НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Радіотехнічний факультет
(повна назва інституту/факультету)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва кафедри)

«На правах рукопису»
УДК 004.2

«До захисту допущено»
Завідувач кафедри
(підпис) Є.А. Мелін (ініціали, прізвище)
"___" грудня 2018 р.

Магістерська дисертація

за спеціальністю 172 Телекомунікації та радіотехніка
(код і назва спеціальності)
за спеціалізацією Інтелектуальні технології мікросистемної радіоелектронної техніки

на тему: „Оптимізація маршрутизації на основі використання даних"

Виконав (-ла): студент (-ка) 6 курсу, групи ДІ-371мп
(шифр групи)

Шевченко Владислав Олегович (прізвище, ім'я, по батькові) В. Шев (підпис)

Науковий керівник проф., д.т.н. О.П. Яненко (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з охорони праці к.т.н., доцент Каштанов С.Ф.
(назва розділу) (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент РОС Ч.О. Малєєвська Н.О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент В. Шев
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет радіотехнічний

Кафедра радіоконструювання та виробництва радіоапаратури

Рівень вищої освіти – другий (магістерський)

Спеціальність 172 – телекомунікації та радіотехніка

Спеціалізація інтелектуальні технології мікросистемної радіоелектронної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Є.А. Нелін
(підпис) (ініціали, прізвище)
«30» вересня 2017 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Шевченко Владислав Олегович
(прізвище, ім'я, по батькові)

1. Тема дисертації Оптимізація маршрутизації на основі використання даних,
науковий керівник дисертації Олексій Пилипович Яценко проф. д.т.н,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «6» грудня 2018 р. № 4094-с

2. Термін подання студентом дисертації 30 листопада 2018 року

3. Об'єкт дослідження мережевий додаток моделювання оптимального маршруту

4. Предмет дослідження алгоритми пошуку оптимального маршруту, використання даних, програмний код на Java, Java Script, сервер на базі мікрос, бази даних

5. Перелік завдань, які потрібно розробити Розглянути основні алгоритми пошуку оптимального маршруту, провести аналіз даних алгоритмів, написати код обраного алгоритму, використовуючи сучасні технології і фреймворки: Java, Spring, Java Script. На основі отриманих даних аналізу розробити додаток, що використовує дані та прораховує маршрути.

6. Орієнтовний перелік ілюстративного матеріалу презентація

7. Орієнтовний перелік публікацій 1 публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	доцент Коштанов С.Ф.		

9. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Пошук джерел інформації	10.09.17 - 15.11.17	Аналіз
2	Огляд предметної області	16.07.18 - 15.08.18	Розділ 1
	Аналіз існуючих рішень		
3	Аналіз та вибір алгоритмів	16.10.18 - 31.10.18	Розділ 2
4.	Проведення розробки програми, розв-	01.10.18 - 31.10.18	Розділ 3
	язки алгоритм побудови маршруту		
5.	Підготовка звітної документації	01.10.18 - 01.12.18	

Студент В.Шев — (підпис) В.О. Шевченко (ініціали, прізвище)

Науковий керівник дисертації _____ (підпис) _____ (ініціали, прізвище)

Реферат

Структура й обсяг дипломної роботи:

Магістерська дисертація: 81 с., 17 рис., 8 табл., 4 додатка, 15 джерел.

Ключові слова: TSP, geolocation, Java, JavaScript, Spring, Ajax, Hibarnate, Телекомунікація клієнта з сервером по TCP/IP, HTTP.

Актуальність теми в аналізі алгоритмів, оптимальних для поставленого завдання з використанням сучасних фреймворків, що надасть приріст в продуктивності.

Мета дослідження є необхідність створення мережевого додатку, для пошуку і геокодування оптимального шляху.

Для реалізації поставленої мети були сформульовані такі **завдання дослідження**: дослідити предметну область типового рішення побудови оптимального маршруту; створити математичну модель існуючих алгоритмів TSP з описом їх переваг та недоліків; розробити мережевий додаток з використанням сучасних мов програмування та фреймворків.

Об’єкт дослідження — геокодування, побудова маршрутів, телекомунікація клієнта та сервера**.**

Предмет дослідження — мержевий додаток гекодування і побудови оптимальних маршрутів.

Методи дослідження: При вирішенні задач роботи застосовувались наступні методи: математичне моделювання алгоритмів TSP, програмування середовищы Intellij за допомогою Java 8, тестування Junit.

Наукова новизна одержаних результатів. Наукова новизна полягає в розробці методики оптимізації розрахунку TSP та телекомунікації клієнта з сервером по TCP/IP і HTTP протоколах.

Практичне значення одержаних результатів виражається в потенціалі для подальшого розвитку і комерціалізації додатку.

ASTRACT

Structure and volume of the diploma

Master dissertation: 76 p., 17 fig., 8 tabl. 4 aplication, 15 sources.

Keywords. TSP, geolocation, Java, JavaScript, Spring, Ajax, Hibarnate, Telecommunications client with server with heips of TCP/IP, HTTP.

Relevance of the topic. In the analysis of algorithms optimal for a given task using modern frameworks, which will provide an increase in productivity.

The aim of the study is the need to create a network application for the search and geocoding of the optimal path.

To accomplish this goal, the following **research objectives** were formulated: to investigate the subject area of a typical solution for constructing an optimal route; create a mathematical model of existing TSP algorithms with a description of their advantages and disadvantages; Develop a network application using modern programming languages and frameworks.

The object of the research is geocoding, route construction, customer and server telecommunications.

Subject of research is geocoding, route construction, customer and server telecommunications.

Research methods: is solving problems of work, the following methods were used: mathematical modeling of TSP algorithms, programming of Intellij environments using Java 8, Junit testing.

Scientific novelty of the obtained results. The scientific novelty is to develop a methodology for optimizing TSP calculation and client telecommunications with the server through TCP / IP and HTTP protocols.

The practical value of the results obtained is expressed in the potential for further development and commercialization of the application.

ЗМІС

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ....4
ВСТУП....5
РОЗДІЛ 1 ОГЛЯД ЗАДАЧІ ОПТИМАЛЬНОЇ ПОБУДОВИ МАРШРУТУ....7
1.1 Типова задача клієнта....7
1.2Аналіз існуючих сервісів і їх недоліки....9
1.3 Постановка задачі....13
1.4Висновки по Розділу 1....14
РОЗДІЛ 2 МАТЕМАТИЧНА ОСНОВА ПРОЕКТУ....14
2.1.Аналіз алгоритмів пошуку оптимального шляху....14
2.2 Точні алгоритми....15
2.3 Неточні алгоритми....19
2.4 Directions API from Google Maps....23
2.5 Генетичний алгоритм....24
2.6 Висновки по розділу 2....30
РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ....31
3.1.Огляд використовуваних в розробці програми інструментів....31
3.2 Логіка роботи додатку....37
3.3 Аналіз отриманих даних та тестування додатку....49
3.4 Висновок по розділу 3....51
РОЗДІЛ 4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ....52
4.1 Опис ідеї проекту....52
4.2 Технологічний аудит ідеї проекту....53
4.3 Аналіз ринкових можливостей запуску стартап-проекту....54
4.4 Розроблення ринкової стратегії проекту....57
4.5 Розроблення маркетингової програми стартап-проекту....59
4.7 Висновки за розділом....60
РОЗДІЛ 5 ОХОРОНА ПРАЦІ ТА ПОЖЕЖНА БЕЗПЕКА....61
5.1 Аналіз небезпечних і шкідливих чинників....61
5.2 Заходи щодо поліпшення умов праці при роботі з комп’ютером....71
5.3 Пожежна безпека....72
ВИСНОВОК....77
СПИСОК ЛІТЕРАТУРИ....78
ДОДАТОК А Технічне завдання....81

1 Підстава для виконання роботи.....82
2 Мета і призначення.....82
2.1 Об’єкт та предмет дослідження.....82
2.2 Мета роботи.....82
2.3 Задачі, які потребують вирішення.....83
3 Вихідні данні для проведення роботи.....83
3.1 Підручники.....83
4 Вимоги до виконання.....83
5 Етапи та термін виконання.....84
6 Очікувані результати та порядок реалізації.....85
7 Матеріали, які подаються по закінченню дипломної роботи.....86
8 Порядок приймання дисертації та її етапів.....86
9 Вимоги до розроблюваної документації.....86
10 Приблизний зміст дипломної роботи.....86
ДОДАТОК Б Перелік публікацій за темою магістерської дисертації.....88

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

TSP (Travelling salesman problem)	завдання побудови оптимального маршруту
GPS (Global Positioning System)	система глобального позиціонування
ORM (Object-Relational Mapping)	об'єктно-реляційне відображення
JDBC (Java DataBase Connectivity)	з'єднання з базами даних на Java
HTML (HyperText Markup Language)	мова розмітки гіпертекстових документів
CSS (Cascading Style Sheets)	каскадні таблиці стилів
JAVA SE (Java Standard Edition)	

JAVA EE (Java Enterprise Edition)	
API (Application Programming Interface)	інтерфейс програмування Додатків
SQL (Structured query language)	мова структурованих запитів
DOM (Document Object Model)	модель документа об'єкта
JSON (JavaScript Object Notation)	об'єктній запис JavaScript

ВСТУП

Завдання побудови оптимального маршруту, вперше піднята в 1832 році народами Греції. Вони висловлювали пропозиції щодо дій торгових агентів, а саме – як він повинен вести себе і що повинен робити для того, щоб доставляти товар і мати успіх у своїх справах – поради старого кур'єра, є актуальною і на сьогоднішній день – розробляються нові методи розв'язання задачі, реалізуються програми, які дозволяють працювати з кількістю вузлів близьким до мільйону за прийнятний час. Незгасний інтерес до цього завдання обумовлений різноманітністю застосувань її на практиці. Пошук оптимального шляху широко використовується у всіх завданнях транспортної логістики, на виробництві – в вигляді задач мінімізації часу переналагодження.

Питання оптимізації маршруту піднімаються в працях таких відомих діячів математики як Вільям Роуен Гамільтон, Джордж Данциг, Річард Карп,

Девід Аплгейт, Герхард Райнельт.[1] Незважаючи на велику теоретичну базу, на жаль, представлено не так багато додатків, що дозволяють людям, далеким від математики і програмування, використовувати існуючі розробки для вирішення практичних завдань. Користувач, який хоче обійти n-ну кількість місць за мінімальний час, використовуючи такі популярні картографічні сервіси як «Яндекс Карти» [2] і «Google Карти» [3], не може вирішити поставлене завдання, так як маршрут в додатку будується по заздалегідь заданому в певному порядку списку місць. Мета даної роботи полягає в інтегруванні алгоритму завдання розрахунку оптимального маршруту між n-ною кількістю місць, з можливістю корегування побудованого маршруту клієнтом, віртуального перегляду вулиць та місць по кліку на карті, отримання повної інформації щодо пересування між пунктами призначення .

Для досягнення зазначеної мети поставлено такі завдання:

1. Розглянути основні алгоритми пошуку оптимального маршруту;

2. Провести аналіз даних алгоритмів і вибрати один з них для практичної реалізації;

3. Написати код обраного алгоритму, використовуючи сучасні технології і фреймфорки : Java, Spring, JavaScript, авторизацію, реєстрацію клієнта з використанням Spring Sequrity і базу даних MySql;

4. На основі отриманих даних аналізу розробити додаток, що дозволяє користувачам ввести пункт відправлення, призначення і проміжні пункти свого маршруту, вибрати тип поїздки, отримати оптимальний шлях, наочно показаний на Google Maps карті з можливістю корегування, візуальним переглядом вулиць, місць маршруту по кліку на певне місце на карті та отриманням широкої інформації для пересування по маршруту .

Актуальність даної роботи полягає в поставленої мети – аналізу алгоритмів, оптимальних для поставленого завдання, і створенні зручного

додатки з пошуку оптимального шляху для гідів, екскурсоводів, кур'єрів, туристів, які потребують в швидкій побудові оптимального маршруту через n- ну кількість місць. Практична значимість роботи виражається в потенціалі для подальшого розвитку і комерціалізації додатку.

РОЗДІЛ 1 ОГЛЯД ЗАДАЧІ ОПТИМАЛЬНОЇ ПОБУДОВИ МАРШРУТУ

Перший розділ складається з огляду предметної області, що включає в себе опис та історію типового рішення побудови оптимального маршруту, а також аналізу вже існуючих додатків. Ставиться завдання диплома, формулюються вимоги до її реалізації.

1.1 Типова задача клієнта

Комбінаторика - це розділ математики, що вивчає комбінації і їх кількість, складені з тих чи інших умов з заданих об'єктів. Вперше питання, які вивчає комбінаторика, були підняті в працях математиків Стародавньої Греції, Індії та Китаю. Інтерес до предмету збільшився в 19-му і 20-му століттях у зв'язку з розвитком теорії графів. У комбінаторики є багато застосувань в інших областях математики, включаючи теорію графів, програмування, криптографію і теорію ймовірності. Серед провідних математиків цій галузі можна виділити Блеза Паскаля (1623-1662), Якоб

Бернуллі (1654-1705), Леонхард Ейлера (1707-1783) і Пала Ердеша (1913-1996) [4].

Завдання побудови оптимального маршруту – класична задача комбінаторики. Щоб наочніше показати її значимість, нижче наведена коротка історія вирішення поставленого завдання.

Математичні проблеми, пов'язані із завданням побудови маршруту, були вивчені в 1800-их роках ірландським математиком сером Вільямом Роуеном Гамільтоном і британським математиком Томасом Пенінгтон Кіркменом. У 1857 Гамільтон створив гру Ікосіан, в правилах якої сказано, що учасники повинні з'єднати 20 точок додекаедру так, щоб кожна точка використовувалася не більше одного разу, також кінцева точка шляху повинна збігатися з вихідною. Ця гра лягла в основу появи гамільтонова графа.[5]

Завдання пошуку оптимально маршруту було вперше розглянута з математичної точки зору в 1930-их роках математиком і економістом Карлом Менджером у Відні. У 1940-их статистики Механобіс, Джессен, Гош і Маркс намагалися знайти застосування цій задачі в сільськогосподарському секторі, що призвело до її популяризації – починаючи з середини 1950х роках методи розв'язання задачі стали публікуватися в наукових журналах.

Хоча завдання побудови маршруту проста для розуміння, її вкрай складно вирішити. [6] У 1972 Річард М. Короп показав, що завдання Гамільтона циклу належить до класу NP-повних задач (Nondeterministic Polynomial time), що має на увазі NP-складність завдання TSP. [7] Це відкриття дало наукове пояснення очевидною обчислювальною складністю знаходження оптимальних маршрутів.

Методи рішення TSP ставали більш складними, кількість міст зростала. Нижче представлена коротка історія етапів рішення задачі знаходження оптимальних маршрутів.

У 1954 Данциг, Фалкерсона і Джонсон видали опис методу для рішення TSP і практично проілюстрували його, вирішивши завдання з 49 містами, з'єднавши таким чином міста кожного штату Америки. У 1971 році Гельд і Карпо вирішили задачу пошуку оптимального шляху між 64 містами. Пізніше в 1977 році, Мартин Гротчел побудував шлях між 120 німецькими містами. У 1973 Лін і Керниган знайшли застосування завдання TSP в інженерії - вони поєднали 318 пунктів, отриманих в результаті різання лазером. [8]

У 1987 році Голландія і Гроешел була вирішена задача з 666 містами, пізніше в цьому ж році Падберг і Рінальді знайшли оптимальний тур, зв'язуючи вже 2392 міста. У 1994 році кількість міст збільшилася до 7397, через 10 років кількість міст досягло до 24978. У 2006 було отримано рішення задачі оптимального розрахунку маршрутів з 85900 містами. [8]

1.2 Аналіз існуючих сервісів і їх недоліки

Популярні картографічні сервіси типу Яндекс.Карти і Google Maps не пропонують користувачам можливість пошуку оптимально шляху. При введенні декількох координат сервіс вибудовує маршрут в тому порядку, в якому дані були введені. Користувачі можуть вибирати засоби пересування (на машині, пішки, на наземному транспорті), але всі ці зміни впливають виключно на варіанти побудови маршруту між його фіксованими точками.

Аналіз, проведений шляхом порівнянням десятків як російськомовних, так і зарубіжних картографічних сервісів показує, що серед найпопулярніших варіантів тільки у одного доступна функція побудови оптимально шляху, і далеко не завжди вона працює коректно. Нижче представлений детальний звіт про найбільш гідні уваги сервіси і їх недоліки. Серед російськомовних сервісів в першу чергу можна виділити Meganavigator [9]. Він включає в себе конструктор карт, візуалізацію GPS треків і побудова оптимальних маршрутів. Сервіс являє собою сайт з вбудованою яндекс картою і полями

для введення пунктів маршруту (Рис 1.1).Тестування функціоналу показало наявність істотних недоліків:

1. Незважаючи на те, що Meganavigator позиціонує себе як сервіс для побудови автомобільних, велосипедних і пішохідних маршрутів, на сайті відсутня вибір засобу пересування: режим, виставлений за замовчування, відповідає пересуванню на автотранспортному засобі;
2. У сервісу не передбачені підказки при введенні адреси, що ускладнює роботу з ним;
3. При побудові маршруту перший пункт вибирається довільно із заданого списку адрес. Немає можливості задавати адресу як перший за замовчуванням;
4. При виборі пунктів на інтерактивній карті без введення адрес в текстові поля, маршрут не будується.

