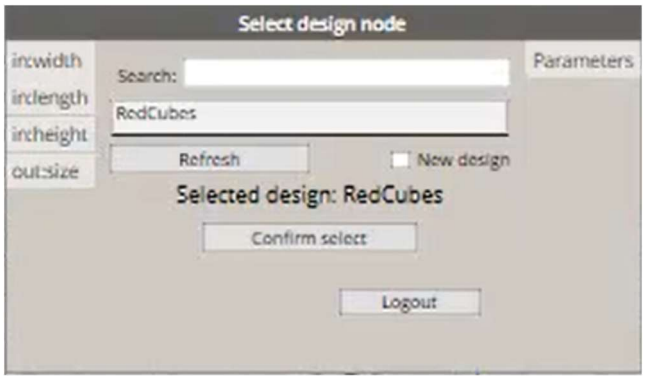

Рисунок 4.19 – Select design вузол з одним дизайном

Як видно, після вибору дизайну автоматично підтягуються його параметри, які вже не можна змінити.

За отримання списку дизайнів відповідальний той самий запит, що і для WEB-сайту. Список параметрів обраного дизайну отримується так

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs/[design_id]/parameters'
-X 'GET'
-H ' Authorization : Bearer [host_access_token]'
```

Converter вузол не має елементів управління, а лише вхідні і вихідні параметри самого вузла (рис. 4.20)

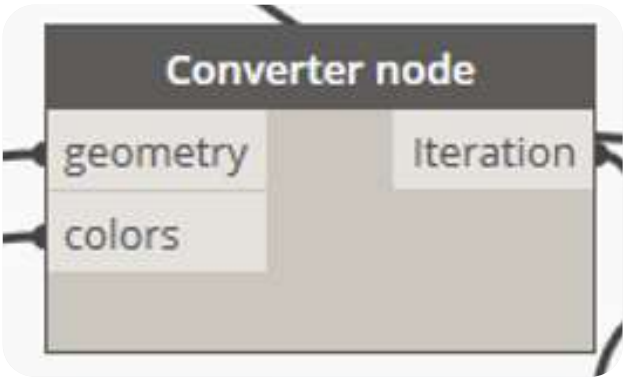

Рисунок 4.20 – Converter вузол

Цей вузол приймає список геометрій і список кольорів, тріангулює, індексує буфер вершин та створює ітерацію у вигляді JSON файлу.

За тріангуляцію відповідає Dynamo Mesh Toolkit. Він розбиває геометрію на точки та з’єндує їх буфером вершин. Індексація буферу вершин описана далі. Точки та індекси вершин заносять до JSON файлу. Також до нього додється інформація про колір та матеріал. Приклад JSON файлу


```
"geometries": [

  {
    "uuid": "0c36bac6-7d5e-41c1-a48c-f152bfa3e807",
    "type": "Geometry",
    "data": {
      "vertices": [ vertices_array ],
      "normals": [],
      "uvs": [],
      "faces": [ faces_array ],
      "scale": 1.0,
      "visible": true,
      "castShadow": true,
      "receiveShadow": false,
      "doubleSided": true
    },
    "materials": null
  },
 "materials": [
  {
    "uuid": "a15fc798-59bb-49e0-849c-25c0a904d90e",
    "name": null,
    "type": "MeshBasicMaterial",
    "color": "0x0000FF",
    "opacity": 1.0,
    "transparent": true,
    "wireframe": false,
    "side": 2
  },
 ],
 "metadata": {
  "type": "Object",
  "version": 4.3,
  "generator": "DynamoJsonExporter"
 },
 "object": {
  "uuid": "f44484ba-14d3-41db-abc5-b96108501b3e",
  "name": null,
  "type": "Scene",
  "matrix": [ matrix ],
  "children": [
    {
      "uuid": "bdc16457-0cb1-4190-8c14-05a078546b02",
      "name": "mesh0",
      "type": "Mesh",
      "matrix": [matrix],
      "children": null,
```

```
      "geometry": "0c36bac6-7d5e-41c1-a48c-f152bfa3e807",
      "material": "a15fc798-59bb-49e0-849c-25c0a904d90e",
      "userData": { "layer": "Default" }
     }
    ],
    “Geometry": null,
    "material": null,
    "userData": { "layers": "Default" }
  }
}
```


Зображення створюється за допомогою OpenGL. До нього подається масив точок, де з них створюється та рендириться геометрія.

Далі йде вузол send design, який відправляє дані на сервер (рис. 4.21)

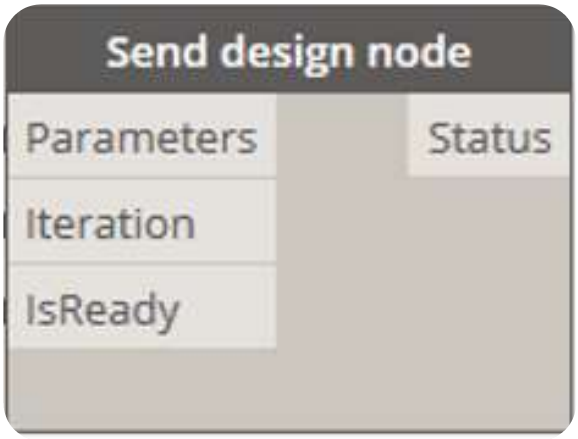

Рисунок 4.21 – Send design вузол

Вхідні дані:

- Parameters – спиоск параметрів, який передає Select design вузол;
- Iteration – ітерація, яку повертає Converter вузол;
- IsReady – це параметр який приймає bool значення, та служить як стопер. для автоматичного режиму Dynamo Studio (в автоматичному режимі вузли спрацьовують при зміні будь-якого параметру)

Якщо створюється новий дизайн, то робиться спочатку такий запит

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs'
  -X 'POST'
  -H ' Authorization : Bearer [host_access_token]'
  -d '[parameters]'
```

Спиок параметер виглядяє як строка, параметри розділені комою.

У свою чергу WEB-api сервер створює CSV файл для ітерацій на сервері та робить запит на створення папки у BIM360 під дизайн.

```
curl -v 'https://developer.api.autodesk.com/data/v1/projects/[project_id]/folders’
 -X 'POST'
 -H ' Authorization : Bearer [autodesk_access_token]
   Content-type: application/vnd.api+json'
-d '[Body]’
```

Тіло запиту задане у JSON форматі і має такий паттерн:

```
{
 "jsonapi": {
  "version": "1.0"
 },
 "data": {
  "type": "folders",
  "attributes": {
   "name": "",
   "extension": {
    "type": "folders:autodesk.bim360:Folder",
    "version": "1.0"
   }
  },
  "relationships": {
   "parent": {
    "data": {
     "type": "folders",
     "id": ""
    }
   }
  }
 }
}
```

Після створення дизайну додаток надсилає ітерації:

```
curl -v 'https://[host] /api/v1/users/[user_id]/designs/[design_id]/iterations’
 -X 'POST'
 -H ' Authorization : Bearer [host_access_token]'
 -d ‘ [Parameters]‘
  --data-binary ‘[img],[model]’
```

В тілі запиту надходять параметри та файли зображення і моделі.

Пісдя цього WEB-api сервер надсилає ці ітерації на BIM360:

```
curl -X POST -H "Content-Type: application/vnd.api+json" -H "Accept: application/vnd.api+json" -H
"Authorization: Bearer nFRJxzCD8OOUr7hzBwbr06D76zAT"
"https://developer.api.autodesk.com/data/v1/projects/b.cGVyc29uYWw6d2l/storage"
-d '[Body]'
```

Тіло запиту задане у JSON форматі і має такий паттерн:

```
{
 "jsonapi": { "version": "1.0" },
 "data": {
```

```
    "type": "objects",
    "attributes": {
      "name": "[iteration_name]"
    },
    "relationships": {
      "target": {
        "data": {
          "type": "folders",
          "id": "[design_folder_id]"
        }
      }
    }
  }
}
```

Це ствоює місце для зберігання файлу. Потім потрібно завантажити в це місце файл:

```
curl -X PUT
-H "Authorization: Bearer nFRJxzCD8OOUr7hzBwbr06D76zAT"
--data-binary [file] "https://developer.api.autodesk.com/oss/v2/buckets/[bucket]/objects/[file_id]"
```

Бакет та ідентифікатор файлу отримується з попереднього запиту. Залишається лише створити першу версію файла:

```
curl -X POST
 -H "Authorization: Bearer nFRJxzCD8OOUr7hzBwbr06D76zAT"
-H "Content-Type: application/vnd.api+json"
 -H "Accept: application/vnd.api+json"
"https://developer.api.autodesk.com/data/v1/projects/b.cGVyc29uYWw6d2l/items"
-d '[Body]'
```

Тіло запиту задане у JSON форматі і має такий паттерн:

```
{
  "jsonapi": { "version": "1.0" },
  "data": {
    "type": "items",
    "attributes": {
      "displayName": "[FILE_NAME]",
      "extension": {
        "type": "items:autodesk.bim360:File",
        "version": "1.0"
      }
```

```
    },
    "relationships": {
      "tip": {
        "data": {
          "type": "versions",
          "id": "1"
        }
      },
      "parent": {
        "data": {
          "type": "folders",
          "id": "[FOLDER_ID]"
        }
      }
    }
  },
  "included": [
    {
      "type": "versions",
      "id": "1",
      "attributes": {
        "name": "[FILE_NAME]",
        "extension": {
          "type": "versions:autodesk.bim360:File",
          "version": "1.0"
        }
      },
      "relationships": {
        "storage": {
          "data": {
            "type": "objects",
            "id": "[OBJECT_ID]"
          }
        }
      }
    }
  ]
}
```


Після цього сервер записує отримані від додатку параметри та отримання від Autodesk Forge посилання на зображення та модель ітерації до CSV- файлу.

Якщо ж потрібно додати ітерацію до вже існуючого дизайну, будуть виконуватись лише запити для створення ітерацій.

Вузол send design повертає статус відправлення. Ось приклад взаємодії вузлів додатку і геометрії Dynamo: звичайний куб з прозорого матеріалу пофарбований у червоний колір (рис. 4.22)

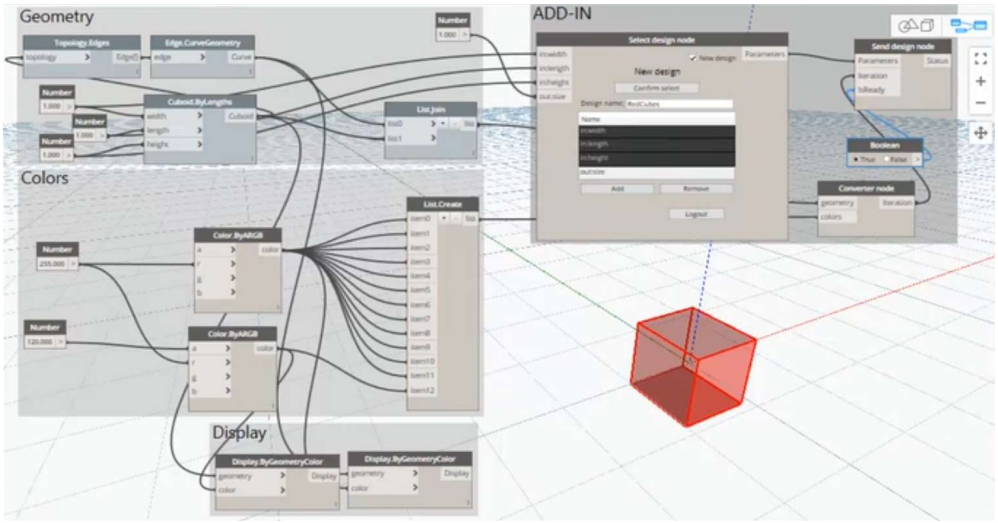

Рисунок 4.22 – Приклад роботи додатку

Також було зроблено інсталятор за допомогою WIX. Він має простий інтерфейс та встановлює додаток у потірбну для Dynamo Studio папку (рис. 4.23)

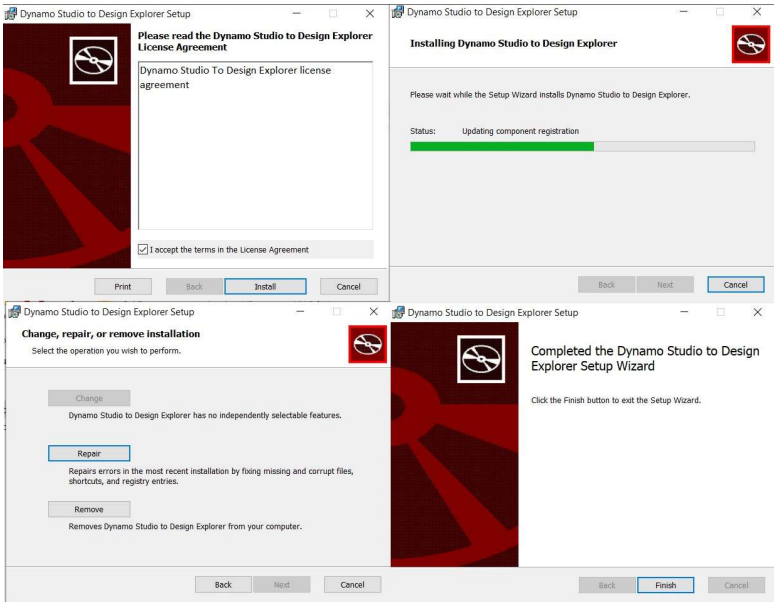

Рисунок 4.23 – Інсталятор

4.3 Індексація буферу вершин

Для опису алгоритму використаємо наступний псевдокод з поясненням

```
faceIndx = 0; // ітератор індексу
VertexBuffer = new array(); // буфер вершин
VertexIndecies = new array() // буфер індексів
for (i = 0; i<points.length; i ++)
{
   if (!VertexBuffer.Contains(points[i])) // якщо точки не існує в буфері
// симплексів
    {
        VertexBuffer.Add(points[i]); // Додати точку до буфера вершин
        VertexIndecies.Add(faceIndx); // Додати індекс до буфера індексів
        faceIndx++; // ітерування індексу
    }
```

```
    else // в іншому випадку
    {
        VertexIndecies.Add(VertexBuffer.IndexOf(point)); // Додати
      // індекс існуючої точки до буфера індексів

    }
}
```

Завдяки цьому алгоритму можна зекономити багато місця на хмарі. Побудуємо гістограму з тестовими даними (рис. 4.24)

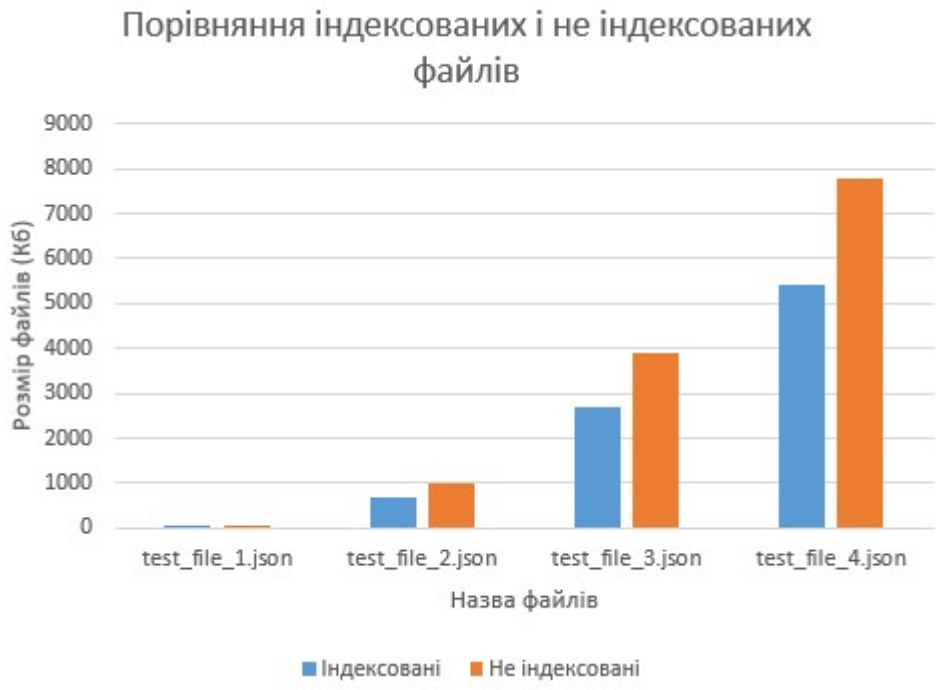

Рисунок 4.24 – Гістограма порівняння

На гістаграмі можна побачити вплив індексації на розмір файлів, і чим більший об’єм геометрії тим більший розрив можна побачити між індексованими вершинами і не індексованими.

