

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

Факультет прикладної математики

Кафедра прикладної математики

«До захисту допущено»

Завідувач кафедри

О.Р. Чертов

(підпис)

“ ” 2015 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності 6.040301 «прикладна математика»

на тему: «ППП для побудови оптимального маршруту транспортного засобу»



Виконав: студент 4 курсу, групи КМ-11

Кукса Микола Юрійович

(підпис)

Керівник: доцент кафедри Прикладної математики, к.т.н., доцент,
Олефір О.С.

(підпис)

Консультант з нормоконтролю: старший викладач Мальчиков В.В.

(підпис)

Рецензент: доцент кафедри Обчислювальної техніки ФІОТ, к.т.н.,
доцент, Клименко І.А.

(підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

(підпис)

Київ – 2015 року

**Національний технічний університет України
«Київський політехнічний інститут»**

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 6.040301 “Прикладна математика”

ЗАТВЕРДЖУЮ
Завідувач кафедри

О.Р. Чертов
(запис)

« ____ » 2015 р.

**ЗАВДАННЯ
на дипломну роботу студенту
Куксі Миколі Юрійовичу**

1. Тема роботи «ППП для побудови оптимального маршруту транспортного засобу»,

керівник роботи Олефір Олександр Степанович к.т.н. доцент,

затверджені наказом по університету від «19» травня 2015 р. № 1039-С

2. Термін подання студентом роботи «12» червня 2015 р.

3. Вихідні дані до роботи:

- транспортна мережа міста Києва (автомобільні дороги, обмеження встановленні правилами дорожнього руху);
- назви вулиць, будинків, географічних об'єктів міста Києва, їх географічні координати, опис, характеристики.

4. Зміст роботи:

- провести аналіз існуючих систем побудови оптимального маршруту та їх програмних інтерфейсів;

- визначити джерело картографічних даних, програмний інтерфейс для використання в роботі системи;
- визначити математичну модель задачі;
- проаналізувати алгоритми вирішення задачі;
- виконати програмну реалізацію системи побудови оптимального транспортного маршруту;
- виконати тестування розробленого програмного продукту.

5. Перелік ілюстративного матеріалу:

- структура географічних даних, що використовуються в роботі;
- математична модель предметної області;
- архітектура системи;
- ERD-діаграма бази даних системи;
- структурна схема взаємодії модулів програмного забезпечення;
- знімки екранних форм роботи програми;
- знімки екранних форм тестування програмного забезпечення.

6. Консультанти розділів проекту (роботи):

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	ст.викладач Мальчиков В.В.		

7. Дата видачі завдання «15» жовтня 2014 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз існуючих систем	12.11.2014	
2.	Вибір джерела картографічних даних	10.12.2014	
3.	Визначення математичної моделі транспортної мережі	30.12.2014	
4.	Огляд алгоритмів побудови оптимального маршруту	02.02.2015	
5.	Вибір алгоритмів розв'язку поставленої задачі	20.02.2015	
6.	Проектування програмних засобів	03.03.2015	
7.	Програмна реалізація системи	10.04.2015	
8.	Тестування розробленого програмного продукту	30.04.2015	
9.	Підготовка матеріалів для пояснівальної записки дипломної роботи	06.05.2015	
10.	Підготовка графічної частини дипломної роботи	20.05.2015	
11.	Оформлення дипломної роботи	10.06.2015	

Студент

М.Ю. Кукса

(підпис)

Керівник роботи

О.С. Олефір

(підпис)

АНОТАЦІЯ

Дана дипломна робота присвячена розробці програмної системи побудови оптимального маршруту транспортного засобу.

В роботі розглядаються існуючі системи побудови оптимального маршруту, порівнюються їх основні функціональні можливості та програмні інтерфейси для інтеграції з іншими системами. Також в рамках дипломної роботи розглядається математична модель транспортної мережі, розглядаються алгоритми пошуку оптимального маршруту.

