Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження та реалізація методів 3D моделювання з використанням 2D зображень (тема)

Виконав: студент 2 курсу, групи ІПЗм-18-4___
_____Федосенко_____Н.В.
_____
(прізвище, ініціали)
спеціальності 121- Інженерія програмного забезпечення____
(код і повна назва спеціальності)
____Освітньо-наукової програми_____
(тип програми)
____Інженерія програмного забезпечення_____
(повна назва освітньої програми)

Керівник _____ проф. Білоус Н.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук
Кафедра Програмної інженерії
Рівень вищої освіти – другий (магістерський)

Спеціальність 121-Інженерія програмного забезпечення
(код і повна назва)
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Федосенко Нікіті Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та реалізація методів 3D моделювання з використанням 2D зображень

затверджена наказом по університету від «27» 03 2020 р. № 473

2. Термін подання студентом роботи до екзаменаційної комісії 05 травня 2020 р.

3. Вихідні дані до роботи *моделі та алгоритми генерації 3D моделей, методи розпізнання складних об'єктів на зображеннях, пояснювальна записка. Використовувати ОС macOS, середовище об'єктно-орієнтованого проектування Xcode.*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд існуючих методів управління складними об'єктами, етапи аналізу, реалізація методів створення 3D зображень з 2D зображень.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка *
1.	Аналіз предметної галузі	27 січня 2020 р.	
2.	Огляд існуючих методів	10 лютого 2020 р.	
3.	Методи створення 3D моделей з 2D зображень	09 березня 2020 р.	
4.	Підготовка пояснювальної записки	23 березня 2020 р.	
5.	Спецчастина	06 квітня 2020 р.	
6.	Підготовка презентації та доповіді	14 квітня 2020 р.	
7.	Попередній захист	20 квітня 2020 р.	
8.	Нормоконтроль, рецензування	05 травня 2020 р.	
9.	Занесення диплома в електронний архів	09 травня 2020 р.	
10.	Допуск до захисту у зав. кафедри	10 травня 2020 р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання _____ _____2020 р.

Студент _____
(підпис)

Керівник роботи _____ __ доц. Білоус Н.В.___
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Атестаційна робота магістра: 77 с., 18 рис., 4 табл., 30 джерел.

PYTHON, 3D МОДЕЛЮВАННЯ, PYCHARM, КОМП'ЮТЕРНИЙ ЗІР, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ.

Об'єктом дослідження є алгоритми комп'ютерного зору створені, щоб аналізувати зображення для подальшого використання даних для генерації об'ємних моделей.

Метою роботи є реалізація методів 3D моделювання, для подальшого дослідження та вдосконалення існуючих методів, що допоможе швидше виконувати операцію моделювання об'єктів у сферах архітектури, дизайну, кіно, інженерії тощо.

Методи розробки базуються на засобах комп'ютерного зору та штучного інтелекту, на методах згорткових нейронних мереж та технологіях Python, Keras, Tensorflow, базі даних MongoDB та хмарному сховищі AWS S3, та середовищі розробки PyCharm. Результатом роботи є алгоритм, що аналізує вхідні зображення та створює 3D модель об'єктів.

PYTHON, 3D SIMULATION, PYCHARM, COMPUTER VISION, CONVOLUTIONAL NEURAL NETWORKS.

The object of the research is the computer vision algorithms for analyzing images and using this data for further 3D modelling.

The aim is to implement 3D modelling techniques for further research and improvement of existing methods, which will help to perform faster object modeling operations in the fields of architecture, design, cinema, engineering and more.

Development methods are based on computer vision and artificial intelligence, convolutional neural network techniques and Python, Keras, Tensorflow technologies, MongoDB database and AWS S3 cloud storage, and PyCharm development environment. The result is an algorithm that analyzes the input images and creates a 3D model of the objects.

ЗМІСТ

ВСТУП

У сучасному світі, використання 3D моделювання застосовується у великій кількості предметних областей. Об’ємна модель є невід’ємною частиною презентації концепту об’єкта, продукта тощо.

Тривимірна графіка реального часу на сьогоднішній день ефективно застосовується в багатьох предметних областях. Потужність комп'ютерних обчислень дозволяє обробляти досить складні сцени в режимі реального часу без втрати швидкості і якості відображення [1].

Ці можливості привели до появи інтересу до тривимірної візуалізації з боку фахівців з різних сфер діяльності. Так, в області архітектури і містобудування віртуальні будівлі з прогулянками по приміщеннях і віртуальні міста знаходять все більш широке застосування. Фотореалістична реконструкція об'єктів дозволяє на етапі проектування ефективно працювати із замовником, використання 3D моделей в процесі навчання, в музейних, реставраційних, рекламних, комерційних проектах також є сучасним і перспективним.

Особливе значення 3D технологія набуває в задачах інтерактивного проектування інженерних підсистем в системах автоматизації, іменованих «розумний будинок» (smart house).

Інтерактивна 3Д візуалізація почала застосовуватись або ж успішно може застосовуватись за такими напрямками:

– дизайн інтер’єру, екстер’єру;

– архітектура та містобудування;

– туристичний бізнес (проектування готелів, ресторанів, відпочинкових баз тощо);

– культура (реставрація пам’яток культури, замків, старовинних приміщень, проектування музеїв, театрів, галерей тощо);

– розважальна індустрія (створення ігор на основі архітектурної візуалізації реалістичної якості, читання книжок, перегляд телебачення, слухання радіо просто під час інтерактивної 3Д візуалізації);

– кіноіндустрія;

– реклама та 3D презентація;

– медицина (проектування органів та внутрішньої оболонки людини з подальшим відтворення руху імплантантів чи нанороботів всередині);

– інженерія (системи автоматизації, IT індустрії - взаємодія із програмними додатками, smart house тощо).

Актуальність. З даного переліку областей можливого застосування інтерактивної візуалізації слідує важливість аналізу даної технології та актуальність створення 3D інтерактивних додатків у відповідних напрямках використання [2]. Отже тема та засоби для виконання роботи є актуальними.

Зв'язок роботи з програмами наукових досліджень кафедри ПІ. Дана робота долідужє методи машиного навчання та нейроних мереж, що напряму зв’язано з напрямом створення систем штучного інтелекту.

Мета і задачі дослідження. Метою атестаційної роботи є аналіз та реалізація існуючих методів створення 3D моделей з використанням 2D зображень, та знаходження можливих шляхів поліпшення цих методів, для створення більш швидкого та продуктивного рішення.

Методи дослідження. Нейроні мережі для генерації 3D моделей з набору 2D зображень, використання методів вейвлет-перетворень, реконструкція 3D моделей з мап глибини зображень.

Публікації. Під час дослідження проблеми створення 3D моделей з 2D зображень, було написано наукові тезиси з порівнянням існуючих методів генерації 3D моделей. Тезіси було відправлено та буде опублікувано у рамках 15 Міжнародної наукової та технічної конференції у місті Львів. Текст публікації присутній у додатку В.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Дослідження і аналіз методів створення 3D моделей з 2D зображень

Сфера інформаційних технологій є одною з найпередовіших галузей у нашому столітті. Такий швидкий розвиток технологій обумовлено тим, що нові технології спрощують життя у багатьох сферах життєдіяльності.

Наука, бізнес, розваги – усе це залежить насамперед від інформаційних технологій. За допомогою них ми можемо прискорювати багато процесів які потребують складних обчислювань.

Сьогодні існує багато проблем при взаємодії людини та пристрою або комп'ютеру. По перше це відволікання людини на цей пристрій та втрата уваги у реальному світі. Також усі пристрої досить сильно зав'язані на енергії що їх живить та акумуляторах. Ще одним недоліком є те, що деякі системи або пристрої є незручними для використання у повсякденному житті, адже вони можуть потребувати додаткового обладнання, чи є дуже великими для пересування [3]. Наступним фактором незручності та проблеми використання є фізичні обмеження або недоліки людини. Не усі пристрої проектуються з урахуванням цих факторів, що призводить до проблем та навіть конфліктів.

Якщо взяти технології навігаторів, як приклад системи, з якою взаємодіє дуже багато людей, можна знайти певні недоліки:

– постійне відволікання на пристрій-навігатор, як вже зазначалося вище та як наслідок втрата уваги;

– значна похибка у точності навігаторів та відображення інформації;

– помилкові або застарілі дані, які можуть не оновлюватися досить довго;

– високе енергоспоживання таких пристроїв, адже йде прямий зв'язок із системами супутників та інтернету;

– досить важкий для розуміння інтерфейс програми навігатора, який може відштовхнути потенційних користувачів від програмного забезпечення або продукту.

Усі вищезазначені проблеми може бути вирішено за допомогою використання технології доповненої реальності. Перш за все доповнена реальність має інтуїтивно зрозумілий інтерфейс для роботи з нею, а також вона не відволікає користувача від дійсної реальності. Великі компанії вже почали розробку окулярів доповненої реальності, що у подальшому дасть змогу мігрувати усі поточні рішення, які застосовуються на смартфонах на ці пристрої. Це вирішить багато проблем, пов’язаних із користуванням тих або інших додатків та представить новий інтерфейс взаємодії комп’ютера та людини у повсякденному житті.

Впровадження інформаційних технологій у усі сфери життя дали потужний поштовх у створенні нових комп’ютерів та інтерфейсів взаємодії з нами. Це дивовижно, адже менш ніж п’ятдесят років тому люди навіть уявити не могли, як комп’ютер та обчислення тісно увійдуть у повсякденне життя звичайного користувача. Взаємодія комп’ютера та людини сьогодні є невід’ємною частиною кожного з нас, саме тому ця галузь є популярною та обговорюваною.

Взаємодія людини та комп’ютера зараз швидко розвивається. Регулярне і систематичне вдосконалення технологій призвело до появи абсолютно нових парадигм призначених для користувача інтерфейсу комп’ютеру. Глобалізація в сучасному суспільстві стала тією самою тенденцією, яка поширилася і на комп'ютерні науки. Віртуальну реальність торкнулася та сама глобалізація, що призвело до появи нового терміну – «доповнена реальність». Основна маса розроблених технологій зосереджена в більшості випадків на взаємодії людини і використанні ним пристроїв (стаціонарний комп'ютер, ноутбук, смартфон тощо). Доповнена реальність, в свою чергу, на підставі використання деяких конкретних комп'ютерних технологій, робить можливим поліпшення інтерфейсу людини і його взаємодії з реальним навколишнім світом.

Актуальний етап досліджень доповненої реальності стартував вже давно. За кордоном видавалося дуже багато робіт на дану тему, хто вивчає її актуальність і неймовірний потенціал. Дуже довго подібні технології здавалися важко реалізованими і залишалися більш гіпотетичними, ніж реальними. Коли ж розвиток споживчої електроніки досяг рівня, здатного забезпечити масове впровадження даної технології, доповнена реальність стала однією з найшвидших у розвитку галузей інформаційних технологій.

На теперішній час ця технологія є однією з найбільш цікавих тем для досліджень. Доказом цього є неймовірна хвиля популярності однієї з ігор для смартфонів – «Pokémon Go» від компанії Nintendo в 2016 році.

Доповнена реальність (від англ. Augmented Reality, AR – «розширена реальність») – це реальність із вбудованими (розширеними) в неї можливостями, яку сприймає людина. Тобто це те, що виходить в результаті введення в поле сприйняття будь-яких сенсорних даних, з метою доповнення відомостей про оточення [4].

Однак, якщо звернутися до витоків, то інструменти доповненої реальності пробував створити ще в п’ятнадцятому столітті архітектор Філіппо Брунеллески. Він малював об'єкт, який доповнював інший, існуючий в реальному світі і пропонував дивитися на нього через дзеркало з отвором. А в 1994 році Пол Милграм створив спектр реальностей. В цей спектр входить: фізичне навколишнє середовище (всі об'єкти реальні, їх можна відчути), доповнена реальність, доповнена віртуальність і, за підсумком, повністю віртуальна реальність (всі об'єкти і середовище згенеровано комп'ютером).

Том Коделл, дослідник в корпорації побудови авіаційної техніки Boeing, ввів термін «доповнена реальність» в 1990. Він застосував термін до приладу з цифровим дисплеєм, який «звертав» робітників під час збирання електричних проводів в літаках. У той час «доповнена реальність» визначалася як схрещування віртуальної і фізичної реальності, де цифровий візуальний ефект додавали до реального світу, щоб поліпшити його сприйняття людиною.

Існують також синоніми терміну «доповнена реальність»: «розширена реальність», «поліпшена реальність», «збагачена реальність», «збільшена реальність». Але таке використання термінів в глобальному сенсі не є правильним. Наприклад, терміни «розширена реальність», «збільшена реальність», «збагачена реальність» застосовні лише для позначення певних форм реалізації та аспектів практичного застосування доповненої реальності.

На сьогоднішній день доповнена реальність має дуже великий спектр застосування. Її використовують, перш за все, у військовій області, області медицини, авіації і мобільних технологій.