В середньому можна зекономити 30% об’єму пам’яті.

ВИСНОВКИ

В результаті виконання роботи було проведено аналітичний огляд, обрано методи вирішення поставленої задачі, спроєктовано функціональні процеси і виконано програмну реалізцію

Розроблено інформаційну технологію керування проєктами візуального програмування для платформи Dynamo Studio. Було створено WEB-api сервер, WEB-сайт та додаток для Dynamo Studio.

Веб-ресурс має привабливий та зручний у використанні інтерфейс, який дозволяє у повний мірі досліджувати параметричні ітерації дизайнів.

3D моделі та зображення надходять с хмари BIM360, тому сервер не матиме проблем зі зберіганням великих об’ємів пам’яті, і при цьому 3D моделі містять у собі тріангульовану геометрію з індексованими буферами вершин, що також зменшує об’єм зберігаємих даних.

В роботі були використані такі технології, як HTML, CSS, JavaScript, а також їх бібліотеки та фреймворки такі, як Bootstrap, Jquery. Були задіяні такі серверні технології, як ASP.NET MVC та ASP.NET WEB API. Для додатку було використано Dynamo Studio API

Як було сказано у вступі, цю систему можна буде використовувати для швидкого створення багатовимірної параметричної моделі дизайну, та її дослідження, що полегшить роботу 3D-проєктуальникам та менеджерам цих проєктів.

Після тестування WEB-частину буде інтегровано до веб-системи компанії AMC Bridge LLC.

СПИСОК ЛІТЕРАТУРИ

1. Переваги ASP.NET [електронний ресурс], URL: https://studopedia.ru/16_2491_perevagi-ASPNET.html

2. Разработка веб систем [електронний ресурс], URL: http://webviki.by/web-system/

3. Bootstrap. Sleek, intuitive, and powerful front-end framework for faster and easier web development [електронний ресурс], URL: http://getbootstrap.com/2.3.2/index.html

4. Ташков, П.А. Веб-мастеринг на 100%: HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка / П. А. Ташков. — 2-е изд., испр. — СПб. : Питер, 2010. —512 с.

5. Design Explorer | CORE studio [електронний ресурс]. URL: http://core.thorntontomasetti.com/design-explorer/

6. Хоган, Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения /Б. Хоган ; пер.Е. Матвеев. — 2-е изд. — СПб. : Питер, 2014. — 320 с.

7. Никольский, А.П. JavaScript на примерах. Практика, практика и только практика, изд. -СПб.: Наука и Техника, 2017. -272 с.

8. What is ASP.NET ? | .NET [електронний ресурс], URL: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet.

9. Симпсон К. ES6 и не только. — СПб.: Питер, 2017. — 336 с.

10. Фрімен А., Pro Entity Framework Core 2 for ASP.NET Core MVC/ А.Фрімен - Каліфорнія, 2018. – 1-5с.

11. Ajax – Вікіпедія [електронний ресурс], URL: https://uk.wikipedia.org/wiki/AJAX

12. Красковський, Р.О. Інформаційний веб-сервіс “Розташування структурних підрозділів Сумського державного університету” [Текст]: робота на здобуття кваліфікаційного ступеня бакалавра; спец.: 6.040302 - інформатика / Р.О. Красковський; наук. керівник В.В. Ємельяненко. - Суми: СумДУ, 2019.

13. Autodesk. Forge Documantation [електронний ресурс]. URL: https://forge.autodesk.com/developer/documentation

14. Design Explorer | GitHub [електронний ресурс]. URL: https://github.com/tt-acm/DesignExplorer

15. Тріангуляція_(геометрія) [електронний ресурс], URL: https://uk.wikipedia.org/wiki/Тріангуляція_(геометрія)

16. Developing for Dynamo [електронний ресурс], URL: https://developer.dynamobim.org/03-Development-Options/3-1-getting-started.html

17. Петковіч Д, Microsoft SQL Server 2019: A Beginner's Guide/ Д. Петковіч –вид 7.- США, 2019 – 3 с.

18. Геффелфінгер Р., Java EE 8 Application Development/ Р. Геффелфінгер – Бірмінгем, 2017 - 1-20с.