Робота виконана на 97 аркушах, має посилання на список використаних літературних джерел з 12 найменувань, а також 9 рисунків та 6 додатків.

Наведені результати можуть бути використані при створенні систем пошуку оптимального маршруту, картографічного та географічного програмного забезпечення.

Ключові слова: оптимальний маршрут, транспортний засіб, програмний інтерфейс, система.

ABSTRACT

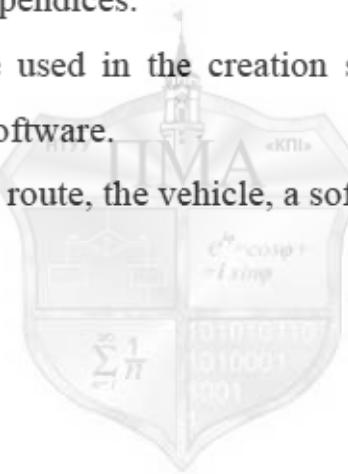
This degree work is dedicated to developing the software systems for constructing an optimal route of the vehicle.

The paper considers the existing system of constructing an optimal route, the comparative analysis of their main functionality and application programming interfaces to integrate with other systems. It is also defined a mathematical model of the transport network, consider algorithms for finding the optimal route.

Work carried out on 97 pages, links to literary sources used list of 12 names, 9 figures and 6 appendices.

The results can be used in the creation systems of finding the optimal route and geographical software.

Keywords: optimal route, the vehicle, a software interface, the system.



ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	8
ВСТУП	9
1 ПОСТАНОВКА ЗАДАЧІ.....	10
2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ	11
2.1 Он-лайн сервіси побудови оптимального маршруту	12
2.2 Висновки до розділу	18
3 АНАЛІЗ СТРУКТУРИ КАРТОГРАФІЧНИХ ДАНИХ OpenStreetMap .	20
3.1 Базові типи географічних даних в OpenStreetMap	21
3.2 Інформаційна схема об'єктів	24
3.3 Висновки до розділу	25
4 МАТЕМАТИЧНІ МЕТОДИ РОЗВЯЗКУ ПОСТАВЛЕНОЇ ЗАДАЧІ	26
4.1 Опис математичної моделі	26
4.2 Огляд алгоритмів пошуку на графах	29
4.3 Схеми збереження графів в пам'яті ЕОМ	31
4.4 Залежність вибору алгоритму від типу задачі	32
4.5 Висновки до розділу	34
5 ПРОГРАМНА РЕАЛІЗАЦІЯ.....	35
5.1 Архітектура системи.....	35
5.2 Структура модулів програмного забезпечення.....	36
5.3 Опис розроблених програмних засобів	37
5.4 Опис структури бази даних системи.....	39
5.5 Графічний інтерфейс програми	41
5.6 Тестування програмного забезпечення.....	43
5.7 Висновки до розділу	45
ВИСНОВКИ.....	46
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	47

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API (application program interface) – прикладний програмний інтерфейс, набір процедур та функцій, що призначення для зв'язків різних програм.

БД – база даних.

JSON (JavaScript Object Notation) – текстовий формат обміну даних побудований на основі мови JavaScript. Порівняно з XML має компактнішу форму опису даних.

ПЗ – програмне забезпечення.

ППП – пакет прикладних програм.

СКБД – система керування базою даних.

XML (eXtensible Markup Language) – розшириована мова розмітки, призначена для опису та передачі структурованих даних.

ВСТУП

У сучасному світі, внаслідок урбанізації, розвитку бізнесу, наземний транспорт відіграє дуже важливу роль. Безперечно, транспортні засоби роблять життя комфортнішим та простішим, але їх використання пов'язане з багатьма проблемами та недоліками, які останнім часом стали дуже актуальними.

До основних проблем пов'язаних з використанням транспорту належать: забруднення атмосфери вихлопними газами, шум, високий рівень аварійності.

