МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерні науки
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів побудови туристичних маршрутів з урахуванням вподобань та обмежень користувача
(тема)

Виконав:
Студент 2 курсу, групи ІПЗм-19-1
Пітюкова М.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Шевченко О.Л.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерні науки
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова програма
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав.кафедри _____
(підпис)
« 26 » березня 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Пітюкової Марії Олегівни
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів побудови туристичних маршрутів з урахуванням вподобань та обмежень користувача»
затверджена наказом університету від 26.03.2021 № 385 Ст
2. Термін подання студентом роботи до екзаменаційної комісії «11» травня 2021 р.
3. Вихідні дані до роботи *методи побудови туристичних маршрутів, розробка мобільного додатку для формування пішохідних туристичних маршрутів*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної галузі, аналіз існуючих систем, постановка задачі, математична модель, аналіз існуючих алгоритмів, характеристика обраного алгоритму, опис прийнятих програмних рішень, опис роботи програмної системи, висновоки, додатки*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, ілюстрацій (слайдів) *мета роботи, постановка задачі, методи і алгоритми, опис отриманих результатів, інтерфейс програмної системи, демонстраційні матеріали*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Шевченко О.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	26.03 – 02.04.2021	*Виконано*
2	Огляд існуючих методів та алгоритмів	03.04 – 10.04.2021	*Виконано*
3	Характеристика обраного методу	11.04 – 18.04.2021	*Виконано*
4	Підготовка пояснювальної записки	19.04 – 04.05.2021	*Виконано*
5	Спецчастина	19.04 – 04.05.2021	*Виконано*
6	Підготовка презентації та доповіді	19.04 – 04.05.2021	*Виконано*
7	Нормоконтроль, рецензування	04.05 – 07.05.2021	*Виконано*
8	Занесення матеріалів в електронний архів	07.05.2021	*Виконано*
9	Попередній захист	10.05.2021	*Виконано*
10	Допуск до захисту у зав. кафедри	11.05.2021	*Виконано*

Дата видачі завдання 26 03 2021р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шевченко О.Л.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 76 с., 12 рис., 1 табл., 24 джер.

АЛГОРИТМИ, БАЗА ДАНИХ, КРИТЕРІЇ, МЕТОДИ, МОБІЛЬНИЙ ДОДАТОК, ОПТИМІЗАЦІЙНА ЗАДАЧА, ПОДОРОЖ, ТУРИСТИЧНИЙ МАРШРУТ, ASP.NET, C#, JAVA, MICROSOFT SQL SERVER, VISUAL STUDIO 2019.

Об’єктом дослідження є методи побудови індивідуальних пішохідних туристичних маршрутів із зазначеними критеріями користувачами.

Метою роботи є аналізування існуючих алгоритмів для побудови маршрутів, подальшого його вибору згідно критеріїв користувача та реалізація системи.

Методи розробки: багатокритеріальний аналіз, класичний генетичний алгоритм, технології такі, як Android SDK, Java, C#, ASP.NET.

В результаті роботи було досліджено різні методи побудови маршрутів, проведено аналіз та моделювання предметної області, розроблено схему бази даних та програмно реалізовано основні функції мобільного додатку.

ALGORITHM, DATABASE, CRITERIAS, METHODS, MOBILE APPLICATION, OPTIMIZATION PROBLEM, MODELING, ROYALTY, TOURIST ROUTE, ASP.NET, C#, JAVA, MICROSOFT SQL SERVER, VISUAL STUDIO 2017.

The object of research is the methods of construction of individual hiking tourist routes with the specified criteria by users.

The aim of the work is to analyze the existing algorithms for the construction of routes, its subsequent selection according to user criteria and system implementation.

Development methods: multicriteria analysis, classical genetic algorithm, technologies such as Android SDK, Java, C #, ASP.NET.

As a result of work various methods of construction of routes were investigated, the analysis and modeling of subject area is carried out, the scheme of a database is developed and the main functions of the mobile application are software implemented.

Я, Пітюкова Марія Олегівна, студент гр. ІПЗм-19-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів побудови туристичних маршрутів з урахуванням вподобань та обмежень користувача», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Яке б століття не було – людина завжди потребує яскравих емоцій, нових почуттів, того самого адреналіну, завдяки якому має бути подальший прогрес в її житті. Тому і кличуть людину нові пригоди, бажання пізнати незвідане. І саме подорож дає можливість побачити світ та отримати цей необхідний новий досвід. Це і навколишнє середовище – цікаві місця, галереї, музеї; і інші люди, з їх культурою та способом життя; і природні пам'ятки культури – озера, ліси, луги, гори тощо.

За останнє десятиліття кількість громадян України, що подорожують, зросла майже у 9 разів. Із року в рік зростає кількість подорожуючих за кордон та по своїй країні. На зміну кількості подорожуючих впливає такі фактори як, девальвація гривні, погодні умови, складний підготовчий процес тощо.

Деякі українці відкладають подорож, бо їм буває складно належним чином підготуватися до поїздки. Треба перегортати величезну кількість сайтів у пошуках цікавого у містах подорож, придбати квитки, знайти місце для ночівлі, скласти список потрібних речей, зібрати їх, приготувати їжу в дорогу і тому подібне. Тому люди можуть витратити багато часу аби віднайти найказкові, найцікавіші місця, сформувати відповідний маршрут з них так, щоб він був найбільш комфортний, швидким і оптимальним. А часу як завжди замало.

Тоді існує потреба у звертанні до певних сервісів заради необхідних послуг, які тим чи іншим шляхом допоможуть впоратися з турботами перед подорожжю. Але, на жаль, кожен з підручних варіантів націлений на якусь одну проблему: чи то речі, які необхідно взяти з собою, складання списку з ними; чи то просто інформаційні сайти, або блоги, зі збірками різноманітної інформації про визначні місця, які можна відвідати. Присутні в мережі Інтернету також сайти з картами, що містять у собі об'єкти, блоги, що розповідають про ці самі речі і радять, куди варто відправитися першою чергою; і сайти – створювачі маршрутів, які приймають

критерії, такі як точку відправлення і точку прибуття, та потім віддають побудований маршрут між цими точками.

Однак у більшості випадків доводиться самому знаходити необхідні ресурси, помічати бажані місця для відвідування, переходити на сервіси для формування маршрутів та витрачати багато часу для побудови оптимального напрямку. Тобто відсутні сервіси, що об’єднують одразу кілька дій між собою та мають можливість прокласти маршрут для певного міста з декількох визначних місць, та при цьому ж запропонувати користувачу обрати необхідні параметри – місто, дати приїзду та від’їзду та інтереси щодо виду місць.

Метою даної кваліфікаційної роботи є аналіз існуючих алгоритмів для побудови маршруту, подальшого його вибору згідно критеріїв користувача та також проектування і створення програмного продукту, який дозволив би автоматизувати даний процес створення напрямків для подорожей у районі міста. Створений додаток посприяє вивченню місцевості та цікавих, для користувача, місць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Комбінаторика – це розділ математики, присвячений вивченню розв'язанню задач вибору та розташування елементів деякої, зазвичай, скінченної множини відповідно до заданих правил [1].

Вперше питання, які вивчає комбінаторика, були підняті в працях математиків Стародавньої Греції, Китаю та Індії. Але з розвитком теорії графів в XIX – XX століттях інтерес до комбінаторного аналізу збільшився.

У комбінаторики є багато застосувань в інших областях математики, таких як теорію графів, як вже було сказано, програмування, криптографію і теорію ймовірності. Серед самих провідних математиків того часу можна виділити Блеза Паскаля, Якоба Бернуллі, Леонарда Ейлера і Пала Ердеша, внесок у розвиток яких виміряти не можливо.

Людина – натура різностороння. Тому існує безліч способів отримання кошт на життя. І тоді постає головне питання – на що направити отримане. Зрозуміло, що це можуть бути як витрати на необхідне – здоров’є, їжу та інше, так і на отримання певної моральної насолоди – незабутні враження, захоплюючі спогади, неповторні відчуття, чудові моменти. На те, про що можна згадати у будь-який момент свого життя, розповісти близьким людям та добрим знайомим. Та при цьому обов’язково відкрити величезні альбоми з фотографіями, що зберігають в собі кожну неперевершену хвилину тих цінних та чудових моментів. І тоді в допомогу стає ЇЇ величність – подорож. Саме після неї залишаються вічнопам’ятні спогади, що зігрівають серце навіть через деякий час та надихають вже на нові круті мандри. Тому все більше і більше людей починають подорожувати задля відчуття того присмаку незабутньої свободи та відкритих у всю очей на неймовірний світ навколо нас.

При чому, за останньою статистикою, якщо 10 років назад більша частина українців подорожувала виключно територією України, а менша частина вирушала

шукати пригод закордон, то зараз ситуація змінилася. На теперішній час, кількість українців, що не бояться перетнути кордон збільшилася майже у 3 рази та перевершила кількість людей, що залишаються подорожувати в рамках однієї рідної України.

Українці не бояться опинитися на чужій незнайомій землі та спілкуватися з іноземцями, бо прагнуть пізнати їх та їхню культуру, яка повністю чи частково відрізняється від української, бо це дозволяє розширити своє сприйняття світу.

Завдання комівояжера – це класична задача комбінаторики, основна суть якої полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу [2].

Задача побудови оптимального маршруту за наявними картографічними даними є окремим випадком задачі комівояжера. Вперше піднята в 1832 році у книзі «Комівояжер – як він повинен вести себе і що повинен робити для того, щоб доставляти товар і мати успіх у своїх справах – поради старого кур'єра». Дане завдання є актуальним і в повсякдення – розробляються нові методи розв'язання задачі, реалізуються програми, які дозволяють працювати з кількістю вузлів близьким до мільйона за припустимий час. Незгасний інтерес до цього завдання обумовлено численною кількістю застосувань її на практиці. Пошук оптимального шляху широко застосовується у всіх завданнях транспортної логістики, а на виробництві – у вигляді задач мінімізації часу переналагодження і при роботі діропробивні преси.

Проте, в більшості випадків розробка маршруту стикається з проблемою часткової або повної нестачі часу на дослідження місцевості, відсутністю попередньої організації дійства та відсутністю будь-якого універсального алгоритму для її вирішення. Тому завдання аналізу існуючих алгоритмів для побудови маршрутів та подальшого його вибору, що задовольняє індивідуальним конкретним вимогам людини, дуже актуальна, як з огляду на широке коло зацікавлених в її вирішенні осіб, так і за браком діючих програм, здатних вирішити цю задачу.

Дана кваліфікаційна робота присвячена аналізу вже існуючих алгоритмів для побудови маршрутів, подальшого його вибору згідно критеріїв користувача, а також визначенню вимог, проектуванню і створенню системи, яка дозволила б автоматизувати даний процес створення шляхів для подорожей у рамках міста.

Тобто рішенням даного програмного продукту повинна бути система, яка надає функціональність обрання найбільш відповідного алгоритму, отримання згенерованого індивідуального маршруту на кожен день подорожі згідно обраних користувачем критеріїв з можливістю його подальшого редагування, включаючи додавання, видалення та переміщення пунктів та можливістю ознайомлення з детальною інформацією про визначні місця і їхнього рейтингу, а також їх оцінка.

1.2 Виявлення проблем та актуалізація рішень

Такі широковідомі картографічні сервіси, як Google Maps, 2GIS, не пропонують користувачам можливість пошуку оптимально шляху. Мова йде про те, що при включенні декількох координат сервіс вибудовує маршрут так, що зберігається порядок, в якому дані були введені. Користувачі мають можливість обирати засоби пересування – на машині, пішки чи на наземному транспорті, але всі зміни впливають виключно на варіанти побудови маршруту між його фіксованими точками.

Проведений шляхом порівнянням десятків зарубіжних картографічних сервісів аналіз показує, що серед широко вживаних варіантів тільки у одного доступна функція побудови оптимально шляху, та далеко не завжди вона працює так як треба. Нижче представлені широко вживані сервіси з їх недоліками та короткою характеристикою.

В першу чергу можна виділити російський сервіс «MegaNavigator» [3] – для побудови автомобільних, велосипедних та пішохідних маршрутів. Також включає в себе конструктор карт, візуалізацію GPS треків. Сервіс, що являє собою сайт з

вбудованою картографічною службою «Яндекс.Карты» та полями для введення пунктів маршруту (див. рис. 1.1).