Доповнену реальність широко використовують в медицині, так як це значно спрощує процес проведення складних операцій і допомагає пацієнтам краще розуміти, що відбувається з їх організмом. Технологію використовують також для наочного навчання студентів-медиків. Технологія застосовується у військовій справі: на бойовому транспорті (літаки, машини, танки) і в обмундируванні самих військових (шоломи, зброя). Один із прикладів цього застосування доповненої реальності – проектування на лобовому склі кабіни літака інформації для пілота (швидкість, погодні зміни і інші показники).

Доповнену реальність активно використовують у інформаційному просторі. Поширення так званих браузерів доповненої реальності – зокрема, Wikitude, Layar, blippAR і інших, виводить технологію в масове використання. У газети, буклети, рекламні проспекти, журнали, карти поміщаються зображення-мітки для подальшої візуалізації цифрових об'єктів. У ролі додаткової інформації виступають всі види цифрових даних: текст, зображення, відео, звук або тривимірні об'єкти, статичні або анімовані. Реалізація відбувається за допомогою спеціальних додатків для смартфонів.

Існує досить багато додатків, які дозволяють за допомогою доповненої реальності отримати розширені відомості про оточення: браузери доповненої реальності, ігри, інформаційні програми.

Зростаюча популярність технології серед споживачів пов'язана з тим, що розвиток технологій дозволив ці функції реалізувати. Наприклад, обчислювальна потужність і набір датчиків в смартфонах дозволяють виробляти накладення будь-яких цифрових даних на отриманий в реальному часі звук або зображення. В іграх на все ті ж смартфони вже давно використовуються методи доповненої реальності. Наприклад, ще в 2004 році була випущена гра «Mosquitos», в якій зображення накладалося на відео, яке надходить до екрану з камери пристрою. Як уже було згадано вище, вибуховий успіх мала гра «Pokémon Go», що привело до перенесення на додану реальність таких ігор як «Minecraft», «Super Mario Bros» та інших, менш популярних ігрових додатків.

Великі кампанії вже давно почали працювати над доповненою реальністю. Деякі з цих проектів вже встигли зачинитись із-за складності реалізації, або неготовності користувачів до таких технологій. Корпорація Google працювала над окулярами «Project Glass» – одна з перших спроб вивести доповнену реальність в маси в 2013 році, на даний момент розробка припинена.

Одночасно з окулярами розроблялася платформа для доповненої реальності «Project Tango», яка була випущена у 2016 році та показала потужних потенціал цієї технології. Проект «Project Tango» – це технологія побудови трьох вимірної моделі навколишнього середовища за допомогою смартфону або планшету. Група ATAP створила 5-дюймовий смартфон, оснащений стереокамерой, сенсорами і програмним забезпеченням, які відстежують стан смартфона в 3D-просторі, а також сканують навколишній світ в реальному часі зі швидкістю 250 тисяч вимірів в секунду. Все це об'єднується в єдину 3D-модель за допомогою унікального процесора Myriad 1, розробленого стартапом Movidius. Коли модель готова, пристрій може перманентно визначати своє місце розташування всередині неї.

Microsoft в 2016 році випустила «Hololens», а влітку 2017 року Apple випустила платформу «ARKit» і відразу за нею компанія Google випустила свій аналог – «ARCore». Аналогічні розробки ведуть інші великі компанії, а також багато початківці компанії, так як ця тема дуже перспективна.

Ринок напрямків доповненої та віртуальної реальності знаходиться на дуже ранній стадії розвитку, але при цьому є вкрай інвестиційно привабливим, тому що він потенційно може стати дуже розвиненим та навіть перевершити ринок мобільних пристроїв. Загалом, в індустрію за минулий 2016 рік було інвестовано понад двох мільярдів доларів. Це величезна можливість для стартапів та інвесторів. Зараз основна виручка генерується шоломами віртуальної реальності і контентом, який для них створюється. Але картина буде змінюватися – велика ставка буде зроблена на доповнену реальність, так як на її розвиток виділяються величезні ресурси передовими компаніями в світі електроніки та інформаційних технологій. Навіть Ілон Маск згадував у своїх інтерв'ю про перспективи і можливості, які нам може відкрити доповнена реальність. За його словами, доповнена реальність та пристрої, пов'язані з нею, можуть повністю замінити ринок смартфонів.

Прогнозовані гігантські розміри ринку породжують величезні можливості для створення власного бізнесу. Згадаємо Apple і їх мобільні додатки: iPhone зробив революцію на ринку смартфонів. Хто заробив на цьому більше всіх? Розробники, які робили додатки під операційну систему даних смартфонів. Так як на той момент ринок додатків був дуже маленький, попит на них був величезний за рахунок дистрибуції цих пристроїв. Люди купували і качали практично все, що потрапляло на очі – популярними ставали навіть ігри, які раніше здавалися приреченими на провал.

Така поведінка ринку може скластися і на ринку технологій доповненої реальності. Це великі можливості для користувачів, компаній та власне фахівців інформаційних технологій усіх галузей.

У основі доповненої реальності лежить технологія комп'ютерного зору, адже пристрій повинен аналізувати навколишнє середовище і доповнювати його.
Комп'ютерний зір – це набор методів, які займаються вилученням інформації з зображень, і важливо відзначити, що зображення можуть бути самих різних типів. Це можуть бути фотографії, потокове або статичне відео, набори фото або медичний томографічний знімок.

Останнім часом дуже популярними алгоритмами є ті, що працюють із зображеннями і поєднують колірну інформацію про точки і їх положення в просторі. Так сталося, тому що сенсори, які отримують подібну інформацію, стали доступнішими, а розуміти фізичну реальність за допомогою таких сенсорів роботам або комп'ютерами стало набагато легше і швидше. Саме приріст потужності комп’ютерів надав таку змогу.

Інформація, яку різні алгоритми витягають із зображень в комп'ютерному зорі, також може мати різну природу, тобто різний тип. Деякі алгоритми просто розбивають зображення на частини, які відповідають окремим об'єктам або частин об'єктів.

Другий важливий розділ комп'ютерного зору – це побудова тривимірних моделей із зображень, наприклад з набору фотографій або з відео. Для наочності візьмемо приклад із тисячі фотографій, зроблених різними людьми в різний час, що відобразили Таймс Сквер в Нью Йорку. З даних фотографій буде будуватися тривимірна модель площі. Щоб це зробити комп'ютеру потрібно зіставити частини сцени, тобто зрозуміти, що на різних зображеннях різні фрагменти відповідають одним і тим же об'єктам в тривимірному світі. Це робиться за принципом відбору унікальних фрагментів і поширення сформованого знання на менш унікальні фрагменти. Комп'ютер для цього використовує алгоритми оптимізації, а ось що для цього використовують люди або тварини, коли будують в мозку тривимірні карти, ще до кінця не відомо. В результаті такого моделювання комп'ютер отримує фактично тривимірний зліпок сцени, і розуміє становище камери в момент, коли відбувалася фото та відео зйомка. Надалі ця інформація може використовуватися різними способами. Наприклад, для вимірювань, так званої фотограмметрії. Або отриману тривимірну модель можна використовувати в додатку комп'ютерної графіки в процесі створення фільмів (спецефекти: поєднання реальних і віртуальних об'єктів). Крім цього тривимірна модель активно використовується в комп'ютерних іграх, додатках реальної і доповненої реальності.

Комп’ютерний зір сьогодні займається рішенням проблеми розуміння зображень та що на них відбувається. Це називається високорівневим комп'ютерним зором. Цей розділ займається завданнями та інструкціями до комп’ютеру, що дуже легко сформувати. Наприклад, «виписати всі об'єкти з даного фото, визначити їх клас і місце розташування на фото». Або дано відео, на якому потрібно визначити, яким видом діяльності зайняті об'єкти [5].

Саме завдяки вищесказаному, є можливість створювати нові ідеї, програми та продукти, які дадуть змогу користувачеві відчути та отримати нові враження від взаємодії з його пристроєм. А швидкість створення нових технологій та конкуренція між компаніями-гігантами сприяє гарному розвитку цієї галузі, що дає змогу багатій кількості людей займатися технологіями, пов’язаними з доповненою та віртуальною реальністю.

Отже, можна зробити висновок про доцільність використання продукту який підтримує доповнену реальність. Ця технологія дозволяє створити новий досвід використання мобільних додатків, покращити взаємодію людини та комп’ютера, покращити існуючи системи та технології.

1.2 Виявлення проблем та актуалізація рішень.

Існує велика кількість методів та алгоритмів створення 3D моделей з 2D зображень, тому нижче будуть розглянуті тільки найбільш популярні та актуальні методи, принципи їх роботи та реалізації для подальшої реалізації та поліпшення.

1.2.1 Трьохвимірний метод Хафа

Перетворення Хафа - алгоритм, застосовуваний для вилучення елементів з зображення. Використовується в аналізі зображень, цифровій обробці зображень і комп'ютерному зорі. Призначений для пошуку об'єктів, що належать певного класу фігур, з використанням процедури голосування. Процедура голосування застосовується до простору параметрів, з якого і виходять об'єкти певного класу фігур по локального максимуму в так званому накопичувальному просторі (accumulator space), яке будується при обчисленні трансформації Хафа [6].

При автоматизованому аналізі цифрових зображень дуже часто виникає проблема ідентифікації простих фігур, таких як прямі, кола або еліпси. У багатьох випадках використовується алгоритм пошуку кордонів як предобробки для отримання точок, що знаходяться на кривій в зображенні. Призначення перетворення Хафа – вирішити проблему угруповання граничних точок шляхом застосування певної процедури голосування до набору параметризованих об'єктів зображення.

У найпростішому випадку перетворення Хафа є лінійним перетворенням для виявлення прямих. Пряма може бути задана рівнянням y = mx + b і може бути обчислена по будь-якій парі точок (x, y) на зображенні. Головна ідея перетворення Хафа – врахувати характеристики прямій не як рівняння, побудоване по парі точок зображення, а в термінах її параметрів, тобто m - кутового коефіцієнта і b - точки перетину з віссю ординат. Виходячи з цього пряма, задана рівнянням y = mx + b, може бути представлена у вигляді точки з координатами (b, m) в просторі параметрів [7].

Алгоритм перетворення Хафа використовує масив, званий акумулятором, для визначення присутності прямої y = mx + b. Розмірність акумулятора дорівнює кількості невідомих параметрів простору Хафа. Наприклад, для лінійної трансформації потрібно використовувати двовимірний масив, так як є два

невідомих параметри: m і b. Два вимірювання акумулятора відповідають квантованим значенням параметрів m і b. Для кожної точки і її сусідів алгоритм визначає, чи достатня «вага» кордону в цій точці. Якщо так, то алгоритм обчислює параметри прямої і збільшує значення в осередку акумулятора.

Потім, знайшовши осередки акумулятора з максимальними значеннями, зазвичай пошуком локального максимуму в просторі акумулятора, можуть бути визначені найбільш підходящі прямі. Найпростіший спосіб - це порогова фільтрація. Однак в різних ситуаціях різні методи можуть давати різні результати. Так як отримані прямі не містять інформацію про довжину, наступним кроком є знаходження частин зображення, відповідних знайденим прямим. Більш того, через помилки на етапі визначення меж фігур в просторі акумулятора також будуть міститися помилки. Це робить пошук відповідних ліній нетривіальним.

Одним із прикладів класичного підходу до виявлення тривимірного об'єкту є тривимірний метод Хафа. Даний метод став популярний в застосуванні для двовимірних зображень, де в якості областей інтересу використовує в основному контури об'єктів. При роботі в тривимірному форматі метод виділяє в тривимірному зображенні особливі точки для скорочення обчислювальних витрат. Такі точки виділяються за допомогою спеціального алгоритму кластеризації. Процедура голосування відбувається в акумуляторному просторі з урахуванням тільки виділених особливих точок [8].

Вектор нормалі обмежений довжиною в один символ і визначається двома кутами орієнтації, φ і ψ, як показано на рисунку 1.1.

В результаті виходять локальні максимуми в тих областях, де потенційно може знаходиться шуканий об'єкт. Ще одна складність з додаванням третього виміру, крім зростання обчислювального навантаження, - це ймовірність різної орієнтації сцени і шуканого об'єкта. Ця проблема вирішується за допомогою введення спеціальних векторів, що забезпечують інваріантність до обертання і повороту [9]. Трохи більш проробленим є метод геометричної зв'язності.

Основною відмінністю від тривимірного методу Хафа є інший алгоритм пошуку особливих точок, які об'єднуються в так звані особливі області і переводяться в формат, що описується спеціальним індексом форм. Отримані області у вигляді значень індексів записуються в двовимірні гістограми, де і відбувається процедура голосування по всіх локальних околицях, що містяться в тестовому об'єкті.