19. PHP – Вікіпедія [електронний ресурс]. URL: https://uk.wikipedia.org/wiki/PHP

ДОДАТОК

Код WEB-api серверу

```
using System.Web.Http;

namespace DynamoStudioToDesignExplorerWebApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate:
"{controller}/{action}/{id}/{designId}/{guidShare}",
                defaults: new { id = RouteParameter.Optional, guidShare =
RouteParameter.Optional, designId = RouteParameter.Optional }
            );
        }
    }
}
```

```
using DynamoStudioToDesignExplorerWebApi.Infrastructure;
using DynamoStudioToDesignExplorerWebApi.Models.ModelFactory;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using System.Net.Http;
using System.Web.Http;

namespace DynamoStudioToDesignExplorerWebApi.Controllers.BaseControllers
{
    public class BaseApiController : ApiController
    {
        private ModelFactory _modelFactory;
        private ApplicationUserManager _AppUserManager = null;

        protected ApplicationUserManager AppUserManager
        {
            get
            {
                return _AppUserManager ??
Request.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
        }

        private ApplicationRoleManager _AppRoleManager = null;

        protected ApplicationRoleManager AppRoleManager
        {
            get
            {
                return _AppRoleManager ??
Request.GetOwinContext().GetUserManager<ApplicationRoleManager>();
            }
        }

        public BaseApiController()
        {
        }

        protected ModelFactory TheModelFactory
        {
            get
            {
                if (_modelFactory == null)
                {
                    _modelFactory = new ModelFactory(this.Request,
this.AppUserManager);
                }
                return _modelFactory;
            }
        }

        protected IHttpActionResult GetErrorResult(IdentityResult result)
        {
            if (result == null)
            {
                return InternalServerError();
            }

            if (!result.Succeeded)
```

```
            {
                if (result.Errors != null)
                {
                    foreach (string error in result.Errors)
                    {
                        ModelState.AddModelError("", error);
                    }
                }

                if (ModelState.IsValid)
                {
                    return BadRequest();
                }

                return BadRequest(ModelState);
            }

            return null;
        }
    }
}
```

```
using System;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Web.Configuration;
using System.Web.Http;
using DynamoStudioToDesignExplorerWebApi.Controllers.BaseControllers;
using DynamoStudioToDesignExplorerWebApi.Models;
using DynamoStudioToDesignExplorerWebApi.Models.JsonModels;
using Microsoft.AspNet.Identity;
using Newtonsoft.Json;

namespace DynamoStudioToDesignExplorerWebApi.Controllers
{
    [RoutePrefix("")]
    public class AccountsController : BaseApiController
    {
        private DataContext db = null;

        public AccountsController():base()
        {
            db = new DataContext();
        }

        [HttpPost]
        [Route("users")]
        public async Task<IHttpActionResult> CreateUser()
        {
            string code = await Request.Content.ReadAsStringAsync();
            User user;
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                var content = new StringContent(
                    code,
                    Encoding.UTF8
                );
                var response = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/GetUserInfoFromAutodesk", content);
                string responseString = await
response.Content.ReadAsStringAsync();
                var autodeskUserModel =
JsonConvert.DeserializeObject<AutodeskUserModel>(responseString);
                user = new User()
                {
                    Id = autodeskUserModel.userId,
                    Email = autodeskUserModel.emailId,
                    UserName = autodeskUserModel.userName,
                    FirstName = autodeskUserModel.firstName,
                    LastName = autodeskUserModel.lastName,
                    RefreshToken = autodeskUserModel.refreshToken,
                    AccessToken = autodeskUserModel.accessToken,
                    Expires =
DateTime.Now.AddSeconds(autodeskUserModel.expires - 60)
                };
            }
```

```
            var userResp = db.Users.Find(user.Id);
            if (userResp != null)
            {
                IdentityResult changePasswordResult = await
this.AppUserManager.ChangePasswordAsync(user.Id,
userResp.RefreshToken,user.RefreshToken);
                userResp.RefreshToken = user.RefreshToken;
                await db.SaveChangesAsync();
                if (!changePasswordResult.Succeeded)
                {
                    return GetErrorResult(changePasswordResult);
                }
                Uri locationHeader1 = new
Uri(Url.Link(nameof(GetUserById), new { id = user.Id }));
                return Created(locationHeader1,
TheModelFactory.Create(user));
            }

            IdentityResult addUserResult = await
this.AppUserManager.CreateAsync(user, user.RefreshToken);
            var currentUser = AppUserManager.FindByName(user.UserName);

            this.AppUserManager.AddToRoles(currentUser.Id, new string[] {
"User" });

            if (!addUserResult.Succeeded)
            {
                return GetErrorResult(addUserResult);
            }

            Uri locationHeader = new Uri(Url.Link(nameof(GetUserById),
new { id = user.Id }));

            return Created(locationHeader, TheModelFactory.Create(user));
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        [Route("user/{id}", Name = "GetUserById")]
        public async Task<IHttpActionResult> GetUserById(string Id)
        {
            var user = await this.AppUserManager.FindByIdAsync(Id);

            if (user != null)
            {
                return Ok(this.TheModelFactory.Create(user));
            }

            return NotFound();

        }

        [Authorize(Roles = "Admin")]
        [HttpGet]
        [Route("users")]
        public IQueryable<User> GetUsers()
        {
            return db.Users;
        }
```

```
        [Authorize(Roles = "Admin")]
        [HttpDelete]
        [Route("users/{id}", Name = "DeleteUserById")]
        public IHttpActionResult DeleteUser(string id)
        {
            User user = db.Users.Find(id);
            if (user == null)
            {
                return NotFound();
            }
            foreach (Design design in db.Designs.Where(x => x.UserId ==
id)) {
                db.Designs.Remove(design);
            }
            db.Users.Remove(user);
            db.SaveChanges();
            return Ok(user);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }

        private bool UserExists(string id)
        {
            return db.Users.Count(e => e.Id == id) > 0;
        }
    }
}
```

```
using Autodesk.Forge;
using DynamoStudioToDesignExplorerWebApi.Controllers.BaseControllers;
using DynamoStudioToDesignExplorerWebApi.Models;
using DynamoStudioToDesignExplorerWebApi.Models.JsonModels;
using Microsoft.AspNet.Identity;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Web.Configuration;
using System.Web.Http;

namespace DynamoStudioToDesignExplorerWebApi.Controllers
{
    public class AuthenticationController : BaseApiController
    {
        private DataContext db = null;

        public AuthenticationController() : base()
        {
            db = new DataContext();
        }

        [HttpPost]
        public async Task<AutodeskUserModel> GetUserInfoFromAutodesk()
        {
            var bearer = await GetNewAutodeskTokens();
            using (var client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["AUTODESK__URI"]);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                HttpResponseMessage response = await
client.GetAsync("/userprofile/v1/users/@me");
                string responseString = await
response.Content.ReadAsStringAsync();
                var autodeskUserModel =
JsonConvert.DeserializeObject<AutodeskUserModel>(responseString);
                autodeskUserModel.refreshToken = bearer.refresh_token;
                autodeskUserModel.accessToken = bearer.access_token;
                autodeskUserModel.expires = bearer.expires_in;
                return autodeskUserModel;
            }
        }

        [HttpPost]
        public async Task<dynamic> GetNewAutodeskTokens()
        {
            string code = await Request.Content.ReadAsStringAsync();
            ThreeLeggedApi threeLeggedApi = new ThreeLeggedApi();
```

```
            dynamic bearer = await
threeLeggedApi.GettokenAsync(WebConfigurationManager.AppSettings["FORGE_C
LIENT_ID"],

WebConfigurationManager.AppSettings["FORGE_CLIENT_SECRET"],
"authorization_code", code,
                WebConfigurationManager.AppSettings["REDIRECT_URI"]);
            return bearer;
        }

        [HttpPost]
        private async Task<dynamic> RefreshAutodeskTokens(string id)
        {
            string userId = string.Empty;
            if (id != null)
            {
                userId = id;
            } else
            {
                userId = User.Identity.GetUserId();
            }
            string refreshToken;
            var userResp = db.Users.Find(userId);
            refreshToken = userResp.RefreshToken;
            ThreeLeggedApi threeLeggedApi = new ThreeLeggedApi();
            dynamic bearer = await
threeLeggedApi.RefreshtokenAsync(WebConfigurationManager.AppSettings["FOR
GE_CLIENT_ID"],
            WebConfigurationManager.AppSettings["FORGE_CLIENT_SECRET"],
"refresh_token", refreshToken);
            if (userResp != null)
            {
                await this.AppUserManager.ChangePasswordAsync(userId,
userResp.RefreshToken, bearer.refresh_token);
                userResp.RefreshToken = bearer.refresh_token;
                userResp.AccessToken = bearer.access_token;
                userResp.Expires =
DateTime.Now.AddSeconds(bearer.expires_in - 60);
                await db.SaveChangesAsync();
            }
            return bearer;
        }

        [HttpPost]
        public async Task<IHttpActionResult> GetAutodeskImage(string Id,
int DesignId = -1, string GuidShare = null)
        {
            if (GuidShare == null || DesignId == -1)
            {
                if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
                {
                    return BadRequest("Access denied");
                }
            } else
            {
                var shareLink = db.ShareLinks.Where(sl => sl.GUID ==
GuidShare && sl.DesignId == DesignId).SingleOrDefault();
                if(shareLink == null)
```

```
                {
                    return BadRequest("Access denied");
                }
            }
            string imgUri = await Request.Content.ReadAsStringAsync();
            using (HttpClient client = new HttpClient())
            {
                if (db.Users.Find(Id).Expires.CompareTo(DateTime.Now) ==
-1)
                {
                    dynamic bearer = await RefreshAutodeskTokens(Id);
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                } else
                {
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", db.Users.Find(Id).AccessToken);
                }

                var response = await client.GetAsync(imgUri);
                byte[] arr = await
response.Content.ReadAsByteArrayAsync();
                return Ok(arr);
            }
        }

        [HttpPost]
        public async Task<IHttpActionResult> GetAutodeskJson(string Id,
int DesignId = -1, string GuidShare = null)
        {
            if (GuidShare == null || DesignId == -1)
            {
                if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
                {
                    return BadRequest("Access denied");
                }
            }
            else
            {
                var shareLink = db.ShareLinks.Where(sl => sl.GUID ==
GuidShare && sl.DesignId == DesignId).SingleOrDefault();
                if (shareLink == null)
                {
                    return BadRequest("Access denied");
                }
            }
            string jsonUri = await Request.Content.ReadAsStringAsync();
            using (HttpClient client = new HttpClient())
            {
                if (db.Users.Find(Id).Expires.CompareTo(DateTime.Now) ==
-1)
                {
                    dynamic bearer = await RefreshAutodeskTokens(Id);
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                } else
                {
```

```
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", db.Users.Find(Id).AccessToken);
                }

                var response = await client.GetAsync(jsonUri);
                return await UserChange(Id, response);
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpGet]
        public async Task<IHttpActionResult> GetAutodeskHubs(string id)
        {
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["AUTODESK__URI"]);
                dynamic bearer = await RefreshAutodeskTokens(id);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                var response = await client.GetAsync("/project/v1/hubs");
                return await UserChange(User.Identity.GetUserId(),
response);
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpGet]
        public async Task<IHttpActionResult> GetAutodeskProjects(string
Id, string UserId)
        {
            string hubId = Id;
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["AUTODESK__URI"]);
                var bearer = await RefreshAutodeskTokens(UserId);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                var response = await client.GetAsync("/project/v1/hubs/"
+ hubId + "/projects");
                return await UserChange(User.Identity.GetUserId(),
response);
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        public async Task<IHttpActionResult>
GetAutodeskProjectFolders(string Id)
        {
            string projectIdAndRootFolder = await
Request.Content.ReadAsStringAsync();
            string projectId = projectIdAndRootFolder.Split(';')[0];
            string folderId = projectIdAndRootFolder.Split(';')[1];
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["AUTODESK__URI"]);
```

```
                dynamic bearer = await RefreshAutodeskTokens(Id);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                var response = await client.GetAsync("/data/v1/projects/"
+ projectId + "/folders/" + folderId + "/contents");
                return await UserChange(User.Identity.GetUserId(),
response);
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        public async Task<IHttpActionResult> SetUserAutodeskProject()
        {
            using (DataContext db = new DataContext())
            {
                string projectIdAndRootFolder = await
Request.Content.ReadAsStringAsync();
                string projectId = projectIdAndRootFolder.Split(';')[0];
                string folderId = projectIdAndRootFolder.Split(';')[1];
                db.Users.Find(User.Identity.GetUserId()).Project =
WebConfigurationManager.AppSettings["AUTODESK__URI"] +
                    "/data/v1/projects/" + projectId;
                db.Users.Find(User.Identity.GetUserId()).Folder =
folderId;
                await db.SaveChangesAsync();
            }
            return Ok();
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        public async Task<IHttpActionResult>
CreateUserAutodeskFolder(string id)
        {
            if (User.Identity.GetUserId() != id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            string folderName = await
Request.Content.ReadAsStringAsync();
            User user;
            using (DataContext db = new DataContext())
            {
                user = db.Users.Find(id);
            }
            using (HttpClient client = new HttpClient())
            {
                string jsonDataString = String.Format("{0}",
                    new
StreamReader(HttpContext.Current.Server.MapPath("~/Patterns/create_folder
_pattern.json")).ReadToEnd());
                var folderModel =
JsonConvert.DeserializeObject<FolderModelAutodesk>(jsonDataString);
                folderModel.data.attributes.name = folderName;
                folderModel.data.relationships.parent.data.id =
user.Folder;
```

```
                string jsonData =
JsonConvert.SerializeObject(folderModel);
                var content = new StringContent(
                        jsonData,
                        Encoding.UTF8
                        );
                content.Headers.Remove("Content-Type");
                content.Headers.Add("Content-Type",
"application/vnd.api+json");
                if (db.Users.Find(id).Expires.CompareTo(DateTime.Now) ==
-1)
                {
                    dynamic bearer = await RefreshAutodeskTokens(id);
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                }
                else
                {
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", db.Users.Find(id).AccessToken);
                }
                var response = await client.PostAsync(user.Project +
"/folders", content);
                string responseasdasdas = await
response.Content.ReadAsStringAsync();
                return await UserChange(id, response);
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        public async Task<IHttpActionResult> UploadFileToAutodesk(string
id, string userId)
        {
            if (User.Identity.GetUserId() != userId &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            User user;
            using (DataContext db = new DataContext())
            {
                user = db.Users.Find(userId);
            }
            if (!Request.Content.IsMimeMultipartContent())
            {
                return BadRequest();
            }
            var provider = new MultipartMemoryStreamProvider();
            await Request.Content.ReadAsMultipartAsync(provider);
            string csvInfo = String.Empty;
            string jsonLink = String.Empty;
            string pngLink = String.Empty;
            string jsonAutodeskId = String.Empty;
            string pngAutodeskId = String.Empty;
            foreach (var file in provider.Contents)
            {
                if (file.Headers.ContentDisposition.Name.Trim('\"') ==
"info")
```

```
                {
                    csvInfo = await file.ReadAsStringAsync();
                    continue;
                }
                var filename =
file.Headers.ContentDisposition.FileName.Trim('\"');
                if (Path.GetExtension(filename) != ".png" &&
Path.GetExtension(filename) != ".json")
                {
                    return BadRequest("Only png and json format");
                }
                byte[] fileArray = await file.ReadAsByteArrayAsync();
                ByteArrayContent binContent = new
ByteArrayContent(fileArray);
                using (HttpClient client = new HttpClient())
                {
                    client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/vnd.api+json"));
                    if
(db.Users.Find(userId).Expires.CompareTo(DateTime.Now) == -1)
                    {
                        dynamic bearer = await
RefreshAutodeskTokens(userId);
                        client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                    }
                    else
                    {
                        client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", db.Users.Find(userId).AccessToken);
                    }

                    string jsonDataString = String.Format("{0}",
                        new
StreamReader(HttpContext.Current.Server.MapPath("~/Patterns/create_storag
e_pattern.json")).ReadToEnd());
                    var storageModel =
JsonConvert.DeserializeObject<StorageModelAutodesk>(jsonDataString);
                    storageModel.data.attributes.name =
Path.GetFileName(filename);
                    storageModel.data.relationships.target.data.id = id;
                    string jsonData =
JsonConvert.SerializeObject(storageModel);
                    var content = new StringContent(
                            jsonData,
                            Encoding.UTF8
                            );
                    content.Headers.Remove("Content-Type");
                    content.Headers.Add("Content-Type",
"application/vnd.api+json");
                    var response = await client.PostAsync(user.Project +
"/storage", content);

                    if (!response.IsSuccessStatusCode)
                    {
                        return BadRequest("Autodesk storage ERROR:\n" +
response.ToString());
                    }
                    else
```

```
                    {
                        string responseString = await
response.Content.ReadAsStringAsync();
                        var idFileModel =
JsonConvert.DeserializeObject<IdFileModelAutodesk>(responseString);
                        string idFile =
idFileModel.data.id.Split('/')[1];

                        response = await
client.PutAsync(WebConfigurationManager.AppSettings["AUTODESK__URI"] +
"/oss/v2/buckets/wip.dm.prod/objects/" + idFile, binContent);
                        if (!response.IsSuccessStatusCode)
                        {
                            return BadRequest("Autodesk put file
ERROR:\n" + response.ToString());
                        }
                        responseString = await
response.Content.ReadAsStringAsync();
                        var objectModel =
JsonConvert.DeserializeObject<ObjectModelAutodesk>(responseString);
                        string idObj = objectModel.objectId;
                        if
(file.Headers.ContentDisposition.Name.Trim('\"') == "jsonFile")
                        {
                            jsonLink = objectModel.location;
                        }
                        else if
(file.Headers.ContentDisposition.Name.Trim('\"') == "pngFile")
                        {
                            pngLink = objectModel.location;
                        }
                        jsonDataString = String.Format("{0}", new
StreamReader(HttpContext.Current.Server.MapPath("~/Patterns/create_versio
n_pattern.json")).ReadToEnd());
                        var versionModel =
JsonConvert.DeserializeObject<VersionModelAutodesk>(jsonDataString);

versionModel.included.ToArray()[0].relationships.storage.data.id = idObj;

versionModel.included.ToArray()[0].attributes.name =
Path.GetFileName(filename);
                        versionModel.data.attributes.displayName =
Path.GetFileName(filename);
                        versionModel.data.relationships.parent.data.id =
id;
                        jsonData =
JsonConvert.SerializeObject(versionModel);
                        content = new StringContent(
                                jsonData,
                                Encoding.UTF8
                                );
                        content.Headers.Remove("Content-Type");
                        content.Headers.Add("Content-Type",
"application/vnd.api+json");

                        response = await client.PostAsync(user.Project +
"/items", content);
                        if (!response.IsSuccessStatusCode)
                        {
```

```
                            return BadRequest("Autodesk create version
ERROR:\n" + response.ToString());
                        }
                        else
                        {
                            string responseMessage = await
response.Content.ReadAsStringAsync();
                            var idItemModel =
JsonConvert.DeserializeObject<IdFileModelAutodesk>(responseMessage);
                            if
(file.Headers.ContentDisposition.Name.Trim('\"') == "jsonFile")
                            {
                                jsonAutodeskId = idItemModel.data.id;
                            }
                            else if
(file.Headers.ContentDisposition.Name.Trim('\"') == "pngFile")
                            {
                                pngAutodeskId = idItemModel.data.id;
                            }
                            await UserChange(userId, response);
                        }
                    }
                }
            }
            csvInfo += "," + pngLink + "," + jsonLink + "," +
pngAutodeskId + "," + jsonAutodeskId;
            return Ok(csvInfo);
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        public async Task<IHttpActionResult>
DeleteUserFileAutodesk(string id)
        {
            if (User.Identity.GetUserId() != id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            string fileIdAutodesk = await
Request.Content.ReadAsStringAsync();
            using (HttpClient client = new HttpClient())
            {
                if (db.Users.Find(id).Expires.CompareTo(DateTime.Now) ==
-1)
                {
                    dynamic bearer = await RefreshAutodeskTokens(id);
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                }
                else
                {
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", db.Users.Find(id).AccessToken);
                }
                string jsonDataString = String.Format("{0}", new
StreamReader(HttpContext.Current.Server.MapPath("~/Patterns/delete_versio
n_pattern.json")).ReadToEnd());
```

```
                var deleteVersionModel =
JsonConvert.DeserializeObject<DeleteVersionAutodesk>(jsonDataString);
                deleteVersionModel.data.relationships.item.data.id =
fileIdAutodesk;
                string jsonData =
JsonConvert.SerializeObject(deleteVersionModel);
                var content = new StringContent(
                    jsonData,
                    Encoding.UTF8
                    );
                content.Headers.Remove("Content-Type");
                content.Headers.Add("Content-Type",
"application/vnd.api+json");
                string projectUri = String.Empty;
                using (DataContext db = new DataContext())
                {
                    projectUri = db.Users.Find(id).Project;
                }
                var response = await client.PostAsync(projectUri +
"/versions", content);
                return await UserChange(id, response);
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        public async Task<IHttpActionResult>
DeleteUserFolderAutodesk(string id)
        {
            if (User.Identity.GetUserId() != id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            string idDesign = await Request.Content.ReadAsStringAsync();
            using (HttpClient client = new HttpClient())
            {
                string projectUri = string.Empty;
                string folderId = string.Empty;
                using (DataContext db = new DataContext())
                {
                    projectUri = db.Users.Find(id).Project;
                    folderId =
db.Designs.Find(int.Parse(idDesign)).Folder;
                }
                if (db.Users.Find(id).Expires.CompareTo(DateTime.Now) ==
-1)
                {
                    dynamic bearer = await RefreshAutodeskTokens(id);
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", bearer.access_token);
                }
                else
                {
                    client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", db.Users.Find(id).AccessToken);
                }
```

```
                string jsonDataString = String.Format("{0}", new
StreamReader(HttpContext.Current.Server.MapPath("~/Patterns/delete_folder
_pattern.json")).ReadToEnd());
                var deleteFolderModel =
JsonConvert.DeserializeObject<DeleteFolderAutodesk>(jsonDataString);
                deleteFolderModel.data.id = folderId;
                string jsonData =
JsonConvert.SerializeObject(deleteFolderModel);
                var content = new StringContent(
                    jsonData,
                    Encoding.UTF8
                    );
                content.Headers.Remove("Content-Type");
                content.Headers.Add("Content-Type",
"application/vnd.api+json");
                var method = new HttpMethod("PATCH");
                var request = new HttpRequestMessage(method, new
Uri(projectUri + "/folders/" + folderId))
                {
                    Content = content
                };

                HttpResponseMessage response = new HttpResponseMessage();
                try
                {
                    response = await client.SendAsync(request);
                }
                catch (TaskCanceledException e)
                {
                    return BadRequest("AUTODESK DELETE ERROR: " +
e.ToString());
                }

                return await UserChange(id, response);
            }
        }

        private async Task<IHttpActionResult> UserChange(string userId,
HttpResponseMessage response)
        {
            if (response.IsSuccessStatusCode)
            {
                string responseMessage = await
response.Content.ReadAsStringAsync();
                try
                {
                    return Ok(JObject.Parse(responseMessage));
                } catch(JsonReaderException e)
                {
                    return Ok("ok");
                }
            }
            else
            {
                return BadRequest(response.ToString());
            }
        }

}}
```

```
using DynamoStudioToDesignExplorerWebApi.Controllers.BaseControllers;
using DynamoStudioToDesignExplorerWebApi.Models;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using Microsoft.AspNet.Identity;
using System.Web.Configuration;
using System.Text;
using System;
using Newtonsoft.Json;
using DynamoStudioToDesignExplorerWebApi.Models.JsonModels;
using DynamoStudioToDesignExplorerWebApi.Models.Results;
using System.Collections.Generic;

namespace DynamoStudioToDesignExplorerWebApi.Controllers
{
    [RoutePrefix("users")]
    public class DesignsController : BaseApiController
    {

        private DataContext db = null;

        public DesignsController() : base()
        {
            db = new DataContext();
        }

        [Authorize(Roles = "User,Admin")]
        [HttpGet]
        [Route("{id}/designs")]
        public IHttpActionResult GetAllUserDesigns(string Id)
        {
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            return Ok(db.Designs.Where(d => d.UserId == Id));
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        [Route("{id}/designs")]
        public async Task<IHttpActionResult> CreateUserDesigns(string Id)
        {
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            if (!Request.Content.IsMimeMultipartContent())
            {
                return BadRequest();
            }
            var provider = new MultipartMemoryStreamProvider();
```

```
            string root =
HttpContext.Current.Server.MapPath("~/Designs/");
            await Request.Content.ReadAsMultipartAsync(provider);
            string responseContent = "ERROR";

            foreach (var file in provider.Contents)
            {
                var filename =
file.Headers.ContentDisposition.FileName.Trim('\"');
                if (Path.GetExtension(filename) != ".csv")
                {
                    return BadRequest("Only csv format");
                }
                byte[] fileArray = await file.ReadAsByteArrayAsync();

                using (HttpClient client = new HttpClient())
                {
                    client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                    var content = new StringContent(
                    Path.GetFileName(filename),
                    Encoding.UTF8
                    );
                    client.DefaultRequestHeaders.Authorization =
Request.Headers.Authorization;
                    var response = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
                        "/authentication/CreateUserAutodeskFolder/" + Id,
content);
                    if (!response.IsSuccessStatusCode)
                    {
                        return BadRequest("Autodesk ERROR:\n" +
response.ToString());
                    }
                    else
                    {
                        responseContent = await
response.Content.ReadAsStringAsync();
                    }
                }
                if (!Directory.Exists(root + Id))
                {
                    Directory.CreateDirectory(root + Id);
                }
                using (FileStream fs = new FileStream(root + Id + @"\" +
filename, FileMode.Create))
                {
                    await fs.WriteAsync(fileArray, 0, fileArray.Length);
                }
                var idFolderModelAutodesk =
JsonConvert.DeserializeObject<IdFolderModelAutodesk>(responseContent);
                Design design = new Design
                {
                    UserId = Id,
                    Path = root + Id + @"\" + filename,
                    Folder = idFolderModelAutodesk.data.id,
                    StartDate = DateTime.Now.ToString(),
                    ChangeDate = DateTime.Now.ToString()
                };
```

```
                db.Designs.Add(design);
                db.SaveChanges();
                return Ok(design.Id);
            }
            return Ok(responseContent);
        }

        [HttpGet]
        [Route("{id}/designs/{designId}/{guidShare:guid?}")]
        public IHttpActionResult GetUserDesignsById(string Id, int
DesignId,string guidShare = null)
        {
            if (guidShare == null)
            {
                if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
                {
                    return BadRequest("Access denied");
                }
            }else
            {
                ShareLink shareLink = db.ShareLinks.Where(sl => sl.GUID
== guidShare && sl.DesignId == DesignId).SingleOrDefault();
                if (shareLink == null)
                {
                    return BadRequest("Access denied");
                }
            }
            Design design = db.Designs.Where(d => d.UserId == Id && d.Id
== DesignId).SingleOrDefault();
            if (design != null)
            {
                var fileInfo = new FileInfo(design.Path);
                return new FileResult(fileInfo.FullName);
            }
            else
            {
                return BadRequest("Design not found");
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpDelete]
        [Route("{id}/designs/{designId}")]
        public async Task<IHttpActionResult> DeleteUserDesigns(string Id,
int DesignId)
        {
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }

            Design deleteDesign = db.Designs.Find(DesignId);
            if (File.Exists(deleteDesign.Path))
            {
                File.Delete(deleteDesign.Path);
            }
            using (HttpClient client = new HttpClient())
```

```
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                client.DefaultRequestHeaders.Authorization =
Request.Headers.Authorization;
                var content = new StringContent(
                    DesignId.ToString(),
                    Encoding.UTF8
                    );
                var response = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/DeleteUserFolderAutodesk/" + Id, content);
                string responseString = await
response.Content.ReadAsStringAsync();
                if (response.IsSuccessStatusCode)
                {
                    ShareLink shareLink = db.ShareLinks.Where(sl =>
sl.DesignId == DesignId).SingleOrDefault();
                    if(shareLink != null)
                    {
                        db.ShareLinks.Remove(shareLink);
                    }
                    db.Designs.Remove(deleteDesign);
                    db.SaveChanges();
                    return Ok(responseString);
                }
                else
                {
                    return BadRequest("Error: " + responseString);
                }
            }
        }

        [Authorize(Roles = "User,Admin")]
        [HttpPost]
        [Route("{id}/designs/{designId}/iterations")]
        public async Task<IHttpActionResult>
CreateUserDesignsIteration(string Id, int DesignId)
        {
            User user;
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            else
            {
                user = db.Users.Find(Id);
            }
            Design currentDesign = db.Designs.Find(DesignId);
            string contentLine = String.Empty;
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                client.DefaultRequestHeaders.Authorization =
Request.Headers.Authorization;
                var response = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
```

```
"/authentication/UploadFileToAutodesk?id=" + currentDesign.Folder +
"&userId=" + Id, Request.Content);
                if (!response.IsSuccessStatusCode)
                {
                    string error = await
response.Content.ReadAsStringAsync();
                    return BadRequest("Autodesk ERROR:\n" + error);
                }
                else
                {
                    contentLine = await
response.Content.ReadAsStringAsync();
                }
            }
            File.AppendAllText(currentDesign.Path, contentLine.Trim('\"')
+ "\n");
            currentDesign.ChangeDate = DateTime.Now.ToString();
            await db.SaveChangesAsync();
            return Ok("Iteration added");
        }

        [Authorize(Roles = "User,Admin")]
        [HttpDelete]
        [Route("{id}/designs/{designId}/iterations/{iterationId}")]
        public async Task<IHttpActionResult>
DeleteUserDesignsIteration(string Id, int DesignId, int IterationId)
        {
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            Design currentDesign = db.Designs.Find(DesignId);
            string[] readText = File.ReadAllLines(currentDesign.Path);
            string jsonAutodeskId = String.Empty;
            string pngAutodeskId = String.Empty;
            using (StreamWriter file = new
StreamWriter(currentDesign.Path, false))
            {
                for (int i = 0; i < readText.Length; i++)
                {
                    if (i != IterationId)
                    {
                        file.WriteLine(readText[i]);
                    }
                    else
                    {
                        string[] deletedIteration =
readText[i].Split(',');
                        jsonAutodeskId =
deletedIteration[deletedIteration.Length - 1];
                        pngAutodeskId =
deletedIteration[deletedIteration.Length - 2];
                    }
                }
            }
            using (HttpClient client = new HttpClient())
            {
```

```
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                client.DefaultRequestHeaders.Authorization =
Request.Headers.Authorization;
                var content = new StringContent(
                    jsonAutodeskId,
                    Encoding.UTF8
                    );
                var response = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/DeleteUserFileAutodesk/" + Id, content);
                if (!response.IsSuccessStatusCode)
                {
                    string error = await
response.Content.ReadAsStringAsync();
                    return BadRequest("Autodesk ERROR:\n" + error);
                }
                content = new StringContent(
                    pngAutodeskId,
                    Encoding.UTF8
                    );
                response = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/DeleteUserFileAutodesk/" + Id, content);
                if (!response.IsSuccessStatusCode)
                {
                    string error = await
response.Content.ReadAsStringAsync();
                    return BadRequest("Autodesk ERROR:\n" + error);
                }
            }
            currentDesign.ChangeDate = DateTime.Now.ToString();
            await db.SaveChangesAsync();

            return Ok("Iteration deleted");
        }

        [Authorize(Roles = "User,Admin")]
        [HttpGet]
        [Route("{id}/designs/{designId}/parameters")]
        public async Task<IHttpActionResult> GetDesignParameters(string
Id, int DesignId)
        {
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            Design design = db.Designs.Where(d => d.UserId == Id && d.Id
== DesignId).SingleOrDefault();
            if (design == null)
            {
                return BadRequest("Design not found");
            }
            string parametersLine = string.Empty;
            using (StreamReader file = new StreamReader(design.Path))
            {
                parametersLine = file.ReadLine();
            }
```

```
            List<string> parameters = parametersLine.Split(',').ToList();
            parameters.Remove("img");
            parameters.Remove("threeD");
            parameters.Remove("IDAUTODESKPNG");
            parameters.Remove("IDAUTODESKJSON");
            return Ok(parameters);
        }

        [Authorize(Roles = "User,Admin")]
        [HttpGet]
        [Route("{id}/designs/{designId}/share")]
        public async Task<IHttpActionResult> ShareDesign(string Id, int
DesignId)
        {
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            Design design = db.Designs.Where(d => d.UserId == Id && d.Id
== DesignId).SingleOrDefault();
            if (design == null)
            {
                return BadRequest("Design not found");
            }
            ShareLink shareLink = db.ShareLinks.Where(sl => sl.DesignId
== DesignId).SingleOrDefault();
            if (shareLink == null) {
                string guid = Guid.NewGuid().ToString();
                shareLink = new ShareLink()
                {
                    GUID = guid,
                    DesignId = DesignId
                };
                db.ShareLinks.Add(shareLink);
                await db.SaveChangesAsync();
            }
            return Ok(shareLink.GUID);
        }

        [HttpGet]
        [Route("{id}/designs/{designId}/share/{guidShare}")]
        public IHttpActionResult ShareDesignExists(string Id, int
DesignId, string GuidShare)
        {
            Design design = db.Designs.Where(d => d.UserId == Id && d.Id
== DesignId).SingleOrDefault();
            if (design == null)
            {
                return BadRequest("Design not found");
            }
            ShareLink shareLink = db.ShareLinks.Where(sl => sl.DesignId
== DesignId).SingleOrDefault();
            if (shareLink == null)
            {
                return BadRequest("Not shared");
            }
            return Ok("Exists");
        }
```

```
        [Authorize(Roles = "User,Admin")]
        [HttpDelete]
        [Route("{id}/designs/{designId}/share")]
        public async Task<IHttpActionResult> DeleteShareDesign(string Id,
int DesignId)
        {
            if (User.Identity.GetUserId() != Id &&
!User.IsInRole("Admin"))
            {
                return BadRequest("Access denied");
            }
            Design design = db.Designs.Where(d => d.UserId == Id && d.Id
== DesignId).SingleOrDefault();
            if (design == null)
            {
                return BadRequest("Design not found");
            }
            ShareLink shareLink = db.ShareLinks.Where(sl => sl.DesignId
== DesignId).SingleOrDefault();
            if (shareLink != null)
            {
                db.ShareLinks.Remove(shareLink);
                await db.SaveChangesAsync();
            }
            return Ok("Share deleted success");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