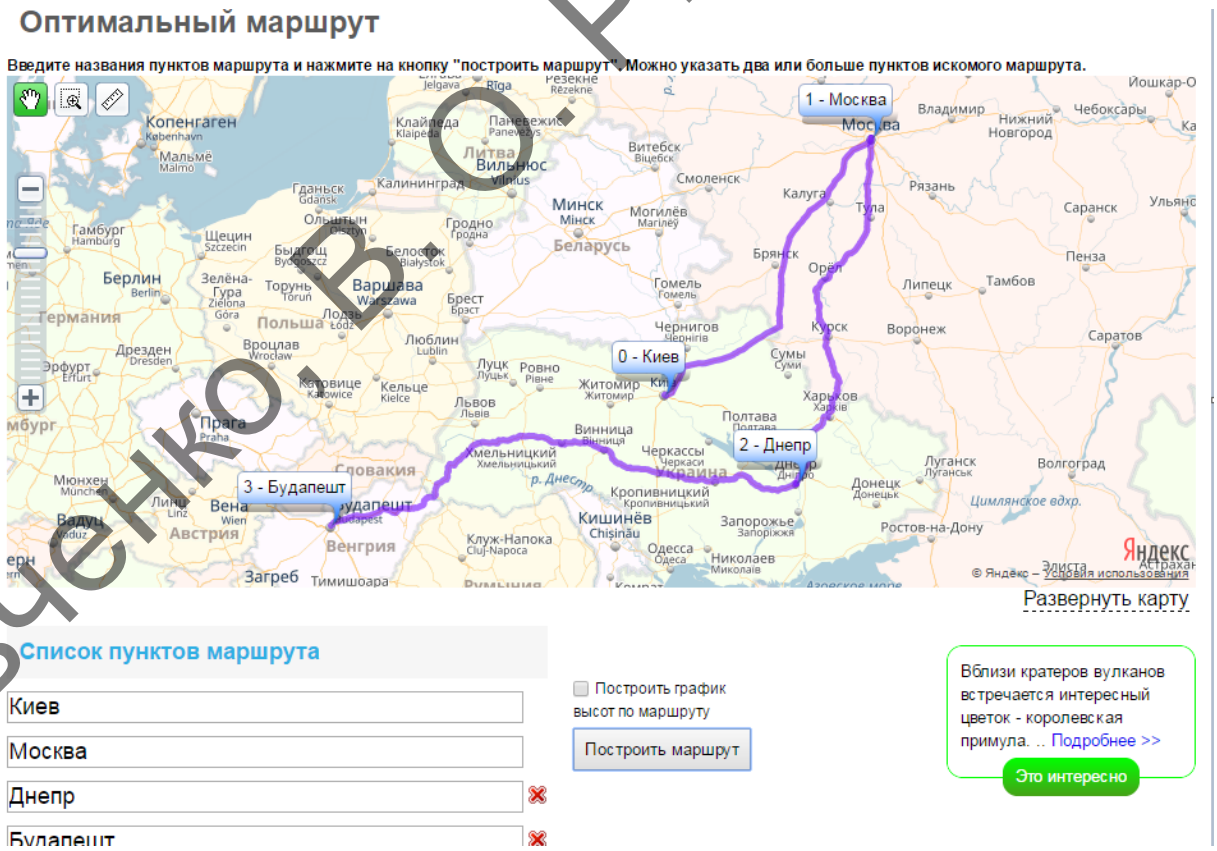

Рисунок 1.1 – Сервіс MegaNavigator

Наступний російськомовний сервіс Логіст [10] має менший кількістю недоліків, але всі вони істотні:

1. Серед засобу пересування можливий вибір між автомобілем (Побудова маршруту по проїжджих дорогах) і вертольотом (прямий шлях між пунктами маршруту);
2. Сервіс в першу чергу розрахований для вирішення завдань логістики і побудови маршруту перевезень продукції між містами (Рис. 1.2).

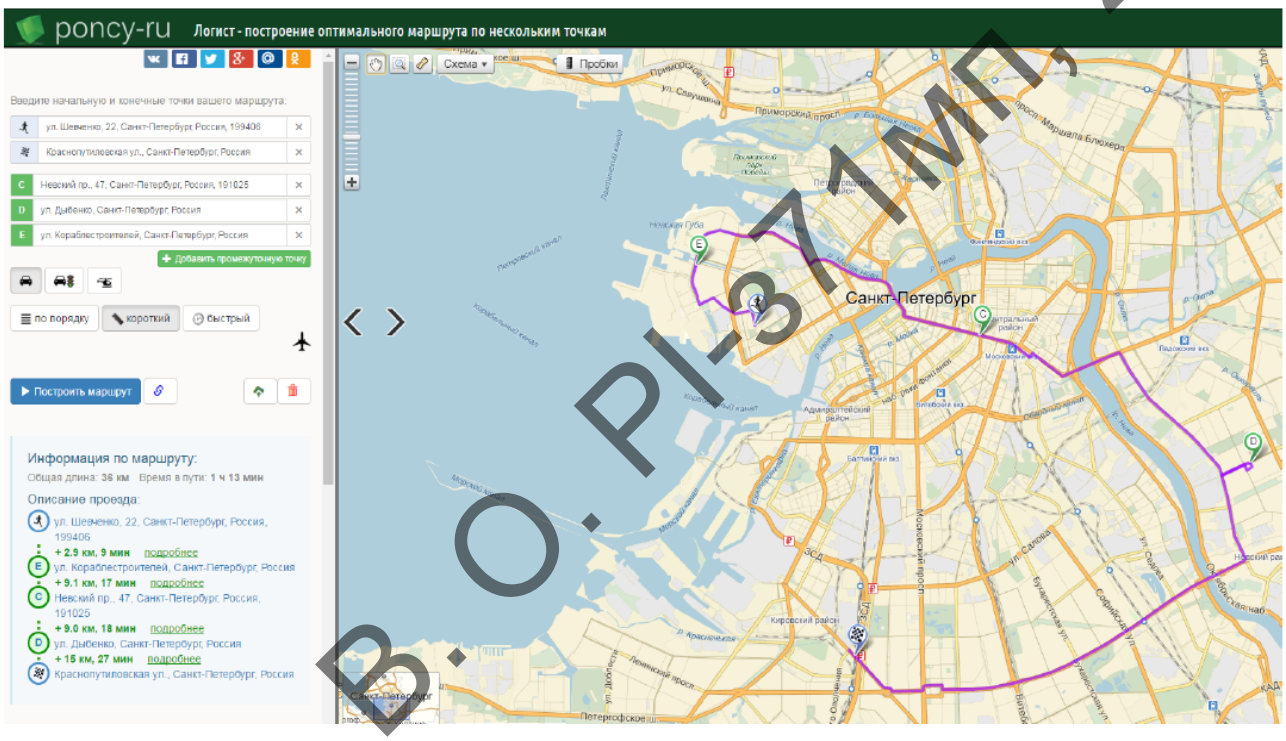

Рисунок 1.2 – Сервіс Логіст.

Серед зарубіжних картографічних сервісів з функцією побудови оптимального маршруту найбільш популярний Speedy Route [11]. В описі продукту сказано, що сервіс розраховує найбільш ефективний по кількості витрат палива маршрут між декількома локаціями, де вихідна і кінцева точка єдині (Рис. 1.3). В першу чергу можна виділити такі недоліки як:

1. Speedy Route розрахований виключно для автомобілістів;
2. При введенні даних виникають труднощі перекладу російськомовних, українських назв на англійську мову;
3. У сервісу передбачено мінімальну кількість пунктів маршруту - 5; маршрути, що складаються з 4 пунктів побудувати

4. неможливо;
5. На сайті представлена урізана версія карти для ознайомлення, для доступу до повної версії потрібно оформити платну підписку.

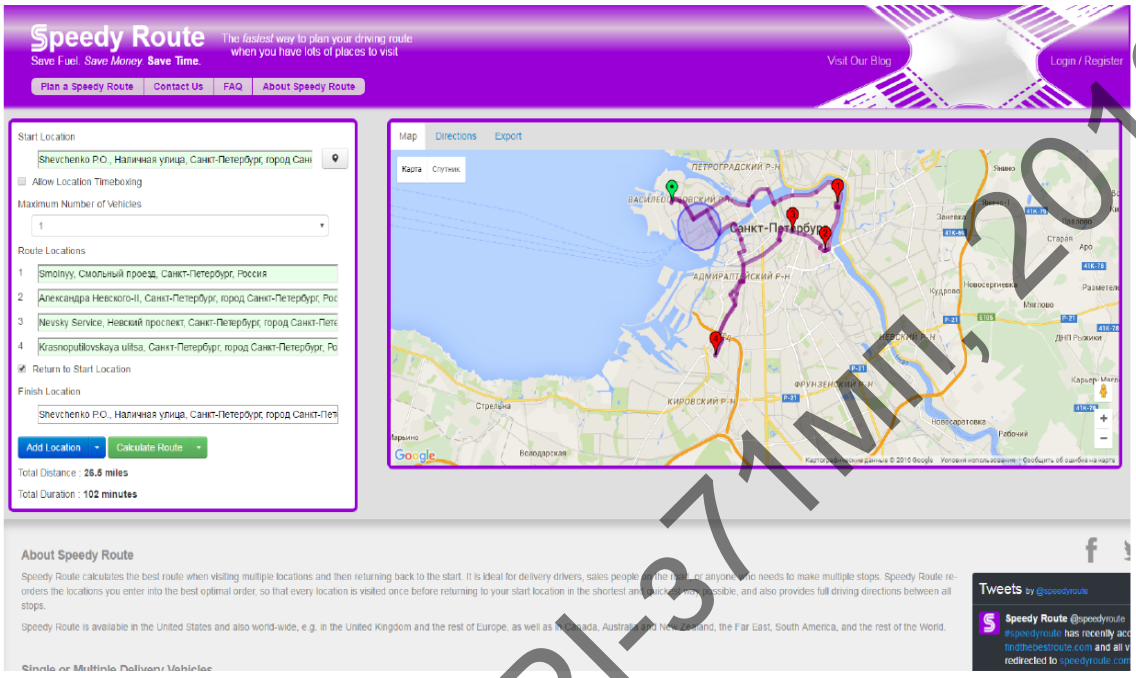

Рисунок 1.3 – Сервіс Speedy Rout

Підсумовуючи все вище зазначене можна сказати, що поки не існують сервіси, які б будували оптимальні пішохідні маршрути усередині міста, чи з'єднували б глобальні точки, надавали візуальне представлення будь–якого елемента маршруту і одночасно були зручними та зрозумілими для користувачів при взаємодії з ними.

1.3 Постановка задачі

Аналіз вже існуючих додатків, представлений в попередньому розділі, допомагає сформулювати вимоги, які можна висунути для розробки додатку по розрахунку оптимального маршруту по запиту клієнта .

1. Функціонал програми полягає в побудові оптимального маршруту за заданими даними, виведення його на інтерактивній карті, виведення інформації щодо пересування по маршруту, візуалізація будь–якого елемента на маршруту;

2. Додаток буде реалізовано на англійській мові для розширення аудиторії веб – сервісу, в подальшому можуть бути додані інші мови;
3. Одне з ключових параметрів сервісу це його доступність – найбільш вдалим рішенням буде реалізація програми в як веб-сайту;
4. Важливо мати можливість реєстрації та авторизації нових клієнтів, з розширення ролів.
5. У зв'язку зі стрімкістю розвитку науки і постійним знаходженням нових методів і оптимізації старих, сервіс повинен бути спроектований таким чином, щоб обсяг робіт по зміни оптимізаційного алгоритму був мінімальний, повинен мати сервер написаний на Java і клієнт написаний на Java Script, Html, CSS, Bootstrap.

1.4 Висновки по Розділу 1

У цьому розділі був представлений огляд предметної області, представлені опис і коротка історія формування завдання пошуку оптимального маршруту, наведений аналіз недоліків існуючих додатків. За результатами огляду було поставлено завдання диплома з вимоги до її реалізації.

РОЗДІЛ 2 МАТЕМАТИЧНА ОСНОВА ПРОЕКТУ

У цьому розділі буде проведений аналіз існуючих алгоритмів TSP з описом їх переваг та недоліків, буде детально розібрано застосування генетичних алгоритмів до вирішення завдання пошуку оптимального маршруту, проаналізовано переваги використання Google Maps, і показано, чому саме Geolocations Api обрано як інструмент розробки програми.

2.1. Аналіз алгоритмів пошуку оптимального шляху.

Алгоритми для вирішення завдання пошуку оптимального шляху можна розділити на точні (exact algorithm) і неточні (non-exact algorithm). Точні алгоритми включають в себе перебір всіх можливих варіантів, в окремих

випадках рішення можуть бути швидко знайдені, але в цілому здійснюється перебір *n*! циклів. Другі в загальних випадках застосовуються для задач, які неможливо вирішити точно (обчислення певних інтегралів, рішення нелінійних рівнянь, витяг квадратного кореня ...), якщо існуючі точні рішення вимагають значних і невиправданих витрат часу при високій складності задачі, і як частина більш складного алгоритму, за допомогою якого завдання вирішується точно. [12]

2.2 Точні алгоритми

У свою чергу існує дві групи точних алгоритмів - одна з них використовує методи релаксації лінійного програмування TSP : алгоритм Гомори, метод внутрішньої точки, метод гілок і меж. Друга, менша група, використовує методи динамічного програмування. Характерна особливість методів обох груп - гарантія знаходження оптимальних рішень при загальній трудомісткості процесу.

Повний перебір (Brute Force)

Один з найбільш очевидних методів вирішення задачі розрахунку оптимального маршруту є метод повного перебору або грубої сили. Його суть полягає в переборі всіх можливих варіантів шляхів, алгоритм вирішення можна записати як:

1. Визначити загальне число можливих гамільтонових контурів;
2. Визначити вагу кожного гамільтонова контуру, склавши вага всіх його ребер;
3. Вибрати гамильтонов контур з мінімальною вагою, який і буде оптимальним.

Метод повного перебору має низку переваг - він гарантує знаходження рішення задачі TSP, при цьому він прямолінійний і простий в виконанні. У той же час, алгоритм вважається неефективним при роботі з великим обсягом

даних, так як для знаходження оптимального маршруту вимагає знайти вага (*n* - 1)! гамільтонових контурів.

Метод гілок і меж (Branch and Bound)

Метод гілок і меж часто використовується для знаходження оптимального рішення задач комбінаторної оптимізації. Його суть полягає в розбитті множини на підзадачі і виключення неоптимальні рішень.

Нехай граф *V* містить всі міста, П - безліч всіх перестановок міст, що покриває всі можливі рішення. Розглянемо перестановку π ∈ П, в якій кожному місту призначається наступник - *i* для π*i* міста. Таким чином, тур можна записати як (1, π(1), π(π(1)), ..., 1). Якщо число міст в турі *n*, тоді перестановку називають циклічною. Завдання про призначеннях ставить перед собою мету знайти циклічні перестановки, а задача розрахунку оптимального маршруту переслідує ту ж мету, але з обмеженням, що у цих перестановок повинна бути мінімальна вартість.

Метод гілок і меж впершу чергу знаходить рішення задачі про призначення, вартість якої для *n* міст досить велика і асимптотично дорівнює *O* (*n*1). [15] Якщо був знайдений повний тур, то отримане значення також є рішенням завдання знаходження оптимального шляху. В іншому випадку проблема поділяється на декілька підгалузей, кожна з яких виключає деякі дуги підтура, таким чином виключаючи сам підтур. Метод, за допомогою якого вираховується, яку дугу слід видалити, називають правилом розгалуження. Важливе зауваження - не повинно існувати дубльованих підзадач, їх загальна кількість має бути мінімізованою. Кількість можливих рішень дорівнює (*n* - 1)! / 2, для n = 50 це приблизно 3 × 1062. Цей метод найбільш часто використовується при кількості вузлів від 40 до 60. [17] Необхідність цілком вирішувати завдання лінійного програмування у всій області допустимих рішень можна вважати головним недоліком вищеописаного методу. Для задач з великим об'ємом даних метод гілок і

кордонів є невиправдано трудомістким, в той же час алгоритм є надійним методом вирішення цілочислових задач.

Алгоритм Гомори (The Cutting Plane)

У 1954 році була представлена робота Данцига, Фалкерсона і Джонсона, описує новий метод розв'язання задачі пошуку оптимального маршруту, який також може бути використаний для вирішення будь-якої проблеми

$$minimize\ c^T x\ subject\ to\ x \in S$$

де $c \neq 0$, S - кінцеве підмножина деякого RU, і таким чином це ми зможемо знайти точки S. Це ітераційний алгоритм - кожне повторення починається з лінійної програмної релаксації. Запишемо у вигляді формули:

$$minimize\ c^T x\ subject\ to\ Ax \leq b$$

де багатогранник P, визначений як $\{ x : Ax \leq b \}$, містить S і обмежений. Так як P обмежений, ми можемо знайти оптимальне рішення x* як екстремальну точку P. Якщо x * належить S, то оптимальне рішення знайдено (2.2); в іншому випадку деякий лінійне нерівність задовольняє всі точки S і порушує x *. Така нерівність називають алгоритмом Гоморі, який докладно описав його 1958 методом відтинають площин або просто відсікань. [17]

Даний метод використовується для побудови точних або наближених задач, особливо часто зустрічається в поєднанні з методом гілок і меж і тоді називається методом гілок і відсікань. Обидва методи засновані на рішенні послідовності релаксувати підзадачі лінійного програмування. В алгоритмі Гоморі релаксовані під задачі поступово покращують апроксимацію цілих завдання, зменшуючи об'єм оптимального рішення. Якщо оптимальність не вдалося отримати, тоді шукається наближене рішення з похибкою. У методу відсікань є перевага над методом гілок і меж – перші більш зручні для апаратного обчислення, так як для їх рішення не потрібен великий обсяг оперативної пам'яті для зберігання дерева рішень.[18]

Метод динамічного програмування (Dynamic Programming)

Розглянемо задачу з n містами і відстанями d_{ij} між будь-якими двома містами, шлях починається і закінчується в місті n_0. TSP може бути вирішена за час $O(n!)$ методом повного перебору, але алгоритм динамічного програмування дозволяє скоротити час до $O(n^2 2^2)$.