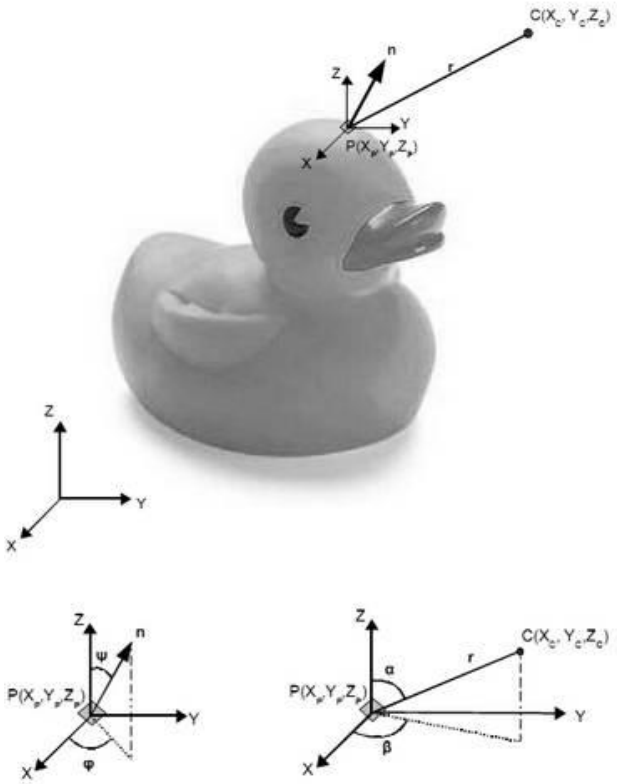

Рисунок 1.1 – параметри, включені в 3D метод Хафа

Існує велика кількість робіт з розпізнавання об'єктів в тривимірному хмарі точок, одержуваних при допомоги LiDAR і стереокамери, що використовують комбінацію різних індивідуальних ознак і дескрипторів з класифікацією методами машинного навчання [10]. Також широко поширені методи семантичної сегментації, де замість окремих класифікаторів використовуються структуровані класифікатори. До недоліків цього методу можна віднести той факт, що

використання хмар точок вимагає для обчислень околиці точок, що часто стає обчислювально нерозв'язною задачею з великою кількістю точок.

1.2.2 2.5D CNN

Архітектура згорткових нейронних мереж призначена і використовується для ефективного розпізнавання зображень, де чергуються згорткові шари (англ. convolutions) з нелінійними функціями активації (ReLU або гіперболічний тангенс tanh) і шари об'єднання або підвибірки (pooling layers).

На відміну від мережі прямого поширення, де кожен вхідний нейрон з'єднується з вихідним нейроном в наступному шарі, в згорткових мережах для отримання вихідних значень застосовуються згортки над кожним вхідним шаром. В операції згортки використовується матриця невеликого розміру, яка проходить по всьому поточному шарі, формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Ця матриця називається ядром згортки; вона використовується для різних нейронів вихідного шару [11].

Як відомо, багатошарові штучні нейронні мережі отримують вхідні дані (наприклад один вектор), після чого трансформують інформацію, проводячи її через ряд прихованих шарів. Кожен прихований шар складається з безлічі нейронів, де кожний нейрон має сильний зв'язок з усіма нейронами в попередньому шарі і де нейрони в якості одного шару повністю незалежні один від одного і не мають спільних з'єднань.

Останній повнозв'язну шар називається вихідним шаром, і в налаштуваннях класифікації він демонструє число класів. Звичайні штучні нейронні мережі (Рис. 1.2) погано масштабуються у випадку з зображеннями великих розмірів. Так, в системі комп'ютерного зору CIFAR-10, картинки становить [32 × 32 × 3] (32 - ширина, 32 - висота, 3 - канали кольорів), тому один повністю підключений нейрон в першому прихованому шарі звичайної нейронної мережі має вагу 32 * 32 * 3 = 3 072. Здається, що це значення можна змінювати, але повнозв’язна структура не

масштабується для великих зображень. Картинка обсягом більше, наприклад, [200 × 200 × 3], приведе до того, що повністю підключений нейрон буде важити 120 000. Крім того, потрібно залучити кілька таких нейронів, що призведе до додавання параметрів. Недоліком повнозв'язності буде величезна кількість параметрів, що швидко привести до перенавчання [12].

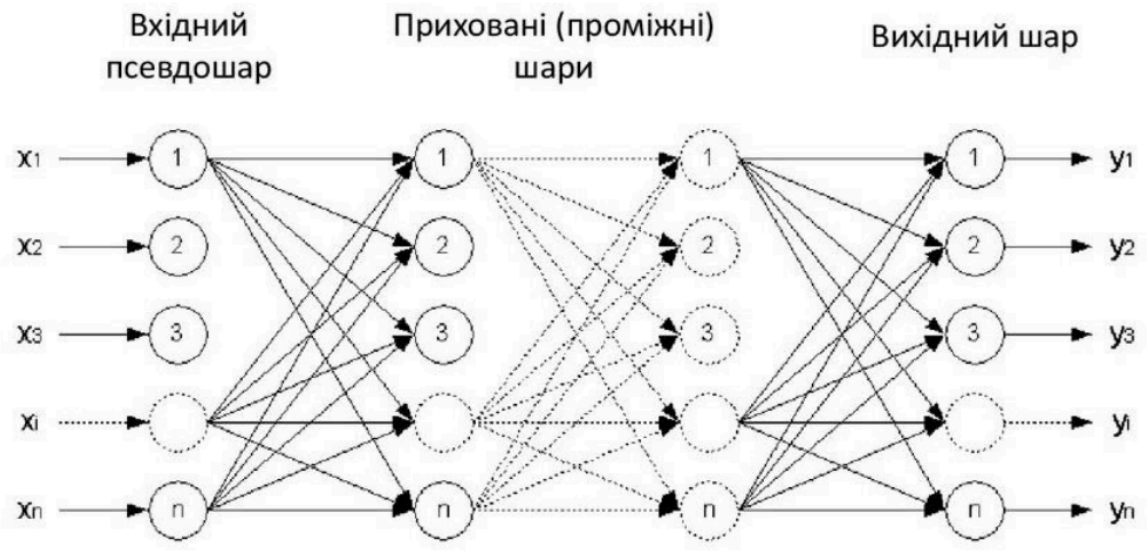

Рисунок 1.2 – Приклад багатошарової нейронної мережі

Згорткові нейронні мережі (Рис 1.3) користуються тим, що ввідні дані складаються з зображень, і вони обмежують побудову мережі більш розумним шляхом. На відміну від звичайної нейронної мережі, шари ЗНМ складаються з нейронів, розташованих в 3-х вимірах: ширині, висоті і глибині, тобто у вимірах, які формують об'єм. Наприклад, зображення на вході CIFAR-10 є вхідними активаційними об'ємами, а об'єм сформований вимірами 32 × 32 × 3. Як буде описано далі, нейрони будуть підключені тільки до невеликої області шару. Крім того, результуючий вихідний шар для даної системи комп'ютерного зору складе 1x1x10, оскільки до кінця побудови ЗНМ зображення перетвориться в єдиний вектор оцінок класу, розташованих по вимірюванню глибини [13].

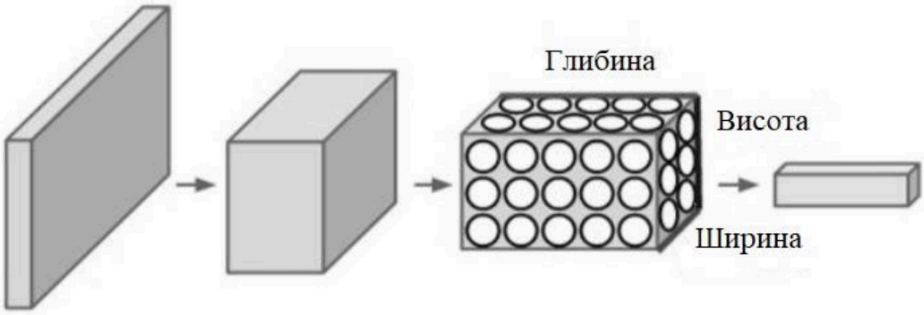

Рисунок 1.3 – Згорткова нейронна мережа

Надихнувшись успішним застосуванням згортальних нейронних мереж для вирішення задач розпізнавання на двовимірних зображеннях, деякі автори розширили їх використання для стерео даних. такі підходи обробляють канал з «глибиною» як додатковий канал, поряд зі звичайними каналами R, G, B. Однак, при цьому не в повній мірі використовується геометрична інформація в тривимірних даних, що ускладнює інтеграцію між зоровими точками.

Для LiDAR даних запропоновані ознаки, локально отримані на даних з поданням 2.5D, а деякі роботи досліджують даний підхід в поєднанні з різновидом так званого навчання без вчителя [14]. В роботі запропонована кодування, яка ефективно використовує інформацію про глибину, але підхід все одно двумерноорієнтірованний. Виходить більш точне уявлення про навколишнє середовище.

1.3 Аналіз існуючих рішень

Перед розглядом існуючих систем слід зазначити, що систем для 3D моделювання з використанням 2D зображень багато і тому далі будуть наведені лише деякі з них. Для аналізу був використаний метод тестування по стратегії

чорного ящика [5]. В ході аналізу існуючих рішень для генерації 3D моделей були проаналізовані існуючі системи і виявлені їх недоліки.

1.3.1 Веб-сервіс Selva 3D

Selva 3D це онлайн-сервіс для генерації 3D-файлів безпосередньо з зображень. Користувач системи має можливість завантажувати зображення у форматах PNG, JPG або GIF. На рисунку 1.4 наведено інтерфейс сервісу Selva 3D.

Рисунок 1.4 – Інтерфейс веб-сервісу Selva 3D

Сервіс пропонує зрозумілий інтерфейс та швидке генерування 3D-файлів з майже усіх форматів зображень. Після завантаження зображення, сервіс надає можливість внести мінімальні корегування у 3D модель перед зберіганням. Функція отримання згенерованих 3D моделей у максимальній якості коштує 3 долара США за кожне зображення.

1.3.2 Веб-сервіс Smoothie 3D

Веб-сервіс Smoothie 3D надає можливість створення 3D моделей з зображень. Головною метою сервісу є спрощення процесу створення моделей для друкування на 3D принтерах або використання в системах доповненої реальності.

На рисунку 1.5 наведено інтерфейс сервісу Smoothie 3D.

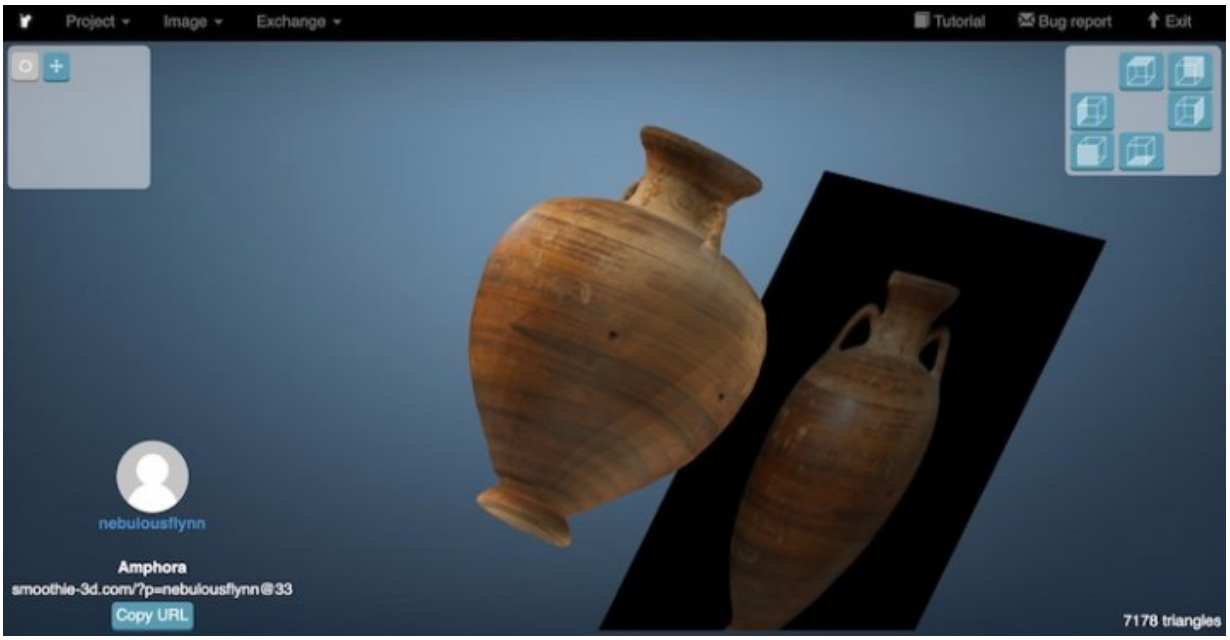

Рисунок 1.5 – Веб-сервіс Smoothie 3D

За допомогою сервісу Smoothie 3D користувачі можуть завантажити зображення та мають велику кількість налаштувань 3D моделі для отримання найкращого результату.

Веб-сервіс Smoothie 3D пропонує велику кількість можливостей для створення 3D файлів з зображень. До переваг веб-сервісу відноситься потужний функціонал та безкоштовність користування сервісом. До основних недоліків сервісу можна віднести складність внесення корегувань та необхідність витратити багато часу на вивчення механіки роботи сервісу, та обов’язкова реєстрація для користування сервісом.

1.3.3 Додаток Insight3d

Insight3d це додаток для платформ Windows та Linux, що дозволяє створювати 3D-файли із серії зображень форматів PNG, JPG. Інструмент вимірює положення в просторі, а також оптичні параметри камери. У результаті, Insight3d генерує текстуровану полігональну модель.