```
using DynamoStudioToDesignExplorerWebApi.Models;
using System.Linq;
using System.Threading.Tasks;
using System.Web.Http;

namespace DynamoStudioToDesignExplorerWebApi.Controllers
{
    [RoutePrefix("authentication")]
    public class RefreshTokensController : ApiController
    {
        DataContext db = null;
        public RefreshTokensController()
        {
            db = new DataContext();
        }

        [Authorize(Users = "Admin")]
        [Route("")]
        public IHttpActionResult Get()
        {
            return Ok(db.RefreshTokens.ToList());
        }

        [AllowAnonymous]
        [Route("revoketoken")]
        public async Task<IHttpActionResult> Delete(string tokenId)
        {
            var token = await
db.RefreshTokens.FindAsync(Helper.GetHash(tokenId));
            bool result = false;
            if (token != null) {
                db.RefreshTokens.Remove(token);
                result = await db.SaveChangesAsync() > 0;
            }

            if (result)
            {
                return Ok();
            }
            return BadRequest("Token Id does not exist");

        }
    }
}
```

```
using DynamoStudioToDesignExplorerWebApi.Infrastructure;
using Microsoft.AspNet.Identity.EntityFramework;
using System.Collections.Generic;
using System.Net.Http;
using System.Web.Http.Routing;

namespace DynamoStudioToDesignExplorerWebApi.Models.ModelFactory
{
    public class ModelFactory
    {
        private UrlHelper _UrlHelper;
        private ApplicationUserManager _AppUserManager;

        public ModelFactory(HttpRequestMessage request,
ApplicationUserManager appUserManager)
        {
            _UrlHelper = new UrlHelper(request);
            _AppUserManager = appUserManager;
        }

        public RoleReturnModel Create(IdentityRole appRole)
        {

            return new RoleReturnModel
            {
                Id = appRole.Id,
                Name = appRole.Name
            };
        }

        public UserReturnModel Create(User appUser)
        {
            return new UserReturnModel
            {
                Id = appUser.Id,
                UserName = appUser.UserName,
                FirstName = appUser.FirstName,
                LastName = appUser.LastName,
                Email = appUser.Email,
                RefreshToken = appUser.RefreshToken,
                Project = appUser.Project,
                Folder = appUser.Folder,
                Roles = _AppUserManager.GetRolesAsync(appUser.Id).Result
            };
        }
    }

    public class RoleReturnModel
    {
        public string Id { get; set; }
        public string Name { get; set; }
    }

    public class UserReturnModel
    {
        public string Id { get; set; }
        public string UserName { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
```

```
        public string Email { get; set; }
        public string RefreshToken { get; set; }
        public string Folder { get; set; }
        public string Project { get; set; }
        public IList<string> Roles { get; set; }
    }
}
```

```
using Microsoft.Owin.Security;
using Microsoft.Owin.Security.DataHandler.Encoder;
using System;
using System.Configuration;
using System.IdentityModel.Tokens;
using Thinktecture.IdentityModel.Tokens;

namespace DynamoStudioToDesignExplorerWebApi.Providers
{
    public class CustomJwtFormat :
ISecureDataFormat<AuthenticationTicket>
    {

        private readonly string _issuer = string.Empty;

        public CustomJwtFormat(string issuer)
        {
            _issuer = issuer;
        }

        public string Protect(AuthenticationTicket data)
        {
            if (data == null)
            {
                throw new ArgumentNullException("data");
            }

            string audienceId =
ConfigurationManager.AppSettings["as:AudienceId"];

            string symmetricKeyAsBase64 =
ConfigurationManager.AppSettings["as:AudienceSecret"];

            var keyByteArray =
TextEncodings.Base64Url.Decode(symmetricKeyAsBase64);

            var signingKey = new HmacSigningCredentials(keyByteArray);

            var issued = data.Properties.IssuedUtc;

            var expires = data.Properties.ExpiresUtc;

            var token = new JwtSecurityToken(_issuer, audienceId,
data.Identity.Claims, issued.Value.UtcDateTime,
expires.Value.UtcDateTime, signingKey);

            var handler = new JwtSecurityTokenHandler();

            var jwt = handler.WriteToken(token);

            return jwt;
        }

        public AuthenticationTicket Unprotect(string protectedText)
        {
            throw new NotImplementedException();
        }
    }
}
```

```
using DynamoStudioToDesignExplorerWebApi.Infrastructure;
using DynamoStudioToDesignExplorerWebApi.Models;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using Microsoft.Owin.Security.OAuth;
using System.Collections.Generic;
using System.Security.Claims;
using System.Threading.Tasks;

namespace DynamoStudioToDesignExplorerWebApi.Providers
{
    public class CustomOAuthProvider : OAuthAuthorizationServerProvider
    {

        public override async Task
GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext
context)
        {
            var allowedOrigin =
context.OwinContext.Get<string>("as:clientAllowedOrigin");

            if (allowedOrigin == null) allowedOrigin = "*";
            context.OwinContext.Response.Headers.Add("Access-Control-
Allow-Origin", new[] { allowedOrigin });

            var userManager =
context.OwinContext.GetUserManager<ApplicationUserManager>();

            User user = await userManager.FindAsync(context.UserName,
context.Password);

            if (user == null)
            {
                context.SetError("invalid_grant", "The user name or
password is incorrect.");
                return;
            }
            ClaimsIdentity identity = await
user.GenerateUserIdentityAsync(userManager, "JWT");
            identity.AddClaim(new Claim(ClaimTypes.Name,
context.UserName));
            identity.AddClaim(new Claim("sub", context.UserName));

            var props = new AuthenticationProperties(new
Dictionary<string, string>
                {
                    {
                        "as:client_id", (context.ClientId == null) ?
string.Empty : context.ClientId
                    },
                    {
                        "userName", context.UserName
                    }
                });

            var ticket = new AuthenticationTicket(identity, props);
            context.Validated(ticket);
        }
```

```
        public override Task TokenEndpoint(OAuthTokenEndpointContext
context)
        {
            foreach (KeyValuePair<string, string> property in
context.Properties.Dictionary)
            {
                context.AdditionalResponseParameters.Add(property.Key,
property.Value);
            }

            return Task.FromResult<object>(null);
        }

        public override Task
ValidateClientAuthentication(OAuthValidateClientAuthenticationContext
context)
        {

            string clientId = string.Empty;
            string clientSecret = string.Empty;
            Client client = null;

            if (!context.TryGetBasicCredentials(out clientId, out
clientSecret))
            {
                context.TryGetFormCredentials(out clientId, out
clientSecret);
            }

            if (context.ClientId == null)
            {
                context.Validated();
                context.SetError("invalid_clientId", "ClientId should be
sent.");
                return Task.FromResult<object>(null);
            }
            using (DataContext db = new DataContext())
            {
                client = db.Clients.Find(context.ClientId);
            }

            if (client == null)
            {
                context.SetError("invalid_clientId",
string.Format("Client '{0}' is not registered in the system.",
context.ClientId));
                return Task.FromResult<object>(null);
            }

            if (client.ApplicationType ==
Models.ApplicationTypes.NativeConfidential)
            {
                if (string.IsNullOrWhiteSpace(clientSecret))
                {
                    context.SetError("invalid_clientId", "Client secret
should be sent.");
                    return Task.FromResult<object>(null);
                }
                else
```

```
                {
                    if (client.Secret != clientSecret) //HelperGetHash
                    {
                        context.SetError("invalid_clientId", "Client
secret is invalid.");
                        return Task.FromResult<object>(null);
                    }
                }
            }

            if (!client.Active)
            {
                context.SetError("invalid_clientId", "Client is
inactive.");
                return Task.FromResult<object>(null);
            }

            context.OwinContext.Set<string>("as:clientAllowedOrigin",
client.AllowedOrigin);

context.OwinContext.Set<string>("as:clientRefreshTokenLifeTime",
client.RefreshTokenLifeTime.ToString());

            context.Validated();
            return Task.FromResult<object>(null);
        }
        public override Task
GrantRefreshToken(OAuthGrantRefreshTokenContext context)
        {
            var originalClient =
context.Ticket.Properties.Dictionary["as:client_id"];
            var currentClient = context.ClientId;

            if (originalClient != currentClient)
            {
                context.SetError("invalid_clientId", "Refresh token is
issued to a different clientId.");
                return Task.FromResult<object>(null);
            }