Позначимо підзадачу: нехай S буде підмножиною міст, що містить 1 і як мінімум ще одне місто, j буде містом відмінним від 1, позначимо через C (S, j) найкоротший шлях, що починається в 1,18 проходить всі міста S і закінчується в j. Запишемо алгоритм у вигляді псевдо-коду.

for all j do $C(\{1, j\}, j) := d_{ij}$

for s: = 3 *do* (*the size of the subsets considered this round*)

for all subsets S of {1, ... *n*} *of size n and containing* 1 *do*

for all $j \in S, j \neq 1$ *do*

$C(S, j) := min_{i != j, i \in S} [C(S - \{j\}\ i + d_{ij})]$

$opt := min_{j != 1} [C(\{1,2..n\}, j) + d_{i1})]$ [19]

Даний метод використовується для підвищення ефективності обчислювальних повторень, зберігаючи проміжні результати і знову використовуючи їх при необхідності.

2.3 Неточні алгоритми

В цілому алгоритми даної групи пропонують потенційно неоптимальні, але швидкі рішення. У свою чергу наближені алгоритми можна розділити на дві категорії: наближені (Approximation Algorithms) і евристичні (Heuristic Algorithms).

Алгоритм Крістофідеса (Christofides 'Algorithm)

Алогорітм Крістофідеса використовується для вирішення метричних TSP -з додатковою умовою, що для матриці відстаней виконано нерівність трикутника:

$$\forall i, j, k \; d_{ik} \geqslant d_{ij} + d_{ik}$$

Велика частина евристичних алгоритмів належать до 2-наближеному класу. Професор Нікос Крістофайдс в 1976 році допрацював один з існуючих алгоритмів (метод подвійного мінімального остовного дерева, $O\,(n^2\ log_2\,(n))$) так, що час вирішення завдання не перевищує оптимальне час більш ніж на 3/2. [21]

Рішення оригінального алгоритму можна записати так: знайти мінімальне дерево з безлічі всіх міст, продублювати всі ребра і побудувати Ейлером граф, побудувати гамільтонів цикл, пройшовши кожен вузол тільки один раз і вибираючи найліпших шлях, що веде з кожного вузла.

Алгоритм Крістофідеса складається з послідовності наступних дій:

1. Визначити мінімальне дерево з безлічі всіх міст;
2. Визначити паросочетание з мінімальною вагою безлічі вершин
3. непарного степеня і побудувати Ейлером граф;

4. Визначити ейлерів обхід і побудувати гамільтонів цикл, уникаючивідвідуваних вузлів. [20]

Основна відмінність - додаткове обчислення паросполучення з мінімальною вагою. Ця частина також найбільш трудомістка, тому час виконання алгоритму зростає до O (n^3). Проведені тести показали, що алгоритм Крістофідеса на 10% вище нижньої межі Хелд-Карп. [21]

Алгоритм найближчого сусіда (NearestNeighbour)

Один з найпростіших евристичних методів рішення TSP. головне правило алгоритму - завжди вибирати сусіднє місто (сусіда). Рішення завдання складається з наступних кроків:

1. Вибрати будь-яке місто;
2. Визначити сусіднє місто, не включений в маршрут, і перейти в нього;
3. Перевірити чи залишилися міста, не включені в маршрут, якщо
4. Відповідь позитивна - повторити другий крок.
5. Щоб завершити тур додати ребро між останнім обраним містом і першим.

У загальному випадку трудомісткість рішення задачі дорівнює O (n^2). Нижня межа вартості оптимального маршруту на 10% вище нижньої межі Хелд–Карп. [22]

Жадібний алгоритм (Greedy)

Щоб вирішити TSP використання жадібний алгоритм, ми досліджуємо всі ребра, виходять з міста-вузла, і вибираємо n найкоротших дуг. Якщо ті n самих коротких дуг формують гамільтонов цикл, тоді ми знайшли оптимальне Рішення. [24]

Трудомісткість рішення задачі жадібним алгоритмом дорівнює O (n^2). Нижня межа вартості оптимального маршруту вище нижньої межі Хелд-Карп на 15-20%.

Алгоритм Керніган - Ліна (Lin-Kernighan)

Алгоритм Керніган - Ліна вважається одним із найбільш ефективних методів пошуку оптимальних або майже оптимальних рішень задачі побудови маршрутів. Однак розробка і реалізація алгоритму лише проста, так як алгоритм складається з безлічі кроків, більшість з яких сильно впливає на роботу алгоритму. [24] Створення алгоритму Керніган було натхненне наглядом, що статичне K в оптимальний метод не дає найкраще Рішення. З'явилася ідея використовувати різні стадії оптимальний методу у виконанні евристичного алгоритму. На практиці було показано, що практично неможливо заздалегідь передбачити яке K слід використовувати, щоб досягти кращого компромісу між трудомісткістю і якістю рішення. Лін і Керниган прибрали цей недолік, ввівши оптимальну змінну, таким чином значення K змінюється під час виконання алгоритму. [25] Трудомісткість при цьому дорівнює O ($n^{2.2}$). [21]

Алгоритм пошуку з заборонами (TabuSearch)

Головна проблема алгоритму найближчого сусіда полягає в частому застряванні в точці локального оптимуму. Цього можна уникнути, застосувавши алгоритм пошуку із заборонами, в 1977 році запропонований Ф. Гловером. Даний метод дозволяє переходити від одного локального оптимуму до іншого в пошуку глобального оптимуму, після переходу ребро потрапляє в список заборон і повторно не використовується, крім випадків, коли воно може поліпшити побудований оптимальний шлях. На практичному рівні заборонений набір зберігається як комбінація раніше відвідуваних кроків, який дозволяє побудувати подальший шлях щодо поточного рішення і сусідніх вузлів. [26]

Головним недоліком цього методу є його час виконання – трудомісткість алгоритму оцінюється як O (n^3). [21]

Мурашиний алгоритм (Ant Colony Optimization)

Мурашиний алгоритм – ефективний поліноміальних алгоритм, розроблений на базі поведінки справжніх мурах. Вперше його принципи були описані в 1991 Марко Дориго. Мурахам властиво співпрацювати в пошуках харчового ресурсу, тому вони залишають слід хімічної речовини, феромонів, на їх шляху від гнізда до джерела їжі. [25] Цей тип невербальної комунікації називають стігмергія – стимуляція, заснована на досвіді попередніх мурах і спрямована на підвищення продуктивності. [27]

Для вирішення завдання побудови оптимального маршруту як правило використовують близько 20 мурах. Їх розміщують у випадкові міста і відправляють в інші міста. Їм не дозволяють двічі відвідувати одине і теж місто, тільки якщо вони не завершують маршрут. Той мураха, який вибрав найкоротший тур, буде залишати слід феромонів обернено пропорційний довжині маршруту. Цей слід феромонів буде зчитуватися наступним мурахою при виборі міста, і з великою ймовірність він піде тим же шляхом, ще сильніше зміцнивши слід. Цей процес буде багаторазово повторений поки не буде знайдено маршрут, досить короткий, який вважатиметься оптимальним.

Серед недоліків алгоритму хочеться виділити, що перше отримане рішення може виявитися одним з найгірших в плані оптимізації, однак при повторному рішенні метод видає досить точний результат. [28]

Нижня межа Хелд-Карпа (The Held-Karp Lower Bound)

Найпоширеніший спосіб виміряти ефективність евристичного алгоритму для вирішення TSP - це порівняти результати з нижньою гранню Хелд-Карпа. Ця нижня межа є рішенням TSP, знайденим за поліноміальний час за допомогою симплекс–методу. Нижня грань Хелд-Карпа приблизно 0.8% нижче оптимальної тривалості туру. [29] У той же час вона гарантовано не перевищує оптимальний час більш ніж на 2/3. [21]

Станом на 2015 алгоритм Крістофідеса вважається самим ефективних методом для вирішення завдання побудови оптимального маршруту на загальних метричних просторах, хоча відомі кращі наближення для приватних випадків. Також добре в тестах себе показали алгоритм Керніган-Ліна і жадібний евристичний алгоритм. [21]

2.4 Directions API from Google Maps

Google Maps - це веб-сервіс, який надає детальну інформацію про географічні регіони і об'єктах по всьому світу. Сервіс дозволяє вибрати між стандартним дорожнім, супутниковим і гібридним режимами перегляду карт. У деяких містах також представлена функція Google Street View зі стереосферіческімі панорамами планувальник маршрутів пропонує маршрути для водіїв, велосипедистів, пішоходів і пересуваються громадським транспортом. Сервіс Google Maps для мобільних пристроїв надає можливість визначення місця розташування для автомобілістів, що використовують систему глобального позиціонування (GPS) місце розташування мобільного пристрою разом з даними з бездротових і стільникових мереж. Як бонус система також пропонує карти Місяця, Марса для любителів астрономії. Одною з найбільш значущих розробок було створення відкритого Google Maps API, який дозволяє розробникам використовувати карти для сторонніх сервісів і управляти ними за допомогою Java, JavaScript. Маршрути можна розраховувати (з використанням різноманітних видів транспорту) за допомогою об'єкта DirectionsService. Цей об'єкт взаємодіє зі службою Google Maps API Directions Service, яка отримує запити маршрутів і повертає розраховані результати. Повернуті маршрути можна обробляти самостійно або використовувати об'єкт DirectionsRenderer для їх відображення на карті.

Алгоритм побудови оптимального маршруту цього сервісу базується на принципі алгоритму найближчого сусіда, з певними модифікаціями від розробників Google.

2.5 Генетичний алгоритм

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 пристосовані до умов навколишнього середовища, мають більше шансів на виживання і продовження роду, і навпаки – непристосовані особини піддаються знищення. Основою відбору є мутації генів і їх комбінації, що формуються при розмножені 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 і дають потомство. Випадкові мутації також можуть впливати на розвиток популяції.

Ми можемо позначити завдання оптимізації як задачу знаходження функції $f(x_1, x_2,..., x_n)$, 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 $f(x\_1, x\_2,..., x\_n) \geq 0$, область є обмеженою. Параметри функції записується як рядки, що складаються з бітів. Рядок, отриманий конкатанацією рядків, називається

особиною: [18]

1010 10110 101.. . 10101

$$|x1|x2|\ x3|...|xn|$$

Генетичний алгоритм універсальний, так як тільки функція пристосованості та кодування рішень залежать від умов поставленої завдання. 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 алгоритму можна розбити на три етапи:

1. Пропорційний залежно від пристосованості відбір особин поточного покоління, які мають право виробляти потомство, і формування з них проміжної популяції;
2. 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.
3. Відбувається мутація отриманого покоління, що не допускає передчасної збіжності. Кожен біт особини з імовірністю не більше 1% записується як протилежну початковій. [18]

Як заздалегідь заданих критеріїв отримання оптимального рішеннях можуть бути певне число зміни поколінь або сходження популяції - умова виконується, якщо всі рядки є майже ідентичними і знаходяться в області екстремуму. Рішенням алгоритму буде особина з найбільшим значенням функції пристосованості.

При використанні генетичного алгоритму для вирішення завдання пошуку оптимального шляху виконуються всі перераховані вище кроки, пристосованість особини фактично служить мірою довжини маршруту. Існують кілька різних реалізацій класичного генетичного алгоритму в залежності від підходу до етапів схрещування і мутації. Наприклад : спочатку є n пунктів вони всі з'єднані між собою дорогами, тобто можна потрапити з одного в будь-який інший, у кожного свій маршрут, логічно уявити всі ці

дороги у вигляді масиву. У особини є набір хромосом - це одновимірний масив - він представляє собою послідовність обходу всіх міст, причому в результаті повинні повернутися в теж місто, з якого вийшли. наприклад – 5 міст, у особини масив 2-3-1-4-5-2 або 5-1-4-3-2-5 і т.д.,є покоління – в якому є k індивідів, у кожного з них є власний шлях обходу (можуть повторюватися). Початкові значення цього обходу заповнюються випадковим чином, але за правилами, наведеними вище. Далі ми організуємо зміну поколінь: істоти можуть розмножуватися і іноді мутувати. Мутувати мають не часто, як і в реальності, досить просто реалізується. Приклад: у істоти набір хромосом 2-3-1-4-5-2 випадковим чином вибираємо 2 міста і міняємо їх місцями – припустимо другий і четвертий – отримуємо у цієї істоти набір став 2-4-1-3-5-2. Для всіх істот в кожному поколінні вважаємо його пристосованість - в нашому випадку найбільш пристосований, це той, у кого довжина шляху обходу міст найменша. Більш пристосованих схрещуємо один з одним. Схрещування в генетичному алгоритмі зазвичай теж просто реалізується - беруться 2 істоти - їх набір хромосом розривається на 2 частини і однією частиною обмінюються один з одним, але стосовно до задачі комівояжера є деякі труднощі – ми не можемо так зробити, тому що у нас кожне місто обходиться тільки один раз, тобто номер одного міста може бути в масиві 2-3-1-4-5-2 тільки один раз (виключаючи крайні) тому треба застосовувати один з методів:

1. Частково відображувальне схрещення
2. Впорядковане схрещування
3. Циклічну схрещення

Деякі з них дали чудові показники при тестуванні, сильно перевершивши алгоритм Керніган-Ліна. Але також, як і у алгоритму пошуку з заборонами трудомісткість процесу створює проблему при досить великому кількості вузлів. [18] Дослідження, представлене в статті "Comparison of Algorithms for Solving Traveling Salesman Problem "[30] показує результати

порівняння трьох методів: алгоритму найближчого сусіда, генетичного і

жодібного алгоритмів. Як приклад можна скористуватися картою Сполучених Штатів Америки, областю 9.9 мільйонів км^2, Представлена на рисунку 2.2. Міста були обрані довільно.

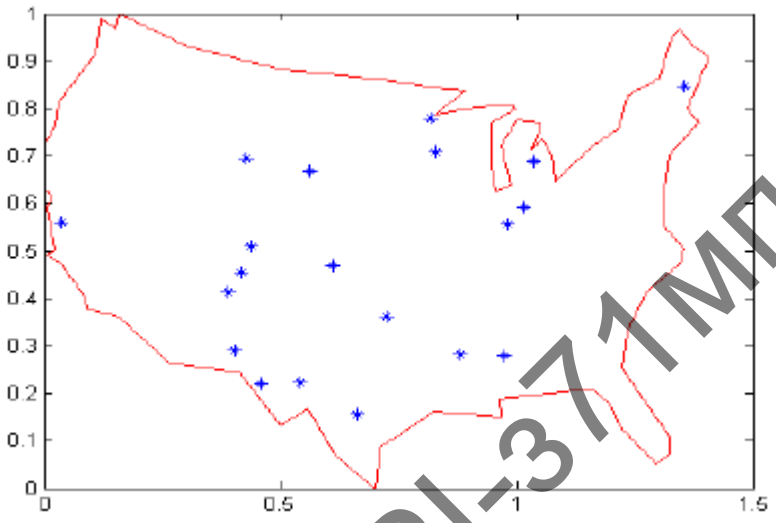

Рисунок 2.2 – Карта Сполучених Штатів Америки [30]

Оптимальне рішення показано на Рис. 2.3 із загальною довжиною маршруту 4616 км. Це рішення має високу складність і складається з найбільшого числа ітерацій

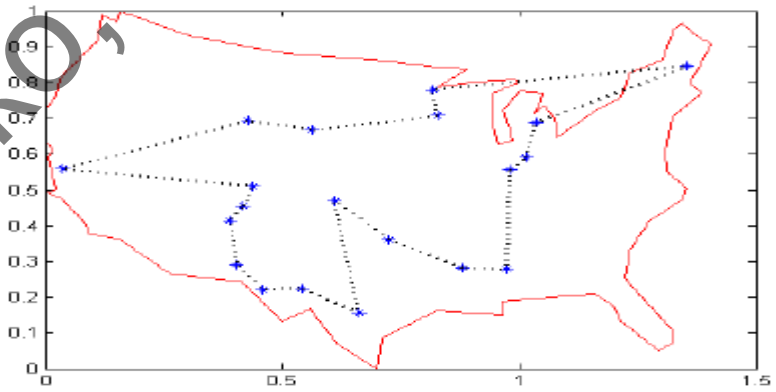

Рисунок 2.3 – Результат побудови маршруту [30]

Вирішуючи ту ж задачу методом найближчого сусіда, був отриманий маршрут, показаний на малюнку 2.4. Його довжина становить 15800км.