Для мінімізації шкоди від використання транспортних засобів інженери з різних країн постійно працюють над вдосконаленням різних моделей автомобілів, зменшуючи витрату палива, розробляючи гібридні моделі автомобілів і т.д. З іншого боку розробники інформаційних систем працюють над вдосконаленням існуючих та розробкою нових систем, які допомагають водіям будувати оптимальний за деякими критеріями маршрут.

Дана робота присвячена розробці системи побудови найкоротшого маршруту транспортного засобу. Враховуючи те, що не всі існуючі системи мають можливість їх безкоштовного використання або задають жорсткі обмеження кількості безкоштовних запитів за певний проміжок часу, дана розробка є досить актуальною. Також актуальність даної розробки пояснюється тим, що існуючі безкоштовні системи з відкритим вихідним кодом мають досить високі системні вимоги.

Дана система може використовуватися звичайними користувачами та організаціями, що надають послуги пошуку певних об'єктів інфраструктури, та додатково шукають оптимальний маршрут до них.

1 ПОСТАНОВКА ЗАДАЧІ

Дана робота присвячена розробці системи побудови оптимального маршруту транспортного засобу. В якості критерію оптимальності використовується довжина маршруту.

Система розрахована на використання у місті Києві.

Вхідні дані – точки маршруту, що задаються у форматі повних географічних назв. Створювана система повинна працювати з маршрутами, що не мають самоперетинань, тобто кожна точка може входити до заданого маршруту лише один раз.

Результат – оптимальний маршрут, що задається послідовністю ділянок транспортної мережі та відстанню між ними. Також в результаті роботи програмного продукту обчислюється та відображається поточна довжина маршруту в проміжних точках, що задав користувач.

Маршрут не може складатися менше ніж з 2 точок. Створювана система повинна мати можливість працювати з маршрутами, що можуть мати довільну кількість точок та не давати користувачу можливість задати маршрут, що має менше двох точок.

Архітектура створюваної системи має бути гнучкою та передбачати можливість розширення функціоналу за рахунок введення нових критеріїв оптимальності маршруту.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Останнім часом з'явилася досить велика кількість програмних та програмно-апаратних систем, які призначені для розв'язання проблеми побудови оптимального транспортного маршруту. Це можна пояснити наступними чинниками:

- а) різкий розвиток географічних інформаційних систем, які акумулюють та оброблюють дані про наявні маршрути, та географічні об'єкти;
- б) розвиток, удосконалення та зниження вартості GPS-датчиків та навігаторів, які з великою точністю можуть відслідковувати географічні координати об'єктів;
- в) розвиток мобільних пристройів, які характеризуються портативністю та технічними показниками, що є достатніми для запуску на них різних програм та додатків;
- г) постійне збільшення кількості автомобілів, проблема заторів;
- д) зростання цін на паливо збільшує попит на системи побудови та планування маршруту.

Серед найбільш поширеніших систем побудови оптимального маршруту можна виділити наступні системи:

- он-лайн сервіси та додатки, які можна використовувати через браузер комп'ютера або іншого пристрою з доступом до мережі Інтернет;
- вбудоване програмне забезпечення GPS-навігаторів, що реалізує функцію побудови маршруту;
- програми та додатки для мобільних пристройів;
- програмне забезпечення для використання на стаціонарних персональних комп'ютерах.

Програмне забезпечення GPS-навігаторів, як правило тісно пов'язане з електронними картами, які завантажуються в пам'ять пристрою. Основні параметри, які використовуються при побудові маршруту: відстань між об'єктами. Деякі з навігаторів відображають дані про затори, та дозволяють враховувати їх при побудові маршруту. Дані про затори отримуються від власних або сторонніх сервісів, що передаються через мережу. Карти та ПЗ для таких пристрій як правило є платними.

Програми для мобільних пристрій, як правило по функціоналу подібні до GPS-навігаторів, або он-лайн сервісів на основі яких вони побудовані. Найчастіше в своїй роботі використовують програмні інтерфейси он-лайн сервісів.