На рисунку 1.6 наведено інтерфейс додатку Insight3d.

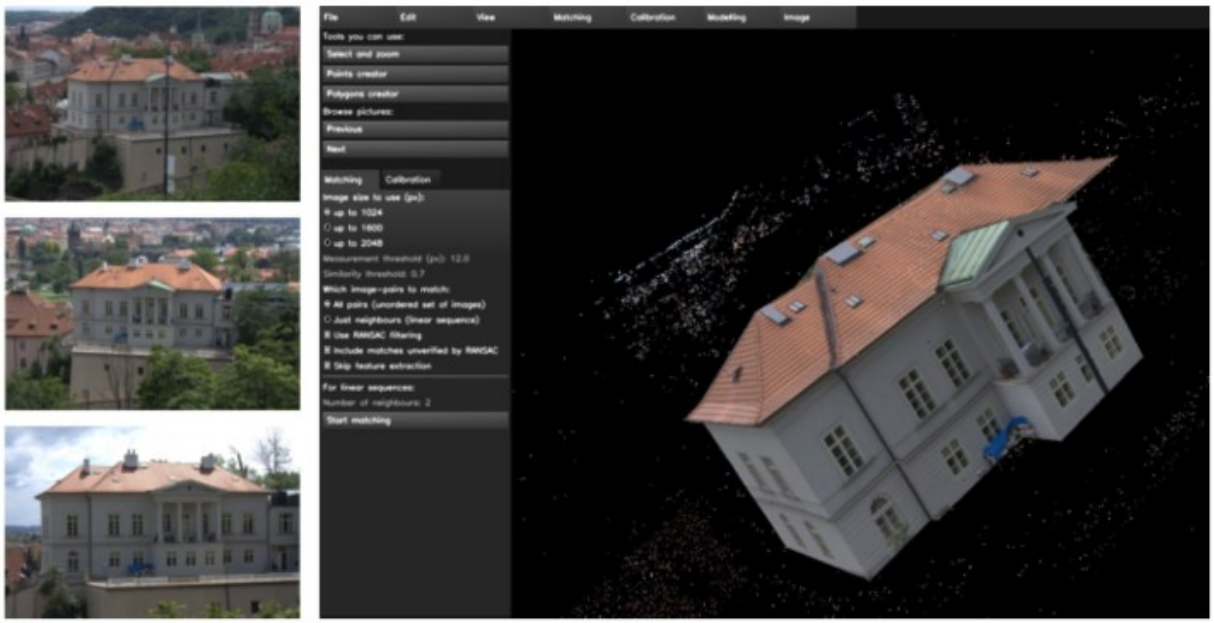

Рисунок 1.6 – Додаток Insight3d

Даний інструмент є доволі потужним та видає гарний результат, але за наявністю серії зображень об'єкта. До недоліків можна віднести обмеженість у використанні, що зв'язана з прив'язкою до платформа Linux та Windows та обмеженність на завантаження якісних фото.

1.4 Постановка задачі дослідження

Перед початком роботи для подальшого проектування програмної системи необхідно визначитися з вимогами до неї. В даному випадку необхідно виділити функціональні властивості, які мають бути реалізовані в готовій системі.

Метою роботи є проектування веб-додатку для генерації 3D моделей з 2D зображень. Ця система повинна дати змогу користувачам завантажити зображення та отримати згенеровану 3D модель.

Результатом роботи повинна стати система для створення об’ємних моделей з 2D зображень, що включає в себе сервер для збереження та обробки завантажених файлів і веб-клієнт з функціями завантаження зображень, перегляду моделей згенерованих різними методами та можливістю зберегти файли моделей.

У ході розробки програмного забезпечення потрібно реалізувати трьох вимірний метод Хафа та алгоритм створення 3D зображень з використанням згорткових нейроних мереж.

За рівнем доступу до системи користувачі повинні поділятися на:

а) неавторизований користувач – повинен мати доступ лише до базової інформації про додаток як систему, а також до функціоналу авторизації та реєстрації;

б) авторизований користувач – має доступ до:

1) адмін панелі користувача;
2) перегляд завантажених зображень та згенерованих 3D моделей;
3) функціоналу завантаження зображень.

За предметом зацікавленості користувачі поділяються на ролі:

– представник компанії – зацікавлений в інтеграції системи на свій веб-ресурс;

– користувач, що зацікавлений у статистичних даних інтеграції для виділення настроїв та побажань відвідувачів веб-сайтів з інтеграцією системи.

Отже, в результаті слід розробити програмну систему, що включає в себе веб-клієнт користувача, модуль генерації 3D зображень, а також сервер для надання API, зберігання та обробки даних. Система надасть змогу користувачам завантажувати зображення та через додаток переглядати та мати можливість зберегти файли 3D моделей.

Для реалізації програмної системи буде використано мову программування Javascript та фреймворк Vue.js, створений для розробки веб-клієнтів систем. Сервер обробки клієнтських даних та надання API буде створений з використанням мови програмування Python та фреймворку Flask. Для реалізації модулю обробки 2D зображень та генерації 3D моделей буде використано мову програмування Python та фреймворку штучного інтелекту та машиного навчання Keras. Більш детально використані технології буде розглянуто у наступних розділах.

1.5 Висновки до розділу 1

В розділі «Аналіз предметної галузі» спершу було досліджено ситуацію у сфері обробки зображень та генерації 3D моделей. Далі було розглянуто існуючі методи та алгоритми для розпізнавання об’єктів на зображеннях для подальшої генерації об’ємних моделей. Було розглянуто переваги та недолікі кожного з методів, приведені приклади використання. Детально розглянуті проблеми аналогів і предметної області. Після цього була поставлена задача створення веб-додатку, що надасть змогу користувачам завантажувати зображення та отримувати згенеровані з використанням різних методів 3D моделі. Було описано ролі користувачів системи та який функціонал повинен бути реалізован для кожної з груп. Все це надає змогу перейти до формування вимог до розроблюваного веб-додатку.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Вимоги до функціональності веб-клієнту

Веб-клієнт повинен задовольняти заступні функціональні вимоги:

– веб-клієнт повинен підтримувати завантаження нових зображень у систему;

– веб-додаток повинен підтримувати перегляд та зберігання згенерованих для користувача 3D моделей;

2.2 Вимоги до функціональності серверу

Сервер розроблюваної програмної системи повинен задовольняти наступні функціональні вимоги:

– сервер повинен забезпечувати авторизацію доступу до даних для запобігання несанкціонованого доступу;

– сервер повинен забезпечувати обробку завантажених користувачами зображень;

– сервер повинен забезпечувати збереження інформації про користувачів та згенеровані 3D моделі;

2.3 Вимоги до інтерфейсу

Розроблювана програмна система повинна задовольняти наступні вимоги до інтерфейсу:

– програмне забезпечення повинно складатися з серверної і клієнтської частини;

– кінцевий продукт повинен бути представлений у вигляді .py файлів (сервер), .html, .js, .css (web-клієнт) файлів;

– повинна бути реалізована генерація 3D моделей з 2D зображень.

2.4 Операційні вимоги

Розроблювана програмна система повинна задовольняти наступні операційні вимоги:

– для роботи програмної системи необхідний веб-браузер Chrome від компанії Google не пізніше версії 52, або веб-браузер Mozilla Firefox від компанії Mozilla Foundation не пізніше версії 48;

– для роботи сервера ПС необхідно щоб були розгорнуті SQL база даних PostgreSQL, Redis, встановлена платформа Python не пізніше 3.0.0.

2.5 Вимоги до ресурсів

Для експлуатації серверної частини програмної системи повинно бути достатньо наступного програмного забезпечення: 8192 Мб ОЗУ, IntelCore i3-4340, 5 Гб дискового простору для розгортання SQL бази даних PostgreSQL, запуску in-memory бази даних Redis та розгортання серверу на платформі Python [15].

2.6 Вимоги до документації

Розроблюване програмне забезпечення повинно супроводжуватися наступними документами (відповідно до методичних рекомендацій): Розрахунково-пояснювальна записка (відповідно до методичних рекомендацій, програмне забезпечення повинно супроводжуватися пояснювальною запискою, згідноз «ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».

2.7 Вимоги до середовищ виконання і платформ

Розроблювана програмна система повинна задовольняти наступні вимоги до середовища виконання і платформ:

– експлуатація серверної частини ПО повинна бути можлива на будь-який ЕОМ, з технічними характеристиками не нижче мінімальних (2048 Мб ОЗУ і процесор Intel Core i3-4340 або рівний йому за продуктивністю), і з встановленою Node.js версією 6.10.0 або пізніше;

– експлуатація веб-клієнта повинна бути можлива в браузерах: IE версії 10 або вище, Microsoft Edge, Mozilla Firefox версії 48 або вище, Google Chrome версії 52 або вище.

2.8 Висновки до розділу 2

В розділі «Формування вимог до програмної системи» було сформовано вимоги до функціональності веб-клієнту та серверу, а також основні вимоги до проектування системи.

Далі були описані вимоги до стеку технологій, а прийняті рішення будуть описані більш детально в розділах проектування системи та опис прийнятих програмних рішень. Все це надає змогу приступити до етапу проектування Веб-додатку для створення 3D-моделей з 2D-зображень.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Структура системи

У ході дослідження існуючих сервісів для створення 3D моделей з зображень, була виявлена закономірність, яка полягає в тому, що користувач повинен обов’язково реєструватися у системі та витрачати на це певний час. Тому доцільно створити систему, яка б зменшила витрати часу на введеня персональної інформації та надавала можливість генерувати 3D моделі швидко. Система буде розподілена на два компоненти: веб-клієнт, сервер [16]. Також у системі буде два сховища даних призначених для різних цілей: Redis, MongoDB. Користувач, в основному, буде взаємодіяти з веб-клієнтом для завантаження зображень та отримання створених 3D моделей. Також, у користувача буде можливість створити сторінку у системі та зберігати інформацію про завантажені зображення, створені моделі та редагувати 3D моделі.

3.2 UML проектування програмного забезпечення

Моделювання розроблюваної системи проводиться з використанням мови UML (unified modeling language) для побудови діаграм, що допоможуть відобразити функціональність та внутрішню структуру системи. UML – мова графічного опису для об’єктного моделювання в галузі розробки програмного забезпечення [17]. UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, що називається UML-моделлю. При розробці системи були створені діаграми наступних типів:

- діаграма варіантів використання;
- діаграма класів;

– діаграма розгортання.

На рисунку 3.1 приведена діаграма варіантів використання.

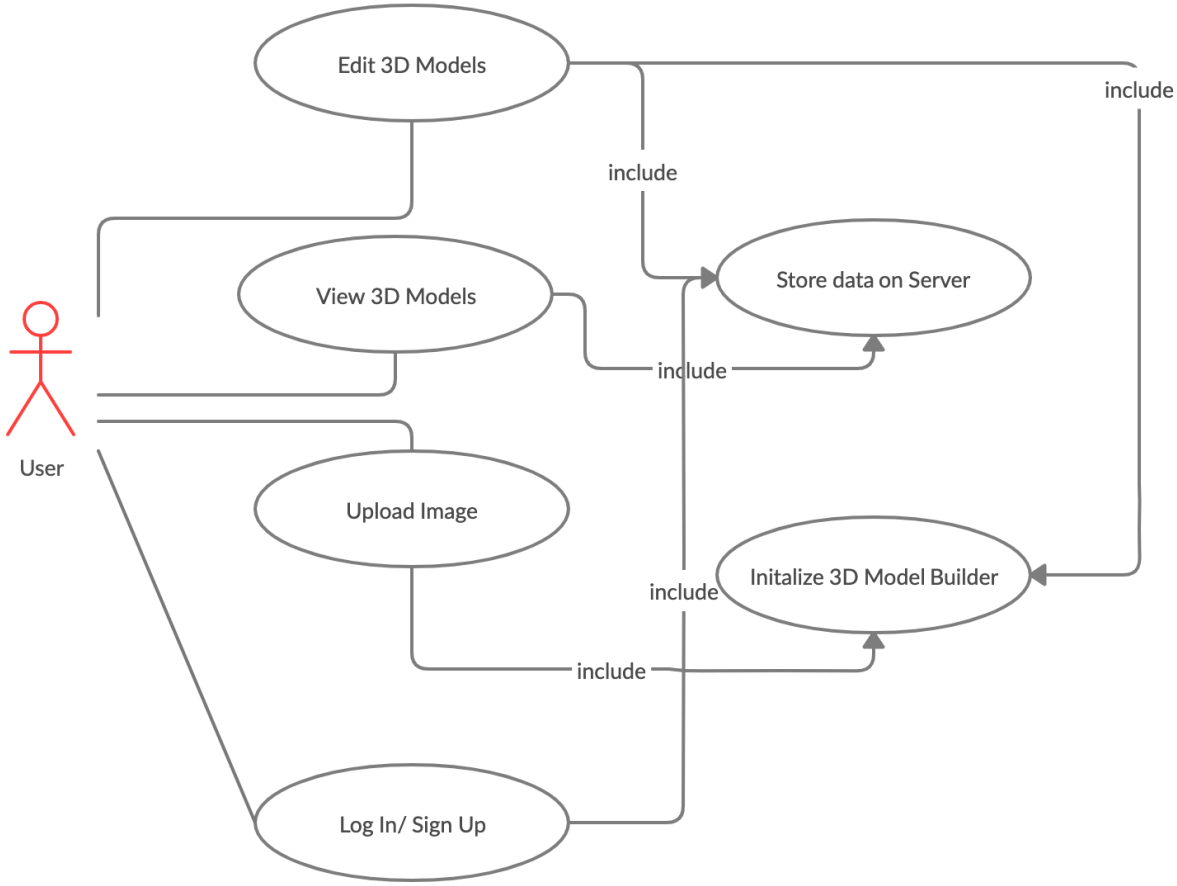

Рисунок 3.1 – Діаграма варіантів використання

Діаграма показує, що в системі 1 користувач:

– користувач – це людина, що використовує веб-додаток для створення 3D моделей;

Основна дія, яку виконує користувач, це завантаження зображень до системи для подальшої генерації системою 3D моделей. Користувач може як завантажити зображення напряму, або створити акаунт, щоб надалі бачити усі завантажені зображеня та створені 3D моделі. Функціонал редагування та перегляду 3D моделей доступний лише авторизованим користувачам системи.