            // Change auth ticket for refresh token requests
            var newIdentity = new
ClaimsIdentity(context.Ticket.Identity);
            //newIdentity.AddClaim(new Claim("newClaim", "newValue"));

            var newTicket = new AuthenticationTicket(newIdentity,
context.Ticket.Properties);
            context.Validated(newTicket);

            return Task.FromResult<object>(null);
        }
    }
}
```

```
using DynamoStudioToDesignExplorerWebApi.Models;
using Microsoft.Owin.Security.Infrastructure;
using System;
using System.Linq;
using System.Threading.Tasks;

namespace DynamoStudioToDesignExplorerWebApi.Providers
{
    public class SimpleRefreshTokenProvider :
IAuthenticationTokenProvider
    {
        public void Create(AuthenticationTokenCreateContext context)
        {
            throw new NotImplementedException();
        }

        public async Task CreateAsync(AuthenticationTokenCreateContext
context)
        {
            var clientid =
context.Ticket.Properties.Dictionary["as:client_id"];

            if (string.IsNullOrEmpty(clientid))
            {
                return;
            }

            var refreshTokenId = Guid.NewGuid().ToString("n");

            using (DataContext db = new DataContext())
            {
                var refreshTokenLifeTime =
context.OwinContext.Get<string>("as:clientRefreshTokenLifeTime");

                var token = new RefreshToken()
                {
                    Id = Helper.GetHash(refreshTokenId),
                    ClientId = clientid,
                    Subject = context.Ticket.Identity.Name,
                    IssuedUtc = DateTime.UtcNow,
                    ExpiresUtc =
DateTime.UtcNow.AddMinutes(Convert.ToDouble(refreshTokenLifeTime))
                };

                context.Ticket.Properties.IssuedUtc = token.IssuedUtc;
                context.Ticket.Properties.ExpiresUtc = token.ExpiresUtc;

                token.ProtectedTicket = context.SerializeTicket();

                var existingToken = db.RefreshTokens.Where(r => r.Subject
== token.Subject && r.ClientId == token.ClientId);

                if (existingToken.Count() > 4)
                {
                    db.RefreshTokens.Remove(existingToken.OrderBy(r =>
r.IssuedUtc).FirstOrDefault());
                    var res = await db.SaveChangesAsync() > 0;
                }
```

```
                db.RefreshTokens.Add(token);

                var result = await db.SaveChangesAsync() > 0;
                if (result)
                {
                    context.SetToken(refreshTokenId);
                }

            }
        }

        public void Receive(AuthenticationTokenReceiveContext context)
        {
            throw new NotImplementedException();
        }

        public async Task ReceiveAsync(AuthenticationTokenReceiveContext
context)
        {

            var allowedOrigin =
context.OwinContext.Get<string>("as:clientAllowedOrigin");
            context.OwinContext.Response.Headers.Add("Access-Control-
Allow-Origin", new[] { allowedOrigin });

            string hashedTokenId = Helper.GetHash(context.Token);

            using (DataContext db = new DataContext())
            {
                var refreshToken = await
db.RefreshTokens.FindAsync(hashedTokenId);

                if (refreshToken != null)
                {

context.DeserializeTicket(refreshToken.ProtectedTicket);
                    db.RefreshTokens.Remove(refreshToken);
                    await db.SaveChangesAsync();
                }
            }
        }
    }
}
```

```
using DynamoStudioToDesignExplorerWebApi.Models;
using Microsoft.AspNet.Identity.EntityFramework;
using System.Data.Entity;

namespace DynamoStudioToDesignExplorerWebApi
{
    public class DataContext : IdentityDbContext<User>
    {
        public DataContext() : base("DataContext",throwIfV1Schema: false)
        {
            Configuration.ProxyCreationEnabled = false;
            Configuration.LazyLoadingEnabled = false;
        }

        public static DataContext Create()
        {
            return new DataContext();
        }

        public DbSet<Client> Clients { get; set; }
        public DbSet<RefreshToken> RefreshTokens { get; set; }
        public DbSet<Design> Designs { get; set; }
        public DbSet<ShareLink> ShareLinks { get; set; }
    }
}
```

Код WEB-сайту

```
using System.Web.Mvc;
using System.Web.Routing;

namespace DynamoStudioToDesignExplorer
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id
= UrlParameter.Optional }
            );

            routes.MapRoute(
                name: "Iteration",
                url: "{controller}/{action}/{userId}/{id}/{guidShare}",
                defaults: new { guidShare = UrlParameter.Optional }
            );
        }
    }
}
```

```
using DynamoStudioToDesignExplorer.Models.JsonModels;
using DynamoStudioToDesignExplorerWebApi.Models;
using DynamoStudioToDesignExplorerWebApi.Models.ModelFactory;
using Newtonsoft.Json;
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Web.Configuration;
using System.Web.Mvc;

namespace DynamoStudioToDesignExplorer.Controllers.AbstractControllers
{
    public abstract class AbstractController: Controller
    {
        protected string AccessToken { get; set; }
        protected string RefreshToken { get; set; }
        protected int ExpiresIn { get; set; }
        protected bool IsAdmin { get; set; }

        public async Task Logout()
        {
            using (var client = new HttpClient())
            {

                string refreshTokenCookie = Request.Cookies["user's
logins cookie"]["refresh_token"];
                HttpCookie cookieLogin = new HttpCookie("user's logins
cookie");
                cookieLogin.Expires = DateTime.Now.AddDays(-1);
                HttpCookie cookieAccess = new HttpCookie("access
cookie");
                cookieAccess.Expires = DateTime.Now.AddDays(-1);
                Response.Cookies.Add(cookieLogin);
                Response.Cookies.Add(cookieAccess);
                var content = new StringContent(
                   refreshTokenCookie,
                   Encoding.UTF8
                );
                content.Headers.Remove("Content-Type");
                content.Headers.Add("Content-Type", "application/json");

                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                var responseApi = await
client.DeleteAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/revoketoken?tokenId=" + refreshTokenCookie);
                string responseString = await
responseApi.Content.ReadAsStringAsync();

Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"]);
            }
        }

        protected async Task GetInformationFromCookie(HttpCookie cookie)
        {
            if (Request.Cookies["access cookie"] != null)
```

```
            {
                AccessToken = Request.Cookies["access
cookie"]["access_token"];
            }
            else
            {
                await RefreshTokens(cookie["userName"]);
                HttpCookie accessCookie = new HttpCookie("access
cookie");
                accessCookie.Expires =
DateTime.Now.AddSeconds(ExpiresIn);

                accessCookie["access_token"] = AccessToken;
                cookie["refresh_token"] = RefreshToken;
                Response.Cookies.Add(accessCookie);
                Response.Cookies.Add(cookie);
            }
        }
        protected async Task RefreshTokens(string userName)
        {
            using (var client = new HttpClient())
            {
                string contentString = "username=" + userName +
                    "&grant_type=refresh_token&refresh_token=" +
RefreshToken +
                    "&client_id=" +
WebConfigurationManager.AppSettings["explorer_client_id"] +
                    "&client_secret=" +
WebConfigurationManager.AppSettings["explorer_client_secret"].Replace("="
, "%3D");

                var content = new StringContent(
                    contentString,
                    Encoding.UTF8
                );
                content.Headers.Remove("Content-Type");
                content.Headers.Add("Content-Type", "application/json");

                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                var responseApi = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/gettoken", content);
                if (!responseApi.IsSuccessStatusCode)
                {
                    await Logout();
                }
                string responseString = await
responseApi.Content.ReadAsStringAsync();
                var tokenModel =
JsonConvert.DeserializeObject<TokenModel>(responseString);

                AccessToken = tokenModel.access_token;
                RefreshToken = tokenModel.refresh_token;
                ExpiresIn = tokenModel.expires_in;
            }
        }
        protected async Task<User> GetUser(string id)
        {
```

```
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", AccessToken);
                StringContent content = new StringContent("",
Encoding.UTF8);
                var response = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/user/" + id, content);
                var responseString = await
response.Content.ReadAsStringAsync();
                return ResponserToUser(responseString);
            }
        }
        protected User ResponserToUser(string response)
        {
            var userModel =
JsonConvert.DeserializeObject<UserReturnModel>(response);
            User user = new User
            {
                Id = userModel.Id,
                UserName = userModel.UserName,
                FirstName = userModel.FirstName,
                LastName = userModel.LastName,
                Email = userModel.Email,
                RefreshToken = userModel.RefreshToken,
                Folder = userModel.Folder,
                Project = userModel.Project
            };
            if (userModel.Roles.Contains("Admin"))
            {
                IsAdmin = true;
            }
            return user;
        }
    }
}
```

```
using DynamoStudioToDesignExplorer.Controllers.AbstractControllers;
using DynamoStudioToDesignExplorerWebApi.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using System.Web;
using System.Web.Configuration;
using System.Web.Mvc;

namespace DynamoStudioToDesignExplorer.Controllers
{
    public class AdminController : AbstractController
    {
        private async Task<List<User>> getUsersListAsync()
        {
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", AccessToken);
                var response = await
client.GetAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/users");
                var responseString = await
response.Content.ReadAsStringAsync();
                return ResponseToUsersList(responseString);
            }
        }
        private List<User> ResponseToUsersList(string response)
        {
            return JsonConvert.DeserializeObject<List<User>>(response);
        }
        public ActionResult Index()
        {
            return View();
        }

        public async Task<ActionResult> Users()
        {
            if (Request.Cookies["user's logins cookie"] == null)
            {

Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"]);

            }
            else
            {
                User user;
                HttpCookie cookie = Request.Cookies["user's logins
cookie"];
                RefreshToken = cookie["refresh_token"];
                await GetInformationFromCookie(cookie);
                user = await GetUser(cookie["userId"]);
                if(!IsAdmin)
                {
```

```
Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"]);
                }
            }
            List<User> users = await getUsersListAsync();
            return View(users);
        }
    }
}
```

```
using Autodesk.Forge;
using DynamoStudioToDesignExplorer.Controllers.AbstractControllers;
using DynamoStudioToDesignExplorer.Models.JsonModels;
using DynamoStudioToDesignExplorerWebApi;
using DynamoStudioToDesignExplorerWebApi.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Web.Configuration;
using System.Web.Mvc;
namespace DynamoStudioToDesignExplorer.Controllers
{
    public class HomeController : AbstractController
    {
        public void ToAutodeskLogin()
        {
            ThreeLeggedApi threeLeggedApi = new ThreeLeggedApi();
            string bearerThree =
threeLeggedApi.Authorize(WebConfigurationManager.AppSettings["FORGE_CLIEN
T_ID"], "code",
                WebConfigurationManager.AppSettings["REDIRECT_URI"], new
Scope[] { Scope.DataRead, Scope.DataWrite, Scope.DataCreate });
            Response.Redirect(bearerThree);
        }

        private async Task<User> SetUser(string code)
        {
            using (var client = new HttpClient())
            {
                var content = new StringContent(
                        code,
                        Encoding.UTF8
                        );

                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                var responseApi = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/users", content);
                string responseApiString = await
responseApi.Content.ReadAsStringAsync();
                return ResponserToUser(responseApiString);
            }
        }

        private async Task GetToken(User user)
        {
            using (var client = new HttpClient())
            {
                string contentString = "username=" + user.UserName +
                    "&grant_type=password&password=" + user.RefreshToken
+
```

```
                    "&client_id=" +
WebConfigurationManager.AppSettings["explorer_client_id"] +
                    "&client_secret=" +
WebConfigurationManager.AppSettings["explorer_client_secret"].Replace("="
,"%3D");

                var content = new StringContent(
                    contentString,
                    Encoding.UTF8
                );
                content.Headers.Remove("Content-Type");
                content.Headers.Add("Content-Type", "application/json");

                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                var responseApi = await
client.PostAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/gettoken", content);
                string responseString = await
responseApi.Content.ReadAsStringAsync();
                var tokenModel =
JsonConvert.DeserializeObject<TokenModel>(responseString);

                AccessToken = tokenModel.access_token;
                RefreshToken = tokenModel.refresh_token;
                ExpiresIn = tokenModel.expires_in;
            }
        }
        private async Task<Dictionary<string, string>>
GetAutodeskHubs(string id)
        {
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", AccessToken);
                var response = await
client.GetAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/authentication/GetAutodeskHubs/" +id);
                string responseString = await
response.Content.ReadAsStringAsync();
                var hubModel =
JsonConvert.DeserializeObject<HubModelAutodesk>(responseString);
                Dictionary<string, string> hubs = new Dictionary<string,
string>();
                foreach (DataHubAutodesk data in hubModel.data)
                {
                    if
(data.attributes.extension.type.Contains("bim360"))
                    {
                        hubs.Add(data.id, data.attributes.name);
                    }
                }
                return hubs;
            }
        }
```

```
        public async Task<ActionResult> Iterations(string userId, int id,
string guidShare = null)
        {
            if (Request.Cookies["user's logins cookie"] == null &&
guidShare == null)
            {

Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"]);

            }
            else if(Request.Cookies["user's logins cookie"] != null)
            {
                HttpCookie cookie = Request.Cookies["user's logins
cookie"];
                RefreshToken = cookie["refresh_token"];
                await GetInformationFromCookie(cookie);
                using (DataContext db = new DataContext())
                {
                    var share = db.ShareLinks.Where(sl => sl.DesignId ==
id).SingleOrDefault();
                    if (share != null)
                    {
                        ViewData["guidShareLogined"] = share.GUID;
                    }
                }
            }
            if (guidShare != null)
            {
                using (HttpClient client = new HttpClient())
                {
                    client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                    var response = await
client.GetAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/users/" + userId + "/designs/" + id + "/share/" + guidShare);
                    if (!response.IsSuccessStatusCode) {
                        return HttpNotFound();
                    }
                }
                ViewData["guidShare"] = guidShare;
            }
            using (DataContext db = new DataContext())
            {
                ViewData["dName"] =
Path.GetFileName(db.Designs.Find(id).Path);
                ViewData["userId"] = userId;
            }
            return View(id);
        }
        public async Task<ActionResult> Index()
        {
            User user;
            if (Request.QueryString["code"] != null)
            {
                User currentUser = await
SetUser(Request.QueryString["code"]);
                await GetToken(currentUser);
                user = await GetUser(currentUser.Id);
                if(user == null)
```

```
                {
                    return View();
                }
                HttpCookie cookieCode = new HttpCookie("AUTODESK_CODE");
                cookieCode.Value = Request.QueryString["code"];
                cookieCode.Expires = DateTime.Now.AddMinutes(5);

                HttpCookie cookie = new HttpCookie("user's logins
cookie");
                cookie.Expires = DateTime.Now.AddDays(7);

                cookie["userId"] = user.Id;
                cookie["userName"] = user.UserName;

                cookie["refresh_token"] = RefreshToken;
                cookie["is_admin"] = IsAdmin.ToString();
                HttpCookie accessCookie = new HttpCookie("access
cookie");
                accessCookie.Expires =
DateTime.Now.AddSeconds(ExpiresIn);

                accessCookie["access_token"] = AccessToken;

                Response.Cookies.Add(cookieCode);
                Response.Cookies.Add(cookie);
                Response.Cookies.Add(accessCookie);
                if (IsAdmin)
                {

Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"] +
"/Admin/Users");
                }
                else
                {

Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"] +
"/Home/Designs/" + user.Id);
                }
            } else if (Request.Cookies["user's logins cookie"] != null)
            {
                HttpCookie cookie = Request.Cookies["user's logins
cookie"];
                RefreshToken = cookie["refresh_token"];
                await GetInformationFromCookie(cookie);
                user = await GetUser(cookie["userId"]);
                if (IsAdmin)
                {

Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"] +
"/Admin/Users");
                }
                return View();
            }

            return View(new User());
        }
        public async Task<ActionResult> Designs(string id)
        {
            if (Request.Cookies["user's logins cookie"] == null)
```

```
            {
Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"]);
            } else
            {
                User user;
                HttpCookie cookie = Request.Cookies["user's logins
cookie"];
                RefreshToken = cookie["refresh_token"];
                await GetInformationFromCookie(cookie);
                user = await GetUser(cookie["userId"]) ?? new User();
                ViewData["userId"] = id;
                if((user.Project == null || user.Project == String.Empty)
&& id == cookie["userId"])
                {
                    var hubs = await GetAutodeskHubs(id);
                    ViewData["hubs"] = hubs;
                } else
                {
                    using (HttpClient client = new HttpClient())
                    {
                        client.BaseAddress = new
Uri(WebConfigurationManager.AppSettings["WEB_API_URI"]);
                        client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", AccessToken);
                        var response = await
client.GetAsync(WebConfigurationManager.AppSettings["API_VERSION"] +
"/users/" + id + "/designs/");
                        string responseString = await
response.Content.ReadAsStringAsync();
                        return View(ResponseToDesign(responseString));
                    }
                }
            }
            return View();
        }

        private List<Design> ResponseToDesign(string response)
        {
            var designsModel =
JsonConvert.DeserializeObject<List<Design>>(response);
            return designsModel;
        }

        public async Task<ActionResult> UserSettings()
        {
            if (Request.Cookies["user's logins cookie"] != null)
            {
                User user;
                HttpCookie cookie = Request.Cookies["user's logins
cookie"];
                RefreshToken = cookie["refresh_token"];
                await GetInformationFromCookie(cookie);
                user = await GetUser(cookie["userId"]) ?? new User();
                var hubs = await GetAutodeskHubs(user.Id);
                ViewData["hubs"] = hubs;
                @ViewData["userId"] = user.Id;
                return View(user);
```