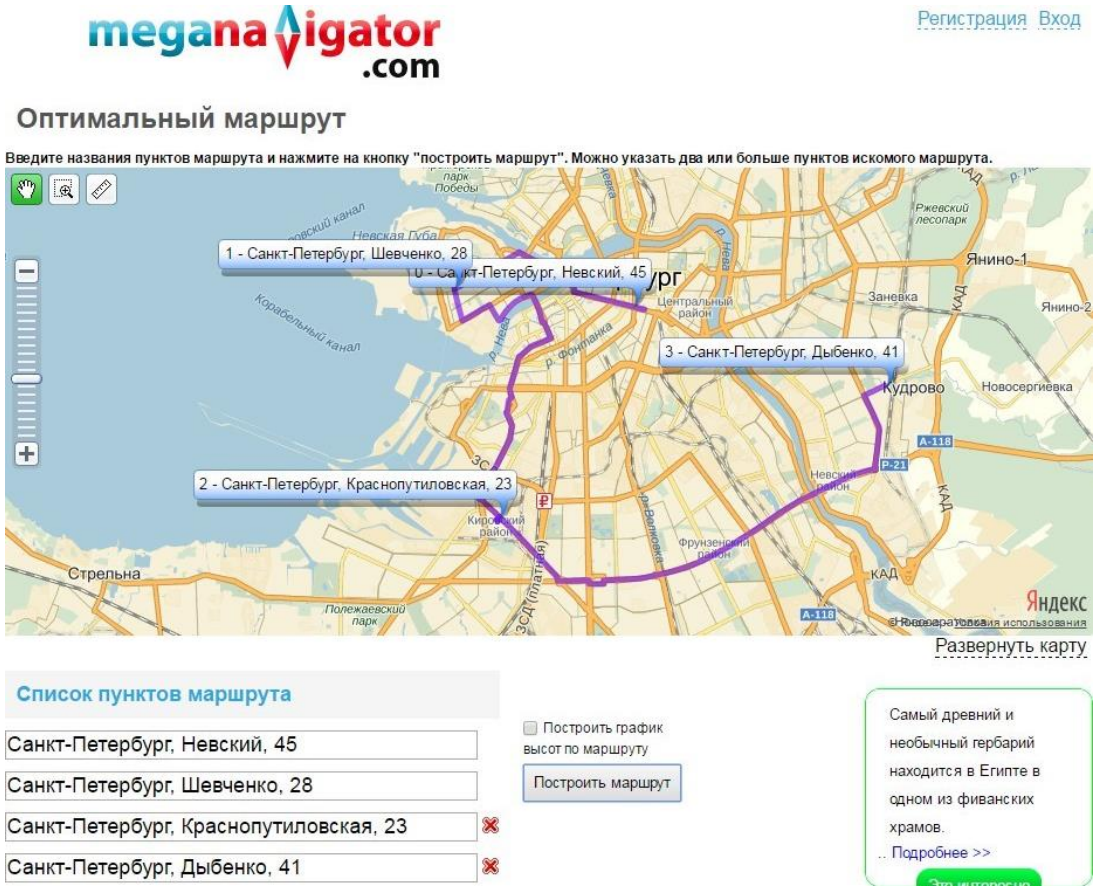

Рисунок 1.1 - Сервіс «MegaNavigator»

Тестування функціональної частини сервісу продемонструвало наявність суттєвих недоліків:

– на сайті відсутня можливість вибору засобу пересування; проте режим, що встановлений за замовчуванням, відповідає пересуванню на автотранспортному засобі;
– при введенні адреси в адресний рядок не передбачені підказки, що безперечно ускладнює роботу з сервісом;
– при побудові маршруту перший пункт обирається довільно із заданого списку адрес;
– немає можливості задавати адресу як першу за замовчуванням;

– при виборі пунктів на інтерактивній карті без введення адрес в текстові поля, маршрут не будується.

Наступний сервіс – Логіст [4], що має меншу кількістю недоліків, але всі вони важливі:

– при обранні засобу пересування можливі два варіанта: автомобілем – побудова маршруту по проїжджих дорогах, та вертольотом – прямий шлях між пунктами маршруту;
– сервіс розрахований для вирішення завдань логістики та побудови маршруту перевезень продукції між містами (див. рис. 1.2).

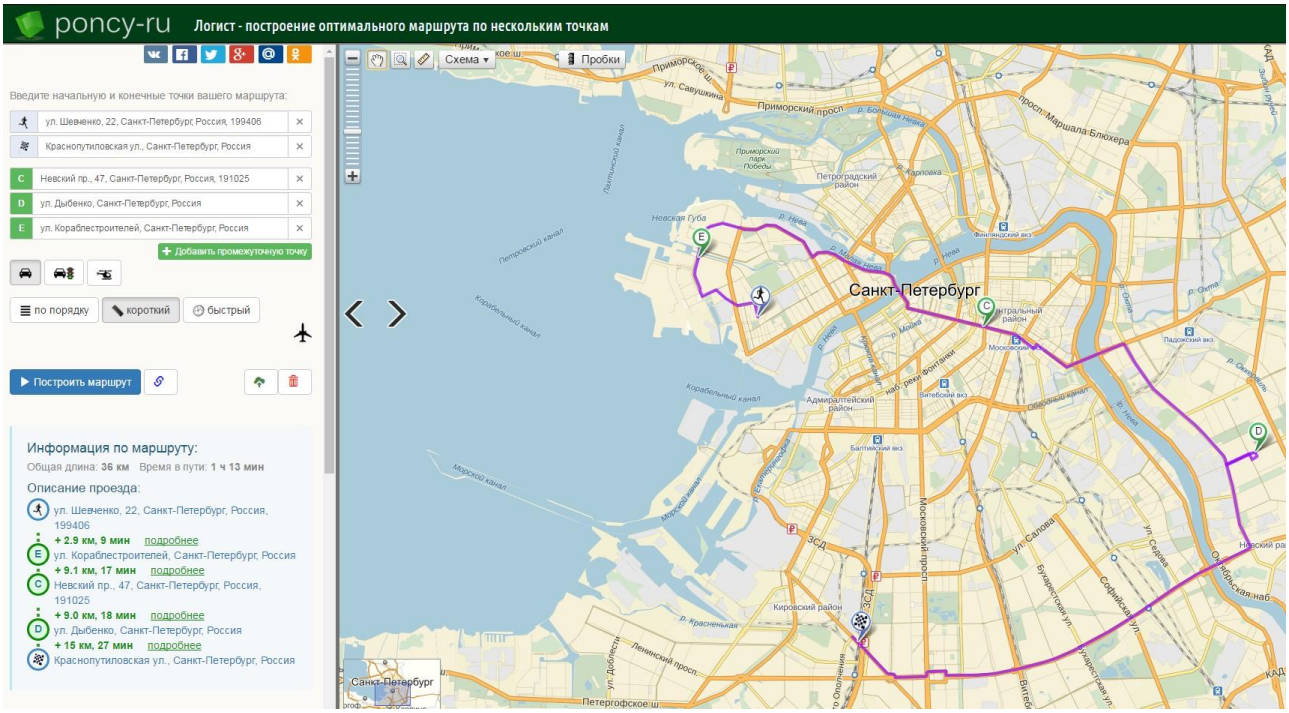

Рисунок 1.2 - Сервіс Логіст

Speedy Route [5] – це картографічний сервіс з функцією побудови оптимального маршруту (див рис. 1.3).

В розділі опису продукта сказано, що основні функції – це розрахунок найбільш кращого маршруту при відвідуванні декількох локацій з подальшим поверненням до початкової точки та повний опис шляху між усіма зупинками.

Більш орієнтований на водіїв-кур'єрів, продавців в дорозі або для тих, кому потрібно зробити кілька зупинок.

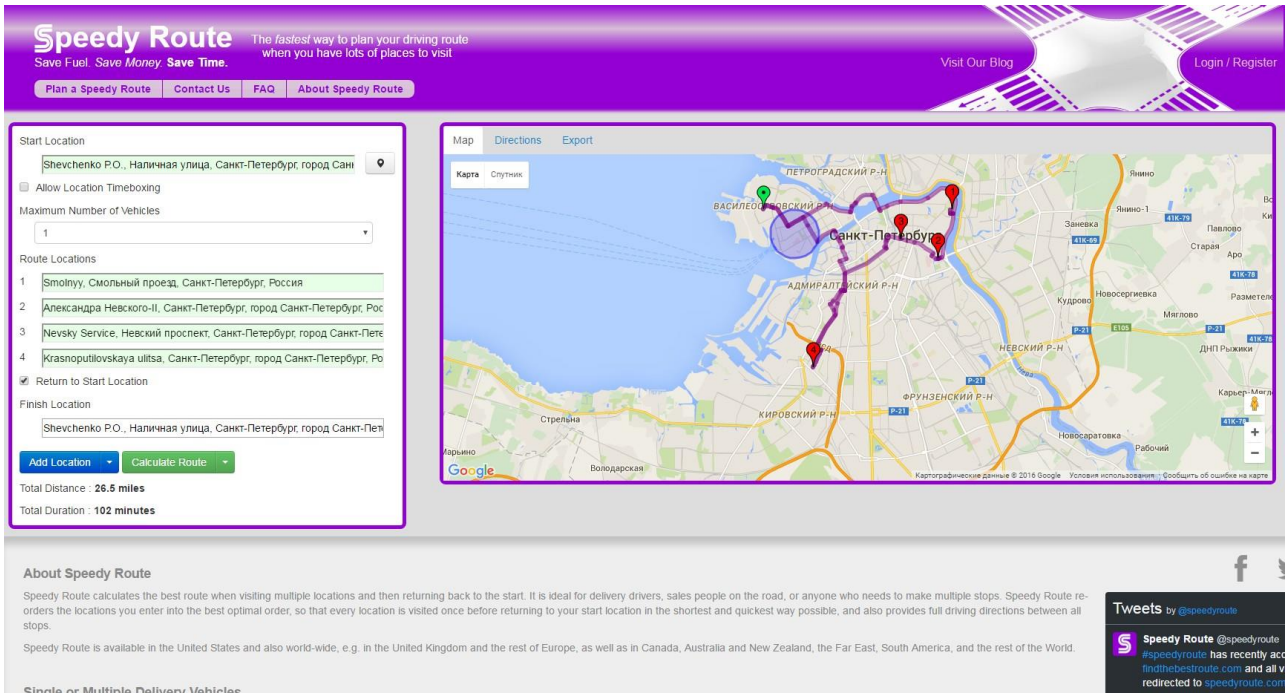

Рисунок 1.3 - Сервіс Speedy Route

Після тестування функціоналу були описані такі недоліки системи, як:

– при введенні даних виникають труднощі перекладу україномовних назв на англійську мову;
– мінімальна кількість точок для побудови маршруту даним сервісом – 5; маршрути, що складаються з 4 пунктів для відвідування, побудувати неможливо;
– на сайті користувачу надана безкоштовна версія карти для ознайомлення, а для доступу до повної версії потрібно оформити платну підписку.

Далі розглянемо мобільний додаток Gogobot [6]. Google назвав даний ресурс соціальною мережею для мандрівників (див. рис. 1.4).

За допомогою сервісу Gogobot зручно планувати поїздки, складати маршрути та розклади на кожен день. Вся інформація синхронізується між кількома пристроями користувача.

За допомогою Gogobot можна отримати доступ до коментарів інших користувачів про ресторани, визначні місця, готелі та розваги. А для того, щоб знайти людей зі схожими інтересами та дізнатися які місця їм подобаються можна використати функцію «Tribes».