Переважна кількість програмного забезпечення для персональних комп'ютерів, що реалізує функції прокладання маршруту представляє собою великі системи, що розраховані на використання на підприємствах та транспортних організаціях. Такі системи як правило дорого коштують, тому не підходять для використання звичайними користувачами. Основний недолік таких систем – ціна та трудності з оновленням картографічних даних.

Серед існуючих систем великий інтерес представляють он-лайн сервіси для побудови маршруту. Такі системи можуть використовуватися безкоштовно, не потребують установки, що робить їх дуже популярними серед звичайних користувачів. Також такі сервіси надають API для розробників програмного забезпечення.

2.1 Он-лайн сервіси побудови оптимального маршруту

Розглянемо декілька найвідоміших на території України інтернет-сервісів побудови оптимального маршруту, що можуть використовуватися

безкоштовно. Для кожного сервісу розглянемо його основні характеристики. Також для кожного сервісу розглянемо наявність безкоштовних програмних інтерфейсів, які надаються розробникам програмного забезпечення.

Під час вибору даних сервісів для аналізу, враховувалися наступні критерії:

- можливість безкоштовного використання;
- повнота та актуальність геоданих для території України;
- можливість використання безкоштовного API;
- наявність та якість документації.

Google Maps – сервіс корпорації Google Inc. Даний сервіс дозволяє прокладати оптимальний автомобільний маршрут в якості критерію оптимальності використовується відстань між об'єктами та час витрачений на маршрут. На картах сервісу є можливість відображення рівню заторів. В опціях побудови маршруту можна обрати варіанти побудови маршруту без руху по шосе, платних дорогах та поромні переправах. Після побудови маршруту Google Maps дає два маршрути: оптимальний та альтернативний. Для кожного маршруту вказуються відстань та час необхідний для руху.[1]

Для розробників програмного забезпечення Google Maps надає програмний інтерфейс (надалі API, application programming interface) під різні платформи та технології розробки [2]:

- JavaScript API (для розробників веб-сайтів);
- Google Maps Android API (інтерфейс для мобільної платформи Android);
- веб-служби API Google карт (програмний інтерфейс, що надає картографічні дані використовуючи протоколи HTTP та HTTPS). Результат виконання запитів оформлюється у форматі JSON або XML (формат результату задається в параметрах запиту).

Розглянемо детальніше API веб-служб, оскільки такий інтерфейс є найбільш універсальний, та не залежить від конкретної платформи. Google Maps надає наступні веб-служби:

а) API маршрутів (побудова маршруту по заданим початковій та відправній точці). Є можливість задавати декілька проміжних точок. Обмеження на використання – 2500 маршрутів за один день, кожен з яких може містити не більше 8 проміжних точок. Для користувачів API Google Карт для організацій діють більш високі квоти: 100 000 маршрутів за один день, кожен з яких може мати до 23 проміжних точок [3];

б) API матриці відстаней (служба, що надає дані про відстані та час в дорозі для початкових та кінцевих точок). Обмеження на використання служби: 100 елементів за один запит, 100 елементів за 10 секунд, 2500 елементів за 24 години. Для користувачів API Google Карт для організацій діють більш високі квоти: 625 елементів за один запит, 1000 елементів за 10 секунд, 100 000 елементів за 24 години [4];

в) API висотних даних. Даний сервіс надає зручний інтерфейс, що дозволяє отримати дані про висоти різних точках земної поверхні. Також існує можливість отримувати висотні дані вздовж вибірки точок на протязі шляху, що дозволяє обчислювати зміну висоти на маршрути. На використання сервісу накладаються наступні обмеження: не більше 2500 та 100 000 запитів в день для звичайних користувачів та організацій відповідно. В кожному запиті дозволяється вказувати до 512 точок, але не більше 25 000 за день (для користувачів API Google Карт для організацій – 1 000 000) [5];

г) API геокодування (відображення адреси географічної точки в географічні координати та навпаки). Обмеження використання – не більше 2500 та 100 000 запитів в день для звичайних користувачів та організацій відповідно [6].