```
            } else
            {

Response.Redirect(WebConfigurationManager.AppSettings["REDIRECT_URI"]);
                return View(new User());
            }
        }
    }
}
```

Код WEB-сайту (сторінки)

Designs.cshtml

```
@model List<DynamoStudioToDesignExplorerWebApi.Models.Design>
@{
    ViewBag.Title = "Designs";
}

<h2>Designs</h2>
@if (ViewData["hubs"] != null)
{
    <div id="hubs-box" class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-
title">@DynamoStudioToDesignExplorer.Resources.Resource.CHOOSE_HUB</h3>
        </div>
        <div class="panel-body">
            @{var hubs = (Dictionary<string, string>)ViewData["hubs"]; }
            @if (hubs.Count != 0)
            {
                foreach (KeyValuePair<string, string> hub in
(Dictionary<string, string>)ViewData["hubs"])
                {
                    <button id="@hub.Key" class="btn btn-default hub-
choose">@hub.Value</button>
                }
            }
            else
            {

<p>@DynamoStudioToDesignExplorer.Resources.Resource.NOT_HAVE_BIM360_ERROR
</p>
                <ul class="no-margin-ul">
                    <li><a href="https://docs.b360.autodesk.com"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.BIM360_L
INK</a></li>
                    <li><a href="https://knowledge.autodesk.com/search-
result/caas/CloudHelp/cloudhelp/ENU/BIM-360-Admin-Help/files/GUID-
9C991F4C-7012-4770-B8F7-A1D45F5D9197-htm.html"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.SETUP_BI
M360_INSTRUCTIONS_LINK</a></li>
                </ul>
            }
        </div>
    </div>

    <div id="projects-box" class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-
title">@DynamoStudioToDesignExplorer.Resources.Resource.CHOOSE_PROJECT</h
3>
        </div>
        <div id="projects" class="panel-body">

        </div>
    </div>
```

```
    <div id="folders-box" class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-
title">@DynamoStudioToDesignExplorer.Resources.Resource.ERROR</h3>
        </div>
        <div id="folders" class="panel-body">

        </div>
    </div>
}
else
{
    <table id="myTable" class="display">
        <thead>
            <tr>
                <th></th>
                <th>Name</th>
                <th>Start Date</th>
                <th>Last Change Date</th>
            </tr>
        </thead>
        <tbody>
            @foreach (DynamoStudioToDesignExplorerWebApi.Models.Design
design in Model)
            {
                <tr>
                    <td id="@design.Id"></td>

<td>@Path.GetFileName(design.Path).Remove(Path.GetFileName(design.Path).L
ength - 4)</td>
                    <td>@design.StartDate</td>
                    <td>@design.ChangeDate</td>
                </tr>
                }
            </tbody>
            </table>
    }

    <script>

    function renderProject(name, projectId, rootFolder) {
        var item = '<button id="' + projectId + ';' + rootFolder + '"
class="btn btn-default project-choose"> ' + name + '</button>'
        $("#projects").append(item);
    }

    function renderFolder(name, projectId, folderId) {
        var item = '<button id="' + projectId + ';' + folderId + '"
class="btn btn-default folder-choose"> ' + name + '</button>'
        $("#folders").append(item);
    }

    function setProject(projectId, folderId) {
        $.ajax({
            method: "POST",
            url: WEB_API_URI +
'@System.Web.Configuration.WebConfigurationManager.AppSettings["API_VERSI
ON"]' +'/authentication/SetUserAutodeskProject',
            dataType: 'text',
```

```
            data: (projectId + ';' + folderId),
            beforeSend: function (xhr) {
                $("#loader").show();
                RefreshToken();
                xhr.setRequestHeader("Authorization", "Bearer " +
accessToken);
            },
            success: function () {
                $("#folders-box").remove();
                $("#projects-box").remove();
                $("#hubs-box").remove();
            },
            complete: function () {
                $("#loader").hide();
                location.reload();
            }
        });
    }

    function setProjectFoldClickAction() {
         $(".project-choose").click(function () {
            var proj = $(this).attr('id');
            $.ajax({
                method: "POST",
                url: WEB_API_URI +
'@System.Web.Configuration.WebConfigurationManager.AppSettings["API_VERSI
ON"]' +'/authentication/GetAutodeskProjectFolders/@ViewData["userId"]',
                data: proj,
                beforeSend: function (xhr) {
                    $("#loader").show();
                    RefreshToken();
                    xhr.setRequestHeader("Authorization", "Bearer " +
accessToken);
                },
                success: function (data) {
                    var projId = proj.split(";")[0];
                    $("#folders").html("");
                    $("#folders-box").css("display", "block");
                    data.data.forEach(function (fold) {
                        if (fold.attributes.name == "Project Files")
                            setProject(projId, fold.id);
                    });

$("#folders").append('<p>@DynamoStudioToDesignExplorer.Resources.Resource
.NOT_SETUP_BIM360_SERVICES</p><ul class="no-margin-ul"><li> <a
href="https://docs.b360.autodesk.com"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.BIM360_L
INK</a></li><li> <a herf="https://knowledge.autodesk.com/search-
result/caas/CloudHelp/cloudhelp/ENU/BIM-360-Admin-Help/files/GUID-
9C991F4C-7012-4770-B8F7-A1D45F5D9197-htm.html"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.SETUP_BI
M360_INSTRUCTIONS_LINK</a></li></ul>');
                },
                complete: function () {
                    $("#loader").hide();
                }
            });
        });
    }
```

```
    function format(id) {
        var row = '<button class="btn btn-info look-design"
onclick="lookDesign(' +
            id + ')">View design</button> <button class="btn btn-danger
delete-design" onclick="deleteDesign(' +
            id + ')">Delete design</button>'
        return row;
    }

    function lookDesign(id) {
        document.location.href =
'@System.Web.Configuration.WebConfigurationManager.AppSettings["REDIRECT_
URI"]' + "/Home/Iterations/@ViewData["userId"]/" + id;
    }

        function deleteDesign(id) {
            var isDelete =
confirm("@DynamoStudioToDesignExplorer.Resources.Resource.CONFIRM_ASK_MES
SAGE");
            if (isDelete) {
                $.ajax({
                    method: "DELETE",
                    url: WEB_API_URI +
'@System.Web.Configuration.WebConfigurationManager.AppSettings["API_VERSI
ON"]' + '/users/' + '@ViewData["userId"]' + '/designs/' + id,
                    beforeSend: function (xhr) {
                        $("#loader").show();
                        RefreshToken();
                        xhr.setRequestHeader("Authorization", "Bearer " +
accessToken);
                    },
                    complete: function () {
                        $("#loader").hide();
                        location.reload();
                    }
                });
            }
    }

    $(document).ready(function () {
        var table = $('#myTable').DataTable({
            "language": {
                "emptyTable":
"@DynamoStudioToDesignExplorer.Resources.Resource.EMPTY_DESIGN_TABLE_MESS
AGE"
            },
            "columns": [
                {
                    "className": 'details-control',
                    "orderable": false,
                    "data": null,
                    "defaultContent": ''
                },
                { "data": "name" },
                { "data": "start" },
                { "data": "change" }
            ],
```

```
            "order": [[1, 'asc']]
        });

        $('#myTable tbody').on('click', 'td.details-control', function ()
{
            var tr = $(this).closest('tr');
            var row = table.row(tr);

            if (row.child.isShown()) {
                row.child.hide();
                tr.removeClass('shown');
            }
            else {
                row.child(format($(this).attr('id'))).show();
                tr.addClass('shown');
            }
        });
        $(".hub-choose").click(function () {

            $.ajax({
                url: WEB_API_URI +
'@System.Web.Configuration.WebConfigurationManager.AppSettings["API_VERSI
ON"]' + '/authentication/GetAutodeskProjects',
                data: 'id=' + $(this).attr('id') +
'&userId=@ViewData["userId"]',
                beforeSend: function (xhr) {
                    $("#loader").show();
                    RefreshToken();
                    xhr.setRequestHeader("Authorization", "Bearer " +
accessToken);
                },
                success: function (data) {
                    $("#projects").html("");
                    $("#projects-box").css("display", "block");
                    $("#folders").html("");
                    $("#folders-box").css("display", "none");
                    data.data.forEach(function (proj) {
                        renderProject(proj.attributes.name, proj.id,
proj.relationships.rootFolder.data.id);
                    });
                    setProjectFoldClickAction();
                },
                complete: function () {
                    $("#loader").hide();
                }
            });
        });



    });

    </script>
```

Index.cshtml

```
@{
    ViewBag.Title = "Home Page";
}
@Styles.Render("~/Content/homeCss")
<div class="pop-up-video">
    <iframe preload="auto" class="video-main"
src="https://www.youtube.com/embed/5qMSFxOu_Dw" frameborder="0"
allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-
picture" allowfullscreen></iframe>
</div>
<div class="intro-container">
    <div class="intro col-12">
        <p class="desc-name">Dynamo Studio to Design Explorer</p>
        <button type="button" class="learn-more btn btn-default"
onclick="showLearn()">Learn More</button>
    </div>
</div>
<div class="video-container">
    <img class="future-video" src="../../images/geom.png">
</div>

<div class="content-explorer">
    <p>Point Cloud Viewer optimizes the use of hardware resources by
cleaning up the data received from 3D Scanner.</p>

    <p>The ultimate goal of the application the possibility to display
arbitrary point cloud while maintaining constant memory footprint and
acceptable frame rate.</p>

    <p>Point cloud viewer is capable of parsing PCD, PTS, PTX, E57 and
XYZ, a widely used open-source point cloud data formats.</p>

    <p>The fully-featured and user-friendly interface ensures an
effective point cloud inspection, viewing it in 3D.</p>
</div>

<footer class="footer-amc">
    <p class="footer-right">Copyright &copy; 2019 AMC Bridge. All rights
reserved. | <a style="color:white"
href="https://www.amcbridge.com">www.amcbridge.com</a></p>
</footer>
<script>
    function showLearn() {
        $(".pop-up-video").css("display","flex");
    }
    $(document).ready(function () {
        $(".pop-up-video").click(function () {
            $(this).css("display","none");
        });
    });
</script>
```

UserSettings.cshtml

```
@model DynamoStudioToDesignExplorerWebApi.Models.User
@{
                /**/

                ViewBag.Title = "User Settings";
}
<h2>User Settings</h2>

<div id="user-info">
    <table class="table">
        <thead>
            <tr>
                <th colspan="2">User info</th>
            </tr>
        </thead>
        <tr>
            <td scope="row">Name</td>
            <td>@Model.UserName</td>
        </tr>

        <tr>
            <td scope="row">First Name</td>
            <td>@Model.FirstName</td>
        </tr>

        <tr>
            <td scope="row">Last Name</td>
            <td>@Model.LastName</td>
        </tr>

        <tr>
            <td scope="row">E-mail</td>
            <td>@Model.Email</td>
        </tr>


    </table>

    <div id="hubs-box" class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-
title">@DynamoStudioToDesignExplorer.Resources.Resource.CHANGE_HUB</h3>
        </div>
        <div class="panel-body">
            @{var hubs = (Dictionary<string, string>)ViewData["hubs"]; }
            @if (hubs.Count != 0)
            {
                foreach (KeyValuePair<string, string> hub in
(Dictionary<string, string>)ViewData["hubs"])
                {
                    <button id="@hub.Key" class="btn btn-default hub-
choose">@hub.Value</button>
                }
            }
            else
            {
```

```
<p>@DynamoStudioToDesignExplorer.Resources.Resource.NOT_HAVE_BIM360_ERROR
</p>
                <ul class="no-margin-ul">
                    <li><a href="https://docs.b360.autodesk.com"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.BIM360_L
INK</a></li>
                    <li><a href="https://knowledge.autodesk.com/search-
result/caas/CloudHelp/cloudhelp/ENU/BIM-360-Admin-Help/files/GUID-
9C991F4C-7012-4770-B8F7-A1D45F5D9197-htm.html"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.SETUP_BI
M360_INSTRUCTIONS_LINK</a></li>
                </ul>
            }
        </div>
    </div>

    <div id="projects-box" class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-
title">@DynamoStudioToDesignExplorer.Resources.Resource.CHOOSE_PROJECT</h
3>
        </div>
        <div id="projects" class="panel-body">

        </div>
    </div>

    <div id="folders-box" class="panel panel-default">
        <div class="panel-heading">
            <h3 class="panel-
title">@DynamoStudioToDesignExplorer.Resources.Resource.ERROR</h3>
        </div>
        <div id="folders" class="panel-body">

        </div>
    </div>
</div>
<div>
    <p>@Model.Role</p>
</div>
<hr />
<footer>
    <p>&copy; @DateTime.Now.Year - AMC Bridge LLC</p>
</footer>

<script>
    function renderProject(name, projectId, rootFolder) {
        var item = '<button id="' + projectId + ';' + rootFolder + '"
class="btn btn-default project-choose"> ' + name + '</button>'
        $("#projects").append(item);
    }

    function renderFolder(name, projectId, folderId) {
        var item = '<button id="' + projectId + ';' + folderId + '"
class="btn btn-default folder-choose"> ' + name + '</button>'
        $("#folders").append(item);
    }
```

```
    function setProject(projectId, folderId) {
        $.ajax({
            method: "POST",
            url: WEB_API_URI +
'@System.Web.Configuration.WebConfigurationManager.AppSettings["API_VERSI
ON"]' +'/authentication/SetUserAutodeskProject',
            dataType: 'text',
            data: (projectId + ';' + folderId),
            beforeSend: function (xhr) {
                $("#loader").show();
                RefreshToken();
                xhr.setRequestHeader("Authorization", "Bearer " +
accessToken);
            },
            success: function () {
                $("#folders-box").remove();
                $("#projects-box").remove();
                $("#hubs-box").remove();
                setTimeout(function () {

alert("@DynamoStudioToDesignExplorer.Resources.Resource.HUB_CHANGE_SUCCES
S_MESSAGE");
                }, 10);
                location.reload();
            },
            error: function () {

alert("@DynamoStudioToDesignExplorer.Resources.Resource.HUB_CHANGE_ERROR_
MESSAGE");
            },
            complete: function () {
                $("#loader").hide();
            }
        });
    }

    function setProjectFoldClickAction() {
         $(".project-choose").click(function () {
            var proj = $(this).attr('id');
            $.ajax({
                method: "POST",
                url: WEB_API_URI +
'@System.Web.Configuration.WebConfigurationManager.AppSettings["API_VERSI
ON"]' +'/authentication/GetAutodeskProjectFolders/@ViewData["userId"]',
                data: proj,
                beforeSend: function (xhr) {
                    $("#loader").show();
                    RefreshToken();
                    xhr.setRequestHeader("Authorization", "Bearer " +
accessToken);
                },
                success: function (data) {
                    var projId = proj.split(";")[0];
                    $("#folders").html("");
                    $("#folders-box").css("display", "block");
                    data.data.forEach(function (fold) {
                        if (fold.attributes.name == "Project Files")
                            setProject(projId, fold.id);
                    });
```

```
$("#folders").append('<p>@DynamoStudioToDesignExplorer.Resources.Resource
.NOT_SETUP_BIM360_SERVICES</p><ul class="no-margin-ul"><li> <a
href="https://docs.b360.autodesk.com"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.BIM360_L
INK</a></li><li> <a herf="https://knowledge.autodesk.com/search-
result/caas/CloudHelp/cloudhelp/ENU/BIM-360-Admin-Help/files/GUID-
9C991F4C-7012-4770-B8F7-A1D45F5D9197-htm.html"
target="_blank">@DynamoStudioToDesignExplorer.Resources.Resource.SETUP_BI
M360_INSTRUCTIONS_LINK</a></li></ul>');
                },
                complete: function () {
                    $("#loader").hide();
                }
            });
        });
    }