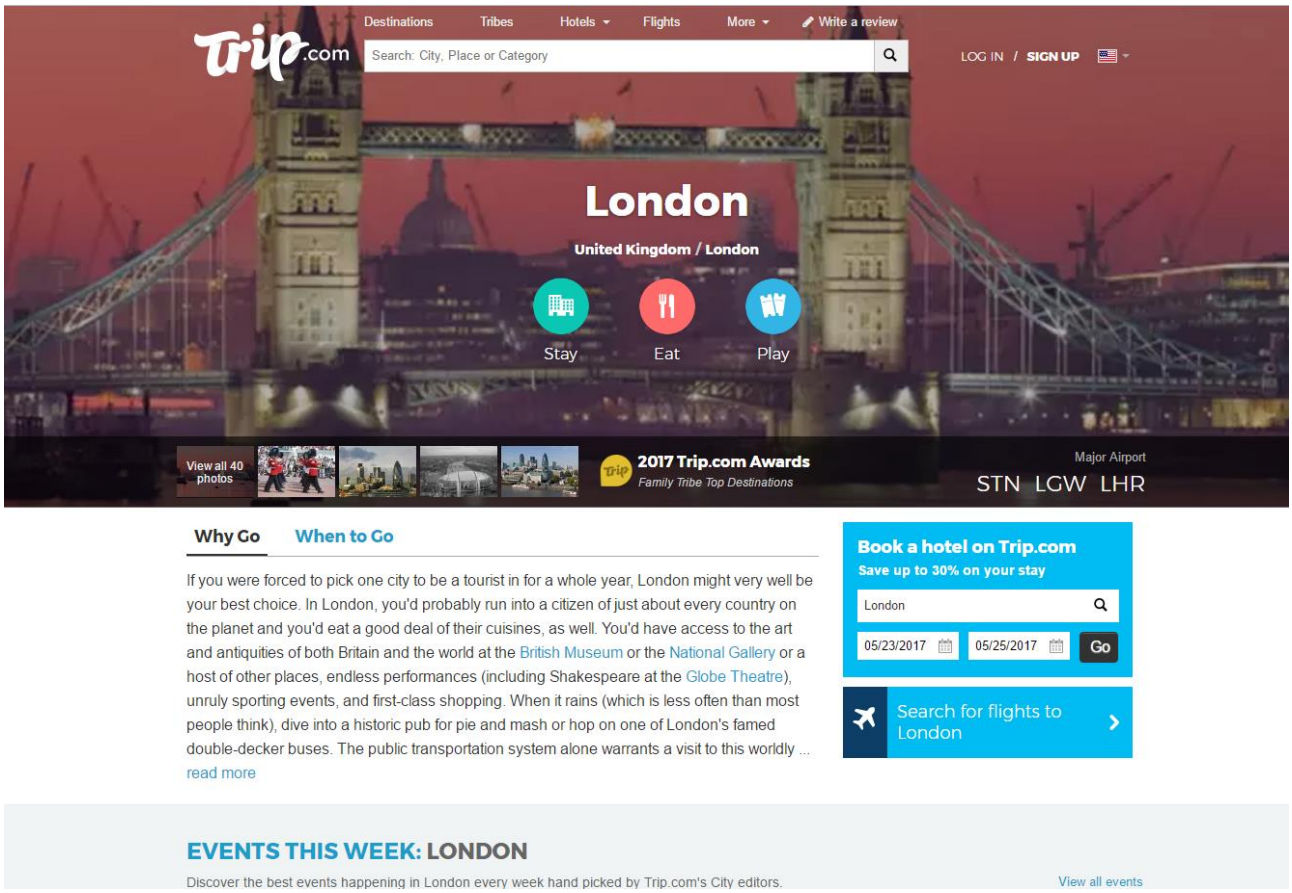

Рисунок 1.4 – Ресурс для туристів Gogobot

Також Gogobot дозволяє створити своєрідний «паспорт» подорожі та поділитися досвідом з усіма користувачами додатку, написати відгуки та опублікувати фотографії.

Gogobot надає тільки інформаційну допомогу, хоч і дуже детальну, цікаву та корисну, використовуючи такі дані, як інтереси, час, локальну погоду та місце знаходження користувача для того, щоб допомогти знаходити найкращі пам’ятки, події та інше.

Triposo Travel Guide [7] – це безкоштовний мобільний додаток, котрий може працювати без прямого підключення до мережі Інтернет та дозволяє

персоналізувати подорож підбираючи визначні місця, розваги, улюблені готелі, активності, ресторани та оформляти бронювання прямо в додатку (див. рис. 1.5).

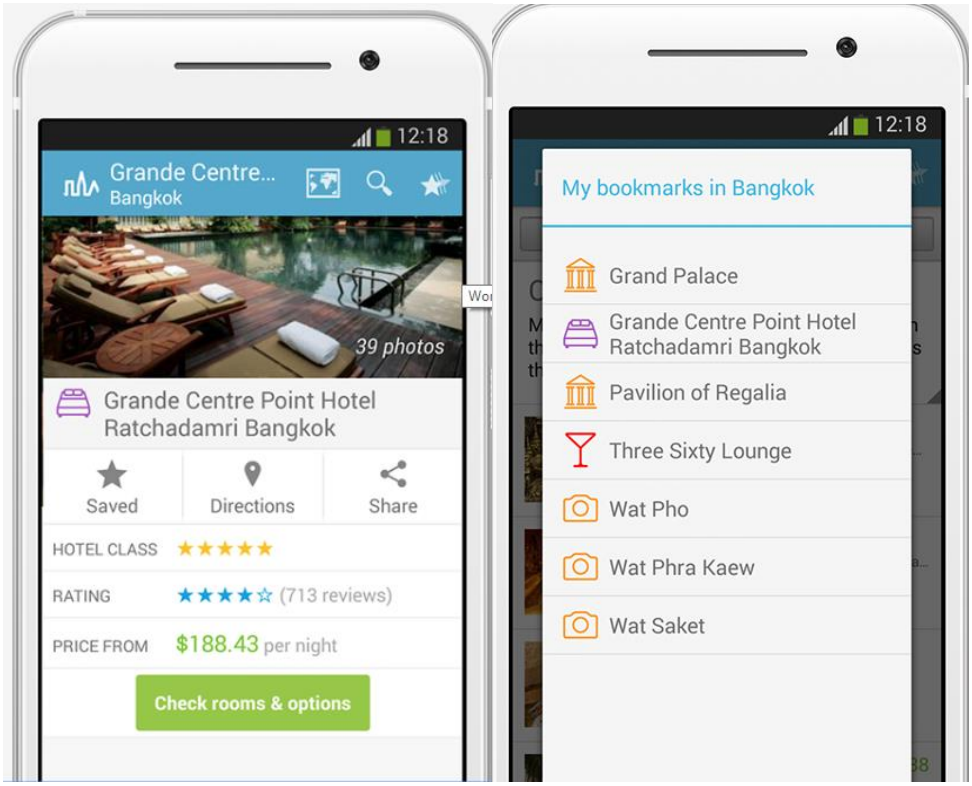

Рисунок 1.5 – Мобільний додаток Triposo Travel Guide

Додаток має доступ до даних відомих таких ресурсів, як World66, Wikivoyage, Wikipedia, Open Street Maps, TouristEye and Flickr, за допомогою котрих розбирає отримані дані та знаходить найбільш релевантну інформацію для мандрівників.

Triposo Travel Guide надає інформаційну послугу та дозволяє зберегти місця, що зацікавили користувача до обраного, проте додаток не надає можливості створити повноцінний маршрут розподілений по днях та у певному порядку.

Інший додаток The Field Trip [8] використовує інформацію про місця, що представляють інтерес для користувача, та, виходячи з місця знаходження користувача, відбирає корисну інформацію (див. рис. 1.6).

Користувач додатку має можливість підключити Bluetooth-аудіопристрій для озвучування інформації вголос, щоб не відволікати водія від дороги.

The Field Trip – це унікальне віконце до різноманітних та унікальних речей навколо. За допомогою даного додатку можливо вивчити все – від місцевої історії до найновіших та найкращих місць для покупок, харчування та розваг.

Користувач обирає напрями, які йому найбільш цікаві та інформація буде з'являтися на телефоні автоматично, як тільки користувач наближається до подібних місць.

Такі сайти, як Thrillist, Zagat і Sunset, додаток використовує для формування рекомендацій, а Songkick і Flavorpill для пошуку місцевої музики. Також додаток доступний на різних мовах та надає інформацію по всьому світу.

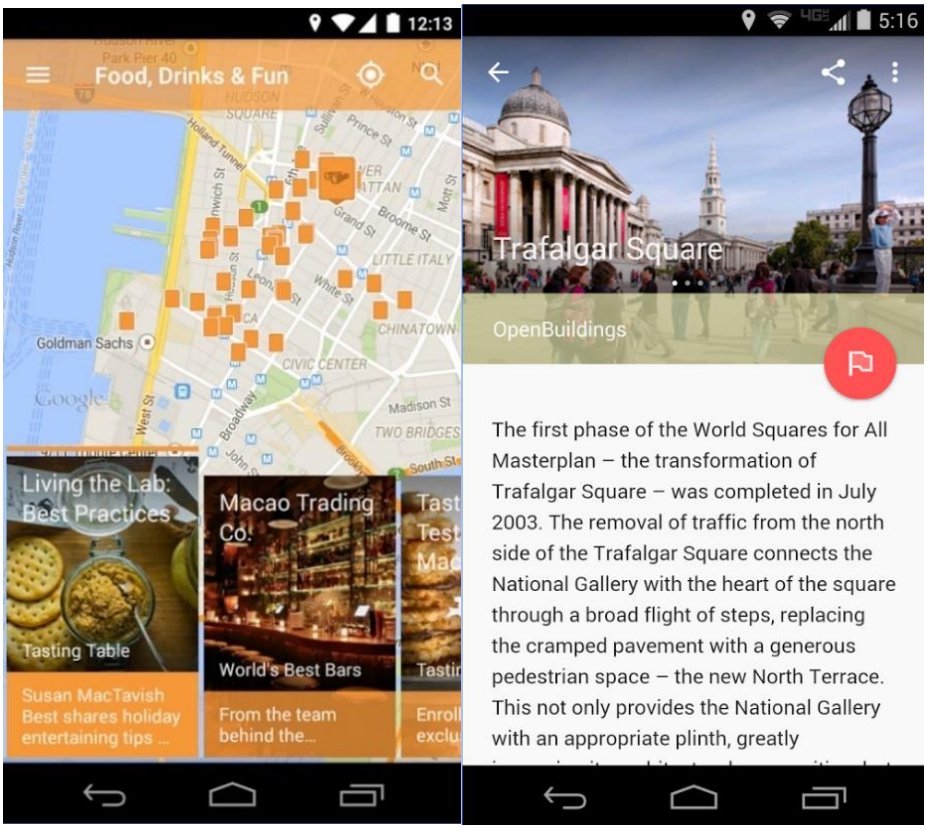

Рисунок 1.6 – Мобільний додаток The Field Trip

Також додаток може працювати у фоновому режимі та має функцію сповіщення при наближенні до місць, що можуть бути цікавими, з деталями про місце. Проте додаток не має можливості планувати та будувати маршрути заздалегідь, аби потім йому слідувати.

Підсумовуючи все вищезазначене можна сказати, що поки не існує сервісу, який би будував оптимальні пішохідні маршрути усередині міста за критеріями користувача та одночасно був зручним і зрозумілим для користувачів при взаємодії з ним.

1.3 Постановка задачі

Задача вибору маршруту, що задовольняє індивідуальним вимогам людини, дуже актуальна, як з огляду на широке коло зацікавлених в її вирішенні осіб, так і за браком діючих програм, здатних вирішити поставлену задачу.

Тому метою даної кваліфікаційної роботи магістра є аналіз наявних алгоритмів для побудови маршрутів, вибір відповідного способу вирішення задачі згідно критеріїв та обмежень користувача та визначення вимог до програмної системи, її проектування і створення.

Основною функціональністю проектованої системи повинна бути генерація, створення та редагування індивідуальних туристичних маршрутів, перегляд детальної інформації про визначні місця і їхнього рейтингу, а також їх оцінка.

Проте розробка будь-якої програмної системи, основною функцією якої є побудова маршруту – чи то пішохідного, чи то транспортного, починається з вибору головних компонентів структури та формування подальших вимог відносно кожного елементу.

Тому можна виділити такі основних етапи для реалізації задачі даної кваліфікаційної роботи:

- обрати критерії якості, що повністю описують вимоги до кожної точки маршруту;
- обрати критерії якості, що повністю описують вимоги до маршруту;
- обрати алгоритм для побудови маршрутів;

– обрати метод прийняття рішень для проведення оцінки;

– обрати відповідний маршрут із побудованих;

– виконати порівняння та оцінку запропонованих маршрутів;

– спроектувати та реалізувати програмну систему.

Необхідно спроектувати та реалізувати прототип програмної системи для побудови індивідуальних туристичних маршрутів.

Додаток має реалізовувати наступні функції:

– процес реєстрації та авторизації користувачів;

– генерація пішохідних маршрутів;

– зберігання та редагування маршрутів;

– перегляд детальної інформації про визначні місця, їх рейтингу.