Необхідно зауважити, що для всіх вищепереліканих веб-сервісів діє обмеження на довжину URL-запиту – 2048 символів, також значним

недоліком є те, що відповідно до умов використання API результати виконання запиту можна відобразити тільки на карті Google. Також всі API є статичними тобто не дозволяють використовувати їх в режимі реального часу, наприклад геокодування географічних назв, що вводить користувач не можна виконати за допомогою даного API.

Основні переваги Google Maps – географічні дані, що постійно оновлюються, широкий набір функцій програмного інтерфейсу, що мають можливість безкоштовного, хоч і досить обмеженого використання, чітка лаконічна документація для розробників, що доступна на офіційному сайті.

Яндекс.Карти – картографічний сервіс пошукової компанії «Яндекс». Компанія була започаткована в Росії, останнім часом стала дуже популярною, та є основним конкурентом Google Maps в країнах СНД.

Даний ресурс дозволяє знаходити географічні об'єкти, будувати маршрути, переглядати інформацію про рівень заторів на автошляхах. Для побудови оптимального маршруту використовується відстань між географічними об'єктами. Перевагою даного сервісу є можливість не тільки відображення рівня затору на картах, але їх встановлення опції «В об'їзд заторів» при побудові маршруту. Серед недоліків прокладання маршруту, наприклад в порівнянні з Google Maps, можна зазначити відсутність опції побудови маршруту без урахування шосе, поромних переправ. Даний недолік може бути досить актуальним для водіїв вантажних транспортних засобів, які мають досить низькі обмеження максимальної швидкості (не можуть рухатися по автомагістралях).

Для розробників програмного забезпечення Яндекс.Карти надає API, який можна використовувати або за допомогою мови JavaScript або використовуючи протоколи HTTP та HTTPS для відправки запитів та отримання результатів. Результати виконання запитів оформлюється у форматі JSON або XML (формат результату задається в параметрах запиту). Недоліком API Яндекс.Карти є те, що більшість функцій доступні

тільки для використання на мові JavaScript, тобто може використовуватися лише у веб-додатках та сайтах.

Серед функцій доступних через HTTP та HTTPS протоколи можна виділити наступні основні функції:

- Геокодер;
- Пошук по «Народній карті» (пошук координат об'єктів народного господарства по їх назвах та ключових словах);
- Static API Карт (статичні зображення карт, які можна отримати по HTTP протоколу). [7]

Для всіх вищенаведених функцій діють обмеження аналогічні тим, що використовуються в API Google Карт, але з дещо більшими квотами безкоштовного використання для звичайних користувачів: 25 000 запитів за добу, порівняно з 2500, що надає Google. Аналогічно попередній системі всі API є статичними.

Основна перевага – детальні картографічні дані, що постійно оновлюються.

OpenStreetMap – он-лайн ресурс створений спільнотою картографів-любителів з усього світу. Даний проект підтримується великою групою добровольців, що збирають і розповсюджують географічні дані зі всього світу, він містить величезну кількість картографічних даних, об'єктів з усього світу. Не комерційний проект, дані якого можна використовувати для своїх цілей.

Є можливість завантажувати карти материків, окремих країн чи регіонів з офіційного сайту проекту та оброблювати і працювати з ними локально на власному сервері чи комп’ютері [8]. Карти завантажуються у файлі з розширенням osm, що на практиці є звичайним XML файлом з описом картографічних даних.

OpenStreetMap не надає безпосередній програмний інтерфейс для розробників, але існує OSRM – проект з відкритим кодом, який побудований на картах завантажених з OpenStreetMap та дозволяє

розгорнати свій картографічний сервіс на власному сервері. Працювати з таким сервером можна через HTTP протокол, результати обробки запитів сервер надає у форматі JSON.

OpenStreetMap – це набір безкоштовних картографічних даних з усього світу. На офіційному сайті проекту є можливість прокладати маршрут, даний сервіс реалізований за допомогою OSRM серверу, що може використовуватися безкоштовно за умови не перевищення певних обмежень на кількість запитів. В разі не виконання умов безкоштовного використання надання послуг припиняється.