    $(document).ready(function () {
        $(".hub-choose").click(function () {

            $.ajax({
                url: WEB_API_URI +
'@System.Web.Configuration.WebConfigurationManager.AppSettings["API_VERSI
ON"]' + '/authentication/GetAutodeskProjects',
                data: 'id=' + $(this).attr('id') +
'&userId=@ViewData["userId"]',
                beforeSend: function (xhr) {
                    $("#loader").show();
                    RefreshToken();
                    xhr.setRequestHeader("Authorization", "Bearer " +
accessToken);
                },
                success: function (data) {
                    $("#projects").html("");
                    $("#projects-box").css("display", "block");
                    $("#folders").html("");
                    $("#folders-box").css("display", "none");
                    data.data.forEach(function (proj) {
                        renderProject(proj.attributes.name, proj.id,
proj.relationships.rootFolder.data.id);
                    });
                    setProjectFoldClickAction();
                },
                complete: function () {
                    $("#loader").hide();
                }
            });
        });



    });
</script>
```

Код додатку Dynamo Studio

```
using System;
using System.Collections.Generic;
using Dynamo.Graph.Nodes;
using ProtoCore.AST.AssociativeAST;
using DynamoStudioToDesignExplorer.Model;
using SimpleDesignExplorerFunction;
using SimpleDesignExplorerFunction.Logs;
using System.Linq;

namespace DynamoStudioToDesignExplorer
{
    [NodeName("Select design node")]
    [NodeDescription("Select design node")]
    [NodeCategory("Design Explorer nodes")]
    [InPortDescriptions("Parameters values")]
    [OutPortTypes("var []")]
    [OutPortDescriptions("Parameters list")]
    [IsDesignScriptCompatible]
    public class SelectDesignNodeModel : NodeModel
    {

        private string _selectedDesignName;
        public string SelectedDesignName
        {
            get
            {
                return _selectedDesignName;
            }
            set
            {
                _selectedDesignName = value;
                OnNodeModified(false);
            }
        }
        private int _selectedDesignId;
        public int SelectedDesignId
        {
            get
            {
                return _selectedDesignId;
            }
            set
            {
                _selectedDesignId = value;
                OnNodeModified(false);
            }
        }
        private List<string> _parametersList;
        public List<string> ParametersList
        {
            get
            {
                return _parametersList;
            }
            set
            {
                _parametersList = value;
```

```
                OnNodeModified(false);
            }
        }
        private List<PortData> ports;
        public SelectDesignNodeModel()
        {
            ports = new List<PortData>();
            OutPortData.Add(new PortData("Parameters", "Parameters
list"));

            RegisterAllPorts();
        }

        public void AddNewPort(string name)
        {
            var port = new PortData(name, "value");
            ports.Add(port);
            InPortData.Add(port);
            RegisterAllPorts();
        }

        public void ChangePortName(string name, int index)
        {
            InPortData[index].NickName = name;
            ports[index] = InPortData[index];
            RegisterAllPorts();
        }

        public void RemovePort(int index)
        {
            InPortData.RemoveAt(index);
            ports.RemoveAt(index);
            RegisterAllPorts();
        }

        public void RefreshPorts()
        {
            foreach (var port in ports)
            {
                InPortData.Remove(port);
            }
            ports = new List<PortData>();
            RegisterAllPorts();
        }

        public override IEnumerable<AssociativeNode>
BuildOutputAst(List<AssociativeNode> inputAstNodes)
        {
            var connectedInput = Enumerable.Range(0, InPortData.Count)
                                           .Where(HasConnectedInput)
                                           .Select(x => new
IntNode(x) as AssociativeNode)
                                           .ToList();
            DesignExplorerLog.Write("Sending parameters from select
design node started");


            if (ParametersList == null || SelectedDesignName == null ||
connectedInput.Count != InPortData.Count)
```

```
            {
                return new[] {
AstFactory.BuildAssignment(GetAstIdentifierForOutputIndex(0),
AstFactory.BuildNullNode()) };
            }

            var parametersList = new List<AssociativeNode>();
            foreach (var parameter in ParametersList)
            {

parametersList.Add(AstFactory.BuildStringNode(parameter));
            }

            if (parametersList.Count != inputAstNodes.Count)
            {
                return new[] {
AstFactory.BuildAssignment(GetAstIdentifierForOutputIndex(0),
AstFactory.BuildNullNode()) };
            }

            var functionCall = AstFactory.BuildFunctionCall(
                new Func<int, List<string>, List<object>, string,
AuthModel>(SelectDesignNodeFunction.GetAuthResult),
                new List<AssociativeNode> {
AstFactory.BuildIntNode(SelectedDesignId),
AstFactory.BuildExprList(parametersList),
AstFactory.BuildExprList(inputAstNodes),
AstFactory.BuildStringNode(SelectedDesignName) }
                );

            return new[] {
AstFactory.BuildAssignment(GetAstIdentifierForOutputIndex(0),
functionCall) };
        }
    }
}
```

```
using Dynamo.Controls;
using Dynamo.Wpf;

namespace DynamoStudioToDesignExplorer
{
    public class SelectDesignNodeModelView:
INodeViewCustomization<SelectDesignNodeModel>
    {
        public void CustomizeView(SelectDesignNodeModel model, NodeView
nodeView)
        {
            var designList = new DesignListLogin();
            nodeView.inputGrid.Children.Add(designList);
            designList.DataContext = model;
        }

        public void Dispose()
        {
        }
    }
}
```

```
using Dynamo.Graph.Nodes;
using ProtoCore.AST.AssociativeAST;
using DynamoStudioToDesignExplorer.Model;
using SimpleDesignExplorerFunction;
using SimpleDesignExplorerFunction.Model;
using System;
using System.Collections.Generic;

namespace DynamoStudioToDesignExplorer
{
    [NodeName("Send design node")]
    [NodeDescription("Send design node")]
    [NodeCategory("Design Explorer nodes")]
    [InPortNames("Parameters", "Iteration", "IsReady")]
    [InPortTypes("AuthModel", "Iteration", "bool")]
    [InPortDescriptions("Parameters list", "Iteration data", "Readying
for send")]
    [OutPortNames("Status")]
    [OutPortTypes("string")]
    [OutPortDescriptions("Sending status")]
    [IsDesignScriptCompatible]
    public class SendDesignNodeModel: NodeModel
    {
        public SendDesignNodeModel()
        {
            RegisterAllPorts();
        }
        public override IEnumerable<AssociativeNode>
BuildOutputAst(List<AssociativeNode> inputAstNodes)
        {
            var functionCall = AstFactory.BuildFunctionCall(
                new Func<AuthModel, Iteration, bool,
string>(SendDesignNodeModelFunction.SendDesign),
                new List<AssociativeNode> { inputAstNodes[0],
inputAstNodes[1], inputAstNodes[2] }
                );
            return new[] {
AstFactory.BuildAssignment(GetAstIdentifierForOutputIndex(0),
functionCall) };
        }
    }
}
```

```
using Autodesk.DesignScript.Geometry;
using Autodesk.DesignScript.Interfaces;
using Autodesk.DesignScript.Runtime;
using Dynamo.Visualization;
using Newtonsoft.Json;
using SimpleDesignExplorerFunction.ImageCreating;
using SimpleDesignExplorerFunction.Model;
using Spectacles.Net.Data;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading;

namespace SimpleDesignExplorerFunction
{
    [IsVisibleInDynamoLibrary(false)]
    public static class ConverterNodeModelFunction
    {
        [IsVisibleInDynamoLibrary(false)]
        public static Iteration Convert(List<Geometry> geometries,
List<DSCore.Color> color)
        {
            var geometryList = new List<SpectaclesGeometry>();
            var materialList = new List<SpectaclesMaterial>();
            var testObject = new SpectaclesObject
            {
                uuid = Guid.NewGuid().ToString(),
                type = "Scene",
                matrix = new double[] { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 1 },
                children = new List<SpectaclesObject>(),
                userData = new Dictionary<string, string>()
            };
            testObject.userData.Add("layers", "Default");

            var verteciesForOpenGL = new List<double>();
            var lineVerteciesForOpenGL = new List<double>();

            foreach (Geometry geometry in geometries)
            {
                if(geometry is Autodesk.DesignScript.Geometry.Point)
                {
                    return null;
                }

                var inputGeometries = new SpectaclesGeometry
                {
                    uuid = Guid.NewGuid().ToString(),
                    type = "Geometry"
                };

                var data = new SpectaclesGeometryData
                {
                    vertices = new List<double>(),
                    faces = new List<int>(),
                    normals = new List<double>(),
                    uvs = new List<double>(),
                    scale = 1,
                    visible = true,
```

```
                    castShadow = true,
                    receiveShadow = false,
                    doubleSided = true
                };
                DefaultRenderPackageFactory f = new
DefaultRenderPackageFactory();
                IRenderPackage rp = f.CreateRenderPackage();
                rp.IsSelected = true;
                TessellationParameters tp = new TessellationParameters {
                    Tolerance = -1,
                    MaxTessellationDivisions = 512
                };
                geometry.Tessellate(rp, tp);
                var verteciesForCurrentJson = new List<double>();
                string typeCurrentGeometry = string.Empty;
                string nameCurrentGeometry = string.Empty;

                if (geometry is Curve)
                {


                    Curve[] curves =
((Curve)geometry).SplitByPoints(((Curve)geometry).PointsAtEqualChordLengt
h(100));

                    foreach (var curve in curves) {
                        data.vertices.Add(curve.StartPoint.X);
                        data.vertices.Add(curve.StartPoint.Y);
                        data.vertices.Add(curve.StartPoint.Z);
                        data.vertices.Add(curve.EndPoint.X);
                        data.vertices.Add(curve.EndPoint.Y);
                        data.vertices.Add(curve.EndPoint.Z);

                        lineVerteciesForOpenGL.Add(curve.StartPoint.X);
                        lineVerteciesForOpenGL.Add(curve.StartPoint.Y);
                        lineVerteciesForOpenGL.Add(curve.StartPoint.Z);
                        lineVerteciesForOpenGL.Add(curve.EndPoint.X);
                        lineVerteciesForOpenGL.Add(curve.EndPoint.Y);
                        lineVerteciesForOpenGL.Add(curve.EndPoint.Z);
                    }


                    typeCurrentGeometry = "Line";
                    nameCurrentGeometry = "line" +
geometries.IndexOf(geometry).ToString();
                }
                else
                {
                    foreach (double verttex in rp.MeshVertices)
                    {
                        verteciesForCurrentJson.Add(verttex);
                        verteciesForOpenGL.Add(verttex);
                    }

                    var pointsLocal = new
List<Autodesk.DesignScript.Geometry.Point>();

                    int faceIndx = 0;
```

```
                    for (int i = 0; i < verteciesForCurrentJson.Count; i
+= 3)
                    {
                        if (i == 0)
                        {
                            data.faces.Add(0);
                        }
                        var point =
Autodesk.DesignScript.Geometry.Point.ByCoordinates(verteciesForCurrentJso
n[i], verteciesForCurrentJson[i + 1], verteciesForCurrentJson[i + 2]);

                        if (!pointsLocal.Contains(point))
                        {
                            pointsLocal.Add(point);

                            if (data.faces.Count % 4 == 0)
                            {
                                data.faces.Add(0);
                            }
                            data.faces.Add(faceIndx);

data.vertices.Add(Math.Round(verteciesForCurrentJson[i], 5));

data.vertices.Add(Math.Round(verteciesForCurrentJson[i + 1], 5));

data.vertices.Add(Math.Round(verteciesForCurrentJson[i + 2], 5));
                            faceIndx++;
                        }
                        else
                        {
                            if (data.faces.Count % 4 == 0)
                            {
                                data.faces.Add(0);
                            }
                            data.faces.Add(pointsLocal.IndexOf(point));
                        }
                    }
                    typeCurrentGeometry = "Mesh";
                    nameCurrentGeometry = "mesh" +
geometries.IndexOf(geometry).ToString();
                }

                var alpha = color[geometries.IndexOf(geometry)].Alpha /
255.0;
                var red =
color[geometries.IndexOf(geometry)].Red.ToString("X2");
                var green =
color[geometries.IndexOf(geometry)].Green.ToString("X2");
                var blue =
color[geometries.IndexOf(geometry)].Blue.ToString("X2");

                var inputMaterials = new SpectaclesMaterial
                {
                    uuid = Guid.NewGuid().ToString(),
                    type = typeCurrentGeometry + "BasicMaterial",
                    color = $"0x{red}{green}{blue}",
                    side = 2,
                    opacity = alpha,
                    transparent = true
```

```
                };
                inputGeometries.data = data;

                geometryList.Add(inputGeometries);
                materialList.Add(inputMaterials);

                var obj = new SpectaclesObject
                {
                    uuid = Guid.NewGuid().ToString(),
                    name = nameCurrentGeometry,
                    type = typeCurrentGeometry,
                    geometry = inputGeometries.uuid,
                    material = inputMaterials.uuid,
                    matrix = new double[] { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 0, 0, 0, 1 },
                    userData = new Dictionary<string, string>()
                };

                obj.userData.Add("layer", "Default");
                testObject.children.Add(obj);
            }

            if (verteciesForOpenGL.Count == 0)
            {
                return null;
            }
            Bitmap image = null;
            double calculatedScale = 1;
            Thread windowThread = new Thread(() => {
                SharpGLPngCreating hiddenWindow = new
SharpGLPngCreating(verteciesForOpenGL,lineVerteciesForOpenGL);
                hiddenWindow.Closed += (object sender, EventArgs e) => {
                    image = hiddenWindow.Image;
                    calculatedScale = hiddenWindow.scale;
                };
                hiddenWindow.ShowDialog();
            });
            windowThread.SetApartmentState(ApartmentState.STA);
            windowThread.Start();
            windowThread.Join();

            // scaling
            foreach(var geometry in geometryList)
            {
                geometry.data.scale = calculatedScale;
            }

            var container = new SpectaclesContainer
            {
                metadata = new Metadata(),
                geometries = geometryList,
                materials = materialList,
                obj = testObject
            };
            container.metadata.generator = "DynamoJsonExporter";
            container.metadata.version = 4.3;
            container.metadata.type = "Object";
            var json = JsonConvert.SerializeObject(container);
            Iteration iteration = new Iteration()
```

```
            {
                JSON = json,
                ImageMap = image
            };
            return iteration;
        }
    }
}
```

```
using Autodesk.DesignScript.Runtime;
using Newtonsoft.Json;
using DynamoStudioToDesignExplorer.Model;
using SimpleDesignExplorerFunction.Model;
using SimpleDesignExplorerFunction.Model.Constants;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using static DynamoStudioToDesignExplorer.Model.AuthModel;
using SimpleDesignExplorerFunction.Logs;
using System.Windows;
using ProtoCore.Runtime;
using SimpleDesignExplorerFunction.Resources;

namespace SimpleDesignExplorerFunction
{
    [IsVisibleInDynamoLibrary(false)]
    public static class SelectDesignNodeFunction
    {
        private static AuthModel authModelStatic;

        [IsVisibleInDynamoLibrary(false)]
        public static bool IsRefreshModel()
        {
            if (Status.OK ==
authModelStatic.RefreshTokens(DesignExplorer.WEB_API_URI))
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        [IsVisibleInDynamoLibrary(false)]
        public static void DeleteAuthModel()
        {
            if (File.Exists(Path.GetTempPath() +
DesignExplorer.DATA_FILE_NAME))
            {
                File.Delete(Path.GetTempPath() +
DesignExplorer.DATA_FILE_NAME);
            }
        }

        [IsVisibleInDynamoLibrary(false)]
        public static bool TryGetCookieFromFile()
        {
            authModelStatic = AuthModel.GetAuthModelFromFile();
            return authModelStatic.IsAccessable;
        }

        [IsVisibleInDynamoLibrary(false)]
        public static void NewAuthModel(string cookie)
        {
            authModelStatic = AuthModel.ParseCookieToAuthModel(cookie);
```