На рисунку 3.2 зображено частину діаграми класів. Діаграма показує лише один з аспектів серверного додатку, та й то не в цілому обсязі, бо для показу, навіть половини усіх класів, немає ніякої можливості. Отже, на діаграму класів були винесені тільки основні класи системи, такі як: AuthenticationController, UserController, UploadController, Analyzer.

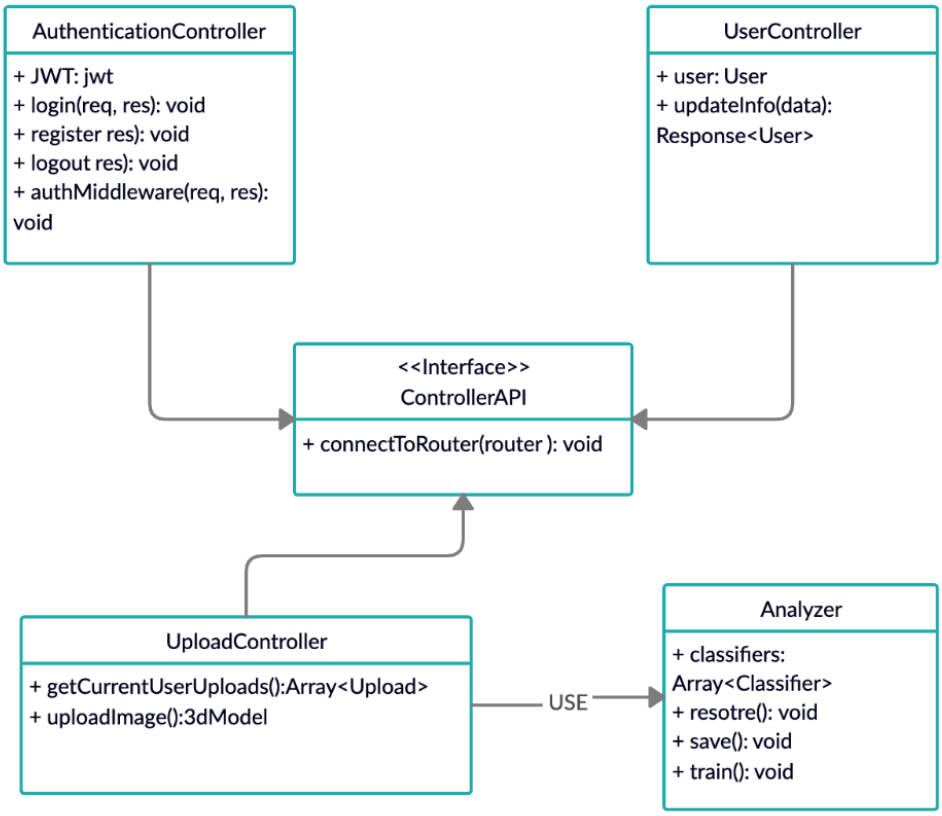

Рисунок 3.2 – Діаграма класів

На діаграмі зображений компонент архітектури MVP, що застосовується на сервері веб-додатку. У ролі моделей виступають схеми бази даних MongoDB, а схема класів контролерів відображена на рисунку 3.2. Тут зображено інтерфейс ControllerAPI, який включає в себе необхідність реалізації методу connectToRouter(), який дозволяє класам контролерів організовувати зв’язок з веб-клієнтом через REST API. Усі класи контролерів реалізовують цей інтерфейс.

Клас UploadsController надає функціонал, необхідний для додавання нових зображень до системи, а також містить такий корисний метод, як getCurrentUserUploads (), який повертає усі зображення користувача системи, які зберігаються у базі даних [18].

Також, необхідні для функціонування системи методи надає клас Analyzer. В цьому класі реалізовані методи машинного навчання, які застосовуються для аналізу та класифікації пошукових запитів. Метод restore() відновлює класифікатори, створені раніше, з бази даних. Метод train() запускає навчання класифікаторів для певної групи.

На рисунку 3.3 зображено діаграму розгортання. Діаграма розгортання демонструє, що система складається з веб-клієнту у браузері та веб-сервера. Крім того, у якості сховищ для даних використовуються Redis та MongoDB.

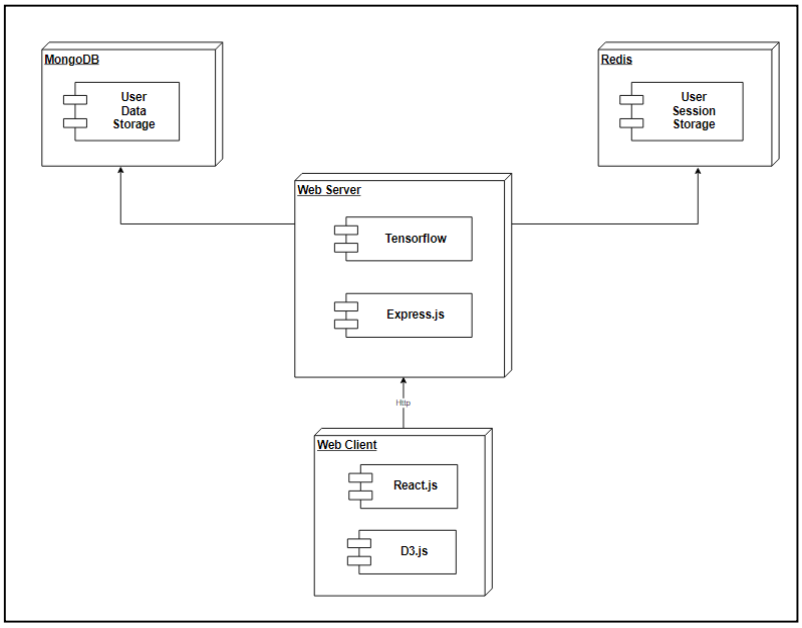

Рисунок 3.3 – Діаграма розгортання

Також, діаграма показує, що веб-клієнт використовує фреймворк React.js для розробки та бібліотеку D3.js для генерування графіків. Веб-сервер, у свою чергу, використовує фреймворка Express.js для розробки HTTP-сервісів та фреймворк Tensorflow для реалізації методів machine learning для аналізу пошукових запитів.

3.3 Проектування архітектури програмного забезпечення

Система буде розподілена на два компоненти: веб-клієнт та сервер [19]. Також у системі буде два сховища даних призначених для різних цілей: Redis, MongoDB. Користувач, в основному, буде взаємодіяти з веб-клієнтом, для перегляду детальної інформації, завантаження зображень та редагування моделей [20].

Загалом, архітектуру системи можна визначити, як багаторівневу багатошарову клієнт-серверну архітектуру [21]. До переваг, якої можна віднести легкість горизонтального масштабування і стійкість до непередбачуваних збоїв, а до недоліків, залежність функціоналу, або навіть доступності клієнтів від роботи серверу.

3.3.1 Порівняльний аналіз систем для розробки веб-клієнта

Існує велика кількість фреймфорків для розробки веб-клієнтів систем. Кожен із них має свої переваги та недоліки. Було вирішено провести аналіз найбільш популярних систем для розробки веб-кліенту. Виділено наступні системи: React.js, Angular.js, Ember.js. Кожна з цих систем реалізує різні принципи та архітектурні рішення. Також, системи мають різну швидкість рендерингу даних у веб-браузері.

Результати порівняння наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння систем для розробки веб-клієнта

Назва	Реалізована архітектура	Швидкість роботи	Документація
React.js	Javascript library	Швидка	Детальна, розбита на модулі
Angular.js	MVM framework	Швидка	Детальна

У результаті аналізу було обрано систему React.js, бо вона надає доволі детальну документацію, швидкий рендеринг веб-сторінок, та дозволяє реалізовувати архітектуру системи самостійно. Компоненто-орієнтований підхід по побудови архітектури клієнтського веб-додатку дозволив створити гнучку, легко масштабовану та модульну кодову базу. Для розробки клієнтського додатку було використано JavaScript фреймворк React.

React – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram та спільнотою індивідуальних розробників. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер, і може бути використане у поєднанні з іншими JavaScript бібліотеками або фреймворками.

3.3.2 Порівняльний аналіз систем для розробки серверу

Сервер обробки клієнтських даних було створено із використанням стандартної та найпоширенішої архітектури для веб-застосувань, тобто Model View

Controller. Були проаналізовані системи для розробки серверної частини системи, а саме Express.js, Koa.js, Sails.js. Результати аналізу систем приведений у таблиці 3.2.

Таблиця 3.2 – Порівняння систем для розробки серверу

Назва	Кількість користувачів	Документація	Продуктивність	Використання ресурсів системи
Express.js	Багато користувачів	Детальна, розбита на модулі	Висока	Низьке
Koa.js	Середня кількість користувачів	Детальна	Висока	Низьке
Sails.js	Середня кількість користувачів	Меньш детальна	Низька	Високе

У результаті аналізу було виявлено, що система Express.js має найбільш детальну та зрозумілу документацію, високу продуктивність відносно інших варіантів, та низьке використання ресурсів. Отже, у якості фреймворка для розробки веб-серверу для обробки клієнтських запитів було обрано Express. Express – це вільний та відкритий веб-фреймворк, написаний мовою JavaScript, який використовує Node.js. Express дозволяє швидко створювати крос-платформові застосунки. Він інтегрується з MongoDB і використовує розподілений дата-протокол та шаблон проектування публікація-підписка, щоб автоматично оновлювати дані на клієнті без необхідності писати відповідний код для синхронізації. Необхідно зазначити, що сама серверна частина клієнтського додатку описуваної програмної системи грає роль прикладного програмного інтерфейсу, що надає доступ користувачеві до документо-орієнтованої системи керування базами даних MongoDB.

3.3.3 Порівняльний аналіз систем баз даних

Вибір бази даних є дуже важливим питанням у проектуванні системи. Це питання залежить від багатьох факторів: як зберігаються дані, швидкість виконування запитів, можливості масштабування. Тому, першим чином, було вирішено проаналізувати існуючі рішення та вибрати найбільш оптимальне для системи аналізу пошукових запитів користувачів.

Перед вибором бази даних, був проведений аналіз існуючих SQL та NoSQL рішень, для знаходження найбільш оптимального варіанту.

Для зберігання даних системи було вирішено використовувати NoSQL бази даних. Основною проблемою SQL баз даних є складність обробки даних під навантаженнями, актуальними у нашій тематиці. Також, SQL бази даних мають не гнучкий дизайн логічної структури.

Отже, було вибрано три NoSQL бази даних та проведений детальний аналіз їх реалізації та функціоналу. Для аналізу були обрані: CouchDB, MongoDB та Redis.

3.3.3.1 База даних CouchDB

CouchDB – розподілена документо-орієнтована система управління базами даних класу NoSQL-систем, що не вимагає опису схеми даних. Запити до CouchDB та індексація даних можуть виконуватися згідно з парадигмою MapReduce, використовуючи для формування логіки вибірки даних мову JavaScript. До переваг даного рішення відноситься легка інтеграція з мовою програмування Javascript, зберігання даних у форматі JSON, та вбудований функціонал для розгортання веб-серверу на протоколі HTTP. До недоліків даного рішення відноситься обмеження на запис до бази у тільки один запит запису у час, відсутність підтримки розподілу обчислень на кілька вузлів.

3.3.3.2 База даних MongoDB

MongoDB – документо-орієнтована система керування базами даних (СКБД) з відкритим сирцевим кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів. MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД. До переваг даного рішення відноситься наявність вбудованих засобів для забезпечення шардінгу, комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови. Також, дана база даних забезпечує ефективне зберігання бінарних даних великих обсягів.