Мова C# буде використана в якості основної мови розробки серверної частини програмного продукту, а JAVA в якості мови розробки мобільного додатку. Сервіс буде розроблений на платформі Microsoft .NET та за допомогою фреймворку Microsoft ASP.NET WEB API. Запити до сервісу повинні дотримуватися протоколу HTTPS та повинні бути оформлені згідно зі стандартом REST. Мобільний додаток буде розроблений на платформі Android.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ПРИЙНЯТТЯ РІШЕННЯ ПРИ ПОБУДОВІ ПІШОХІДНОГО МАРШРУТУ

2.1 Визначення критеріїв якості, що описують вимоги до кожної точки маршруту

На даній стадії необхідно розглянути можливі критерії для кожної зупинки на шляху користувача, виходячи з його побажань та обмежень.

До критеріїв висувають такі вимоги [9]:

- повнота (набір критеріїв має забезпечити адекватність оцінки досягнення мети рішення);
- дієвість, або операціональність (наявність у критерію чіткого, однозначного формулювання);
- можливість декомпозиції (можливість структуризації системи критеріїв);
- достатність (відсутність надмірності);
- мінімальність (набір критеріїв повинен бути мінімально необхідним для здійснення оцінки);
- вимірність (кожен критерій повинен давати кількісну або якісну оцінку ступеня досягнення мети).

До характеристик місця відвідування (далі – точки) віднесемо назву та її географічні координати.

Виділимо критерії y_i:

- час на огляд (в хвилинах);
- тип пам’ятки (один з множини: релігійні, містобудування, природні, археологічні, історичні за періодами, архітектурні, культурна спадщина, монументальне мистецтво);
- доступність місця (один з множини: закрито чи відчинено, на ремонті);
- доступність в сезон (чи можна відвідати точку в задану пору року);
- можливість звукового супроводу екскурсій для людей з вадами зору;

- можливість читання шрифтом Брайля різних інформаційних табличок для людей з вадами зору;
- можливість пояснювання інформації мовою жестів для людей з вадами слуху;
- рейтинг місця.

За допомогою вищевказаних критеріїв характеризується кожна точка, що може бути включена до формування пішохідного маршруту. При задовільнені побажань та обмежень користувача, точка розглядається як потенційна вершина графу, яка входитиме до складу маршруту.

2.2 Визначення критеріїв якості, що описують вимоги до маршруту

Турист має обмеження в часі, тому одним з важливих чинників при побудові є обмеження за часом: це може бути загальний час, необхідний для здійснення усієї подорожі чи обмеження по часу початку або закінчення подорожі.

Не менш значиму роль грає цікавість маршруту. Тобто другий фактор – це зміст маршруту, а саме сукупність всіх, пам'яток та інших цікавих місць, які входять до складу маршруту, з урахуванням їх релевантності для конкретного туриста. Під релевантністю буде розумітися чисельне вираження відповідності до побажань користувача – функція корисності.

Третім важливим критерієм буде довжина маршруту. На скільки протяжну дистанцію готовий подолати турист при обході різних пам'яток.

Для людей з особливими потребами існують спеціально адаптовані до особливих потреб готелі, музеї, якісь будівлі, окремі зони, тощо. І враховуючи ці обмеження можна визначити місця з наявною інфраструктурою та побудувати маршрут, який буде містити доступні для такої категорії місця. І цей критерій повинен бути визначним при врахуванні обмежень користувача.

Таким чином, можна визначити обов’язкові та додаткові критерії, які можуть бути враховані при генерації маршруту x_i:

- максимальна кількість визначних місць, які можна включати в один день подорожі;
- місцевість, в якій здійснюється подорож (один з множини: рівнинна, горбкувата, легкопересічена, середньопересічена, відкрита);
- дата та час початку мандрівки;
- максимальна тривалість маршруту (в хвилинах);
- типи розваг, місць, які можуть бути цікавими для користувача (ресурсні і виснажливі, осмислені і ні, соціально корисні і соціально небезпечні);
- наявність та тип обмежень.

На підставі визначених критеріїв можна визначити місця для відвідування, а далі запропонувати алгоритм знаходження шляху між визначеними точками.

В якості алгоритму для побудови маршруту пропонується використовувати класичний генетичний алгоритм [10], в зв'язку з тим, що раніше проведені дослідження показали високу ефективність даного алгоритму та можливість використовувати його для пошуку найкращого шляху в реальних умовах, а критерієм вибору шляху вважатимемо оцінку часових витрат на проходження шляху з урахуванням заздалегідь введених обмежень, тобто до безлічі можливих місця відвідування (вершин) потрапляють лише ті, які задовільнять вимогам користувача.

2.3 Опис методу прийняття рішень для проведення оцінки маршруту

Прийняття рішення – це комплексний та невизначеним у часі динамічний процес, котрий виникає коли необхідно обрати ідеальний у певному сенсі варіант

серед множини альтернативних варіантів для досягнення заданого або бажаного результату [11].

Процес прийняття рішення у загальному випадку полягає в оцінці припустимих альтернатив та виборі кращої з них за певними заданими критеріями. При цьому припускається, що реалізація будь-якого з варіантів рішень передбачає настання певних наслідків, аналіз та оцінка яких повністю характеризує обраний варіант. Для оцінювання альтернатив та можливих наслідків традиційно використовуються складні аналітичні розрахунки, знання фахівців-експертів, засоби сучасних інформаційних технологій.

З постановки задачі до даної програми системи можна зробити висновок, що в поставленому завданні принцип оптимальності задається множиною критеріальних функцій, тобто маємо багатокритеріальну задачу, котру необхідно звести до скалярної шляхом введення деякого універсального критерію [12].

Існує два методи рішення, що базуються на схемі:

– всі критерії повинні бути приведені до порівнянного безрозмірного вигляду;
– критерії «згортають» в одну цільову функцію, чи іншими словами в узагальнений критерій, враховуючи їх відносну важливість за допомогою вагових коефіцієнтів.

В результаті чого багатокритеріальна задача зводиться до звичайної задачі оптимізації за одним критерієм.

В даній роботі розглянеться більш детально такий вид згортки, як мультиплікативна згортка критеріїв [13].

Мультиплікативною згорткою є добуток критеріїв, при якому перед множенням критеріїв необхідно звести їх до рівня тим більшого, чим більше важливість критерія.

Маємо m альтернатив та n критеріїв. Побудуємо матрицю рішень $n \times m$. Значення для альтернатив мають j індекс та заповнюються в одному стовпчику, а значення для критерію мають i індекс та заносяться в один рядок.

Мультиплікативна згортка з нормалізуючими факторами описується законом, що наведено нижче:

$$Z = max \prod_{i=1}^{m} a_i x_{ij}$$

де a_i – нормуючі множники (див. табл. 1);

x_{ij} – значення критерію для кожної альтернативи в матриці рішень $n \times m$.

Таблиця 1 – Матриця прийняття рішень

	Альтернатива 1	…	Альтернатива $\boldsymbol{m}$
Критерій 1	x_{11}	…	x_{1m}
…	…	…	…
Критерій $\boldsymbol{n}$	x_{1n}	…	x_{nm}

Даний вид згортки базується на аксіомі: «низька характеристика принаймні одного із критеріїв тягне за собою низьке значення функції корисності».

Вагові коефіцієнти, або вага критеріїв – це добірне місце в задачі критеріального упорядкування альтернатив [13]. На основі інтуїтивного ідеї відносної важливості критеріїв доволі часто встановлюється вага. Проте, дослідження показують, що експерт не може прямолінійно надати правильну чисельну вагу критеріям.

Нехай маємо n критеріїв. Коефіцієнт вагою m балів – найважливіша альтернатива, наступна за важливістю – $m-1$ бал, останній – 1 бал. Отримані оцінки ділять на $(1 + 2 + \ldots + m)$ так, що сума всієї ваги дорівнює 1.

Мультиплікативна згортка з нормуючими та ваговими коефіцієнтами описана нижче:

$$Z = max \prod_{i=1}^{m} w_i a_i x_{ij} \quad (1)$$

де w_i – вагові коефіцієнти,

a_i – нормуючі коефіцієнти,

x_{ij} – значення критерію для кожної альтернативи в матриці рішень $n \times m$.

Вагові коефіцієнти w_i вираховуються згідно формули:

$$w_i = \frac{k}{\sum_{j=1}^{m} m}$$

де k – задається експертом для кожної альтернативи окремо.

Використана мультиплікативна згортка необхідна при вирішенні багатокритеріальних задач. Проте як і все, має свої переваги та недоліки.

До переваг можна віднести:

– не потрібно приводити власні критерії;

– завжди визначається одне прийнятне рішення.

До недоліків можна віднести такі:

– труднощі у визначенні вагових коефіцієнтів;

– перемножування різних величин;

– взаємна компенсація значень власних критеріїв.

Дані характеристики необхідно враховувати при роботі з мультиплікативною згорткою.

3 ДОСЛІДЖЕННЯ МЕТОДІВ ПОБУДОВИ МАРШРУТУ

3.1 Аналіз математичного апарату

Розглядається задача побудови маршрутів для відвідування безлічі туристичних точок пішим способом. Дана задача – це завдання пошуку найкоротшого шляху між декількома точками. Найбільш популярною інтерпретацією є подання до вигляду графа, вершини якого представляють точки, а сума, вага ребер, – відстань між точками. Дане завдання є класичною задачею в теорії графів і характеризується високою значимістю, на увазі масовості практичного застосування.

Вихідною інформацією будуть координати локацій міста з географічною інформацією про них.

На основі наявної інформації необхідно створити маршрут прямування. Для цього слід правильно визначити фактори, що впливають на вибір конкретного маршруту бажанням користувача.

У самій загальній постановці завдання комівояжера полягає в пошуку самого вигідного маршруту, що проходить через зазначені міста хоча б по одному разу з наступним поверненням в початкову точку.

Великий вплив на якість знаходження шляху надає спосіб представлення ландшафту. В основному для цього використовується представлення карти в вигляді матриці прохідності. Вибір форми осередку грає велику роль при реалізації алгоритму - впливає на довжину отриманого шляху.

Математично вся розглянута місцевість може бути представлена у вигляді спрямованого графа (див. рис.3.1):

$$G \coloneqq (n, m), i = 1, n;$$

де n – це безліч вершин або вузлів, m – це безліч ребер графа.

Вершини n являють собою перехрестя, перетину доріг, ребра m – об'єднують

їх ділянки доріг. Кожній грані графа призначається своя вартість, відповідна «вартості» переміщення по межі, задана при постановці завдання.

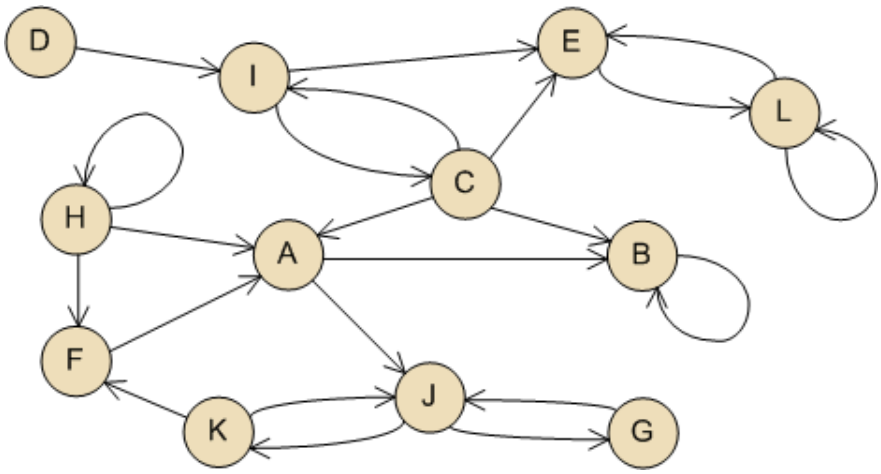

Рисунок 3.1 – Приклад спрямованого графу

Таким чином, для вирішення завдання слід використовувати алгоритми на графах, що дозволяють знайти найкращий шлях через набір вершин.

Також, дана матриця відстаней між вузлами. Відстані між вузлами можна знайти за формулою обчислення відстані між двома точками:

$$S(g_i, g_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Рішенням завдання комівояжера вважається мінімізація (оптимізація) цільової функції, де сумарна відстань повинна бути мінімальною:

$$D(v) = \sum_{i,j \in P} S(g_i, g_j) \rightarrow min$$

Слід враховувати деякі обмеження, які допоможуть уникнути деяких логічних помилок:

– у виконанні завдання комівояжера не розглядаються однакові індекси вершин;

– вершини в графі повинні бути пов'язані;

– кожну вершину пройти можна тільки один раз (Гамільтонів шлях).