```
        }

        [IsVisibleInDynamoLibrary(false)]
        public static List<string> GetParametersList(int designId)
        {
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new Uri(DesignExplorer.WEB_API_URI);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", authModelStatic.AccessToken);
                var response =
client.GetAsync($"{DesignExplorer.API_VERSION}/users/{authModelStatic.Use
rId}/designs/{designId}/parameters").Result;
                string responseString =
response.Content.ReadAsStringAsync().Result;
                return
JsonConvert.DeserializeObject<List<string>>(responseString);
            }
        }

        [IsVisibleInDynamoLibrary(false)]
        public static AuthModel GetAuthResult(int id, List<string>
parameters, List<object> parametersValues, string designName)
        {
            if (parameters == null || parameters.Count !=
parametersValues.Count)
            {
                return null;
            }
            if (designName.Contains(".csv"))
            {
                throw new
Exception(ExplorerResource.DESIGN_NAME_INCLUDE_CSV_ERROR);
            }
            else if (designName.Equals(string.Empty))
            {
                throw new
Exception(ExplorerResource.DESIGN_NAME_EMPTY_ERROR);
            }
            else if (designName.Contains(" "))
            {
                throw new
Exception(ExplorerResource.DESIGN_NAME_INCLUDE_SPACE_ERROR);
            }
            foreach (var sym in Path.GetInvalidFileNameChars())
            {
                if (designName.Contains(sym.ToString()))
                {
                    throw new
Exception(string.Format(ExplorerResource.DESIGN_NAME_INCLUDE_SPECIAL_SYMB
OL_ERROR, sym));
                }
            }

            int inCount = 0;
            int outCount = 0;
            for (int i = 0; i < parameters.Count; i++)
            {
                string currentParameter = string.Empty;
```

```
                string currentParameterValue = string.Empty;

                if (!string.IsNullOrEmpty(parameters[i]) &&
!string.IsNullOrEmpty(parametersValues[i]?.ToString()))
                {
                    currentParameter = parameters[i].Trim();
                    currentParameterValue =
parametersValues[i].ToString().Trim();
                } else
                {
                    throw new
Exception(ExplorerResource.NOT_FILLED_ALL_FIELDS_ERROR);
                }

                if (currentParameter.Contains(",") ||
currentParameterValue.Contains(","))
                {
                    throw new
Exception(string.Format(ExplorerResource.PARAMETERS_INCLUDE_SPECIAL_SYMBO
L_ERROR, ","));
                }
                else if (currentParameter.StartsWith("in:"))
                {
                    inCount++;
                    if (currentParameter.EndsWith("in:"))
                    {
                        throw new
Exception(ExplorerResource.IN_PARAMETER_EMPTY_ERROR);
                    }
                }
                else if (currentParameter.StartsWith("out:"))
                {
                    outCount++;
                    if (currentParameter.EndsWith("out:"))
                    {
                        throw new
Exception(ExplorerResource.OUT_PARAMETER_EMPTY_ERROR);
                    }
                }
            }
            if (inCount == 0)
            {
                throw new
Exception(ExplorerResource.LEAST_ONE_IN_PARAMETER_ERROR);
            }
            if (outCount == 0)
            {
                throw new
Exception(ExplorerResource.LEAST_ONE_OUT_PARAMETER_ERROR);
            }



            string parametersString = string.Empty;
            for (int i = 0; i < parameters.Count; i++)
            {
                parametersString += (parameters[i] + "=" +
parametersValues[i].ToString() + ",");
            }
```

```
            DesignExplorerLog.Write("Getting parameters from inputs
started");
            authModelStatic.DesignId = id;
            authModelStatic.Parameters = parametersString.TrimEnd(',');
            authModelStatic.DesignName = designName;
            authModelStatic.RefreshTokens(DesignExplorer.WEB_API_URI);
            DesignExplorerLog.Write("Getting parameters from inputs
ended");
            DesignExplorerLog.Write("Sending parameters from select
design node ended");
            return authModelStatic;
        }

        [IsVisibleInDynamoLibrary(false)]
        private static List<Design> ResponseToDesign(string response)
        {
            var designsModel =
JsonConvert.DeserializeObject<List<Design>>(response);
            return designsModel;
        }
        [IsVisibleInDynamoLibrary(false)]
        public static List<Design> GetDesignsList()
        {
            using (HttpClient client = new HttpClient())
            {
                client.BaseAddress = new Uri(DesignExplorer.WEB_API_URI);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", authModelStatic.AccessToken);
                var response =
client.GetAsync($"{DesignExplorer.API_VERSION}/users/{authModelStatic.Use
rId}/designs/").Result;
                string responseString =
response.Content.ReadAsStringAsync().Result;
                return ResponseToDesign(responseString);
            }
        }

        [IsVisibleInDynamoLibrary(false)]
        private static User ResponserToUser(string response)
        {
            var userModel =
JsonConvert.DeserializeObject<UserReturnModel>(response);
            User user = new User
            {
                Id = userModel.Id,
                UserName = userModel.UserName,
                FirstName = userModel.FirstName,
                LastName = userModel.LastName,
                Email = userModel.Email,
                RefreshToken = userModel.RefreshToken,
                Folder = userModel.Folder,
                Project = userModel.Project
            };
            return user;
        }
    }
}
```

```
using Autodesk.DesignScript.Runtime;
using DynamoStudioToDesignExplorer.Model;
using SimpleDesignExplorerFunction.Logs;
using SimpleDesignExplorerFunction.Model;
using SimpleDesignExplorerFunction.Model.Constants;
using SimpleDesignExplorerFunction.Resources;
using SimpleDesignExplorerFunction.Waiting;
using System;
using System.IO;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading;
using System.Windows;


namespace SimpleDesignExplorerFunction
{
    [IsVisibleInDynamoLibrary(false)]
    public static class SendDesignNodeModelFunction
    {
        [IsVisibleInDynamoLibrary(false)]
        public static string SendDesign(AuthModel authModel,Iteration
iteration,bool isReady)
        {
            bool isError = false;
            DesignExplorerLog.Write("Sending design started");
            string NodeResult = string.Empty;
            string messageBoxMessage = string.Empty;
            string messageBoxTitle = string.Empty;
            MessageBoxImage level = MessageBoxImage.Information;
            double ownerTop = 0, ownerLeft = 0;
            Application.Current.Dispatcher.Invoke(() =>
            {
                var owner = Application.Current.MainWindow;
                ownerTop = owner.Top + (owner.Height / 2.0);
                ownerLeft = owner.Left + (owner.Width / 2.0);
            });
            Thread windowThread = new Thread(() => {
                if (isReady)
                {
                    if (authModel == null)
                    {
                        NodeResult =
ExplorerResource.SELECT_DESIGN_NODE_NULL_RESULT_ERROR;
                        isError = true;
                        return;
                    }
                    if(iteration == null)
                    {
                        NodeResult =
ExplorerResource.CONVERTER_NODE_NULL_RESULT_ERROR;
                        isError = true;
                        return;
                    }

                    WaitWindow waitWindow = new WaitWindow();
                    waitWindow.Top = ownerTop - (waitWindow.Height /
2.0);
```

```
                    waitWindow.Left = ownerLeft - (waitWindow.Width /
2.0);
                    waitWindow.Show();

                    if (!Directory.Exists(Path.GetTempPath() +
DesignExplorer.DIRECTORY_NAME))
                    {
                        Directory.CreateDirectory(Path.GetTempPath() +
DesignExplorer.DIRECTORY_NAME);
                    }
                    var splitString = authModel.Parameters.Split(',');
                    if (authModel.DesignId == -1)
                    {
                        string parametersName = string.Empty;
                        for (int i = 0; i < splitString.Length; i++)
                        {
                            parametersName +=
splitString[i].Split('=')[0] + ",";
                        }
                        parametersName +=
"img,threeD,IDAUTODESKPNG,IDAUTODESKJSON";
                        string csvFileName = Guid.NewGuid().ToString() +
".csv";
                        string csvFilePath = Path.GetTempPath() +
DesignExplorer.DIRECTORY_NAME + "/" + csvFileName;
                        File.WriteAllText(csvFilePath, parametersName +
"\n");
                        var designId = SendDesign(csvFilePath,
authModel.DesignName, authModel.UserId, authModel.AccessToken);
                        File.Delete(csvFilePath);
                        if (designId.IsSuccessStatusCode)
                        {
                            authModel.DesignId =
int.Parse(designId.Content.ReadAsStringAsync().Result.Trim('"'));
                        } else
                        {
                            messageBoxMessage =
ExplorerResource.CREATE_DESIGN_ERROR;
                            messageBoxTitle =
ExplorerResource.SERVER_ERROR;
                            level = MessageBoxImage.Error;
                            NodeResult =
designId.Content.ReadAsStringAsync().Result;
                            waitWindow.Close();
                            return;
                        }
                    }

                    string jsonFileName = Guid.NewGuid().ToString() +
".json";
                    string jsonFilePath = Path.GetTempPath() +
DesignExplorer.DIRECTORY_NAME + "/" + jsonFileName;
                    File.Create(jsonFilePath).Close();
                    File.WriteAllText(jsonFilePath, iteration.JSON);

                    string parametersValue = string.Empty;
                    for (int i = 0; i < splitString.Length; i++)
                    {
```

```
                        parametersValue += splitString[i].Split('=')[1] +
","";
                    }
                    string pngFileName = Guid.NewGuid().ToString() +
".png";
                    string pngFilePath = Path.GetTempPath() +
DesignExplorer.DIRECTORY_NAME + "/" + pngFileName;
                    iteration.ImageMap.Save(pngFilePath);
                    var result = SendIteration(jsonFilePath, pngFilePath,
parametersValue, authModel.AccessToken, authModel.UserId,
authModel.DesignId);
                    File.Delete(jsonFilePath);
                    File.Delete(pngFilePath);
                    if (result.IsSuccessStatusCode) {
                        NodeResult =
result.Content.ReadAsStringAsync().Result;
                        messageBoxMessage = NodeResult;
                        waitWindow.Close();
                        return;
                    } else
                    {
                        NodeResult =
result.Content.ReadAsStringAsync().Result;
                        messageBoxMessage =
ExplorerResource.CREATE_ITERATION_ERROR;
                        messageBoxTitle = ExplorerResource.SERVER_ERROR;
                        level = MessageBoxImage.Error;
                        waitWindow.Close();
                        return;
                    }
                } else
                {
                    DesignExplorerLog.Write("Sending design is not
ready");
                    NodeResult =
ExplorerResource.NOT_READY_SENDING_WARNING;
                    isError = true;
                    return;
                }
            });
            windowThread.SetApartmentState(ApartmentState.STA);
            windowThread.Start();
            windowThread.Join();
            if (isReady && !isError)
            {
                MessageBox.Show(messageBoxMessage, messageBoxTitle,
MessageBoxButton.OK, level);
            }
            DesignExplorerLog.Write("Sending design ended");
            if(isError)
            {
                throw new Exception(NodeResult);
            }
            return NodeResult;
        }
        [IsVisibleInDynamoLibrary(false)]
        private static HttpResponseMessage SendDesign(string csvFilePath,
string designName, string userId, string token)
        {
```

```
            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri(DesignExplorer.WEB_API_URI);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
                using (var content = new MultipartFormDataContent())
                {
                    content.Add(new StreamContent(
                        new MemoryStream(
                            File.ReadAllBytes(csvFilePath)
                            )
                        ),
                        "design",
                        designName + ".csv"
                        );
                    var result =
client.PostAsync($"{DesignExplorer.API_VERSION}/users/{userId}/designs",
content).Result;
                    return result;
                }
            }
        }
        [IsVisibleInDynamoLibrary(false)]
        private static HttpResponseMessage SendIteration(string
jsonFilePath, string pngFilePath, string parameters, string token, string
userId, int designId)
        {
            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri(DesignExplorer.WEB_API_URI);
                client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
                using (var content = new MultipartFormDataContent())
                {
                    content.Add(new StreamContent(
                        new MemoryStream(
                            File.ReadAllBytes(pngFilePath)
                            )
                        ),
                        "pngFile",
                        Path.GetFileName(pngFilePath)
                        );
                    content.Add(new StreamContent(
                        new MemoryStream(
                            File.ReadAllBytes(jsonFilePath)
                            )
                        ),
                        "jsonFile",
                        Path.GetFileName(jsonFilePath)
                        );
                    content.Add(new StringContent(
                        parameters.TrimEnd(',')
                        ), "info");
                    var result =
client.PostAsync($"{DesignExplorer.API_VERSION}/users/{userId}/designs/{d
esignId}/iterations", content).Result;
                    return result;
                }
            }}}}
```

Код інсталятора додатку Dynamo Studio

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" Name="Dynamo Studio to Design Explorer" Language="1033"
Version="1.0.0.24" Manufacturer="AMC Bridge LLC" UpgradeCode="84261fba-
b00d-4112-928c-d5d32f16e7d7" >
    <Package InstallerVersion="200" Compressed="yes"
InstallScope="perUser" Manufacturer="AMC Bridge LLC"
Description="Exporting Dynamo Studio geometry to Design Explorer"
InstallPrivileges="limited" />
    <Media Id="1" Cabinet="demoApp.cab" EmbedCab="yes" />

    <Property Id="ARPHELPLINK" Value="https://www.amcbridge.com" />
    <Property Id="ARPURLUPDATEINFO" Value="https://www.amcbridge.com" />
    <Property Id="ARPURLINFOABOUT" Value="https://www.amcbridge.com" />

    <WixVariable Id="WixUILicenseRtf" Value="lic.rtf" />

    <CustomAction Id='FooAction' BinaryKey='FooBinary'
DllEntry='CreateDynamoPackagesReg' Execute='immediate'
                  Return='check'/>

    <Binary Id='FooBinary'
SourceFile='..\CustomActionInstaller\bin\Release\CustomActionInstaller.CA
.dll'/>

    <InstallUISequence>
      <Custom Action='FooAction' Before='PrepareDlg'/>
    </InstallUISequence>

    <Property Id="PACKAGESPATH">
      <RegistrySearch Id="TS2018iSetupMain"
                      Root="HKCU"
                      Key="Software\DynamoStudioToDesignExplorer"
                      Name="DynamoExplorerPackagesPath"
                      Type="raw"
                      />
    </Property>
   <SetProperty Id="ProgramFilesFolder"
Value="[AppDataFolder]\[PACKAGESPATH]\packages"
Before="CostFinalize"><![CDATA[NOT Privileged]]></SetProperty>


    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
          <Directory Id="ADDINFOLDER"
Name="DynamoStudioToDesignExplorer">

          </Directory>
      </Directory>
    </Directory>

    <Feature Id="ProductFeature" Title="SimpleSetupProject" Level="1"
Description="TestFull" ConfigurableDirectory="ADDINFOLDER">
      <ComponentGroupRef Id="ProductFilesComponentGroup" />
    </Feature>

    <UI Id="MyWixUI_Minimal">
```

```
      <UIRef Id="WixUI_Minimal"/>
      <UIRef Id="WixUI_ErrorProgressText"/>
    </UI>
  </Product>

</Wix>
```

```
using Microsoft.Win32;
using System.IO;
using System;
using Microsoft.Deployment.WindowsInstaller;

namespace CustomActionInstaller
{
    public class CustomActions
    {
        [CustomAction]
        public static ActionResult CreateDynamoPackagesReg(Session
session)
        {
            session.Log("Begin creating packages path");
            RegistryKey currenUser = Registry.CurrentUser;
            RegistryKey softwareKey = currenUser.OpenSubKey("Software",
true);
            RegistryKey dynamoEplorerKey =
softwareKey.CreateSubKey("DynamoStudioToDesignExplorer");
            var appDataPath =
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
            DirectoryInfo dirInfo = new DirectoryInfo(appDataPath +
@"\Dynamo\Dynamo Core");
            string versionFodler = string.Empty;
            foreach (var item in dirInfo.GetDirectories())
            {
                double res;
                bool isNum = double.TryParse(item.Name, out res);
                if (isNum)
                {
                    versionFodler = item.Name;
                    break;
                }
            }
            dynamoEplorerKey.SetValue("DynamoExplorerPackagesPath",
@"Dynamo\Dynamo Core\" + versionFodler);
            return ActionResult.Success;
        }
    }
}
```