3.3.3.3 База даних Redis

Redis – розподілене сховище пар ключ-значення, які зберігаються в оперативній пам'яті, з можливістю забезпечувати довговічність зберігання за бажанням користувача. Це програмне забезпечення з відкритим сирцевим кодом написане на C. Всі дані у повному обсязі кешуються в оперативній пам'яті. Зберігання всіх даних в оперативній пам'яті дозволяє досягнути значної продуктивності. Отже, до переваг Redis можна віднести швидку обробку операцій, порівняно з іншими аналогами. До недоліків системи можна віднести зберігання даних у оперативній пам’яті, що може привести до втрати даних при збої у комп’ютерній системи.

3.3.3.4 Результати порівняльного аналізу баз даних

В результаті аналізу систем баз даних, було виявлено, що деякі з них не надають готових рішень для масштабування системи, архітектури деяких можуть привести до часткової або повної втрати даних.

У таблиці 3.3 наведено результати порівняння баз даних.

Таблиця 3.3 – Порівняння NoSQL баз даних

Назва	Модель даних	API запитів	Система збереженняданих
CouchDB	Документи	Map/Reduce	Append-only B-tree
MongoDB	Документи	Cursor	B-tree
Redis	Колекції	Collection	In-memory

В ході дослідження у якості бази даних, обрана база MongoDB [22]. Дана база даних забезпечує швидке виконування операцій, вбудовані варіанти для масштабування, та надає зручний інтерфейс для роботи з базою на мові Javascript.

3.4 Проектування бази даних

В ході дослідження у якості бази даних, обрана база MongoDB [23]. Було створено наступні сутності: користувач, зображення, модель. Крім того, для пошуку були створені категоріальні індекси у MongoDB.

Користувач у базі даних описується наступними полями:

а) Id

б) Email

в) Токен доступу

г) Пароль

д) Місце проживання (необов’язково)

Зображення у базі даних описується наступними полями:

– Id

– Назва

– URL

Модель у базі даних описується наступними полями:

– URL моделі

– ID зображення

Місце проживання у базі даних описується наступними полями:

– Країна

– Область (необов’язково)

– Місто (необов’язково)

Загальна схема бази даних зображена на рисунку 3.4.

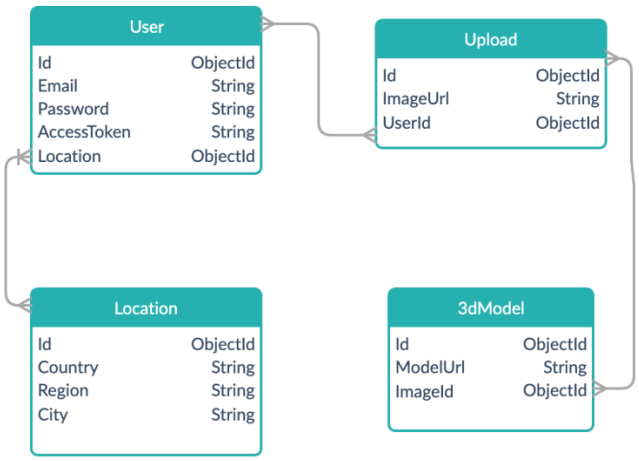

Рисунок 3.4 – ER-діаграма бази даних системи

Схема бази даних демонструє, що кожен користувач має зв’язок з певною кількістю зображень, кожне зображення містить назву та силку до самого зображення, а кожна модель містить інформацію про зображення до котрого вона відноситься та силку на саму модель.

Для збільшення швидкості роботи бази даних, було вирішено додати необхідні індекси на запити, що використовуються найчастіше. Таким чином, був створений індекс на вибірку з таблиці зображень та моделей.

На рисунку 3.5 зображено індекс MongoDB.

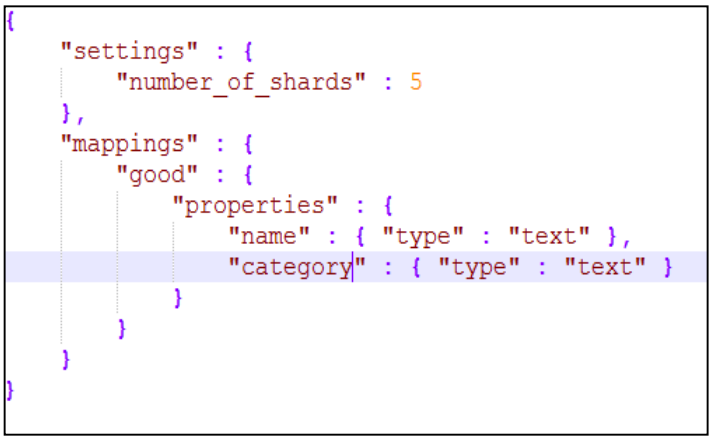

Рисунок 3.5 – Індекс MongoDB

Індекс досить простий і складається з ключового слова і найменування категорії до якої це ключове слово відноситься.

3.5 Вибір алгоритму машинного навчання

Під машинним навчанням в термінах постановки задачі дослідження будемо розуміти процес створення та навчання системи, за допомогою якої в моделі буде реалізовано функцію визначення розташування вказівника мишки. Розглянемо існуючі алгоритми.

3.5.1 Дерево прийняття рішень

Перевагами цього методу вважається той факт, що він стабільно функціонує у разі втрат даних та не вимагає від даних нормалізації. Недоліком є те, що він чутливий до шуму і змін в даних – незначна зміна даних може суттєво змінити дерево. Існує можливість повторного навчання, що вирішується за допомогою обрізки і деревовиних композицій. Ще одним недоліком є той факт, що цей алгоритм відноситься до «жадібних», тому що в кожній вершині вибирається предикат, який найкраще локалізовано.

3.5.2 Метод найближчих сусідів

Перевагою цього алгоритму є простота, тому що основним параметром для його оптимізації є лише кількість сусідів. Він є найбільш підходящим, коли інформативний показник відстані легше знайти між двома символами, ніж визначити відстань для послідовності символів. Також до недоліків слід віднести той факт, що він ресурсномістким, оскільки основні розрахунки проводяться саме під час оцінки, а не шляхом навчання [24].

3.5.3 Лінійна регресія

Лінійна регресія має багато переваг, одна з яких швидке навчання через те, що кількість параметрів досить мала – кількість параметрів дорівнює кількості ознак. Цей алгоритм можна використовувати для великих даних через оптимізацію стохастичного градієнтного спуску, який приймає об’єкти по черзі. Ще одна перевага полягає в тому, що цей метод легко інтерпретувати – чим більша вага, тим важливішим є ця ознака. Разом з тим алгоритм має певні обмеження: лінійна регресія не може досягти високої якості, наприклад, у випадкових лісах, які

вимагають обов’язкову попередню обробку даних. Збільшення масштабів, шум та викиди повинні бути усунені, що, у свою чергу, потребує додаткових ресурсів та знижує швидкодію методу.

3.5.4 Метод опорних векторів

У цьому методі функція завжди має єдиний мінімум, який є глобальним, тобто немає проблемі із застряганням у локальних мінімумах. Корисним наслідком цього алгоритму є виділення допоміжних об’єктів. Зазвичай він добре працює на невеликих і чистих даних. Недоліком методу є те, що опорними об'єктами можуть бути обрані викиди або шум, тому дані повинні бути очищені перед підготовкою класифікатора. Крім того, він не дуже добре працює з великими даними, тому що при великій кількості даних навчання може зайняти багато часу [25].

3.5.5 Випадковий ліс

Цей алгоритм відновлює складні залежності і не допускає перетренувань, оскільки додавання нових дерев до композиції тільки покращує якість. Це не вимагає попередньої обробки даних, тому що це логічний класифікатор. Одним з недоліків є те, що його тренування займає довгий час і будує зберігати далекоглядність протягом тривалого часу через глибину дерев. Крім того, випадковий ліс не дуже добре працює з розрідженими даними у зв'язку з тим, що дерева не пристосовані для цього [26].

3.5.6 Підвищення градієнта

Це машинний метод навчання застосовується для дослідження проблем регресу. Цей алгоритм відновлює складні залежності і не вимагає попередньої обробки даних. Одним з недоліків є те, що під час тренування моделі можуть бути

перепрограмовані; також він не достатньо добре працює з розрідженими даними [26-27].

3.5.7 Нейроні мережі

Нейронна мережа є комплексом лінійних класифікаторів. Перевагами її застосування є те, що градієнтний спуск використовується в нейронних мережах для оптимізації моделі, яка може ефективно працювати з великими даними. Також нейронні мережі можуть відновлювати складні залежності, оскільки вони є сховищем алгоритмів та можуть мати багато нейронів. Особливо ефективними є так звані глибокі нейронні мережі, які останнім часом майже витіснили всі інші методи машинного навчання під час розв’язанні складних завдань, таких як розпізнавання зображень, тексту тощо. Що стосується недоліків цього підходу, то одним з головних недоліків є те, що під час навчання модель зростає, збільшується складність її оптимізації і збільшується час на отримання результатів розрахунків. Також цей метод має схильність до перепрограмування моделі, для запобігання якого застосовуються різні методи регуляризації [27].

3.6 Аналіз бібліотек для машинного навчання

На сьогоднішній день у реалізації нейронних мереж важливу роль відіграють, як правило, спеціалізовані пакети, що надають вже повністю реалізовані базові структури та алгоритми для зручної розробки. Найбільш широко використовуються такі популярні бібліотеки, як Theano, Tensorflow та Torch та ін. Дані пакети реалізують програмний інтерфейс для реалізації низькорівневих розрахунків, тому прийняття рішень, в першу чергу, повинно базуватися на суперечностях про продуктивність та зручність розробки [28].

Варто відзначити, що Torch реалізує інтерфейс для мови Lua, не дуже популярної серед програмістів, що істотно знижує зручність і швидкість розробки. Згідно дослідженням Theano and Torch показують приблизно однакову продуктивність на одну й тому ж наборі даних.

Окрім описаної вище незручності використання бібліотеки Torch, існує ще одна, також пов'язана зі зручністю розробки. Інші два згаданих вище інструменти здатні працювати з бібліотекою більш високого рівня Keras [29], що надає загальні, для Tensorflow [29] і Theano, заздалегідь визначені алгоритми та структури. За результатами порівняння та аналізу документації бібліотек було прийнято рішення використовувати для програмної реалізації згорткової нейронної мережі зв'язок бібліотек Tensorflow і Keras.

Бібліотека Keras [30] - програмний продукт, написаний мовою python, що надає високорівневе API для роботи зі штучними нейронними мережами. Даний продукт створювався з метою запобіганню труднощів в розробці нейронних мереж, пов'язаних із синтаксичними особливостями конкретних пакетів. В січні 2016 року даний інструмент було придбано компанією Google і адаптовано як основне високорівневе надлаштування над бібліотекою Tensorflow.

Бібліотека Theano була написана в першу чергу як розширення мови Python, яка дозволяє ефективно розрахувати математичні вирази, що містять багатомірні масиви. В бібліотеці реалізується базовий набір інструментів для побудови нейронних мереж. Процес створення моделі та визначення її параметрів вимагає написання об'ємного коду, що включає реалізацію класу моделі, самостійного визначення її параметрів, реалізації методів, що визначають функцію помилок, правило обрахування градієнтів, способи зміни ваги нейронів.

Бібліотека Caffe реалізована на мові програмування C++. Топологія нейромереж, вихідні дані та спосіб навчання задаються за допомогою конфігураційних протоколів (застосовується технологія і протокол передачі даних) у прототиповому форматі. Побудова структури мережі виконується з простотою та зручністю. Бібліотека Caffe підтримується досить великою спільнотою розробників

та користувачів і на сьогоднішній день є самою розповсюдженою бібліотекою глибинних навчальних програм широкого кола застосування.

Огляд методів вирішення задач контролю за напрямком зору людини обґрунтував переваги нейромережевого методу. Принцип роботи нейронів штучної мережі дозволяє виявити його активність, що визначається його параметрами – вагами і його функцією активації. Ефективність навчання управлінням складними об’єктами з використанням управління за напрямком зору на початковому етапі досліджень можна досягти саме методом навчання з учителем. Перейдемо до програмної реалізації моделі системи.

3.7 Створення UI/UX системи

При створенні веб сайту, UI/UX архітектура є одним з найважливіших критеріїв поряд з іншими факторами розробки системи. Палітра кольорів, шрифти, меню, іконки і навіть прості кнопки створюють позитивне чи негативне відношення клієнтів до вашої компанії чи продукції.

У даній дипломній роботі, було вирішено використовувати рекомендації до оформлення від компанії Google. Збірна назва таких концепцій – дизайн матеріалів або Material Design цього типу дизайну: у його рамках об'єкти «відповідають» користувачеві тінями і реакцією на дотики. На перший погляд новий концепт занадто «примітивний». Він плоский і багатошаровий. Шари позначені тінями. Ідея така ж, як у фізичної книги або зошиту, і полягає в «гортанні сторінок». Розробники отримали можливість задавати визначення кута окремих елементів інтерфейсу і промальовувати тіні в режимі реального часу.

«Матеріальний» дизайн характеризують жирні яскраві шрифти і адаптивна «палітра», що дозволяє додаткам пристосовувати свої кольори до кольорової гами

відображуваного їм контенту. Згідно перерахованим вище принципам і був сформований графічний інтерфейс веб клієнта Transphere.