Враховуючи доцільність мінімізації часу, який потрібен для проходження маршруту, виразимо час, необхідний для подолання відстані між точками (g_i, g_j):

$$T(g_i, g_j) = \frac{S(g_i, g_j)}{4{,}5}$$

де 4,5 – середнє значення швидкості пішохода [14].

З урахуванням часу на огляд пам’ятки оптимізація цільової функції може біти представлена наступним чином:

$$D(v) = \sum_{i,j \in P} T(g_i, g_j) + \sum_{i \in P} T(g_i) \rightarrow min$$

Можлива ситуація недосягнення обмеження по часу при побудові маршруту, до складу якого входять усі можливі місця відвідування, які задовольняють критеріям, обраним користувачем. Тобто користувач має можливість обрати невідповідні значення можливого часу, який витрачається на маршрут та кількість місць для відвідування. Для досягнення значення обмеження за часом, можна скоротити маршрут, тобто зменшити кількість точок відвідування. При зменшенні кількості точок враховуємо рейтинг кожної з них та виключаємо з маршруту місця з найнижчим рейтингом.

Але, враховуючи змінну кількість вершин графа, пропонуємо наступний критерій оптимізації цільової функції:

$$D(v) = \max_{P} \min_{T} \left(\sum_{i,j \in P} T(g_i, g_j) + \sum_{i \in P} T(g_i) \right)$$

Таким чином, отримаємо перелік маршрутів, які задовольняють обмеженням користувача, але, враховуючи значення кожного з критеріїв оцінки маршруту, для ранжування отриманих маршрутів використаємо мультиплікативну згортку (див. формулу 1), де в якості множини альтернатив виступає множина отриманих маршрутів.

3.2 Визначення рейтингу місця

Детальна інформація про місця дістається з зовнішнього серверу, а потім зберігається у локальній базі даних. Інформація містить в собі такі дані, як адресу, телефон, місце знаходження у координатах, години роботи, фотографії, рівень ціни від 0 до 4 (точна сума грошей для кожного рівня варіюється від регіону до регіону), тип місця (кафе, парк і т.д.), відгуки та загальний рейтинг від 1 до 10.

Програмна система, що розробляється, повинна мати свою власну систему рейтингу. Користувач матиме змогу порекомендувати час, період, пору року для проведення і виставити загальну оцінку.

Має враховуватися оцінка з зовнішнього провайдеру даних та з розроблюваної програмної системи при виведенні оцінки користувача.

Проте на основі відгуків, при виведенні загальної оцінки, мають враховуватися наступні правила:

– кількість відгуків не впливає на рейтинг;

– місце з меншою кількістю позитивних відгуків не має бути вище у рейтингу місця з більшою їх кількістю.

Для визначення рейтингу було взято за основу теорему Байєса [15]. Ідея теореми у тому, щоб нормалізувати рейтинги так, щоб підвищити рейтинг у тих, у кого менша загальна кількість голосів. Такий метод використовує всесвітньо відома система рекомендації фільмів IMDB.

Базова формула виглядає наступним чином:

$$B(r) = \frac{W(a) * a + W(r) * r}{W(a) + W(r)}$$

де $B(r)$ – байєсовський рейтинг;

$W(a)$ – довільне число, яке повинне бути більшим, ніж очікувана кількість оцінок для елементу;

a – середня оцінка для всіх елементів;

$W(r)$ – кількість оцінок для даного елементу;

r – середня оцінка для одного елементу.

Наприклад, три користувача виставили оцінку 10 балів, а середня оцінка для всіх елементів становить 6,5. Тоді:

$$B(r) = \frac{100 * 6{,}5 + 3 * 10}{100 + 3} = \frac{680}{103} = 6{,}6$$

Тобто загальна оцінка для даного елементу становитиме 6,6 балів.

Оскільки зовнішній провайдер даних забезпечує інформацією про відгуки разом з оцінками користувачів, то до загальної кількості оцінок користувачів розроблюваної програмної системи додаються оцінки з провайдеру даних.

3.3 Аналіз алгоритмів для побудови маршрутів

Відомий базовий алгоритм прокладки оптимальних маршрутів заснований на методі динамічного планування Форда-Беллмана на зважених графах у 1956 – 1958 роках.

Безліч алгоритмів, запропонованих в наступні роки, таких як алгоритми Дейкстри, Калаба, A* та інші, в основному є варіаціями базового алгоритму для

приватних постановок, завдяки чому досягається більш висока обчислювальна ефективність даних алгоритмів у порівнянні з базовим алгоритмом.

Використавши необхідні критерії відбору – насиченість, час, довжина – та відповідний алгоритм необхідно прокласти оптимальний маршрут від заданої стартової точки до заданої кінцевої точки.

Сучасні алгоритми дозволяють отримувати якісні результати, але вимагають точного підстроювання численних керуючих параметрів і мають надмірно високою трудомісткістю при великій розмірності задачі – до 1000 вершин і більше. Кожен алгоритм можна розбити на такі етапи:

- задається $V = \{v_0, v_1, \ldots, v_n\}$ – безліч всіх вершин;
- v_0 – це вершина, в якій побудовані маршрути повинні починатися; v_n – це вершина, в якій побудовані маршрути закінчуються за вибором користувача;
- $V' = V \setminus \{v_0\}$ – це безліч з n цільових вершин для відвідування;
- задається симетрична або несиметрична матриця вартостей переїздів між вершинами;
- потрібно побудувати m маршрутів, які починаються в v_0 і закінчуються в v_n, яка обрана користувачем.

Алгоритми, які можуть застосовуватися в розгляді даного завдання:

а) конструктивні алгоритми – виконують поступову побудову рішення, відстежуючи зростання його вартості, але не мають фази подальшого поліпшення.

б) двофазні, або кластерні алгоритми – завдання розбивається на дві частини:

1) угруповання вершин для кожного майбутнього маршруту, тобто кластеризацію;
2) рішення задачі комівояжера для кожної отриманої групи:
   - спочатку кластеризація, потім пошук рішення комівояжера;

– спочатку рішення комівояжера, а потім поділ на кілька маршрутов. Задача комівояжера вирішується для всіх вершин вихідного безлічі.

в) покращуючі алгоритми – спочатку ведеться пошук деякого рішення, а потім робляться спроби обміну вершин (ребер) всередині кожного маршруту або між маршрутами.

Поряд з алгоритмами, що дозволяють безпосередньо визначити шлях [16], слід враховувати різноманітні алгоритми подальшої обробки отриманого маршруту. Вони дозволяють розбити вихідну задачу на кілька підзадач:

– автоматичний розрахунок оптимального маршруту між заданими точками початку і кінця;
– побудова опису маршруту (передача інформації від системи до користувача);
– розпізнавання опису маршруту (передача інформації від користувача до системи);
– відстеження переміщення по заданому маршруту;
– обчислення релевантності кожного окремого об'єкта;
– визначення корисності маршруту – якесь чисельне вираження його змісту, засноване на сукупності релевантних входять до нього об'єктів;
– знаходження відповідного маршруту. Додаткову складність у задачу вносить ту обставину, що невідомо, скільки об'єктів повинно входити в такий маршрут, тобто в принципі їх може виявитися від 0 до N, де N – кількість об'єктів на карті.

Для формування підсумкового рішення використовується генетичний алгоритм, який маніпулює цілими маршрутами, оскільки проведені раніше дослідження показали ефективність даного алгоритму та можливість використовувати його для пошуку шляху в реальних умовах, а критеріями вибору маршруту буде вважатися оцінка часових витрат на проходження шляху з урахуванням заздалегідь введених обмежень.

3.4 Опис генетичного алгоритму

Генетичний алгоритм – це евристичний алгоритм прямого пошуку (метод оптимізації на основі порівняння значень функцій в пробних точках), який використовується для вирішення завдань оптимізації та моделювання шляхом випадкового підбору, комбінування і варіації шуканих параметрів з використанням механізмів, аналогічних природному відбору в природі [17]. Є різновидом еволюційних обчислень, за допомогою яких вирішуються оптимізаційні задачі з використанням методів природної еволюції, таких як успадкування, мутації, відбір і кросинговер. Особливістю генетичного алгоритму є акцент на використання генетичних операторів схрещування і мутації, які дозволяють виробляти еволюцію особин аналогічно.

Генетичні алгоритми належать до методів оптимізації, в основу яких лягли біологічні процеси, що протікають в природі. Чарльз Дарвін у своїй еволюційної теорії ввів визначення природного відбору, згідно з якою особи, більш присолені до умов навколишнього середовища, мають більше шансів на виживання і продовження роду, і навпаки – непристосовані осипи піддаються виборчому знищення.

Початкове покоління формується першим компонентом алгоритму, в подальшому, якщо чисельність особин в популяції виявляється нижче заданої межі, перший компонент використовується для заповнення відсутніх особин. Чисельність особин в популяції є параметром алгоритму і може бути налаштована при необхідності.

Основою відбору є мутації генів і їх комбінації, що формуються при розмноженні і передаються потомству. В ході природного відбору виживають екземпляри з найбільшою функцією пристосованості. Це чисельна характеристика, яка може змінюватися в залежності від умов конкретного завдання. Пристосовані особини схрещуються (кросовер) і дають потомство. Випадкові мутації також можуть впливати на розвиток популяції.

На рисунку 3.2 представлена загальна схема генетичного алгоритму:

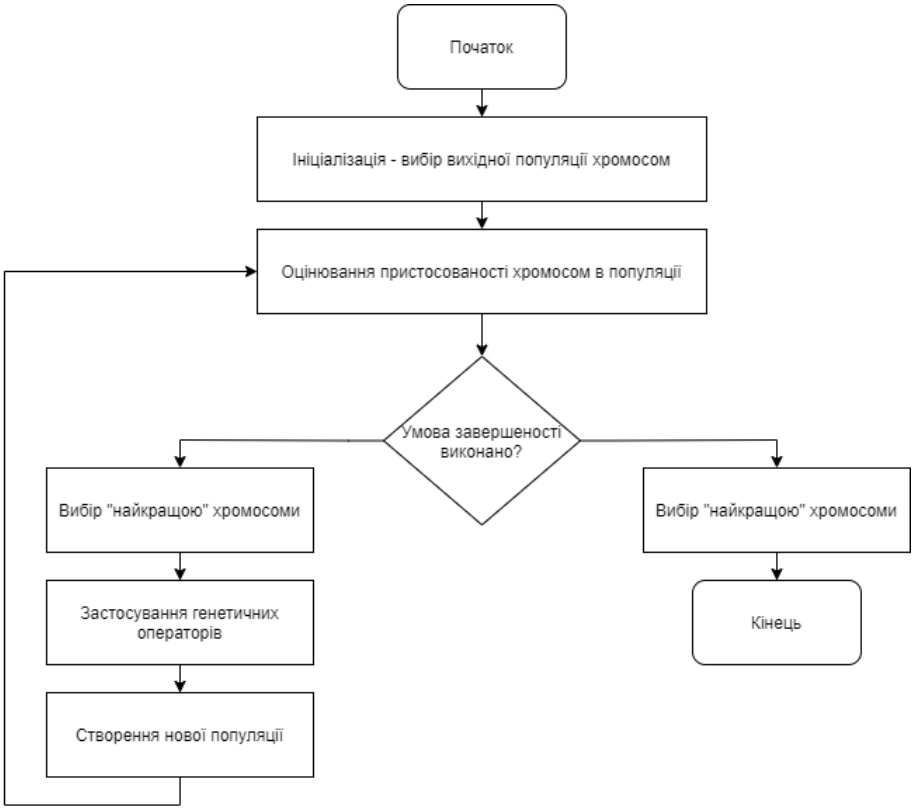

Рисунок 3.2 – Схема генетичного алгоритму

Можна позначити завдання оптимізації як задачу знаходження функції $f\ (x_5, x_6, \ldots, x_{\mathrm{n}})$, що носить назву функція пристосованості, яка дозволяє виділити найбільш пристосованих особин популяції для розмноження і найменш пристосованих, які викреслюються, підвищуючи тим самим пристосованість нового покоління. Потрібно, щоб на області визначення виконувалося нерівність $f\ (x_1, x_2, \ldots, x_{\mathrm{n}}),\ \geq\ 0$, область є обмеженою. Параметри функції записується як рядки, що складаються з бітів. Рядок, отримана конкатенацией рядків, називається особиною (див. рис. 3.3).