OSRM сервер надає наступні API [9]:

- а) `viaroute` – побудова оптимального (по критерію відстані між об'єктами) маршруту;
- б) `nearest` – пошук найближчої дороги, ділянки шляху до точки з заданими координатами;
- в) `locate` – пошук точки, вузла маршруту, шляху з найближчої до заданої точки;
- г) `table` – обчислення відстаней для точок з заданими координатами;
- д) `match` – знаходження маршруту, що найточніше відповідає послідовності географічних координат (які отримуються наприклад з GPS-приймача).

Очевидна перевага OpenStreetMap в поєднанні з OSRM – відсутність будь-яких обмежень, оскільки використовується власний сервер. Однак, якщо немає змоги або необхідності розгорнати власний сервер можна скористатися безкоштовним сервером проекту OSRM, що знаходиться за наступною адресою <http://router.project-osrm.org>. Але в такому разі діють обмеження, які накладають власники серверу. Також можна скористатися наступними безкоштовними серверами, на яких розгорнуто OSRM сервіс: MapQuest Open, Stamen, CartoDB, TMS. Звичайно, в такому випадку також

діють обмеження встановлені власниками. Існують також і комерційні OSRM-сервери, які надають оплачуване використання сервісу.

Основні переваги OpenStreetMap:

- абсолютно безкоштовні картографічні дані без жодних обмежень;
- високий рівень деталізації, опису географічних об'єктів та частоти оновлення даних, що пояснюється постійним збільшенням кількості учасників проекту;
- детальна та об'ємна документація.

2.2 Висновки до розділу

Всі сервіси побудови оптимального маршруту серед розглянутих мають досить високий рівень деталізації та повноти картографічних даних.

Порівнюючи їх API, можна зробити висновок, що Google Maps мають найширший набір безкоштовних функцій доступний через HTTP протокол, але обмеження в кількості запитів, роблять не можливим використання даної системи для деяких користувачів та організацій.

Яндекс.Карти надають більшу кількість запитів доступних безкоштовно, але набір безкоштовних функцій доступних по HTTP протоколу API значно менший порівняно з Google Maps. Таким чином Яндекс.Карти можуть використовуватися лише обмеженим колом користувачів та установ.

OpenStreetMap – представляє набір географічних даних, доступних без обмежень у використанні, але даний проект не надає API для побудови оптимального маршруту. Вирішенням даної проблеми може бути використання проекту з відкритим кодом побудованого на основі даних OpenStreetMap – OSRM. Однак даний проект має високі системні вимоги,

які необхідні для його використання, а безкоштовні сервери, що надають можливість використання API OSRM, мають обмеження в кількості запитів подібні до Google Maps та Яндекс.Карт.

Використання безкоштовного відкритого проекту OSRM дозволяє уникнути обмежень, що надаються безкоштовними API. Але такий підхід має свої недоліки – необхідність власного серверу для розгортання проекту, оскільки його робота вимагає досить багато обчислювальних ресурсів.

Враховуючи вищезазначені фактори, можна зробити висновок, що найоптимальнішим шляхом вирішення поставленої задачі буде робота з безпосередніми даними отриманими від OpenStreetMap. Для подальшої їх обробки та зберігання доцільним буде використати СКБД. Використання СКБД значно зменшує системні вимоги до системи, оскільки немає необхідності постійно тримати в пам'яті ЕОМ великі об'єми даних, які досить швидко можна отримати з БД у разі необхідності. Також при такому підході зручно виконувати оновлення даних.

3 АНАЛІЗ СТРУКТУРИ КАРТОГРАФІЧНИХ ДАНИХ OpenStreetMap

Дані з OpenStreetMap про певну ділянку земної поверхні завантажуються у файлі з розширенням osm, що за структурою є звичайним XML файлом, що містить опис картографічних даних.