Екран автентифікації можна побачити на рисунку 3.6.

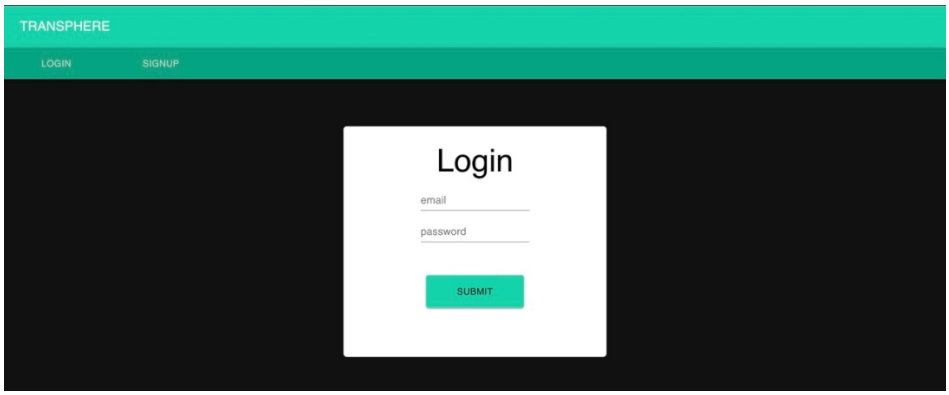

Рисунок 3.6 – Сторінка автентифікації користувачів

Для успішної автентифікації, користувачу потрібно ввести email-адресу та пароль, що були вказані при реєстрації. Після успішної автентифікації, користувачу відкриється головна сторінка веб-додатку. Якщо відвідувач веб-додатку ще не має акаунта у системі, то йому спершу доведеться пройти процес реєстрації.

Екран реєстрації можна побачити на рисунку 3.7.

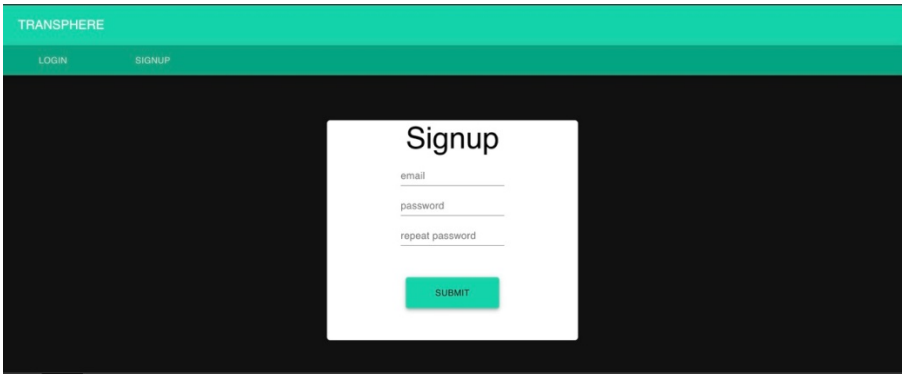

Рисунок 3.7 – Сторінка реєстрації користувача

На головній сторінці користувача розташовується інформація про завантажені зображення та кнопки для завантаження згенерованих 3D моделей. У

наступних версіях системи передбачається інтегрувати перегляд 3D моделей у веб-браузері користувача.

На рисунку 3.8 зображена посадкова сторінка веб-додатку. На цій сторінці користувач має змогу познайомитися з функціями сервісу та перейти до створення 3D моделей.

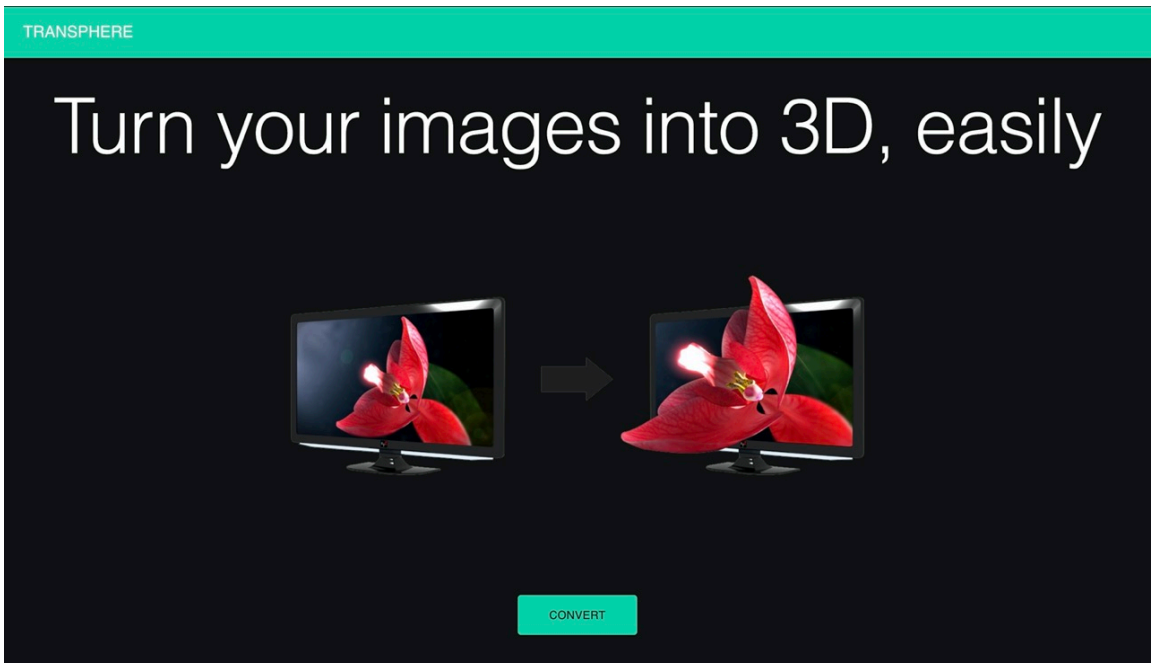

Рисунок 3.8 – Головна сторінка системи

Після натискання на кнопку “Convert” користувач перейде до сторінки завантаження зображень. На рисунку 3.9 зображена сторінка завантаження зображень для подальшої обробки та генерації 3D моделі.

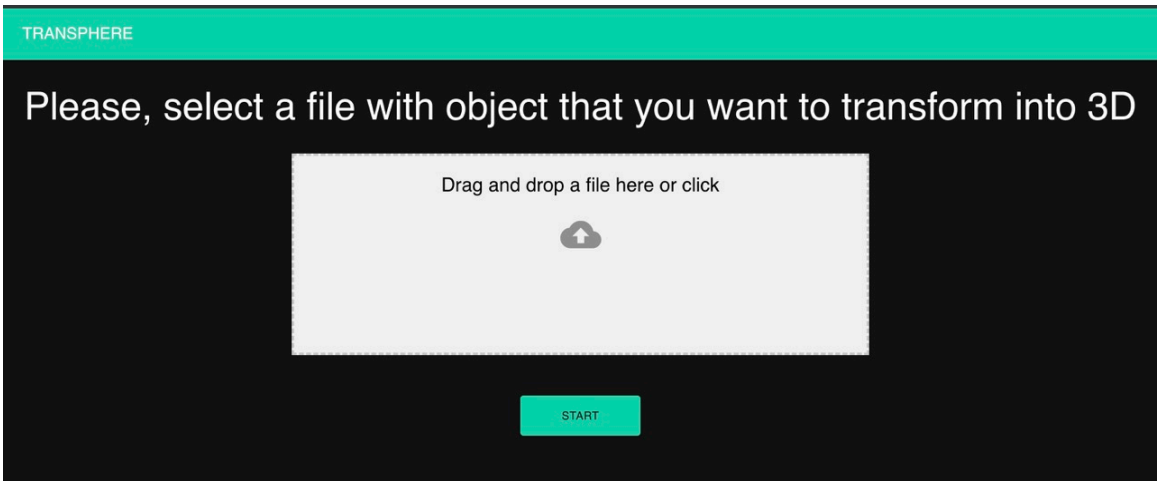

Рисунок 3.9 – Сторінка завантаження зображень

3.8 Висновки до розділу 3

В розділі «Архітектура та проектування програмного забезпечення» спершу було прийнято рішення щодо архітектури системи. Були визначені основні елементи системи та взаємозв’язок між ними.

Після цього було проведено UML-моделювання системи. Були складені діаграми варіантів використання, класів і компонентів. Ці діаграми не відображають системи цілком, а лише дають загальне уявлення про її роботу. З діаграми варіантів використання видно, що система призначена не лише для простих користувачів, а й для дослідників, в першу чергу, маркетологів.

Було розглянуто схему бази даних та індексу MongoDB, які демонструють основні сутності в системі.

Наприкінці було обрано мови для розробки системи, та середовища розробки. Вибір зумовлювався перевагами технологій, сучасними тенденціями і зручністю в розробці. Також було розроблено інтерфейс основних сторінок роботи веб-додатку.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір інструментарію

Для реалізації системи створення 3D моделей з зображень було обрано такі мови програмування, як JavaScript ES6, Python. Платформа Node.js була обрана у якості http-сервера для сервера системи, через свою гарну продуктивність. Був використаний фреймворк Express.js, який надає необхідний функціонал та велику документацію. Для написання веб-клієнту використовується зв’язка React плюс Redux, яка вже зарекомендувала себе, як гарне рішення для розподілу логіки представлення та бізнес логіки на клієнтській частині.

Для розробки обрано операційні систему Mac OS High Sierra, так як Mac OS добре підходить для тестування сервера і розробки web-клієнта.

Що стосується середовища розробки для створення серверу та веб-клієнту, то воно повинно мати наступні характеристики:

– підтримувати мову програмування Python (усі функції, дебагер);

– підтримувати мову програмування JavaScript, стандарту ES6;

– підтримувати інтелектуальне доповнення коду (Code Completion);

– підтримувати інтелектуальну інспекцію коду;

– повинно включати інструменти для розробки та використання реляційних баз даних.

Серед програмних засобів розробки, які мають характеристики, необхідні для розробки сервера та веб-клієнта, доступними є наступні:

– Jetbrains Webstorm v2016.3;

– Jetbrains PyCharm v2017.1.

Для порівняння цих програмних продуктів скористаємося методом варіантних мереж. Оцінимо за п'ятибальною шкалою наступні характеристики програмних продуктів:

– швидкість розробки (5);

– вимоги до обчислювальних ресурсів (3);

– надані можливості (5);

– швидкість роботи розробленого програмного забезпечення (3);

– легкість інсталяції розробленого програмного забезпечення (4);

– зручність експлуатації (3).

В круглих дужках важливість даної характеристики.

Рішення поставленого завдання вибору програмного забезпечення методом варіантних мереж показано в таблиці 4.1.

Таблиця 4.1 – Рішення завдання вибору програмного забезпечення

Середовище розробки	Характеристика						Сума
	а (5)	б (3)	в (5)	г (3)	д (3)	е (4)	
Jetbrains Webstorm v2016.3;	5	4	5	5	4	5	109
Jetbrains PyCharm v2017.1;	5	5	4	5	4	3	99

У результаті проведеного дослідження отримали, що найкращою середою для розробки сервера та веб-клієнта у даному випадку є Jetbrains Webstorm v2016.3.

4.2 Реалізація системи

У першу чергу було реалізовано серверну частину веб-додатку. У якості фреймворка було обрано Express.js. Структуру каталогів проекту сервера можна побачити на рисунку 4.1.

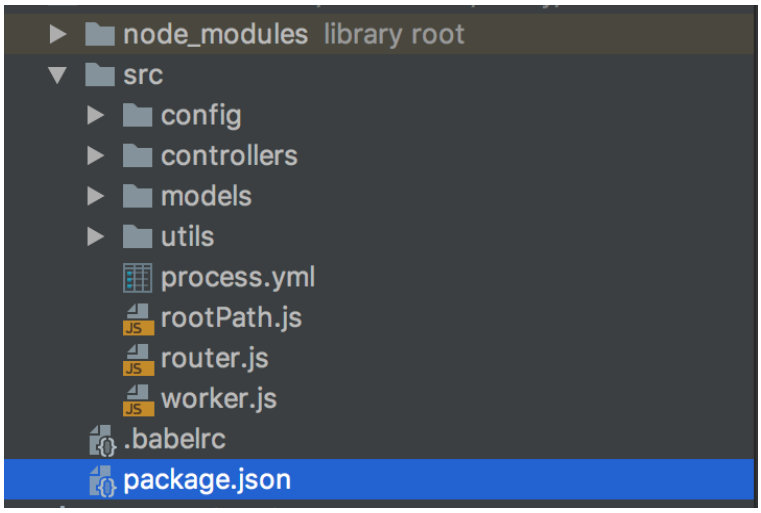

Рисунок 4.1 – Структура каталогів проекту

Як видно з рисунку, у проекті використовується MVC архітектура. У ролі View виступає веб-клієнт тощо. У директорії конфігурацій лежать файли з налаштуваннями бази даних, процесу автентифікації тощо. Також, у серверній частині проекту використовується техніка кластеризації для збільшення швидкодії серверу та відмовостійкісті. При кластеризації сервер системи запускається у декількох процесах. Отже, це приводить до роботи певної кількості однаково працюючих версій системи. Важливим елементом при кластерізації є модуль для балансування навантаження. Цей модуль займається визначенням навантаження на кожному із процесів нашої системи та вибором на який процес направити користувача системи. Для балансування навантаження використовується алгоритм Round Robin.