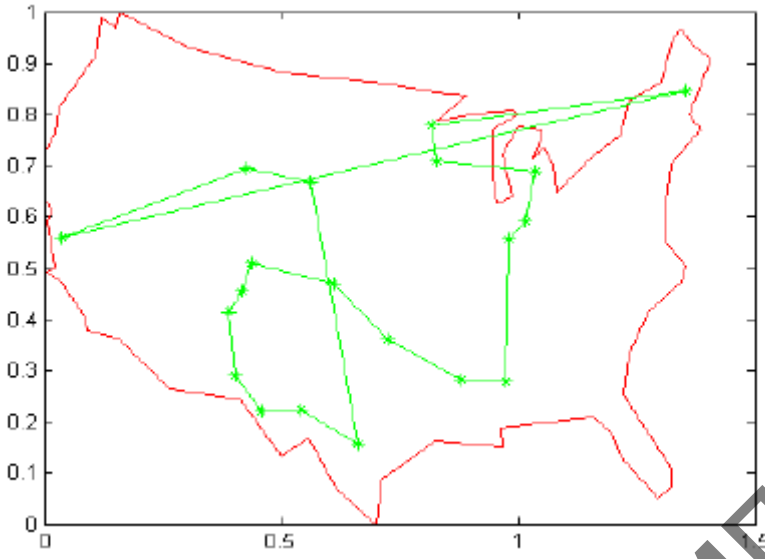

Рисунок 2.4 – Результат роботи алгоритму найближчого сусіда [30]

Рішення TSP для 20 міст з використанням генетичного алгоритму показано на рисунку 2.5. Хоча було скоєно більше ітерацій, ніж в попередньому прикладі, даний алгоритм знайшов більш оптимальний маршрут рівний 11900км

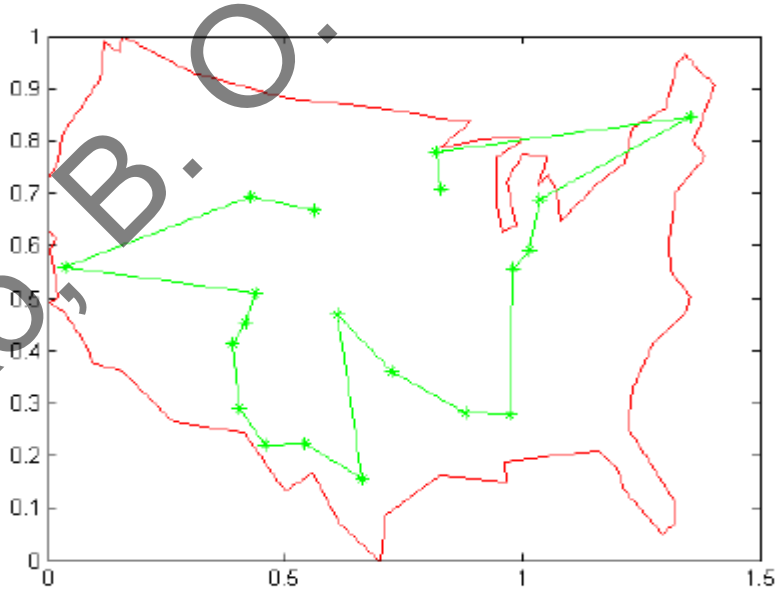

Рисунок 2.5 – Результат роботи генетичного алгоритму [30]

Такий самий приклад TSP було вирішено з використанням жадібного алгоритму. Результат показаний на рисунку 2.6. і має довжину 12900км. Хоча це рішення має незначно скорочення відстані в порівнянні з алгоритмом

найближчого сусіда, у нього все ж більш висока складність і час виконання. Генетичний алгоритм показав себе як самий ефективний метод вирішення TSP з невеликим обсягом даних, але в той же час як самий трудомісткий.

Подібні порівняння було наведено для задач великої розмірності. Результати дослідження, проведені на 2,2 GHz 8GB of 1600MHz DDR3L SDRAM Intel Core i7 Assus і включають в себе порівняння таких параметрів як довжина оптимально шляху в кілометрах, витрачений час в секундах і кількість ітерацій, представлені на Таблиці 3.1.

Порівняння роботи алгоритмів при 100 точках			
Обраний алгоритм	Довжина оптимального маршруту (км)	Час разрахунку(сек)	Кількість ітерацій
Алгоритм найближчого сусіда	26664	2.5	100
Генетичний алгоритм	225479	45	10000
Жадібний алгоритм	23311	0.07	18
Порівняння роботи алгоритмів при 1000 точках			
Алгоритм найближчого сусіда	83938	95.5	1000
Генетичний алгоритм	282866	468	10000
Жадібний алгоритм	72801	127	151

Таблиця 3.1 – Порівняння алгоритму найближчого сусіда, генетичного і жадібних алгоритмів при вирішенні задач з 100 і 1000 містами. [30]

Таблиця 3.1 вище наочно показує, що при кількості міст в задачі комівояжера близькому до 100, генетичний алгоритм програє жадібному за всіма параметрами. При аналізі TSP с 1000 вихідними містами генетичний алгоритм в корені неефективний. [30]

Підсумовуючи сказане вище, можна сказати, що головним недоліком генетичного алгоритму є відсутність гарантії, що отримане рішення є

оптимальний та часові показники на великих об’ємах даних не вражають. Але генетичні алгоритми показує високу ефективність в задачах з невеликою кількістю міст. У ситуаціях, коли в задачах великої розмірності відсутня упорядкованість вхідних даних, генетичний алгоритм є єдиною альтернативою алгоритму повного перебору. Головна його перевага – його можна використовувати для вирішення складних неформалізованих проблем для яких не існує приватних методів вирішення, що дозволяє йому ефективно вирішувати нестандартні завдання. [18]

2.6 Висновки по розділу 2

У розділі 2 було проведено аналіз методів вирішення TSP, представлено короткий опис алгоритмів, перераховані їхні переваги і недоліки. Більш детально був описаний генетичний алгоритм, також підведені підсумки порівняльного дослідження алгоритму з рядом інших. Але в рамках дипломної нам, потрібно будувати оптимальний маршрут з урахуванням пробок на дорозі, з можливістю вибору способу пересування, візуалізацією будь–якого елемента маршруту, тому прийнято рішення використовувати Google Maps APIs[39], а саме Directions API, який базується алгоритмі найближчого сусіда і перебору з певними модифікаціями від розробників компанії Google, зручність використання API в тому, що наявна можливість черпати потрібну інформацію.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

Третя глава описує використані для розробки програмні інструменти, також наводиться алгоритм, який будує оптимальний маршрут по заданих на мапі точкам. В кінці розділу дається аналіз роботи програми.

3.1. Огляд використовуваних в розробці програми інструментів

Додаток реалізовано, як на стороні клієнта так серверна частина. Нижче наведені інструменти, використовувані при написанні, такі як Java SE, JavaEE, фреймворки Spring Boot, Spring Security, Spring Core, Hibernate, база даних MySql, HTML5, CSS3, Bootstrap 3, JavaScript і бібліотека jQueary, Google Maps.

На сьогоднішній день компанія Google є передовою в усіх напрямках розвитку IT, вона надає можливість використовувати власні ресурси стороннім розробникам програмного забезпечення. Задачі розробки маршрутів переміщення також можуть вирішуватись на основі ресурсів Google. Необхідна інформація для побудови оптимального маршруту розміщена в Google Maps Api , а саме в Directions Api в поєднанні з Geocoding Api. За наявності токена, дані ресурси можливо використовувати безкоштовно по 1000 запитів на день, що достатньо для тестування програмного забезпечення.

Java, Spring

Для програмування обираємо мову Java, так я вона є найбільш популярною на IT ринку, використовується як в передових компаніях України, таких як EPAM, Luxoft, Infopulse , так і в таких світових лідерів, як Google. Для ефективного рішення поставленої задачі необхідно використовувати сучасні фреймворки, такі як Spring (рис.2.) , що надає додаткові можливості програмісту в розробці продукту, в поєднані з принципами ООП (об’єктно-орієнтованого програмування) таких як

поліморфізм, наслідування, абстракція, інкапсуляція, за невеликий проміжок часу можна створити дійсно сучасний, гнучкий і головне відкритий до модифікації проект, який далі буде легко супроводжувати .

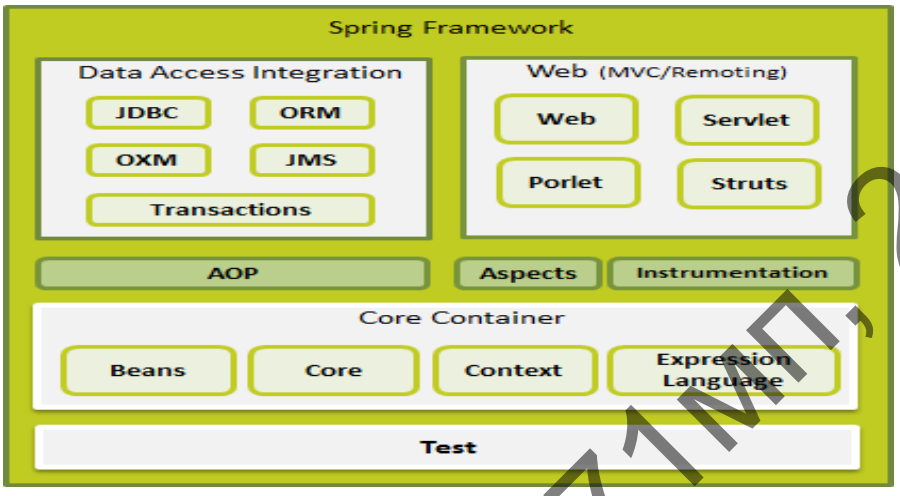

Рисунок.3.1 – Інструменти Spring Framework

Spring Security

Java забезпечує максимальну секюрність даних, на її основі можна захищати дані клієнтів з використанням різних підходів таких як наприклад, протокол авторизації Spring Security (рис. 3.1)

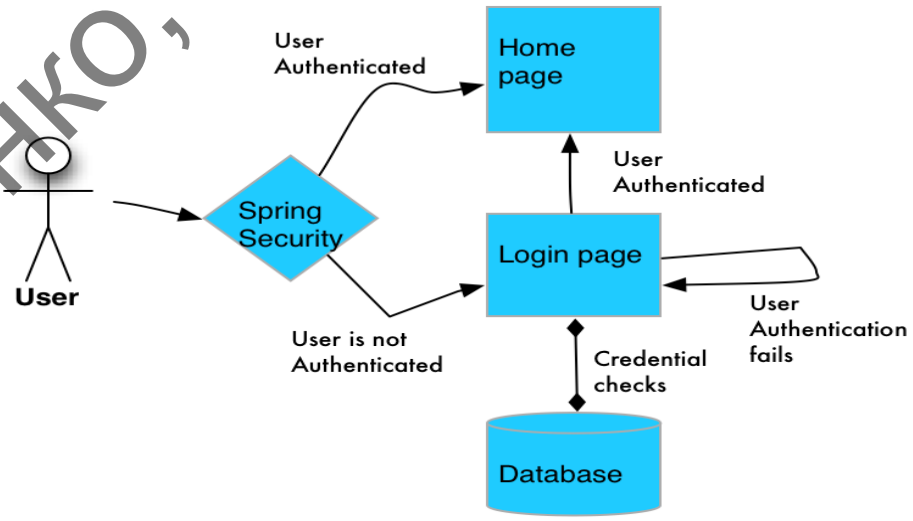

Рисунок 3.1 – Протокол авторизації Spring Security

Spring Security це Java / JavaEE framework, що надає механізми побудови систем аутентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework. Проект був розпочатий Беном Алексом (Ben Alex) в кінці 2003 року під ім'ям «Acegi Security», перший реліз вийшов в 2004 році. Згодом проект був поглинений Spring'ом і став його офіційним дочірнім проектом. Вперше публічно представлений під новим ім'ям Spring Security 2.0.0 в квітні 2008 року.

MySql

Обираємо базу даних для зберігання інформації про клієнта . То може ути sql чи NoSql бази : MySql, PostreSql, Elastic, Mongo. В кожної з них є свої недоліки та переваги. В базі даних NoSql швидкість пошуку певних даних значно вище ніж в SQL базі, до того ж в NoSql базі дані зберігають в форматі JSON, який є досить зручним для обробки. З урахуванням того, що на сервері ми будемо зберігати лише дані про клієнта то оберемо найбільш просту Sql базу даних – це MySql , для рішення задачі дипломної роботи її стабільність, функціонал і швидкість роботи цілком задовольняє вимоги ТЗ.

ORM, Hibernate

ORM (Object-relational mapping) – це відображення об'єктів будь-якої об'єктно-орієнтованої мови в структури реляційних баз даних. Саме об'єктів, таких, якими вони є, з усіма полями, значеннями.

ORM-рішенням для мови Java, є технологія Hibernate, яка не тільки дбає про зв'язок Java класів з таблицями бази даних і типів даних Java в типи даних SQL, але також надає можливість для автоматичної побудови запитів і отримання даних і може значно зменшити час розробки, який зазвичай витрачається на ручне написання SQL і JDBC коду. Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Розробник може використовувати Hibernate як при

розробці з нуля, так і для вже існуючої бази даних. Hibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних. Hibernate забезпечує прозору підтримку збереження даних, тобто їхньої персистентності (англ. persistence) для «POJO»-об'єктів, себто для звичайних Java-об'єктів; єдина сувора вимога до класу, що зберігається — конструктор за замовчанням (Для коректної поведінки у деяких застосуваннях потрібно приділити особливу увагу до методів equals() і hashCode().

Mapping (зіставлення, буквально — картування) Java класів з таблицями бази даних здійснюється за допомогою конфігураційних XML файлів або Java анотацій. При використанні файлу XML, Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання (persistent). У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних. Забезпечуються можливості з організації відношення між класами «один-до-багатьох» і «багато-до-багатьох». На додаток до управління зв'язками між об'єктами, Hibernate також може керувати рефлексивними асоціаціями, де об'єкт має зв'язок «один-до-багатьох» з іншими примірниками свого власного типу даних. Hibernate підтримує відображення користувацьких типів значень. Це робить можливим такі сценарії: Перевизначення типу за замовчуванням SQL, який Hibernate вибирає при відображенні стовпчика властивості. Картування перераховуваного типу Java до колонок БД, так ніби вони є звичайними властивостями. Картування однієї властивості в декілька колонок. Hibernate забезпечує прозоре збереження POJO (Plain Old Java Objects — простих старих об'єктів Java). Єдина сувора вимога для

персистентного класу — конструктор без аргументів, не обов'язково публічний. Для правильної поведінки деяких програм також потрібна особлива увага до методів equals() і hashCode().[1] Колекції об'єктів даних, як правило, зберігаються у вигляді колекцій Java-об'єктів, таких як набір (Set) і список (List). Підтримуються узагальнені класи (Generics), введені в Java 5. Hibernate може бути налаштований на «ледачі» (відкладені) завантаження колекцій. Відкладені завантаження є варіантом за замовчуванням, починаючи з Hibernate 3. Зв'язані об'єкти можуть бути налаштовані на каскадні операції. Наприклад, батьківський клас, Album (музичний альбом), може бути налаштований на каскадне збереження і/або видалення свого нащадка Track. Це може скоротити час розробки і забезпечити цілісність. Функція перевірки зміни даних (dirty checking) дозволяє уникнути непотрібного запису дій в базу даних, виконуючи SQL оновлення тільки при зміні полів персистентних об'єктів.

Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних і конвертації об'єктів, зберігаючи програму актуальною для роботи з будь–якою SQL базою даних, отже якщо в майбутньому виникне потреба використання більш потужної бази даних, то це не спричинить проблем і досить легко і швидко реалізується.

GSON

Дані з інформацією від Google Maps будемо отримувати в форматі Json. Json на сьогоднішній день – загальноприйнятий стандарт обміну даними між сервером і веб-додатком, і, що часто буває з багатьма стандартами, ми можемо прийняти його по дефолту, не заглиблюючись в принципи його роботи. Часто ми не замислюємося про те, які бібліотеки використовуємо для роботи з JSON, хоча, безумовно, між ними є різниця.

Вибір бібліотеки виходячи з швидкості роботи залежить від конкретного випадку: якщо треба часто працювати з великими JSON-файлами, то що

кращою бібліотекою буде Jackson. GSON відчуває найбільші труднощі при роботі з великими файлами. Якщо доводиться мати справу з великою кількістю коротких JSON-запитів, як це часто відбувається в сервісах і розподілених додатках, рекомендують обирати GSON, Jackson справляється з великою кількістю маленьких файлів найгірше. Якщо необхідно працювати як з великими, так і з маленькими файлами, JSON.simple один з кращих : за результатами тестів він займає друге місце для обох типів файлів. Ні Jackson, ні GSON не можуть з належною швидкістю працювати одночасно з різними за розміром файлами. Коли справа стосується швидкості, JSONP навряд чи можна порекомендувати для будь-якого випадку – він працює повільно в порівнянні з іншими доступними бібліотеками, як з великими, так і з маленькими файлами. В рамках дипломної робити від Google Maps Api ми отримуємо велику кількість Json файлів з інформацією про маршрути та певні геолокації, отже обираємо GSON.

JavaScript і JQuery

Розробляючи додаток на стороні клієнта в якості провідної мови було обрано JavaScript динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу. Мова JavaScript

використовується для: написання сценаріїв веб-сторінок для надання їм інтерактивності; створення односторінкових веб-застосунків (ReactJS, AngularJS, Vue.js); програмування на стороні сервера (Node.js); стаціонарних застосунків (Electron, NW.js); мобільних застосунків (React Native, Cordova); сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter); всередині PDF-документів тощо. Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme. [5]. На ньому написано динамічне відображення отриманого оптимального маршруту безпосередньо на самій карті після того як користувач вибере точки маршруту. JavaScript (JS) - об'єктно-орієнтована мова сценаріїв веб-сторінок. Код, написаний на JavaScript, вбудовується в вихідну HTML код і зчитується браузером в міру завантаження веб-сторінки. З його допомогою можна динамічно змінювати як HTML так і CSS код сторінки в залежності від дій відвідувачів сайту або програми. Крім чистого JS в додатку використовується бібліотека jQuery, спрощує взаємодію з деревом DOM - об'єктної моделі HTML-документів.

3.2 Логіка роботи додатку

Додаток являє собою декілька HTML сторінок, з яких клієнт передає інформацію до бек-енду використовуючи JavaScript, його бібліотеку JQuery і технологію Ajax, де за допомогою Java виконується вся логіка, обробка інформації і взаємодія з Directions, Geolocations API від Google Maps .

Перша секція є системою авторизації клієнта пропонується ввести логін і пароль(Рис 3.2) .