1010 10110 101 … 10101

$|x1\ |\ x2\ |\ x3\ |\ \ldots\ |\ xn\ |$

Рисунок 3.3 – Особина

Генетичний алгоритм універсальний, так, як тільки функція пристосованості та кодування рішень залежать від умов поставленого завдання. Також даний алгоритм відображає реальний приклад еволюції в природних популяціях і використовуються як ефективний пошуковий інструмент при еволюційному вирішенні завдань оптимізації.

При виконанні алгоритму враховуються наступні правила: початкова популяція вибирається випадково, кількість її особин залишається незмінним, кожна з них записується як рядок з певною довжиною кодування.

Кожен крок генетичного алгоритму можна розбити на три етапи [18]:

- пропорційний залежно від пристосованості відбір особин поточного покоління, які мають право виробляти потомство, і формування з них проміжної популяції;
- проміжну популяцію ділять на пару і з якоюсь імовірністю схрещують, в результаті в нове покоління потрапляє сама пара або її нащадки при їх присутності. Нащадки формуються з відсічених частин батьківських рядків, розділеної певною точкою.
- відбувається мутація отриманого покоління, що не допускає передчасної збіжності. Кожен біт особини з імовірністю не більше 1% записується як протилежну початковій.

Також даний алгоритм складається з наступних кроків:

- формування початкової популяції (набору рішень);
- оцінка пристосованості нинішнього покоління;
- перевірка умови зупинки алгоритму;
- селекція;
- застосування генетичних операторів (мутація, кросовер);
- оцінка рішень (формування нової популяції);
- відбір найкращого результату.

Пропонується наступна функція пристосованості в розв'язуваної задачі:

$$f_{it} = \begin{cases} P, t \leq t_{max} \\ \varepsilon, t > t_{max} \end{cases}$$

де P – корисність маршруту,

t – час, необхідний для проходу маршруту (обчислюється за допомогою жодного алгоритму розв'язання задачі комівояжера),

t_{max} – час, яким обмежений користувач, $\varepsilon > 0, \varepsilon \approx 0$.

Спосіб кодування рішення наступний: список об'єктів переформовувалися таким чином, щоб розташовані близько один до одного об'єкти були розташовані близько один до одного в новому списку. Новий список, що володіє необхідною властивістю, може бути представлений як порядок обходу всіх об'єктів жадібним алгоритмом для завдання комівояжера.

Таким чином, рішення буде кодуватися так: і-му гену хромосоми буде відповідати той об'єкт, який є і-м в обході всіх об'єктів жадібним алгоритмом для завдання комівояжера. Значення гена і дорівнюватиме 0, якщо і-ий об'єкт з нового списку не включено в маршрут, і дорівнюватиме 1 в іншому випадку.

Розмір популяції, ймовірність мутації, ймовірність схрещування особин є змінними параметрами. Також може варіюватися частка найбільш пристосованих особин, яка згідно зі стратегією елітарності, буде гарантовано переходити в наступне покоління. Для схрещування використовується одноточковий кросовер. Алгоритм завершує роботу після зміни певного числа поколінь, оптимальне значення якого буде також визначено під час тестування алгоритму.

Головною метою системи є формування оптимального маршруту. Щоб вибрати такий маршрут, генетичний алгоритм використовує функцію пристосування, яка оцінює кожен маршрут, і привласнює йому тим більшу оцінку, чим ближче він до оптимуму. В якості оцінки пристосування маршруту використовується час прибуття в точку призначення, взяте з протилежним знаком (завдяки чому менше по модулю значення, вважається великим, чим більша за модулем значення).

Основною функціональністю системи є генерація, створення та редагування

туристичних маршрутів, перегляд детальної інформації про визначні місця та їх рейтинг, а також написання відгуків та оцінювання як місць, так і маршрутів.

У запропонованій моделі довжина генотипу залежить від кількості робіт і виконавців. Кодування особини - десяткова. У генетичному алгоритмі особина – це одна хромосома. Початкова популяція генерується випадковим чином. Основним критерієм генерації популяції – достатня різноманітність особин. Це необхідно для того, щоб популяція не «впала» в найближчий екстремум [19].

Точкою зупинки роботи генетичного алгоритму є досягнення його збіжності, тобто до моменту, коли в популяції операції схрещування і мутації не дають приросту функції пристосованості, інакше відбувається стагнація в популяції. Схема моделі генетичного алгоритму показана на рисунку 3.4.

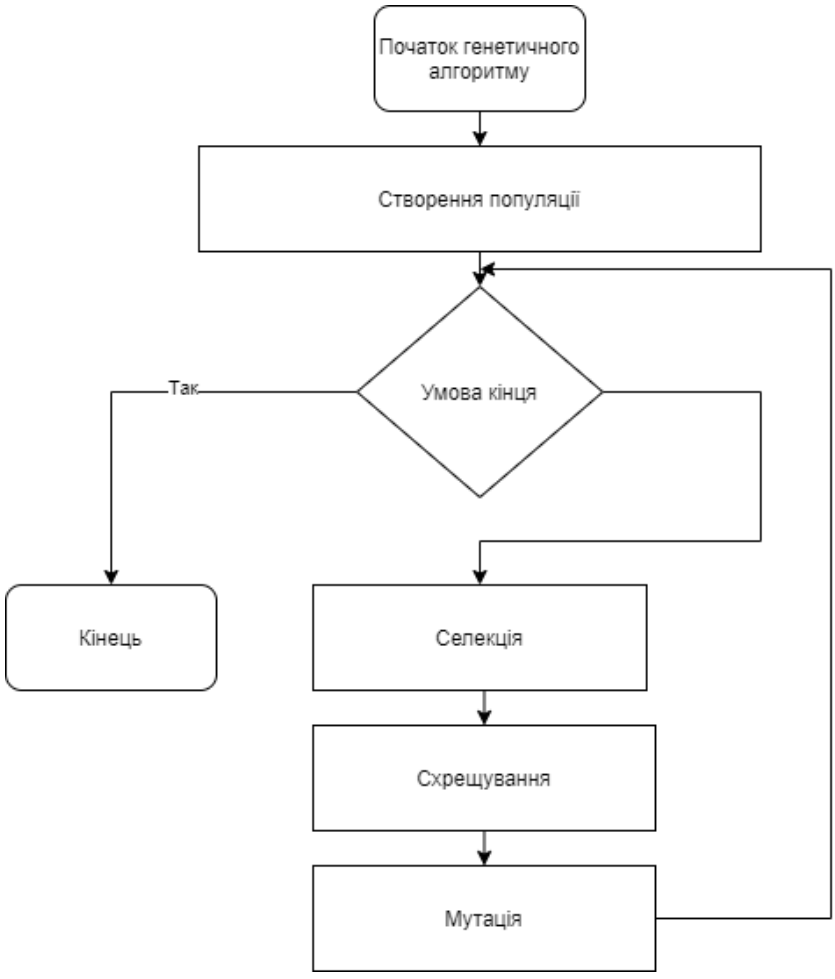

Рисунок 3.4 – Модель генетичного алгоритму

Дана модель відрізняється від класичного генетичного алгоритму цілеспрямованої мутації.

У алгоритмі використовуються наступні стандартні генетичні оператори: пропорційна селекція і модифікований арифметичний крос вдосконалення. При арифметичному схрещуванні обмін генами відбувається по всій хромосомі, але вибір гена батька відбувається за формулою:

$$x + \alpha * |y - x|$$

де x, y – гени батьків, α – константа, $\alpha \in [0,1]$.

Генетичний оператор мутації має спрямований характер, він аналізує значення генів і в залежності від їх значення знижує або підвищує їх значення.

3.5 Висновки

Були визначені критерії для оцінки відповідності точок місцевості побажанням та обмеженням користувача та критерії оцінки побудованих маршрутів.

Для побудови маршруту використаємо генетичний алгоритм.

Для визначення точок, які входять до складу маршруту, застосуємо відповідність значень критеріїв точок та побажань користувача, а при необхідності обмеження кількості точок важливу роль відіграватиме максимізація рейтингу точки.

Для оцінки побудованих маршрутів використаємо метод мультиплікативної згортки з нормуючими та ваговими коефіцієнтами.

4 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 UML проектування програмного забезпечення

Моделювання програмного продукту було проведено з використанням діаграм UML [20]. Кожна діаграма відображує систему з конкретної сторони, і у сукупності це дає змогу отримати потрібну інформацію для розробки системи (архітектура, робота, функціонал).

В рамках дослідження застосування генетичних алгоритмів для вирішення задач побудови маршруту генетичним алгоритмом було розроблено програмне забезпечення на мові Java, що використовує вільно поширювані карти для роботи з картами.

Використання генетичного алгоритму обумовлено високою обчислювальною складністю при побудові найкоротшого маршруту через набір заданих точок. На рисунку 4.1 представлена UML діаграма класів розробленого додатка.

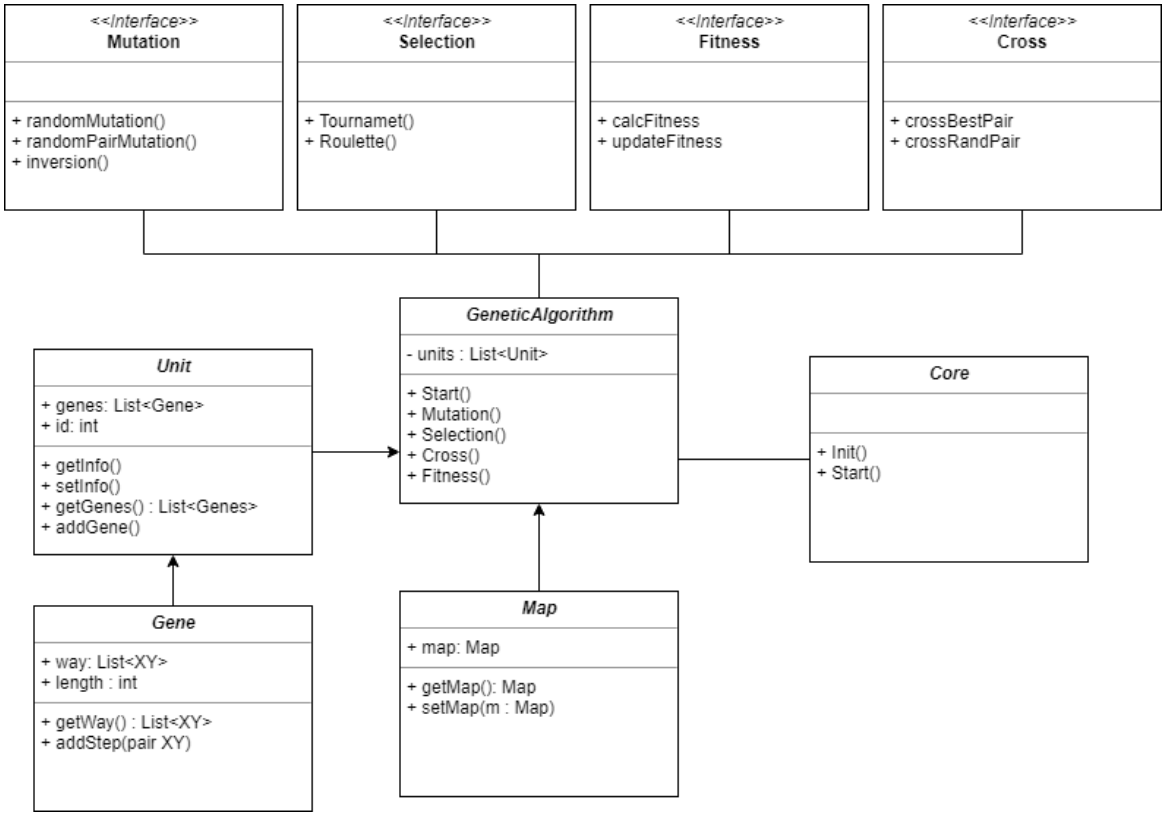

Рисунок 4.1 – Діаграма класів розробленого додатка

Основні класи діаграми:

а) Core – забезпечує налаштування та запуск алгоритму;

б) GeneticAlgotithm – здійснює роботу генетичного алгоритму;

в) Unit – містить опис особини, що піддається еволюції;

г) Gene – ген особини;

д) Map – надає доступ до карти. Основні інтерфейси класу:

е) Mutation – надає набір методів мутації:

1) випадкова мутація одного гена;
2) випадкова мутація пари генів;
3) інверсія випадкового гена;

ж) Selection – надає набір методів для відбору особин:

1) турнірний відбір;
2) відбір методом рулетки;

з) Fitness – обчислення фітнес-функції;

и) Cross – надає набір методів схрещування;

1) схрещування кращих рішень;
2) схрещування випадкових рішень

4.2 Проектування структури зберігання даних

Для повноцінної роботи будь-якої системи необхідно реалізовувати зберігання даних [21]. Було прийнято під час проектування використовувати реляційну модель баз даних. Оскільки база даних такого виду дозволяє відображати інформацію у найбільш простій та зручній формі. Також перевагою реляційної моделі є те, що вона базується на розвиненому математичному апараті, котрий дозволяє коротко описати основні операції над даними.