На ріисунку 4.2 зображена схема роботи системи з балансуванням навантаження.

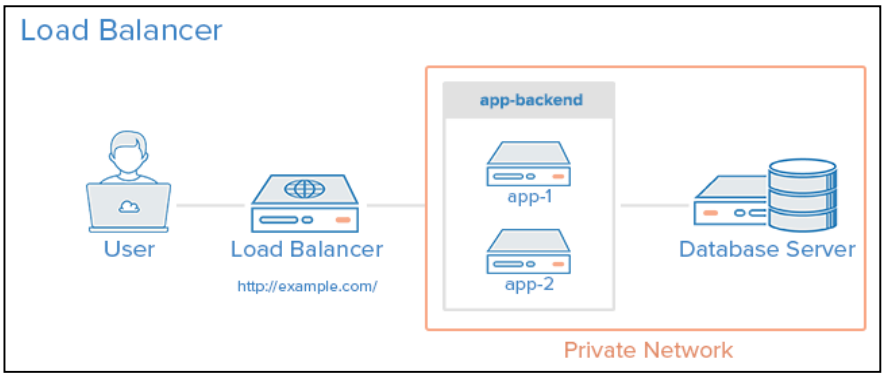

Рисунок 4.2 – Схема роботи системи з балансуванням навантаження

Для балансування навантажень та моніторингу модулів системи було вирішено використовувати open-source модуль PM2. PM2 – це процес менеджер для Node.js проектів, який надає можливість швидко налаштувати розподіл навантажень, проводити моніторинг системи. Налаштування для PM2 можна зберігати у YML файлі. Модуль працює через термінал системи.

Нижче наведено приклад файлу налаштувань для PM2 модуля для запуску 4 процесів системи серверу, виконування процесу балансування, та перезапуску серверу при змінах у коді:

```
apps:
  - script : worker.js
instances: 4
exec_mode: cluster
watch  : true
env    :
      NODE_ENV: development
env_production:
      NODE_ENV: production
```

Для шару збереження даних використовується база даних MongoDB і об’єктно-реляційне відображення Mongoose. Mongoose надає можливість створювати схеми документів, що будуть зберігатися у базі даних, додавати валідацію полів, додавання індексів тощо.

Нижче представлений код для ініціації з’єднання з базою даних, та обробку помилок та перезапуск з’єднання, якщо база даних не була запущена у системі:

```
module.exports = () => {
mongoose.connect(settings.dbPath);
mongoose.Promise = Bluebird;
constdb = mongoose.connection;

db.on('error', (error) => {
console.error(`Mongoose Error ${error}`);
constmongod = exec('mongod');
mongod.stdout.once('data', (data) => {
mongoose.connect(settings.dbPath);
mongod.unref();
        });
    });
db.once('open', () => {
console.log('MongoDB connected');
    });

    return db;
};
```

Для автентифікації відвідувачів системи було використано open-source бібліотеку Passport.js. Passport.js – це Javascript бібліотека для автентифікації користувачів, що використовує зручну модульну архітектуру, та дає можливість використовувати велику кількість різноманітних стратегій автентифікації та реєстрації користувачів.

У серверній частині проекту використовується автентифікація користувачів за email-адресою та паролем. Також, використовується JWT автентифікація. JWT – це стандарт токена доступу, що використовується для верифікації тверджень. JWT складається з трьох частин: заголовка, вмісту і підпису. JWT передається в заголовках HTTP, зазвичай у полі Authorization як Bearer-Token.

Нижче наведений фрагмент коду з додаванням стратегій до модулю Passport.js для проведення JWT автентифікації та автентифікації за email-адресою та паролем:

```
MODULE.EXPORTS = () => {
    PASSPORT.SERIALIZEUSER(USER.SERIALIZEUSER());
    PASSPORT.DESERIALIZEUSER(USER.DESERIALIZEUSER());
    PASSPORT.USE(USER.CREATESTRATEGY());
    PASSPORT.USE(NEW BEARERSTRATEGY(
        (TOKEN, DONE) => {
```

```
            USER.FIND({ACCESSTOKEN: TOKEN}, FUNCTION (ERR,USER){
                IF (ERR) {
                    RETURN DONE(ERR);
                }
                IF (!USER.LENGTH) {
                    RETURN DONE(NULL, FALSE);
                }
                RETURN DONE(NULL, USER[0], {SCOPE: 'ALL'});
            });
        }
    ));
};
```

Система серверу працює з клієнтом через HTTP endpoints, які уявляють собою поєднання частини URL і типу запиту (GET, POST) з певним обробником.

У сукупності роути утворюються API серверу. Далі наведено список частини роутів системи:

– POST api/v1/upload – роут для завантаження зображення;

– GET api/v1/uploads – роут для надання завантажених зображень користувача;

– GET api/v1/integrations/:id/ – роут для надання детальної інформації (включно 3D модель) користувача за id;

– GET api/v1/uploads/date/{dateFrom}/{dateTo} – роут для надання завантажень користувача за проміжком дат;

– POST api/v1/login – роут авторизації;

– POST api/v1/signup – роут для реєстрації нових користувачів;

Останнім було розроблено веб-клієнт. Веб-клієнт уявляє собою single-page application в якому данні управляються за допомогою бібліотеки redux, а усе зв’язане з представленням розроблено у вигляді React.js компонентів. Однією з переваг React.js є наявність великої кількості готових компонентів, які дуже легко додати до власного додатку.

Наприклад, на рисунку 4.3 зображено компонент App відповідальний за роутинг веб-клієнта, та за перевірку даних о користувачі у локальному сховищі браузера.

```
class App extends Component {

    componentDidMount() {
        this.props.checkLocalStorage();
    }

    render() {
        return (
            <MuiThemeProvider>
                <div>
                    <HeaderContainer/>
                    <Switch>
                        <Route exact path="/" component={ContactListContainer}/>
                        <Route exact path="/login" component={LoginContainer}/>
                        <Route exact path="/dashboard/consumer" component={DashboardConsumer}/>
                        <Route exact path="/dashboard/ads" component={DashboardAds}/>
                    </Switch>
                    <RootModalContainer/>
                </div>
            </MuiThemeProvider>
        );
    }
}
```

Рисунок 4.3 – Вихідний код Login ReactJS компонента

Цей компонент використовує інший компонент з відкритим вихідним кодом MuiThemeProvider, що надає можливість використовувати компоненти с бібліотеки React Material для стилізації елементів системи.

4.3 Висновки до розділу 4

В розділі «Опис програмної реалізації» спершу були описані вимоги для середовища розробки системи, операційної системи, необхідного функціоналу.

Потім розглянуто серверну частину системи. За допомогою рисунка було наведено внутрішню структуру сервера програмної системи. Дані пояснення про вміст директорій і призначення окремих модулів. Показане рішення для балансування навантаження на сервер та код запуску модуля для балансування. Також, приведені фрагменти коду для підключення до бази даних та фрагмент коду модуля автентифікації. Показано API розробленого серверу і призначення кожного з пунктів API. Також описано веб-клієнт, технології, які використовувалися для його розробки, їх доцільність та переваги від їх використання.

ВИСНОВКИ

У результаті виконання роботи було проведено докладне дослідження предметної області, обґрунтовано доцільність розробки системи, описані принципи її роботи та виявлено основні функції і реалізована програмна система, яка реалізує створення 3D-моделей с 2D-зображень.

Був проведений аналіз існуючих аналогів, який включав аналіз існуючих сервісів створення 3D-моделей з 2D-зображень, а також аналіз методів класифікації об’єктів та трансформування зображень у 3D-файли.

Для реалізації усього необхідного функціоналу була проаналізована предметна область, виявлені взаємовідносини між основними об’єктами системи. Був проведений аналіз існуючих рішень та виділено переваги та недоліки кожного. При проведенні аналізу предметної області було прийнято рішення створити програмну систему, яка зможе забезпечити створення 3D-моделей з 2D-зображень.

Слід зазначити, що серверна сторона реалізована на платформі Django і написана за допомогою мови програмування Python. Серверна сторона містить шари доступу до даних, логіки додатка, моделей. Веб-клієнт реалізований за допомогою HTML5, CSS і Javascript фреймворку React.js.

Система має середні вимоги до апаратного та програмного забезпечення, забезпечує досить високу швидкість роботи, безпечна у використанні, має гнучку і багатофункціональну архітектуру, що дозволяє її легко модифікувати і додавати нові функції. Має доступний і зручний інтерфейс, взаємодія з яким не викликає труднощів, є доступним для користувачів з різним рівнем комп’ютерної грамотності.

Необхідно відзначити, що програмний продукт навіть після реалізації усіх поставлених на атестаційну роботу задач, для комерційного запуску має дороблюватися.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бейзер Б., Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. – Питер, 2004. – 320 с.

2. Кормен Т., Алгоритмы: построение и анализ – Питер, 2009. – 960 с.

3. Nielsen, M. Neural networks and deep learning – 2014. – 600 с.

4. Maturana, D. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition – 2015 – 90 с.

5. Разлацкий, С.А. Применение метода Хафа для детектирования объектов в трехмерной сцене – 2015 – 580 с.

6. Dosovitskiy A., Springenberg J. T., Learning to generate chairs, tables and cars with convolutional networks – 2017. – 98 с.

7. Eigen D., Puhrsch C., Depth map prediction from a single image using a multi-scale deep network – 2014. – 200 с.

8. V. Lempitsky and Y. Boykov Global optimization for shape fitting // Conference on Computer Vision and Pattern Recognition (CVPR).2007.С.4-5.

9. Configurable Cell Segmentation Solution Using Hough Circles Transform and Watershed Algorithm Proceedings of the International Conference on Advanced Optoelectronics and Lasers, CAOL 2019 | conference-paper DOI: 10.1109/CAOL46282.2019.9019493EID: 2-s2.0-85082014845Часть ISBN: 21601534 21601518

10. Tombari F., Object recognition in 3D scenes with occlusions and clutter by Hough voting // Fourth Pacific-Rim Symposium on Image and Video Technology. 2010.С.2-4.

11. Cardeli L. Typeful programming. Tutorial – Surveys, 1991.- 670 p.

12. Graham I. Object-Oriented: Principles & Practice. Tutorial – Williams, 2004. – 880 p.

13. Klements P., Norsrop L. Software Product Lines: Practices and Patterns. Tutorial – Addison-Wesley, 2002. – 608 p.

14. Larman K. Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development. Tutorial – Williams, 2006. – 736 p.

15. Ахо А., Хопкрофт В., Ульман Д. Структуры данных и алгоритмы. Пособие – Вильямс, 2000. – 384 с.

16. Берг М., Чеонг О., Кревельд М. Вычислительная геометрия. Алгоритмы и приложения. Пособие – ДМК-Пресс, 2016. – 438 с.

17. Гамма Э., Хелм Р., Джонсон Р. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Пособие – Питер, 2007. – 366 с.

18. Гранд М. Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных при помощи UML. Пособие – Новое издание, 2004. – 560 с.

19. Макконнелл С. Совершенный код. Пособие – Питер, 2005. – 896 с.

20. Марк Д. Swift. Разработка приложений в среде Xcode для iPhone и iPad. Пособие – Вильямс, 2016. – 816 с.

21. Мартин Р. Принципы OOD. SOLID. Пособие – Вильямс, 2010. – 690 с.

22. Мухортов В., Рылов В. Объектно-ориентированное программирование, анализ и дизайн. Пособие – Новосибирск, 2010. – 211 с.

23. Нойбург М. Objective-C Основы. Пособие – Вильямс, 2014. – 416 с.

24. Орлов С. Технології розробки програмного забезпечення. Пособие – Питер, 2008. – 464 с.

25. Signal restoration by means of blind deconvolution based on optimization

with wavelet transformation 2016 3rd International Scientific-Practical Conference Problems of Infocommunications Science and Technology, PIC S and T 2016 – Proceedings 2017 | conference-paper DOI: 10.1109/INFOCOMMST.2016.7905324

26. Волегов Д.Б., Юрин Д.В. Предварительное грубое совмещение изображений по найденным на них прямым линиям для построения мозаик, сверхразрешения и восстановления трехмерных сцен – 2008. – 475 с.

27. Gilles Burel, Hugues Henocq. 3D invariants and their application to object recognition // Signal Processing, Vol. 45.1995.C.1-22.

28. Osmani A., Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide – O'Reilly, 2012. – 254 c.

29. Маккинни У. Python и анализ данных – ДМК Пресс, 2015. – 600 с.

30. V. Krylov , G. Shcherbakova, R. Pisarenko, N. Bilous Signal restoration by means of blind deconvolution based on optimization with wavelet transformation // 3rd International Scientific-Practical Conference Problems of Infocommunications Science and Technology.2016.C.3