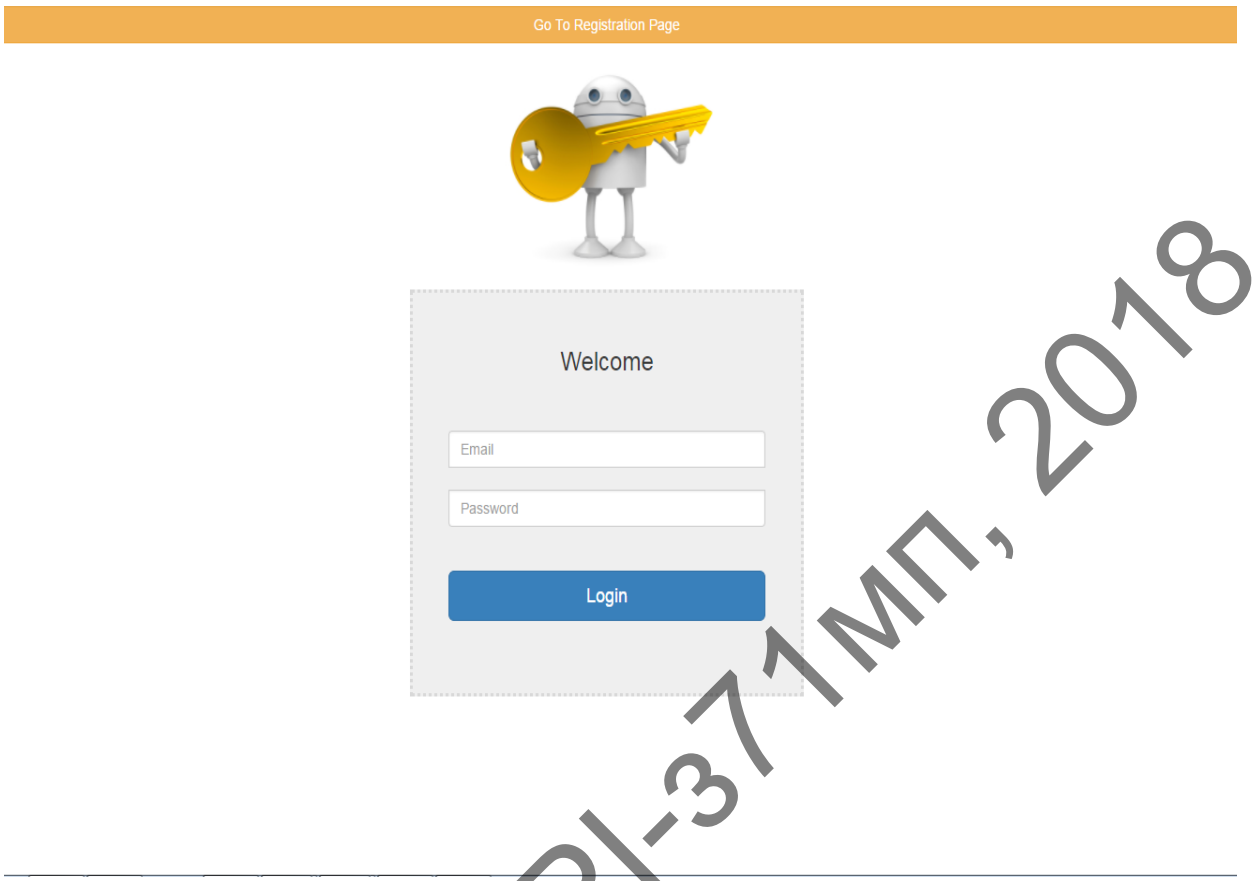

Рисунок 3.2 – Сторінка авторизації клієнта

Якщо це новий клієнт і не зареєстрований в базі, то є можливість перейти до сторінки з реєстрацією нового клієнта (Рис 3.3).

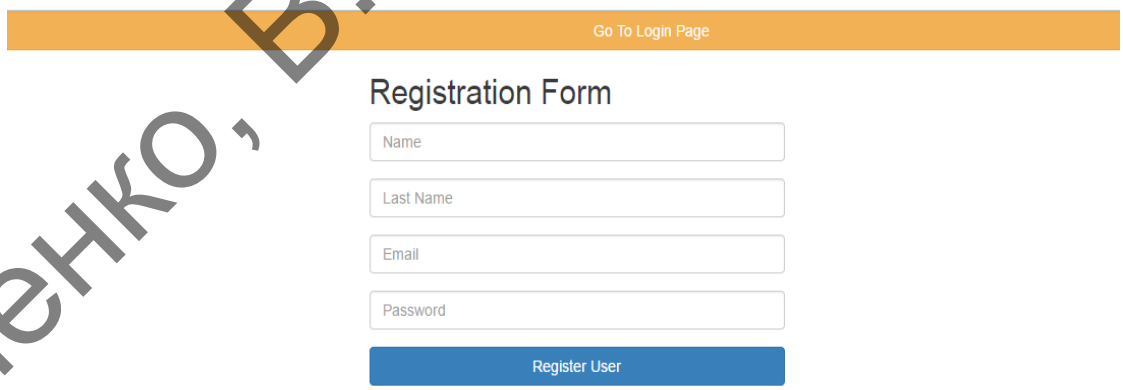

Рисунок 3.3 – Сторінка реєстрації нового клієнта

Реалізація логіки процесу авторизації клієнта зі сторони сервера

Ключові об'єкти контексту Spring Security:

1. SecurityContextHolder, в ньому міститься інформація про поточний контекст безпеки програми, який включає в себе детальну інформацію про користувача Principal працює в даний час з додатком. За замовчуванням SecurityContextHolder використовує ThreadLocal для зберігання інформації, що означає, що контекст безпеки завжди доступний для методів, що виконуються в тому ж самому потоці. Для того що б змінити стратегію зберігання цієї інформації можна скористатися статичним методом класу SecurityContextHolder.setStrategyName (String strategy).
2. SecurityContext, містить об'єкт Authentication і в разі необхідності інформацію системи безпеки, пов'язану із запитом від користувача.
3. Authentication представляє користувача Principal з точки зору Spring Security.
4. GrantedAuthority відображає можливість видачі користувачу в масштабі всього проекту, такі дозволи (як правило називаються «ролі»), наприклад : ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN.
5. UserDetails надає необхідну інформацію для побудови об'єкта Authentication з DAO об'єкті або інших джерел даних системи безпеки. Об'єкт UserDetail включає ім'я користувача, пароль, прапори: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled і Collection – прав (ролей) користувача(Рис 3.4) .
6. UserService, використовується щоб створити UserDetails об'єкт шляхом реалізації єдиного методу цього інтерфейсу, який дозволяє отримати з джерела даних об'єкт користувача та сформувати з нього об'єкт UserDetails(Рис 3.5), який буде використовуватися контекстом Spring Security .


```
@Entity
@Table(name = "user")
public class User {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   @Column(name = "user_id")
   private int id;
   @Column(name = "email")
```

```
    @Email(message = "*Please provide a valid Email")
    @NotEmpty(message = "*Please provide an email")
    private String email;
    @Column(name = "password")
    @Length(min = 5, message = "*Your password must have at least 5 characters")
    @NotEmpty(message = "*Please provide your password")
    private String password;
    @Column(name = "name")
    @NotEmpty(message = "*Please provide your name")
    private String name;
    @Column(name = "last_name")
    @NotEmpty(message = "*Please provide your last name")
    private String lastName;
    @Column(name = "active")
    private int active;
    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"),
inverseJoinColumns =    @JoinColumn(name = "role_id"))
    private Set<Role> roles;
  }
```

Рисунок 3.4 – Код Java об’єкту User

```
@Service("userService")
public class UserServiceImpl implements UserService {

    @Autowired
    private UserRepository userRepository;
    @Autowired
    private RoleRepository roleRepository;
    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @Override
    public User findUserByEmail(String email) {
        return userRepository.findByEmail(email);
    }

    @Override
    public void saveUser(User user) {
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        user.setActive(1);
        Role userRole = roleRepository.findByRole("ADMIN");
        user.setRoles(new HashSet<Role>(Arrays.asList(userRole)));
        userRepository.save(user);
    }
}
```

Рисунок 3.5 – Реалізація UserService

Процесс аутентифікації та авторизації

1. Користувачеві буде запропоновано увійти в систему надавши ім'я (логін або email) і пароль. Ім'я користувача і пароль об'єднуються в екземпляр класу UsernamePasswordAuthenticationToken (екземпляр інтерфейсу

Authentication) після чого він передається екземпляруAuthenticationManager для перевірки.

2. У випадку якщо пароль не збігається з назвою користувача буде викинута помилка BadCredentialsException з повідомленням "Bad Credentials".
3. Якщо аутентифікація пройшла успішно повертає повністю заповнений екземпляр Authentication.
4. Для користувача встановлюється контекст безпеки шляхом виклику методу SecurityContextHolder.getContext (). SetAuthentication (...), куди передається об'єкт який повернув AuthenticationManager.

Після вдалої реєстрації чи авторизації клієнту надається можливість використовувати ресурс з певною роллю, на даний момент додаток є повністю безкоштовним, оскільки це тестова версія, тому клієнту надається одна роль User, в майбутньому в процесі розвитку сервісу можливо створити платні ролі, які надаватимуть клієнту можливість використовувати унікальні можливості додатку, в тестовій версії реалізовано лише одна функція «Distance», яка розраховує дистанції між заданими точками, час і будує динамічний маршрут з урахуванням типу поїздки, можливістю корегування кліком по карті та візуалізацією будь-якої точки маршруту.(Рис 3.6). В майбутньому планується розробка деякого контенту, який дозволить вимірювати висоту гір, чи будь-якої іншої висотки відносно положення положення вільної поверхні Світового океану, вимірювання глибини будь-якої точки водяного ресурсу простим кліком по карті.

Рисунок 3.6 – Домашня сторінка, з переліком можливих функцій проекту

Після того, як клієнт обирає функцію «Distance», пропонується ввести початкову точку, кінцеву, проміжні точки і тип бажаної подорожі, то може бути автомобільна, велосипедна, піша чи з використанням транспорту загального призначення поїздка для динамічної побудови маршруту (Рис. 3.7).

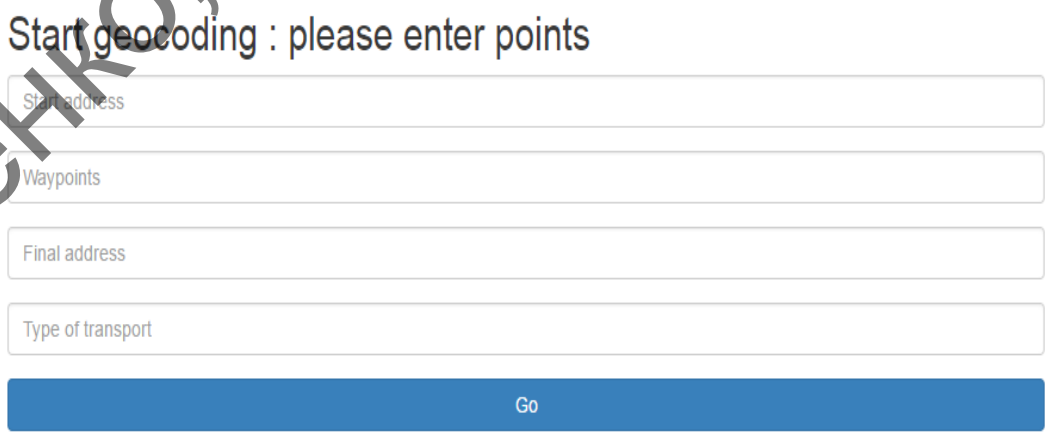

Рисунок 3.7 – Форма для передачі даних серверу

Дані з цієї форми в текстовому вигляді передаються на бек-енд, де за допомогою Java та можливостей Google Maps, а саме Geocoding Api відбувається процес геокодування даних – призначення об'єкту карти (заданому, зазвичай, поштовою адресою або унікальною назвою) певного універсального географічного ідентифікатора наприклад : географічні координати на земній кулі - широта і довгота.(Рис 3.8)

```
public Map<String, Double> getCoordinate(String[] nameOfPoint) {
   GeoApiContext context = new GeoApiContext().setApiKey(appConfig.getApiKey());
   GeocodingResult[] results = null;
   StringBuilder builder = new StringBuilder("");
   for (String item: nameOfPoint) {
      if(item != null && !item.isEmpty()){
         builder.append(item);
      }
   }
   try {   results =
GeocodingApi.newRequest(context).address(String.valueOf(builder)).await();
   } catch (Exception e) {
      e.printStackTrace();
   }
   HashMap<String,Double> location = new HashMap<>();
   Double lng = null;
   Double lat = null;
   if (results.length != 0) {
      Geometry geometry = results[0].geometry;
      lng =  geometry.location.lng;
      lat = geometry.location.lat;
   }
   location.put("lng", lng);  location.put("lat", lat);
return location;}
```

Рисунок 3.8 – Геокодування даних

З боку сервера готується POST запит по протоколу передачі даних HTTP, в параметрах якого вказуємо деяку географічну одиницю до ресурсу Geocoding Api, в процесі геокодування отримуємо значення довготи і широти, які записуємо до Java Collection, в вигляді пар ключ – значення.

Після за допомогою технології Ajax і бібліотек JavaScript отримуємо дані значення широти і довготи на клієнті та позначаємо їх маркерами на карті (Рис 3.9).

```
function makeRequest (method, url, param) {
   return new Promise(function (resolve, reject) {
      var xhr = new XMLHttpRequest();
      var params = JSON.stringify({firstPoint:[param]});
      xhr.open(method, url);
      xhr.setRequestHeader("Content-type", "application/json");
```

```
        xhr.onload = function () {
            if (this.status >= 200 && this.status < 300) {
                resolve(xhr.response);
            } else {
                reject({
                    status: this.status,
                    statusText: xhr.statusText
                });
            }
        };
        xhr.onerror = function () {
            reject({
                status: this.status,
                statusText: xhr.statusText
            });
        };
        xhr.send(params);
    });
}
Promise.all([makeRequest("POST", "/getCoordinate", firstPoint),
    makeRequest("POST", "/getCoordinate", secondPoint)]).then(function (result) {
    var arrayResult = result.map(function (item) {
        return JSON.parse(item);
    })
    var from = arrayResult[0];
    var to = arrayResult[1];
    var map = new google.maps.Map(document.getElementById('map'), {
        zoom: 15,
        center: from
    });
    var marker1 = new google.maps.Marker({
        position:from,
        map: map
    });
```

Рисунок 3.9 – Код відображення поточних точок на карті

В результаті отримуємо карту з точками, які пройшли процес геокодування і в точності відповідають запиту клієнта.(Рис 3.10)

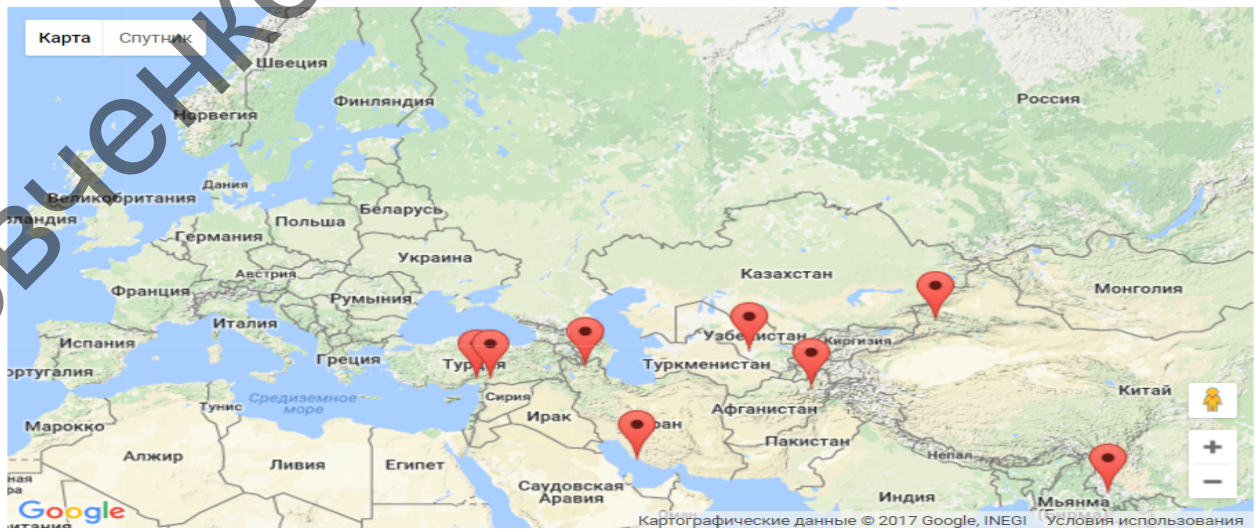

Рисунок 3.10 – Карта з маркерами по запиту клієнта

Далі для побудови маршруту слід розрахувати відстані між точками, використовуючи Directions API, готове рішення від Google. Суть його роботи заключається в тому, що він використовує метод повного перебору :

1. Ініціалізується точка початку маршруту і кінцева точка.
2. Розраховується відстань від першої точки, до кожної наступної.
3. Обирається найближчий сусід.
4. Проходить аналіз ситуації на дорозі, якщо це автомобільна подорож, для цього виконуються запити на інші Google Maps сервіси,які містять подібну інформацію. Якщо пішохідна то аналізується чи наявні пішохідні переходи, якщо велосипедна то наявність велодоріжок, для інвалідів наявність інвалідних спусків і т.д. з урахуванням всих цих критеріїв обирається оптимальний маршрут.
5. Далі ці ж дії повторюються з точки найближчого сусіда.