Схема бази даних для даного сервісу зображена на рисунку 4.2.

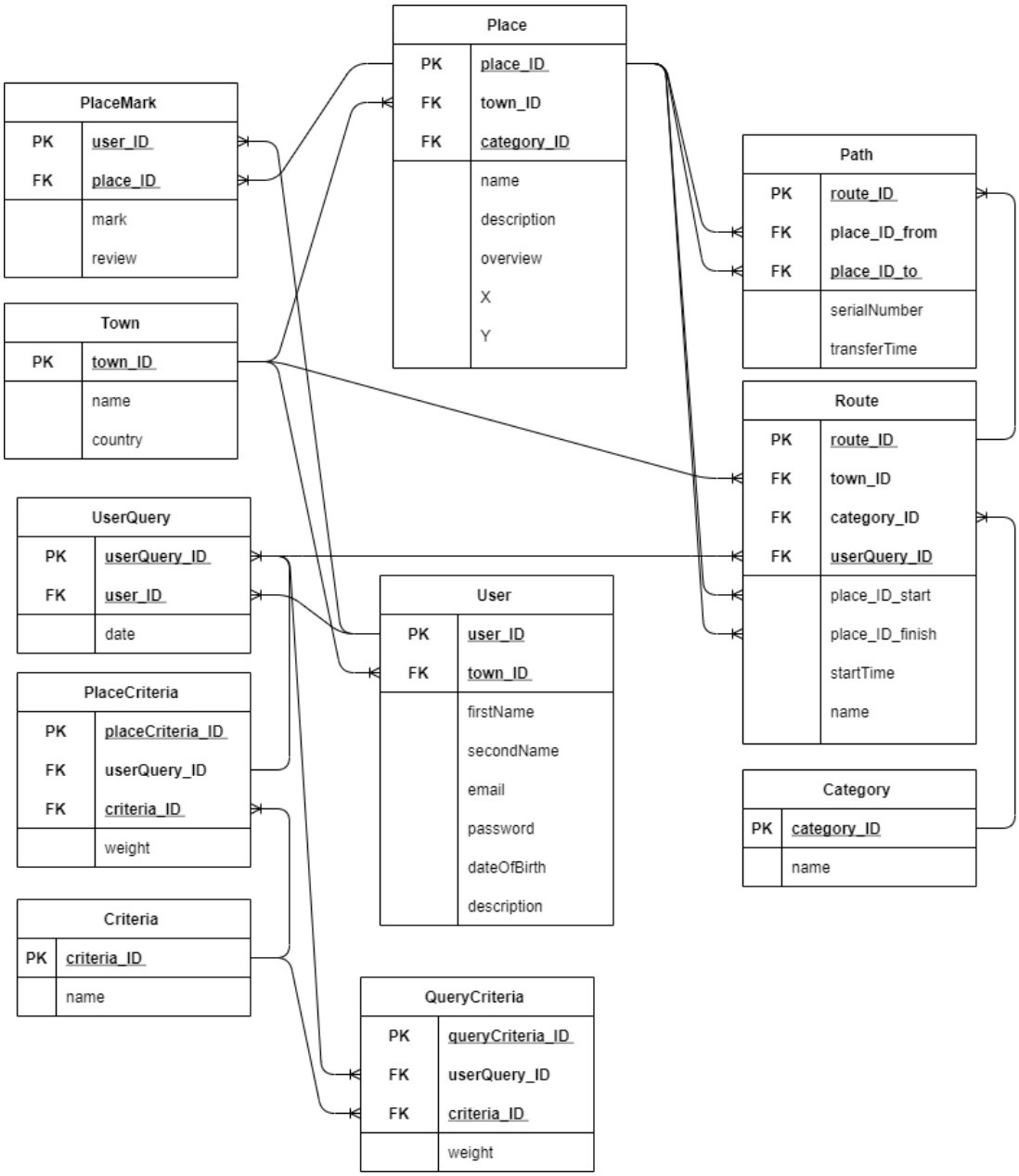

Рисунок 4.2 – Схема бази даних

База даних додатку повинна мати наступні сутності:

а) User – містить всю інформацію про користувача:

1) user_ID – унікальний ідентифікатор користувача (первинний сурогатний ключ);
2) town_ID – ідентифікатор міста, що слугує зовнішнім ключем до таблиці з містами;
3) firstName – ім’я користувача;
4) secondName – прізвище користувача;

5) email – електронна пошта користувача;
6) password – захешований пароль користувача;
7) dateOfBirth – дата народження;
8) description – опис користувача.

б) Route – містить інформацію про створений користувачем індивідуальний маршрут. Має наступні поля:
1) route_ID – унікальний ідентифікатор маршруту (первинний сурогатний ключ);
2) userQuery_ID – ідентифікатор користувача, що слугує зовнішнім ключем до таблиці з користувачами;
3) town_ID – ідентифікатор міста, що слугує зовнішнім ключем до таблиці з містами;
4) category_ID – ідентифікатор категорії, що слугує зовнішнім ключем до таблиці з категоріями;
5) place_ID_start – місце початку подорожі;
6) place_ID_finish – місце закінчення подорожі;
7) startTime – час початку маршруту;
8) name – назва маршруту;

в) Place – зберігає коротку інформацію про місце, яке може бути додане до будь-якого маршруту. Має наступні поля:
1) place_ID – унікальний ідентифікатор місця, слугує ідентифікатором для доступу до інформації в зовнішньому провайдері даних;
2) category_ID – ідентифікатор категорії, що слугує зовнішнім ключем до таблиці з категоріями;
3) town_ID – ідентифікатор міста, що слугує зовнішнім ключем до таблиці з містами;
4) name – назва місця, що зберігається із зовнішнього провайдеру даних;
5) description – короткий опис місця, що зберігається із зовнішнього провайдеру даних;

6) overviewTime;
7) X – довгота координати місця;
8) Y – широта координати місця;

г) Path – містить інформацію про дорогу від точки до точки:
1) route_ID – унікальний ідентифікатор асоціації для з’єднання місця і маршруту (первинний сурогатний ключ);
2) place_ID_from – місце початку мандрівки;
3) place_ID_to – місце кінця мандрівки;
4) serialNumber;
5) transferTime;

д) Category – категорія місця (наприклад: кафе, театр);

е) PlaceMark – оцінка місця користувачем;

ж) Town – місто.

з) Criteria – критерії:
1) criteria_ID – унікальний ідентифікатор критерія;
2) name – назва критерія;

и) PlaceCriteria – зберігає інформацію про критерії місця:
1) placeCriteria_ID – унікальний ідентифікатор критерія місця;
2) place_ID – ідентифікатор місця;
3) criteria_ID – ідентифікатор критерія;
4) weight – вага критерія;

к) UserQuery – зберігає інформацію про маршрути користувача:
1) userQuery_ID – ідентифікатор користувача;
2) user_ID – ідентифікатор користувача;
3) date – дата початку маршруту;

л) QueryCriteria – містить інформацію про критерії маршрута:
1) queryCriteria_ID – унікальний ідентифікатор критерія маршрута;
2) userQuery_ID – ідентифікатор користувача;
3) criteria_ID – ідентифікатор критерія;
4) weight – вага критерія;

5 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

5.1 Опис засобів розробки

Для розробки серверної частини програмної системи було обрано платформу .NET та мову програмування C#. Це об'єктно-орієнтована мова з безпечною системою типізації за допомогою якої розробники програмного забезпечення мають можливість створювати безпечні, сучасні та надійні додатки на платформі .NET [22].

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перезавантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. C# виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів.

C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR.

Для розробки мобільної частини програмної системи було обрано мову програмування Java [23]. Як і C#, це об'єктно-орієнтована мова за допомогою якої можна створювати різні задачі, починаючи від десктопних додатків та закінчуючи великими веб-порталами і сервісами, поєднуючи простий і знайомий синтаксис з надійним і зручним в роботі середовищем розробки.

Програми на Java можуть бути трансльовані в байт-код, що виконується на віртуальній Java-машині (JVM). За допомогою JVM обробляється байт-код та передаються інструкції пристрою, як інтерпретатор. Проте байт-код, на відміну від тексту, обробляється значно швидше.

Найбільш привабливою для використання є платформа .NET [24], оскільки це технологія, яка підтримує створення та виконання нового покоління додатків і

веб-служб XML. Платформа .NET Framework грає роль своєрідної операційної системи всередині операційної системи. В цілому дана платформа замислювалася як альтернатива платформі Java, основою якої також є віртуальна машина.

Для розробки серверної частини програмної системи було обрано технологію ASP.NET Web API [25] та середовище розробки Microsoft Visual Studio 2017.

Платформа ASP.NET Web API, яка є дуже популярною для розробки у сфері Enterprise, дозволяє з легкістю створювати служби HTTPS для широкого діапазону клієнтів, включаючи браузери і мобільні пристрої.

В односторінкових додатках комбінуються стандартні прийоми ASP.NET MVC Framework з Web API, оскільки у контролера API відсутня можливість генерувати HTML-розмітку з view. Інфраструктура ASP.NET MVC Framework виконує необхідне для доставки HTML-змісту користувачеві, включаючи аутентифікацію, авторизацію, вибір і візуалізацію уявлення. Контролер Web API після доставленого HTML-змісту до браузеру буде обробляти запити Ajax, котрі всередині генеруються кодом JavaScript.

За допомогою ASP.NET Web API створюються API за принципами REST. REST, або Representational State Transfer – передача стану через уявлення, – це архітектурний стиль взаємодії компонентів розподіленого додатка в мережі.

Ключовою абстракцією інформації в REST є ресурс. Ресурсом може бути названа будь-яка інформація, як документ або зображення, тимчасові служби, колекція інших ресурсів, невіртуальний об'єкт (наприклад, людина) тощо. Тобто, будь-яка концепція, що може виступати в ролі гіпертекстового посилання автора, повинна вписуватися в визначення ресурсу. Концептуальне відображення для безлічі об'єктів є ресурсом, а не об'єкт, який відповідає відображенню в будь-який конкретний момент часу.

Для розробки мобільної частини програмної системи було задіяно середу розробки Android Studio, систему автоматичного збирання Gradle. Весь програмний код було оптимізовано та зібрано під актуальною версією Android 5.1 Lollipop.

Для роботи з базами даних було вирішено використовувати Entity Framework,

який уявляє з себе об'єктно-оріентовану технологію доступу до даних, будучи object-relational mapping (ORM) рішенням для .NET Framework від Microsoft.

Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища.

5.2 Загальний опис серверної частини

Серверна частина була розроблена на платформі Microsoft .NET за допомогою фреймворку Microsoft ASP.NET WEB API та мови програмування C#. Запити до серверу дотримувалися протоколу HTTPS та були оформлені згідно зі стандартом REST.

Всі сутності в додатку виділені в окремі моделі. Залежно від поставленого завдання та складності програми виділяється різна кількість моделей. Так, для серверної частини мобільного додатку використовувалися 11 моделей: клас User для користувачів додатку, клас Route для маршруту, клас Path для шляху, клас Place для місця, клас PlaceMark для оцінки місця, клас Category для категорій місць, клас Town для міст, клас UserQuery для маршрутів користувачів, клас PlaceCriteria для критеріїв місць, клас Criteria для критеріїв, клас QueryCriteria для критеріїв маршруту.