Код для розрахунку відстані між двома сусідніми точками :

```
public Map<String, String> distanceMatrixSample(List<String> startPoint, List<String>
finalPoint, String mode) throws IOException, JSONException {

   params.put("sensor", "false");// указывает, исходит ли запрос на геокодирование от
устройства с датчиком
   params.put("language", appConfig.getLang());// язык данных

   switch (mode){
      case "walking": params.put("mode", "walking");
      break;

      case "driving": params.put("mode", "driving");
      break;

      case "bicycling": params.put("mode", "bicycling");
      break;
   }

  //  params.put("mode", "walking");// идем пешком, может быть driving, walking, bicycling
и т.д.
   // адрес или координаты отправных пунктов
   params.put("origins", Joiner.on('|').join(startPoint));
   // адрес или координаты пунктов назначения
   // в запросе адреса должны раделяться символом '|'
   params.put("destinations", Joiner.on('|').join(finalPoint));
   final String url = appConfig.getDistanceUrl() + '?' + encodeParams(params);//
генерируем путь с параметрами
   log.info(url); // Можем проверить что вернет этот путь в браузере
   final JSONObject response = JsonReader.read(url);// делаем запрос к вебсервису и
получаем от него ответ
   final JSONObject location = response.getJSONArray("rows").getJSONObject(0);
   final JSONArray arrays = location.getJSONArray("elements");// Здесь лежат все
рассчитаные значения
```

```
    // Ищем путь на который мы потратим минимум времени
    final JSONObject result = Ordering.from(new Comparator<JSONObject>() {
        @Override
        public int compare(final JSONObject o1, final JSONObject o2) {
            final Integer duration1 = getDurationValue(o1);
            final Integer duration2 = getDurationValue(o2);
            return duration1.compareTo(duration2);// Сравниваем по времени в пути
        }
        /**
         * Возвращает время в пути
         *
         * @param obj
         * @return
         */
        private int getDurationValue(final JSONObject obj) {
            try {
                return obj.getJSONObject("duration").getInt("value");
            } catch (final JSONException e) {
                throw new RuntimeException(e);
            }
        }
    }).min(new AbstractIterator<JSONObject>() {// К сожалению JSONArray нельзя
итерировать, по этому обернем его
        private int index = 0;

        @Override
        protected JSONObject computeNext() {
            try {
                JSONObject result;
                if (index < arrays.length()) {
                    final String destionation = finalPoint.get(index);
                    result = arrays.getJSONObject(index++);
                    result.put("address", destionation);// Добавим сразу в структуру и адрес,
потому как его нет в
                    // этом рассчете
                } else {
                    result = endOfData();
                }
                return result;
            } catch (final JSONException e) {
                throw new RuntimeException(e);
            }
        }
    });
    final String distance = result.getJSONObject("distance").getString("text");// расстояние в
километрах
    final String duration = result.getJSONObject("duration").getString("text");// время в пути
    final String address = result.getString("address");// адрес
    log.info(address + "\n" + distance + "\n" + duration);
    //System.out.println(address + "\n" + distance + "\n" + duration);
    Map<String, String> geocodingResult = new HashMap<>();

    geocodingResult.put("distance", distance);
    geocodingResult.put("duration", duration);
    geocodingResult.put("address", address);

    return geocodingResult;
}
```

Після чого результати розрахунків отримуємо на клієнті і за допомогою DirectionsService відображуємо маршрут на карті :

```
var directionsService = new google.maps.DirectionsService;
var directionsDisplay = new google.maps.DirectionsRenderer({
   draggable: true,
   map: map,
   panel: document.getElementById('right-panel')
});
   directionsService.route({
   origin: from,
   destination: to,
   waypoints: location
   travelMode: mode.toUpperCase()
}, function(response, status) {
   if (status === 'OK') {
      directionsDisplay.setDirections(response);
   } else {
      window.alert('Directions request failed due to ' + status);
   }
}
```

В результаті маємо карту з побудованим маршрутом, на якій можемо динамічно змінювати маршрут кліком по карті (Рис 3.11, Рис 3.12)

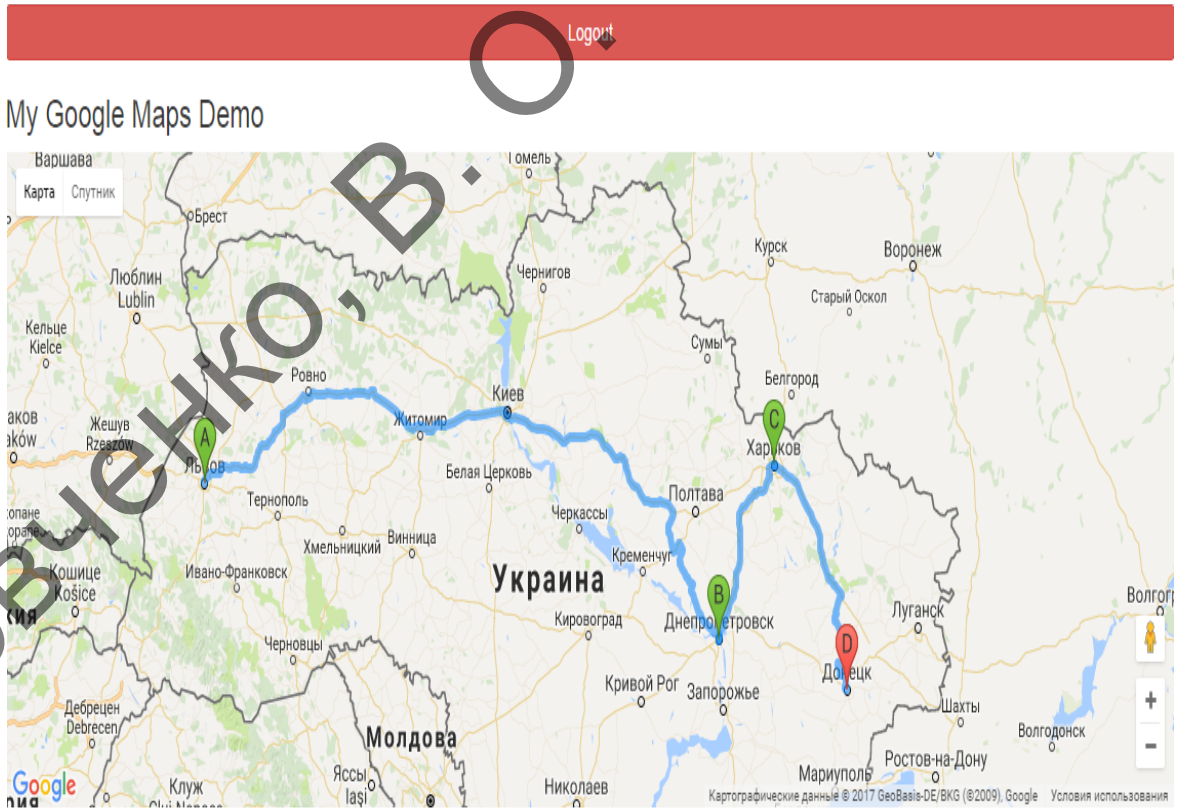

Рисунок 3.11 – Результат роботи сервісу

Нижче розташована панель з детальною інформацією по маршруту. Відстань 1 032 км. і розрахований час 12 год. 53 хв. (Рис 3.13).

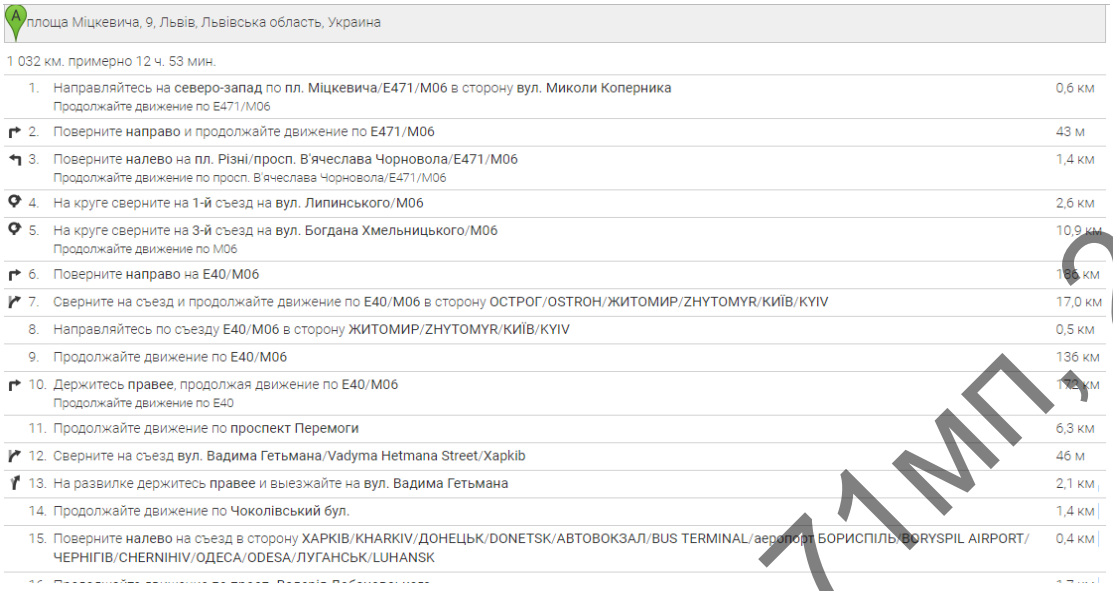

Рисунок 3.13 – Інформація по пересування наданому маршруті

Присутня можливість динамічно змінити маршрут, прямо на карті, наприклад якщо клієнт захоче поїхати не через Київ, а через Білу Церкву по якійсь із причин. Розрахується новий час маршруту, сформується нові підказки по пересування вздовж маршруту і розрахується нова відстань. (Рис.3.12). Відстань 1 042 км, час 14год 13хв .

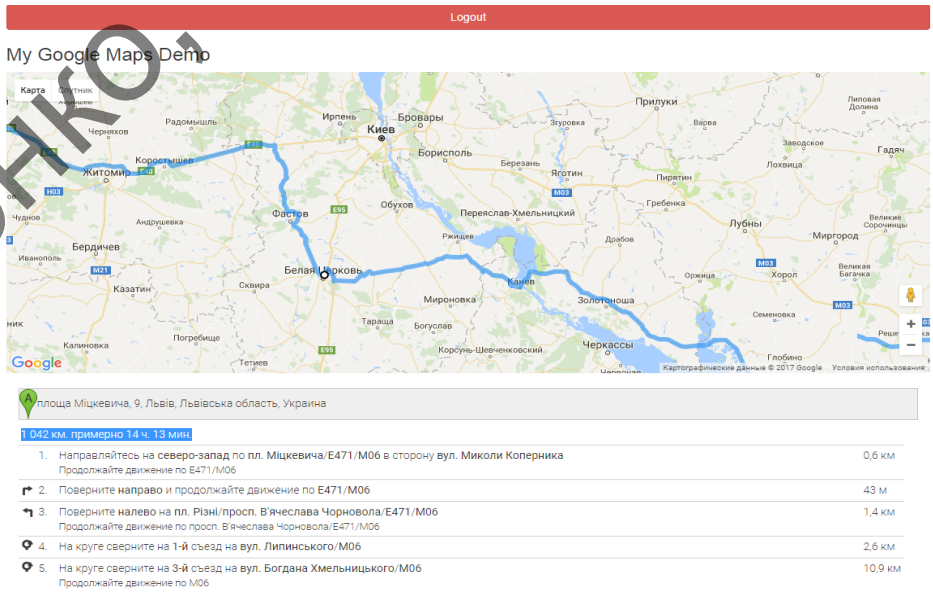

Рисунок 3.12 – Перебудований маршрут

Далі можливо візуалізувати будь-яку частину маршруту, покрути, продивись деякі частини маршруту.(Рис 3.14)

Рисунок 3.14 – Візуалізація окремої ділянки маршруту

3.3 Аналіз отриманих даних та тестування додатку

Для підвищення ефективної роботи веб-сервісу провелось дослідження, метою якого було побудова оптимального маршруту з n-ою кількістю проміжних точок, що гарантують найменшу довжину маршруту при мінімальному витрачений на пошук маршруту часу в секундах. Додаток було протестовано на 2,2 GHz Intel Core i7 Assus при різних значеннях довготи і широти початкової, кінцевою та проміжних точок. Таблиця 3.1 показує обрані результати тестування при різних параметрах.

Результати тестування при різних параметрах				
Кількість точок	Час проходження маршруту	Довжина маршруту	Вид поїздки	Розрахунковий час в мілісекундах
3	10 год. 29 хв.	668 км.	Автомобільна	5259.415
3	131 год.	668 км.	Пішохідна	5159.695
3	7 год. 51 хв.	668 км.	З використанням транспорту загального призначення	5459.193
4	1 год. 14 хв.	5.8 км.	Пішохідна	6459.292
4	27 хв.	6 км.	Велосипедна	9453.123
4	15 хв	7.5 км.	Автомобільна	6451.313
5	17 год. 32 хв.	1073 км.	Автомобільна	8418.993
5	213 год.	1051 км.	Пішохідна	8414.593
5	22 год. 11 хв.	1121 км.	З використанням транспорту загального призначення	8851.151
9	1 год. 53 хв.	7 км.	Пішохідна	8414.593
9	30 хв.	7.4 км.	Велосипедна	9414.593
9	14 хв.	8.6 км.	Автомобільна	8414.593

Таблиця 3.1 – Результати тестування при різних параметрах.

Наведені вище дані показують, що веб додаток функціонує досить добре не залежно від кількості вузлів і довжини маршруту, але помітно збільшують розрахунковий час велосипедні поїздки на території України і можливо їх будувати лиши в деяких містах України .

3.4 Висновок по розділу 3

У розділі 3 були перераховані інструменти, за допомогою яких розроблявся додаток (HTML5, CSS3, Spring, Spring Security, Hibernate, MySql, Bootstrap, JavaScript, jQueary, Google Maps, Directions, Geocoding) і обгрунтовано їх присутність. Наводиться опис взаємодії з додатком зі сторони користувача. Протестовано побудову оптимального маршруту, детально розглянуто реалізацію серверної частини і клієнта, розглянутий вектор розвитку проекту.

РОЗДІЛ 4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Даний розділ має на меті проведення маркетингового аналізу стартап проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

4.1 Опис ідеї проекту

В межах цього підрозділу аналізується зміст ідеї, можливі напрямки застосування, основі вигоди які може отримати користувач товару та відмінності від існуючих аналогів та замінників.

Таблиця 5.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка мережевого додатку для оптимізації маршруту на основі геокодування даних	Виробництво	Збільшення надійності, продуктивновності, швидкості побудови маршрутів
	Наука	

Основним конкурентом розроблюваному проекту є карти Google Maps який дозволяє моделювати маршрут . Пакет Google Maps дозволяє виконувати ті ж речі, що і розроблюваний пакет, але в ньому більш складні вимоги до вихідних даних та відсутність легкої інтеграції в проєкт та можливості змінювати дизайн чи модифікувати пакет функціональності.

Таблиця 5.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	Товари конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент			
1	Простота					✔
2	Дешевизна					✔
3	Швидкодія					✔

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проводиться аудит технології, за допомогою якої можна реалізувати ідею проекту.

Для реалізації цього проекту потрібно вибрати мову програмування та базу даних. Оглянуто три варіанта:

- Oracle PL SQL — база даних компактний багатопотоковий сервер баз даних, характеризується високою швидкістю, стійкістю і простотою використання.
- Мова програмування Java — мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної.
- Мова програмування JavaScript — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценарії ввеб-сторінок, що надає можливість на стороні клієнта(пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній виглядвеб-сторінки.

Таблиця 5.3 — Технологічна здійсненність проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технології
1	Розробка мережевого додатку для оптимізації маршруту на основі геокодування даних	PL SQL	Так	Ні
2		Java	Так	Так
3		JavaScript	Так	Так

Обрана технологія реалізації ідеї проекту:Java, JavaScript

Даний проект можливо реалізувати за допомогою Java для сервера та JavaScript для клієнта, потрібно використовувати всі переваги Java на сервері, такі як швидкість, надійність, безпека і наявніть технологічних фремфорків,

аналогічно JavaScript найкраще рішення для браузера та клієнтської розробки.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

В межах даного підрозділу проводиться визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту. Визначення ринкових можливостей дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Таблиця 5.4 — Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, ум. од.	Невідомий
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Невідома
5	Специфічні вимоги до стандартизації та сертифікації	Існують
6	Середня норма рентабельності в галузі, %	Невідома

За результатами аналізу важно зробити висновок щодо привабливості для входження за попереднім оцінюванням.

Визначимо потенційні групи клієнтів.

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Розробка мережевого додатку для оптимізації маршруту на основі геокодування даних	Логістика	Невідомі	Точність, швидкість обрахунку, адекватність результату

Проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають.

Таблиця 5.6 — Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Новий функціонал ПЗ конкурентів	Впровадження нового функціоналу у Google Maps, аналогічного до розроблюваного у цьому проекті	Вихід з ринку

Таблиця 5.7 Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий функціонал у проекті що розробляється	Додавання нових моделей та можливостей у проект, що розроблюється	Розроблення цього функціоналу

Проведемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
Тип конкуренції — монополістична	Одне підприємство майже зайняло усю нішу	Значний
За рівнем конкурентної боротьби — національне	Дане підприємство відомо по усьому світу	Значний
За галузевою ознакою — внутрішньогалузева	Конкуренція виконується в рамках однієї галузі	Значний
Конкуренція за видами товарів — невідомо		
За характером конкурентних переваг — цінова	Товар даного підприємства має дуже високу вартість	Значний

За інтенсивністю — невідомо

Проведемо більш детальний аналіз умов конкуренції у галузі.