Моделі являють собою прості класи і розташовуються в проекті в каталозі Models. Моделі описують логіку даних. Нижче наведено програмну реалізацію моделі User користувача мобільного додатку:

```
public class User {
[Key]
public int UserId { get; set; }
public int TownId { get; set; }
public string FirstName { get; set; }
```

```
public string LastName { get; set; }
public string DataOfBirth { get; set; }
public string Description { get; set; }
public string Email { get; set; }
public string Password { get; set; }
public virtual PlaceMark PlaceMark { get; set; }
public virtual Route Route { get; set; }
public virtual ICollection<Town> Towns { get; set; }
}
```

Також нижче наведено програмну реалізацію моделі Route мобільного додатку для відображення інформації про маршрут:

```
public class Route {[Key]
public int RouteId { get; set; }
public int UserQueryId { get; set; }
public int TownId { get; set; }
public int CategoryId { get; set; }
public int PlaceIdStart { get; set; }
public int PlaceIdFinish { get; set; }
public DateTime StartTime { get; set; }
public string Name { get; set; }
public virtual Path Path { get; set; }
public virtual ICollection<User> Users { get; set; }
public virtual ICollection<Town> Towns { get; set; }
public virtual ICollection<Place> Places { get; set; }
public virtual ICollection<Category> Categories { get; set; }
}
```

Створений контролер UsersController реалізовує функціонал по роботі з користувачами мобільного додатку на сервері. Далі наведено приклад реалізації дії, що реалізує відповідь на GET-запит в контролері:

```
// GET: api/Users
public IHttpActionResult GetUsers()
{ var users = _unitOfWork.GetUsers();return Json(users); }
```

```
// GET: api/Users/5
[ResponseType(typeof(User))]
public IHttpActionResult GetUser(int id)
{ var user = _unitOfWork.GetUser(id);return Json(user); }
```

Для запиту, при якому сервер приймає дані для зберігання, призначений метод запиту POST. Тобто, іншими словами можна сказати, що POST передає дані, котрі підлягають обробці, наприклад, з форми HTML в ідентифікований ресурс. Дані включені в тіло запиту. Це може привести до створення нового ресурсу або оновлень існуючих ресурсів або того й іншого.

Для забезпечення стабільної роботи із проколом HTTP використовується спеціальна бібліотека OKHTTP. Ця бібліотека підтримує усі сучасні засоби для зручної роботи із протоколом HTTP.

Для отримання даних додатком та коректного їх опрацювання у форматі JSON був розроблений спеціальний клас, лістинг коду якого наведено нижче:

```
public class JSONParser {
public static final MediaType JSON = MediaType.parse("application/json;
charset=utf-8");
OkHttpClient client = new OkHttpClient();static InputStream is = null;
static JSONObject jObj = null;static String jsonAns = null;
JSONObject postRequest(String url, RequestBody json) throwsIOException{
try {
Request request = new Request.Builder().url(url).post(json).build();
Response response =
client.newCall(request).execute();
jsonAns = response.body().string();
} catch (UnsupportedEncodingException e)
{ Log.e("Er:", Log.getStackTraceString(e));
} catch (IOException e) {
Log.e("Er:", Log.getStackTraceString(e));
} catch (Exception e) {
Log.e("Er:", Log.getStackTraceString(e));
} try {
jObj = new JSONObject(jsonAns);
```

```
} catch (JSONException e) {
Log.e("JSON Parser", "Error parsing data " +e.toString()); }
return jObj; } }
```

В даному підрозділі були наведені фрагменти коду програмної реалізації серверної частини.

5.3 Загальний опис мобільного додатку

Як вже було сказано, мобільна частина програмної системи була розроблена в середі розробки Android Studio за допомогою мови програмування Java.

Для вирішення задачі побудови індивідуального пішохідного туристичного маршруту на стороні мобільного додатку було реалізовано генетичний алгоритм.

Приклад формування маршруту зображено на рисунку 5.1.

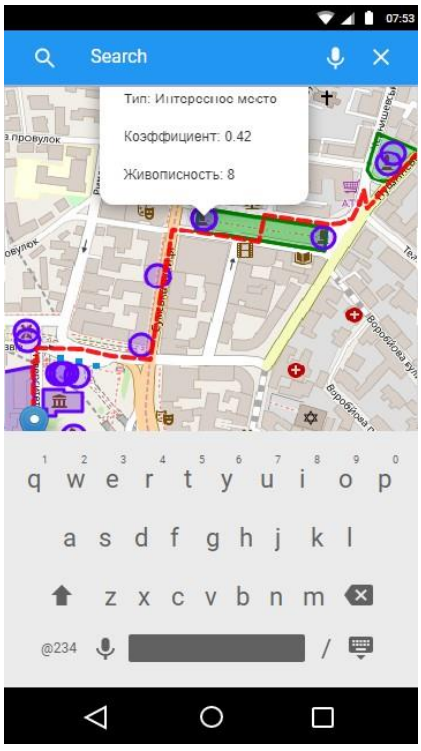

Рисунок 5.1 – Результат побудови маршрута

Код програмної реалізації алгоритму для побудови маршруту наведено в додатку Д.

5.4 Розгортання серверної частини програмного забезпечення

За основну платформу для розгортання серверної частини був обраний Microsoft Azure. Сервіс Web Applications дозволяє легко розгортати ASP.NET додатки у хмарі,а потім легко ними керувати, розширювати та масштабувати, вмикати та вимикатиза необхідністю.

Для систем великих даних, в яких повинні виконуватися як аналітичні операції на весь обсяг збережених даних, так і операції пошуку с швидким часом відгуку, сервіс Kudu видається природним кандидатом в якості движка зберігання даних. KuDu дозволяє подивитися структуру папок та їхній вміст, навіть відредагувати його і відразу побачити результат.

Можна проводити релізи нових версій за допомогою слотів розгортання майже непомітно для користувачів. Наприклад, створити так званий «production environment» – слот для версії, яку використовують користувачі, і «stage environment» – слот для фінальноготестування додатку. Після повного завершення тестування Web Applications дозволяє швидко поміняти місцями ці два слоти, і користувачі майже не помітять різниці.

Для зберігання данних було обрано сервіс Azure SQL Databases, який представляє собою звичайний SQL Server з деякими обмеженнями. Проте ці обмеження зовсім незначні і майже у всіх випадках можна знайти заміну стандартним можливостям SQL Server. Наприклад, Azure SQL Databases не має такої функції звичайного SQL Serveряк SQL Agent, агенту, що дозволяє виконувати певний скрипт на базі даних з певним заданим розкладом. Azure має схожий сервіс, який називається AzureDatabase Jobs, який має такі ж самі функції. А ось тимчасові глобальні таблиці Azure SQL Databases створювати не дозволяє.

ВИСНОВКИ

Під час написання кваліфікаційної роботи було спроектовано та реалізовано прототип програмної системи для побудови індивідуальних туристичних маршрутів.

Було проаналізовано статистику подорожей українців. Було обрано критерії якості, що повністю описують вимоги до кожної точки маршруту та критерії якості, що повністю описують вимоги до маршруту. Було проаналізовано метод прийняття рішень для проведення оцінки. Було проаналізовані існуючі алгоритми для побудови маршрутів та обраний один із найбільш вигідних варіантів. Було сформовано основні вимоги до мобільного додатку з формування туристичних маршрутів за обраними критеріями. Було спроектовано прототип програмного забезпечення, яке задовольняє більшість потреб користувачів. Була описана архітектура програмного забезпечення за допомогою діаграми класів, схеми бази даних.

Прототип програмної системи має дві складові частини: сервер та мобільний додаток. Сервер розроблений на ASP.NET Web Api з використанням Entity Framework для доступу до бази даних. Використовується СУБД MS SQL Server. Сервер дозволяє приймати запити у форматі JSON та відповідає на запити також у форматі JSON. Сервер побудований з використанням тришарової архітектури.

Мобільний додаток написаний на мові програмування Java під операційну систему Android. Мобільний додаток також взаємодіє з сервером у форматі JSON через HTTPS.

Результатом даної роботи є метод побудови туристичних маршрутів з урахуванням вподобань та обмежень користувача та програмна реалізація.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Комбінаторика як наука. Основні задачі комбінаторики. / URL: http://www.rusnauka.com/14_APSN_2008/Pedagogica/32696.doc.htm (дата звернення: 29.03.2021)

2. Задача комівояжера. / URL: http://surl.li/syju (дата звернення: 29.03.2021)

3. Meganavigator. / URL: http://meganavigator.com/ (дата звернення: 03.04.2021)

4. Логист. / URL: http://logist.poncy.ru/ (дата звернення: 03.04.2021)

5. Speedy Route. / URL: https://www.speedyroute.com/ (дата звернення: 03.04.2021)

6. About Trip.com Explore the world. / URL: www.trip.com/about (дата звернення: 03.04.2021)

7. World Travel Guide by Triposo. / URL: https://www.triposo.com/ (дата звернення: 03.04.2021)

8. Field Trip. / URL: https://en.wikipedia.org/wiki/Field_Trip_(application) (дата звернення: 03.04.2021)

9. Трофимова, Л. А. Менеджмент. Методы принятия управленческих решений: учебник и практикум для среднего профессионального образования / Л. А. Трофимова, В. В. Трофимов. — Москва: Издательство Юрайт, 2019. — 335 с.

10. Т. О. Гордієнко, О. О. Мазурова. Дослідження генетичних алгоритмів для пошуку оптимальних шляхів в системі проведення метромарафонів. — Бионика интеллекта. - Харків: ХНУРЕ, - 2019, № 2(93) С. 53-58, 2020.

11. Особливості розв’язання багатокритеріальних задач прийняття рішень у нечіткому середовищі. / URL: https://eprints.oa.edu.ua/6803/1/54.pdf (дата обращения: 09.04.2021).

12. Kuzochkina A., Z. Dudar, Shirokopetleva M. Analyzing and Comparison of NoSQL DBMS International Scientific and Practical Conference «Problems of Infocommunications. Science and Technology» (PIC S&T`2018), October 9-12, 2018.

Proceedings 8632133, pp. 560-565.

13. Кондрук Н. Е. Багатокритеріальна оптимізація лінійних систем: навч. посібник / Н. Е. Кондрук, М. М. Маляр – Ужгород: РА “АУТДОР-ШАРК”, 2019. – 76 с.

14. Передвижение в туристическом походе / URL: http://surl.li/syls (дата обращения: 10.04.2021).

15. Д. Рассел. Джесси Рассел Теорема Байеса – VSD, 2013, – 78 с.

16. Pitiukova M. The justification for the choice of the algorithm for constructing pedestrian tourist routes. // Monografia Pokonferencyjna. Science, research, development #12. Technics and Technology #2. Belgrade(Serbia), Warszawa, 2018 – 108 с.

17. Гладков Л.А. Генетичні алгоритми: навч. посібник / Л.А. Глад- ков, В.В. Курейчик, В.М. Курейчик. - М .: Фізико-математична література, 2010 - 367 с.

18. Pitiukova M., Shevchenko , O. The use of genetic algorithm for formation of pedestrian tourist routes. / Сучасні аспекти та перспективні напрямки розвитку науки: матеріали I міжнародної студентської наукової конференції (Т.2), 16 квітня, 2021рік. Кропивницький, Україна: Молодіжна наукова ліга. С.43-45

19. А.В. Веселов, Л. В. Найханова. Генетический алгоритм для решения задачи о назначениях. / URL: https://elibrary.ru/download/elibrary_39146141_96100900.pdf (дата обращения: 12.04.2021).

20. К. Ламан. Застосування UML та шаблонів проектування. – М.: Видавничий дім «Вильямс», 2004. – 624 с.

21. Дейт К. Дж. Введення в системи баз даних. – М.: Видавничий дім «Вильямс», 8-е видання, 2005. – 1328с.

22. Г.Шилдт. C# 4.0 Повне керівництво. – М.: Видавничий дім «Вильямс», 2011 – 1056 с.

23. Комплекс навчально-методичного забезпечення навчальної дисципліни "Основи програмування на Java" підготовки бакалавра [Електронний ресурс] : спеціальності 121 - Інженерія програмного забезпечення / ХНУРЕ ; розроб. Д. О. Колесников. – Харків, 2017. – 177 с. / URL:

https://catalogue.nure.ua/document=221985.

24. Загальні відомості про платформу .NET Framework / URL: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview (дата звернення 31.05.2019).

25. ASP.NET Web API: Build RESTful web applications and services on the .NET framework / J. Kanjilal. – Packt Publishing, 2013 – 224 c.