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Google Maps	Розрахунок теплообміну	Невідомо	Невідомо	Невідомо
Висновки	Маючи майже монопольне положення на ринку розробник цього ПЗ не буде приділяти уваги розробці	Є можливість виходу на ринок	Невідомо	Невідомо	Невідомо

За результатами аналізу можна зробити висновок, що працювати на даному ринку можна незважаючи на конкурентну ситуацію. Для поширення продукту він повинен володіти рядом факторів, які відрізняють його від існуючого конкурента.

Перелічимо фактори конкурентоспроможності

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Простота	Дана розробка не вимагає від користувача особливих знань у галузі
2	Дешевизна	Поширюється безкоштовно і кожний має можливість користуватися нею
3	Швидкодія	Розраховуються найкращі показники для конкретного проекту

Проведемо аналіз сильних та слабких сторін стартап-проекту.

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали	Рейтинг товарів —конкурентів у порівнянні з проектом, що розробляється						
		1-20	-3	-2	-1	0	+1	+2	+3
1	Простота								
2	Дешевизна								
3	Швидкодія								

Проведемо SWOT-аналіз

Таблиця 5.12 — SWOT-аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
Простота	Невідома компанія
Дешевизна	Відсутність стартового капіталу
Швидкодія	
Можливості:	Загрози:
Розширення функціоналу	Продукти-замінники
Нові технології	

З огляду на SWOT-аналіз можна прийти до висновку що нема потреби розробляти альтернативи ринкового впровадження цього проекту.

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку, а саме опис цільових груп потенційних споживачів.

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Логістика	Готові	Високий	У сегменті значна	Важко

				конкуренція	
2	Екскурсоводи, кур'єрів, гіди	Готові	Високий	У сегменті не значна конкуренція	Важко

Які цільові групи обрано: логістика, екскурсоводи, кур'єрів, гіди

Для роботи в обраних сегментах ринку сформулюємо базову стратегію розвитку.

Таблиця 5.15 — Визначення базової стратегії розвитку

№ п/п	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія ринку
1	Диференційований маркетинг	Простота, дешевизна, швидкодія	Стратегія спеціалізації

Виберемо конкурентну поведінку

Таблиця 5.16 — Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопроходьцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкуренту?	Стратегія конкурентної поведінки
1	Так	Ні	Ні	Заняття конкурентної ніші

Розробимо стратегію позиціонування, що полягає у формуванні ринкової позиції, за яким споживачі мають ідентифікувати проект.

Таблиця 5.17 — Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Точність			

4.5 Розроблення маркетингової програми стартап-проекту

Сформуємо маркетингову концепцію товару, який отримає споживач.

Таблиця 5.18 — Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Розробка мережевого додатку для оптимізації маршруту на основі геокодування даних	Швидке створення, висока точність, швидкість обрахунку, адекватність результату	Швидкодія, безкоштовність, точність

Таблиця 5.19 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
• Товар за задумом	Розробка мережевого додатку для оптимізації маршруту на основі геокодування даних
• Товар у реальному виконанні	Властивості: 5. Простота 6. Дешевизна 7. Швидкодія Якість: тестування на готових фізичних моделях Пакування: відсутнє Марка: відсутня
• Товар із підкріпленням	До продажу: невідомо Після продажу: невідомо

Товар не буде якимось чином захищатись від копіювання та буде поширюватись як є.

Визначимо цінові межі, якими необхідно керуватись при встановленні ціни на товар.

Таблиця 5.20 — Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення

				ціни на товар
1	70-100 тис. ум. од.	До 10 тис ум. од.	Високий	Безкоштовно

Визначимо оптимальну систему збуту

Таблиця 5.21 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Невідома	Вільний доступ до товару	Невідома	Вільний доступ до товару

Розробимо концепцію маркетингових комунікацій

Таблиця 5.22 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Невідома	Інтернет, наукові публікації	Можливості проекту	Донести про можливості проекту	Донесення про можливості та сильні стороні проекту

4.7 Висновки за розділом

За результатами проведеного аналізу можна зробити висновок, що є можливість ринкової комерсалізації проекту оскільки на ринку є попит на таку продукцію. Але оскільки метою цього проекту не є матеріальне збагачення, продукт буде поширюватись вільно, безкоштовно та без обмежень, то комерсалізація проекту не має сенсу.

РОЗДІЛ 5 ОХОРОНА ПРАЦІ ТА ПОЖЕЖНА БЕЗПЕКА

У даному розділі розглянуто питання, пов'язані з аналізом і оцінкою небезпечних та шкідливих чинників, що мають місце при комплексній розробці мереживого додатку по розрахунку оптимального шляху, що використовує радіоканал в якості несучого середовища. Аналіз виконується з урахуванням вимог ДсанПіН 3.3.2.007-98 та ГОСТ 12.1.006-84.

Основна увага в розділу приділяється питанням охорони праці при використанні ЕОМ а також захисту працюючого персоналу від впливу електромаігнітного віпромінювання під час роботи радіо-ЛОМ. До цих потенційно небезпечних і шкідливих для життя і здоров'я людини чинників, можуть бути віднесені:

1. наявність електромагнітного випромінювання радіочастотного діапазону і невикористаного рентгенівського випромінювання;
2. мікроклімат на робочому місці
3. виробничий шум
4. іонний склад повітря
5. електростатичні поля
6. можливість ураження електричним струмом;
7. виникнення пожежо- та вибухонебезпечних ситуацій;

5.1 Аналіз небезпечних і шкідливих чинників

Електробезпека

Вимоги електробезпеки у приміщеннях, де встановлені електронно-обчислювальні машини і персональні комп'ютери (далі — ЕОМ) відображені у ДНАОП 0.00-1.31-99.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загорання внаслідок короткого замикання та перевантаження проводів, обмежувати застосування

проводів з легкозаймистою ізоляцією і, за можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів і прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення.

Використання нульового робочого провідника як нульового захисного провідника забороняється, а також не допускається підключення цих провідників на щиті до одного контактного затискача.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

Неприпустимим є підключення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ до звичайної двопровідної електромережі, в тому числі — з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення персональних ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ слід виконувати за магістральною схемою, по 3—6 з'єднань або електророзеток в одному колі.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог ПВЕ та Правил пожежної безпеки в Україні.

Електромережу штепсельних розеток для живлення персональних ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ при розташуванні їх у центрі приміщення, прокладають у каналах або під знімною підлогою в металевих трубах або в гнучких металевих рукавах. При цьому не дозволяється застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, що містять сірку. Відкрита прокладка кабелів під підлогою забороняється.

Конструкція знімної підлоги повинна бути такою, щоб забезпечувались:

1. вільний доступ до кабельних комунікацій під час обслуговування;
2. стійкість до горизонтальних зусиль при частково знятих плитах;
3. вирівнювання поверхні підлоги за допомогою регулювальних опорних елементів;
4. взаємозамінюваність плит.

Отвори в плитах для прокладання кабелів електроживлення виконуються безпосередньо в місцях встановлення устаткування відповідно до затвердженого технологічного плану розміщення устаткування та його технічних характеристик.

Є неприпустимими:

1. експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;
2. застосування саморобних продовжувачів, які не відповідають вимогам ПВЕ до переносних електропроводок;

3. застосування для опалення приміщення нестандартного (саморобного) електронагрівального обладнання або ламп розжарювання;
4. користування пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;
5. підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною та іншими горючими матеріалами, експлуатація їх зі знятими ковпаками (розсіювачами);
6. використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств-виготовлювачів.

Небезпечний і шкідливий вплив на людей електричного струму виявляється у вигляді електротравм і професійних захворювань.

У помешканні для роботи на обчислювальних машинах відсутні умови, що створюють підвищену небезпеку ураження електричним струмом, оскільки відносна вологість повітря не перевищує 75 %, матеріал підлоги не проводить струм (лінолеум), температура повітря не перевищує 35°С, відсутні хімічно агресивні середовища.

Розрахунок захисного занулення

Розрахувати струм короткого замикання для схеми занулення можна за формулою:

$$I_{КЗ}=\frac{U_C}{r+r_0} \ ,$$

де U_C - розмір живлячої напруги, U_C = 220 В;

r - опір фазного проводу, r = 1,3 Ом;

r_0 - опір нульового проводу, r_0 = 1 Ом.

$$I_{КЗ}=\frac{220}{1+1,3}=96 \text{ А}$$

Для захисту від струму повинна виконуватися умова $I_{КЗ} > 1,4*I_{НОМ}$ (при $I_{КЗ} > 100$ А), тоді:

$$I_{НОМ} \le 68 A$$

Розрахувати напругу доторку можна за формулою:

$$U_{ДОТ}=I_{КЗ} \cdot r_0=96 \cdot 1=96 B$$

У якості захисного пристрою встановлений автомат на 25 А. Припустиме значення напруг доторку (таблиця 3.1.) при часі дії 0,1 с (час спрацьовування автомата) - 500 В, що задовольняє заданим вимогам.

Заходи щодо забезпечення електробезпеки розробляються з урахуванням ГОСТ 12.1.019-79. Електробезпеку необхідно забезпечити конструкцією комп'ютера, технічними способами та засобами захисту, організаційними та технічними заходами.

Забезпечення електробезпеки конструкцією комп'ютера:

1. Розташування та з'єднання електроприладів повинно бути виконані з обліком зручностей і безпеки;

2. Повинна виключатися можливість неправильного приєднання струмоведучих частин електроприладів;

2.1. Конструкція штепсельних розеток і вилок для напруги вище 42 В повинна відрізнятися від конструкції штепсельних розеток і вилок для напруги 42 В та менше;

2.2. Для здійснення з'єднання за допомогою розетки-вилки до розетки повинне підключатися джерело енергії, до вилки - її приймач;

Забезпечення електробезпеки технічними способами:

1. Ізоляція струмоведучих частин;

2. Захисне занулення;

3. Застосування засобів і (або) елементів призначених для автоматичного відключення електроприладу в аварійному режимі (перевантаження, перегрів, коротке замикання).

Електростатичні поля

Відеотермінали на основі електропроміневої трубки є джерелом електростатичних зарядів. Тривале перебування в електричному полі, що створюється цими зарядами може спричинити бронхо-легеневі захворювання, порушення серцево-судинної та нервової систем, ураження шкіри та ін.

Несприятливий вплив електростатичного поля проявляється в тому, що воно здатне притягувати пил, бруд та інші частини, присутні в повітрі, навколо відеотерміналів. Не важко помітити, що після того, як очистити екран відеотерміналу від пилу він досить швидко покривається ним знову. Під час досліджень, проведених Шведським національним інститутом радіаційного захисту на робочих місцях з відеотерміналами вивчався вплив електростатичного поля на інтенсивність осідання ізотопів радону на обличчі оператора. Встановлено, що за концентрації радону в повітрі 100 Бк/м^3 доза радіації за рік зросла приблизно на 50—60%.

Електростатичний заряд зосереджується переважно на відео терміналах з електропроміневою трубкою, зокрема на екрані. Присутність електричного поля, створеного цими зарядами легко можна виявити, якщо піднести до екрана руку; волоски на тильній стороні кисті відразу припіднімаються.

Може навіть відбутися незначний електричний "удар", якщо людина достатньо "заряджена". Така "зарядка" користувача здійснюється, як правило, індуктивним та контактним шляхом. Заряди нагромаджуються на користувачі, підвищуючи тим самим його електричний потенціал.

Проведені дослідження засвідчили значну розбіжність одержаних результатів, залежно від типу відеотерміналу та умов вимірювання. В проведених дослідженнях напруженість електростатичного поля коливалась від 8 до 75 кВ/м.

Електростатичне поле між користувачем та відеотерміналу можна наближено визначити за формулою:

$$E = \frac{V_{\text{ВДТ}} - V_{\text{КОР}}}{l}$$

де E — напруженість електростатичного поля;

$V_{ВДТ}$ — потенціал відеотермінала;

$V_{КОР}$ — потенціал користувача;

l — відстань між відеотерміналом та користувачем.

Відповідно до ДНАОП 0.00-1.31-99 поверхневий електростатичний потенціал відеотермінала не повинен перевищувати 500 В.

Напруженість електростатичного поля на робочих місцях, в тому числі й з ВДТ, не повинна перевищувати 20 кВ/м відповідно до ГОСТ 12.1.045-84 "ССБТ. Электростатические поля. Допустимые уровни на рабочих местах и требования к проведению контроля".

Для запобігання створенню значної напруженості поля та захисту від статичної електрики необхідно:

— встановити нейтралізатори статичної електрики;

— підтримувати в приміщенні з відеотерміналом відносну вологість повітря не нижче 45—50% (чим сухіше повітря тим більше електростатичних зарядів); можна для цього використати навіть побутові зволожувачі;

— застелити підлогу в приміщеннях з відеотерміналами антистатичним лінолеумом і проводити щоденне вологе прибирання;

— складати всі полімерні покриття (чохли) відеотерміналів у найбільш віддаленому від користувачів місці розміщення;

— протирати екран та робоче місце спеціальною антистатичною серветкою або зволоженою тканиною;

— користувачам бажано носити одяг, особливо першого шару з натуральних матеріалів;

— для "зняття" статичного заряду бажано кілька разів на день мити руки та обличчя водою, або час від часу торкатися металевих поверхонь, наприклад, батареї центрального опалення.

Виробниче освітлення

Робота користувачів комп'ютерів характеризується значним напруженням зорового аналізатора, тому виключно важливе значення має забезпечення раціонального освітлення робочих місць. Зоровий дискомфорт може бути викликаний:

1. неправильною орієнтацією робочого місця відносно світлових отворів (вікон);
2. неадекватними світловими характеристиками світильників (та/або)
3. неправильним їх просторовим розташуванням відносно робочих місць;
4. засліплюючою дією яскравих предметів, що знаходяться в полі зору користувача (пряма блискість);
5. дзеркальним відбиттям на екрані предметів з високою яскравістю, що знаходяться за спиною користувача (відбита блискість);
6. неправильним розподілом яскравості в полі зору користувача;

7. засвіченням екрана прямим чи розсіяним світлом світильників або небосхилу через світлові отвори.

У забезпеченні максимально комфортних умов зорової роботи вагома роль належить оптимізації кількісних та якісних показників освітлення.

Нормований рівень освітленості на робочому столі в зоні розташування документа становить 300—500 лк.

Несприятливий вплив на зорову роботу користувача відеотерміналу може здійснювати дзеркальне відбиття на екрані яскравих елементів неправильно розташованих світильників, або ділянок стелі чи вікна, на які подають сонячні промені. Такі дзеркальні відбиття при відносно невеликій яскравості екрана відеотерміналу, можуть викликати практично повну втрату контрасту зображення.

Розрахуємо, втрату відносного контрасту K зображення на екрані відеотерміналу з яскравістю фону $B_Ф$ = 10 кд/м^2 та яскравістю знаків $B_{ЗН}$ = 100 кд/м^2 при накладанні на нього відбиття з яскравістю $B_{ВІД}$ = 500 кд/м^2.

Контраст знаків без впливу дзеркального відбиття становить:

$$K_1 = \frac{(B_{ЗН} - B_Ф)}{B_Ф} = \frac{(100 - 10)}{10} = 9$$

Контраст знаків на екрані при накладанні дзеркального відбиття рівний:

$$K_2 = \frac{[(B_{ЗН} + B_{ВІД}) - (B_Ф + B_{ВІД})]}{(B + B_{ВІД})} = \frac{[(100 + 500) - (10 + 500)]}{(10 + 500)} = 0,176$$

Таким чином контраст знаків на екрані при накладанні дзеркального відбиття зменшився більш ніж у 50 разів.

Ліквідувати контрастопонижуючий вплив дзеркального відбиття на екрані та засвітлень, що викликані високими рівнями розсіяного світла, шляхом підвищення, яскравості знаків, недоцільно, оскільки при цьому

погіршується помітність літер та цифр внаслідок виникнення "розмитості" (нечіткості) їх контурів.

Для забезпечення відносної постійності природного освітлення незалежно від погодних умов чи пори року необхідно вікна обладнати сонцезахисними регульованими жалюзями або світлорозсіюючими шторами з коефіцієнтом відбиття 0,5—0,7.

В якості джерел штучного світла застосовуються люмінесцентні лампи, які краще поєднуються з природним освітленням, аніж лампи розжарювання. Окрім того, вони створюють більш дифузні світлові потоки, через що знижується можливість засліплюючої дії світла, відбитого екраном. Найкраще застосовувати люмінесцентні лампи типу ЛБ, які мають найвищу світловіддачу.

Захист від ЕМВ.

Під час роботи користувачі ЛОМ постійно перебувають від впливом випромінення радіочастотного діапазону, що є несучим середовищем інформаційних потоків.

Згідно з ГОСТ 12.1.006-84 енергетичне навантаження ЕН не повинно перевищувати:

$ЕН_{ЩПЕкд} < 2$ Вт*год/м^2

Припустима щільність потоку енергії при тривалолсті робочого дня в 8 годин дорівнює:

$ЩПЕ_{кд} = ЕН_{ЩПЕкд}/Т = 2/8 = ¼ = 0.25$ Вт/м^2

Фактичне ж значення $ЩПЕ_ф$ дорівнює

$ЩПЕ_ф = Р*\sigma/2*\pi*r^2 = 0.75*0.6/2*3.14*1 = 0.06 < 0.25$

Отже, вимоги щодо захисту працівників від ЕМВ дотримано.

5.2 Заходи щодо поліпшення умов праці при роботі з комп’ютером

При розробці програмного продукту "Мережевий додаток по розрахунку оптимального маргруту", найбільше навантаження приходиться на увагу та зір. Одним із способів зниження навантаження на зір - зменшення шкідливого впливу, джерелом якого є комп'ютер. Необхідно звести їх до мінімуму при забезпеченні максимального виграшу від використання комп'ютера. Для цього необхідних наступні апаратні та програмні засоби:

1. монітор (відеотермінал) повинен мати сертифікат, який підтвержує, що він відповідає нормам MPR II, мати маркірування CE (EN50082 1 – електромагнітна захищеність, EN60950 (IEC950) – безпека продукції);обов'язкове заземлення корпуса комп'ютера, пристроїв із передбаченим конструктивним використанням 3-х штирковою мережною вилкою;

2. розміщення всіх мережних шнурів за межами зони проходу людей (у куті, під підлогою та т.п.);

3. наявності, в моніторі режимів збереження електроенергії EPA/NUTEK (через деякий час, якщо користувач не працює на комп’ютері, монітор виключається);

4. після кожної години роботи на ЕОМ повинна бути перерва біля десятьох хвилин

5. частота оновлення екрану повинна бути більше 100 Гц (практично підтвержено, що очі втомлюються менше);

6. використання блоків безперебійного живлення разом зі стабілізаторами живлення (для вирівнювання напруги і виключення раптових відмовлень техніки в результаті вимикання живлення мережі);

7. використання «розумних» охолоджувальних систем: усі вентилятори повинні мати систему контролю температури (чим більше температура тим більш обертів);

8. використовувати спеціальні комп'ютерні меблі

9. установити в приміщенні кондиціонер, для підтримки постійної температури

5.3 Пожежна безпека

Залежно від особливостей виробничого процесу, крім загальних вимог пожежної безпеки, здійснюються спеціальні протипожежні заходи для окремих видів виробництв, технологічних процесів та промислових об'єктів. Для споруд та приміщень, в яких експлуатуються відеотермінали та ЕОМ такі заходи визначені ДНАОП А.01.001-95 "Правила пожежної безпеки в Україні", ДНАОП 0.00-1.31-99 та іншими нормативними документами.

Будівлі і ті їх частини, в яких розташовуються ЕОМ, повинні бути не нижче II ступеня вогнестійкості. Над та під приміщеннями, де розташовуються ЕОМ, а також у суміжних з ними приміщеннях не дозволяється розташування приміщень категорій А і Б за вибухопожежною небезпекою.

Для всіх споруд і приміщень, в яких експлуатуються відеотермінали та ЕОМ, повинна бути визначена категорія з вибухопожежної і пожежної небезпеки відповідно до ОНТП 24-86 "Определения категорий помещений и зданий по взрывопожарной и пожарной опасности", та клас зони згідно з

Правилами влаштування електроустановок. Відповідні позначення повинні бути нанесені на вхідні двері приміщення.

Сховища інформації, приміщення для зберігання перфокарт, магнітних стрічок, пакетів магнітних дисків слід розміщати у відокремлених приміщеннях, обладнаних негорючими стелажами і шафами. Зберігати такі носії інформації на стелажах необхідно в металевих касетах. В приміщеннях ЕОМ слід зберігати лише ті носії інформації, які необхідні для поточної роботи.

Звукопоглинальне облицювання стін та стель у приміщеннях ЕОМ слід виготовляти з негорючих або важкогорючих матеріалів.

У випадку необхідності проведення дрібного ремонту або технічного обслуговування ЕОМ безпосередньо в машинному залі та неможливості застосування негорючих миючих речовин дозволяється мати не більше 0,5 л легкозаймистої рідини у тарі, що не б'ється та щільно закривається.

Приміщення, в яких розташовуються персональні ЕОМ та дисплейні зали, повинні бути оснащені системою автоматичної пожежної сигналізації з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20м^2 площі приміщення з урахуванням гранично допустимих концентрацій вогнегасної речовини.

Не рідше одного разу на квартал необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами.

Визначення категорії вибухопожежної небезпеки приміщення

Категорію вибухонебезпечної та пожежної небезпеки помешкання визначимо виходячи з виду горючих матеріалів, що знаходяться в ньому. У

помешканні можлива наявність: дерев'яних виробів (меблі), папери (книги, журнали), полімерних матеріалів (ізоляція проводів), тканин і т.д.

Необхідно розрахувати надлишковий тиск вибуху в помешканні для самого несприятливого випадку, коли весь вміст ємності зі спиртом надходить у помешкання.

Спочатку визначимо деякі значення, необхідні для розрахунку надлишкового тиску.

Приміщення необхідно оснастити 2-ма переносними порошковими вогнегасниками міскістю 5 литрів на 400 м^2. Інши типи вогнегасників не рекомендується використовувати в данному приміщенні. (таблиці 8.22 і 8.23 [2]).

Для гасіння пожеж у помешканні можливо застосування порошкових вогнегасників.

Час евакуації відповідає вимогам СНиП 2.01.02 - 85.

Для виявлення початкової стадії пожежі, повідомлення про місце її виникнення застосуємо установку пожежної сигналізації на базі автоматичних пожежних засобів оповіщення. Установки електричної пожежної сигналізації перебувають із засобів оповіщення - датчиків, приймальнею станції, джерела харчування і ліній зв'язку.

Автоматичні засоби оповіщення перетворюють неелектричні фізичні величини в електричні сигнали, що передаються по проводових лініях зв'язку на прийомну станцію. Принцип дії та будова пожежних засобів оповіщення залежать від їх основних характеристик: інерційности, зони дії та конструктивного виконання.

При виборі засобів оповіщення врахуємо необхідну швидкість дії системи пожежного захисту, середовище, у якому буде працювати засіб оповіщення. Для даного типу помешкань рекомендується використовувати

засоби оповіщення типу: ДТЛ та ДИП-1, що у залежності від параметру спрацьовування відносять відповідно до теплових і димових.

Помешкання обладнане чотирма пожежними засобами оповіщення типу ДТЛ (площа, що захищається, - 4 х 15 = 60 м^2); відстань між засобами оповіщення - 2м, що відповідає нормам.

Схеми електричної пожежної сигналізації обладнаємо незалежним резервним джерелом харчування з автоматичним вмиканням у випадку відмови основного джерела харчування.

Припустимі відстані від найбільше віддаленого робочого місця до найближчого евакуаційного виходу та ширина виходу відповідає СНиП 2.01.02-85.

Виникнення пожежі можливо у випадку короткого замикання в ланцюгах електроживлення. У зв'язку з цим необхідно передбачити наступні заходи:

1. Ретельна ізоляція всіх струмоведучих провідників підхожих до робочих місць;
2. Періодичний огляд і перевірка ізоляції;
3. Суворе дотримання норм протипожежної безпеки на робочому місці;
4. Для винятку розливання ЛЗЖ та утворення вибухонебезпечних сумішей, ці рідини необхідно берегти в спеціальному посуді та у спеціальному місці.

У цілому, у робочому помешканні виконані усі вимоги по пожежній безпеці відповідно до НАПБ. А. 01. 001-95 "Правила пожежної безпеки в Україні".

ВИСНОВОК

У даній роботі була досліджена задача побудови оптимального маршруту. Були виявлені недоліки існуючих додатків для пошуку оптимального шляху і за результатами дослідження сформульовані вимоги до розробляється з додатком. При вирішенні завдання пошуку оптимального шляху були вивчені різні алгоритми, був проведений аналіз, в результаті якого був обраний варіант використання сервісу як найбільш задовольняє поставлені мети. Генетичний алгоритм був реалізований на мові JavaScript с використанням Google.Maps і WebGL. Розроблене додаток було успішно протестовано при різних ввідних даних, було досліджено якість результатів при певних параметрах алгоритму, в результаті чого були обрані найбільш ефективні рішення. Мета роботи, яка полягає в створенні зручного інтерактивного додатки з пошуку оптимального шляху для екскурсоводів і туристів, була в повному обсязі досягнута.

СПИСОК ЛІТЕРАТУРИ

1. Geunes J. Operations Planning: Mixed Integer Optimization Models (Operations Research Series). CRC Press, 2014. P. 167–172.
2. Яндекс.Карти. ://yandex.ru/maps
3. Google Maps https://maps.google.com/
4. Cook W. J. In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, 2012. P. 19-39.
5. Applegate D.L., Bixby R.E., Chvátal V., Cook W.J. The Traveling Salesman Problem. Princeton University Press, 2007. P. 44-52.
6. Левітін А. Алгоритми. Введення в розробку і аналіз. Вільямс, 2006. P. 160.
7. Гері М., Джонсон Д. Обчислювальні машини і складнообчислювальні завдання. Світ, 1982.
8. Okano H. Study of Practical Solutions for Combinatorial Optimization Problems. Tohoku University, 2009. P. 14-17.
9. Meganavigator. http://meganavigator.com/
10. Логіст. http://logist.poncy.ru/
11. Speedy Route. https://www.speedyroute.com/
12. Левітін А. Алгоритми. Введення в розробку і аналіз. Вільямс, 2006. 35-36 с.
13. Kona H., Burde A., Dr. Zanwar D. R. A Review of Traveling Salesman Problem with Time Window Constraint // IJIRST - International Journal for Innovative Research in Science & Technology, 2015. Vol. 2, Issue 1. P. 253-254.
14. Tannenbaum P. Excursions in Mathematics. University of Kansas, 2011. P. 25.
15. Clausen J. Branch and Bound Algorithms - Principles and Examples. University of Copenhagen, 1999. P. 5-6.
16. Tollis I. G. Algorithms and Complexity. University of Crete, 2000. P. 140-146.
17. Applegate D., Bixby R., Chvatal V., Cook W. TSP cuts outside the template paradigm. Donet, 2000. P. 1-10.
18. Захарова О.М., Мінашіна І.К. Огляд методів багатовимірної оптимізації // Інформаційні процеси, 2014. Том 14, № 3. стор. 265- 266.
19. Papadimitriou C., Vazirani U. Efficient Algorithms and Intractable Problems. Berkeley EECS, 1999. Chapter 10.

20. Goodrich M., Roberto T. Algorithm Design and Applications, Wiley, 2015. P. 513-514.
21. Nilsson C. Heuristics for the Traveling Salesman Problem. Linkoping University, 2011. P. 1-6.
22. Alsalibi B.A., Jelodar M.B., Venkat I. A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem // International Journal of Computer Science and Electronics Engineering (IJCSEE), 2013. Vol. 1, Issue 1.
23. Gutin G., Yeo A. The Greedy Algorithm for the Symmetric TSP. University of London, 2002. P. 1-2.
24. Gutin G., Karapetyan D. Lin-Kernighan Heuristic Adaptations for the Generalized Traveling Salesman Problem. Royal Holloway London University, 2010. P. 1-5.
25. Gupta R., Chauhan C., Pathak K. Survey of Methods of Solving TSP along with its Implementation using Dynamic Programming Approach. // International Journal of Computer Applications, 2012. Vol. 52, No.4. P. 1-6.
26. Basu S. Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey. Indian Institute of Management Calcutta, 2012. P. 1-8.
27. Dorigo M., Caro G. D. Ant Algorithms for Discrete Optimization // Artificial Life, 1999. Vol. 5, No 2. P. 139-140.43
28. Dorigo M., Gambardella L. M. Ant colonies for the traveling salesman problem. Université Libre de Bruxelles, 1996. P. 1-4.
29. Johnson D. S., McGeoch L. A., Rothberg E. E. Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound. AT & T Bell Laboratories, 1999. P. 1-2.
30. Abdulkarim H.A., Alshammari I. F. Comparison of Algorithms for Solving Traveling Salesman Problem. // International Journal of Engineering and Advanced Technology, 2015. Vol.4, Issue 6. P. 76-79.
31. Роббінс Д.Н. HTML5, 5-е видання. Вільямс, 2015. P. 11-14.
32. HTML5. W3C Recommendation.
33. Макфарланд Д.С. Велика книга CSS3. Символ-Плюс, 2014. P. 10-24.
34. Stylus official site http://stylus-lang.com/
35. Bootstrap official site http://getbootstrap.com/

36. Фленаган Д. Javascript. Детальний керівництво. 6 видання. Символ-Плюс, 2013. P. 21-39.
37. jQuery official site https://jquery.com/
38. Google Maps API https://developers.google.com
39. Мацуда К., Лі Р. WebGL: Програмування тривимірної графіки. ДМК Прес, 2015. P. 26-41.

ДОДАТОК А

Технічне завдання

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Радіотехнічний факультет

ПОГОДЖЕНО

Старший викладач кафедри радіоконструювання та виробництва радіоапаратури КПІ ім. Ігоря Сікорського

_____О.П. Яненко

професор

ЗАТВЕРДЖУЮ

Завідувач кафедри радіоконструювання та виробництва радіоапаратури КПІ ім. Ігоря Сікорського

_____Є.А.Нелін

д.т.н., професор

ТЕХНІЧНЕ ЗАВДАННЯ

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

на виконання НДР

від 01 вересня 2018 р.

«Оптимізація маршрутизації на основі геокодування даних»

«Routing optimization based on data geocoding»

Київ 2018

1 Підстава для виконання роботи

Підставою для виконання роботи є завдання на магістерську дисертацію.

2 Мета і призначення

2.1 Об’єкт та предмет дослідження

Об’єкт дослідження: мережевий додаток моделювання оптимального маршруту.

Предмет дослідження: алгоритми пошуку оптимального маршруту, геокодування даних,програмний код на Java, JavaScript, сервер на базі лінукс, веб клієнт, база даних MySql.

2.2 Мета роботи

Мета даної роботи полягає в інтегруванні алгоритму завдання розрахунку оптимального маршруту між n-ною кількістю місць, з можливістю корегування побудованого маршруту клієнтом, віртуального перегляду вулиць та місць по кліку на карті, отримання повної інформації щодо пересування між пунктами призначення .

2.3 Задачі, які потребують вирішення

2.3.1 Розглянути основні алгоритми пошуку оптимального маршруту.

2.3.2 Провести аналіз даних алгоритмів і вибрати один з них для практичної реалізації.

2.3.3 Написати код обраного алгоритму, використовуючи сучасні технології і фреймфорки : Java, Spring, JavaScript, авторизацію, реєстрацію.

2.3.4 На основі отриманих даних аналізу розробити додаток, що дозволяє користувачам ввести пункт відправлення, призначення і проміжні пункти свого маршруту, вибрати тип поїздки, отримати оптимальний шлях, наочнопоказаний на Google Maps карті з можливістю корегування, візуальним переглядом вулиць, місць маршруту по кліку на певне місце на карті та отриманням широкої інформації для пересування по маршруту.

3 Вихідні данні для проведення роботи

3.1 Підручники

3.1.1 Geunes J. Operations Planning: Mixed Integer Optimization Models (OperationsResearch Series). CRC Press, 2014. P. 167–172.

3.1.2 Cook W. J. In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, 2012. P. 19-39.

3.1.3 Левітін А. Алгоритми. Введення в розробку і аналіз. Вільямс, 2006. P. 160..

3.1.4 Clausen J. Branch and Bound Algorithms - Principles and Examples. University of Copenhagen, 1999. P. 5-6.

4 Вимоги до виконання

Дипломна робота має бути виконана якісно та у встановлений календарним планом термін. Мережевий додаток розробляться на мові програмування джава, має представляти з себе jar файл який буде задеплоїний на сервер Tomcat. Науковий звіт має бути оформлено згідно ДСТУ 3008-2015.

5 Етапи та термін виконання

Етап роботи	Зміст етапу	Термін виконання
Вибір напряму дослідження	Пошук та аналіз літератури. Розроблення, погодження та	Вересень 2018 року

	затвердження ТЗ.	
Теоретичні та експериментальні дослідження	Огляд предметної області, що включає в себе опис та історію типового рішення побудови оптимального маршруту, а також аналізу вже існуючих додатків.	Жовтень 2018 року
Математична основа проекту	У цьому розділі буде проведений аналіз існуючих алгоритмів TSP з описом їх переваг та недоліків, буде детально розібрано застосування генетичних алгоритмів до вирішення завдання пошуку оптимального маршруту і показано, чому саме генетичний алгоритм був обраний в як інструмент розробки програми.	Жовтень – Листопад 2018 року
Практична реалізація	Проведення розробки програми, також розробляється алгоритм, який будує оптимальний маршрут по заданих на мапі точкам.	Жовтень – Листопад 2018 року
Підготовка звітної документації	Узагальнення результатів теоретичних і експериментальних	Жовтень – Листопад 2018 року

	робіт. Оцінювання повноти і якості поставлених завдань. Підготовка комплекту звітної документації.	
Захист дипломної роботи	Представлення дисертації кафедрі. Попередній захист. Захист перед екзаменаційною комісією.	Грудень 2017 року

6 Очікувані результати та порядок реалізації

1. Зробити огляд предметної області, представити опис, поставити завдання на виконня магістерської роботи .

2. Провести аналіз методів вирішення TSP, обрати певний алгоритм вирішення задачі, аргументувати вибір.

3. Детальний аналіз, математична розробка і розрахунок обраного алгоритму побудови оптимального маршруту.

4. Розробити програму моделювання оптимального маршруту на мові Java,JavaScript .

5. Провести тестування розробленого мережевого додатку.

7 Матеріали, які подаються по закінченню дипломної роботи

1. Завдання на дипломну роботу.

2. Технічне завдання.

3. Дипломна робота.

4. Мультимедійна презентація до захисту.

8 Порядок приймання дисертації та її етапів

1. Поетапне узгодження з керівником.

2. Представлення кафедрі.

3. Попередній захист.

4. Захист перед екзаменаційною комісією.

9 Вимоги до розроблюваної документації

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання.

2. ДСТУ 3973-2000. Система розроблення та поставлення продукції на виробництво. Правила виконання науково-дослідних робіт. Загальні положення.

10 Приблизний зміст дипломної роботи

1. Аналітичний огляд.

2.Математична основа проекту, аналіз алгоритмів.

3.Практична розробка додатку.

4.Охорона праці.

5.Економічне обгрунтування.

6.Висновки.

Виконавець _____ _____

(підпис) (розшифровка підпису)