Розглянемо структуру даного файлу детальніше, на рисунку 3.1 наведено основні структурні елементи файлів з розширенням osm.

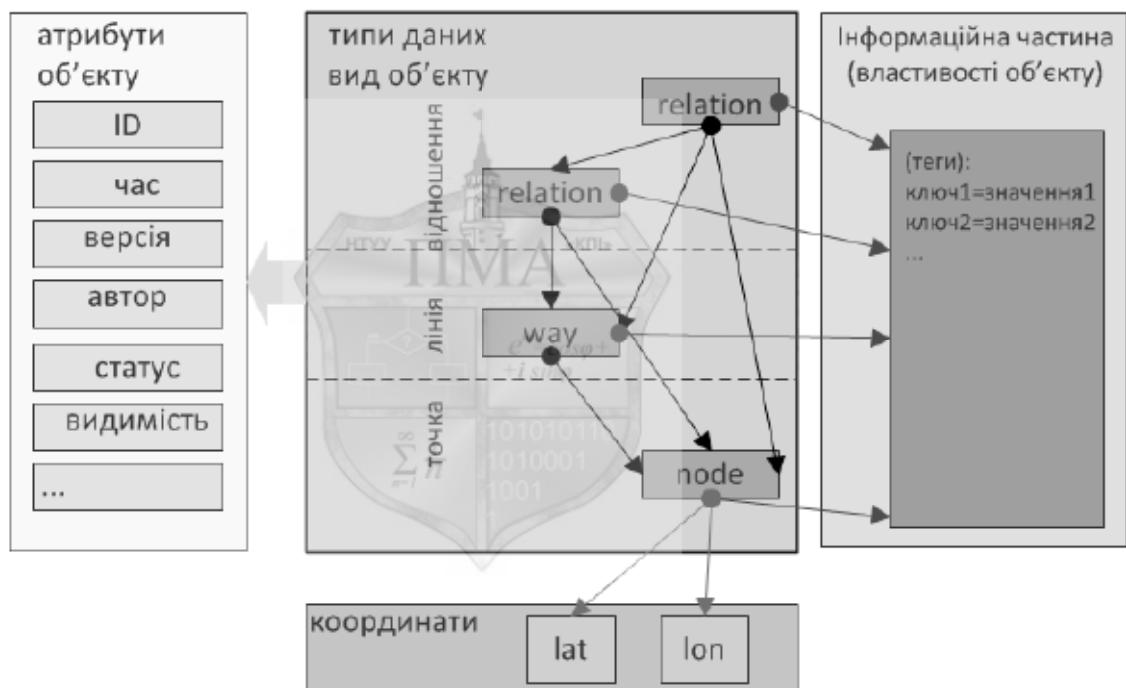


Рисунок 3.1 – Структура osm-файлу

Всі дані можна умовно розбити на три основні групи [10]:

1. Типи даних, що описують у вигляді ієрархічного зв'язку сам об'єкт, як деяку просторову сутність, що має свій кінцевий результат – відомі координати всіх частин об'єкта;
2. Інформаційна частина – це описова характеристика об'єкта, яка не має прямого відношення до просторової географічної структури об'єкта (його назва, фізичні, логічні та інші властивості);

3. Службові атрибути об'єкта, необхідні для організації процесу зберігання і обробки інформації у вигляді набору даних, такі як унікальний ідентифікатор, стан об'єкта в базі, час останньої правки об'єкта в базі і т.д.

3.1 Базові типи географічних даних в OpenStreetMap

Базових типів по суті всього три: точка (node), лінія (way) і відношення (relation). Всі без винятку об'єкти в OSM описуються цими трьома типами даних, після чого наповнюються інформацією за допомогою комбінаціями тегів. Модель даних в OSM будується на ієрархічній структурі посилань, з чого випливає, що будь-який наступний тип даних не містить інформацію, що міститься в попередніх типах а утворює нову сутність, посилаючись на деяку множину об'єктів попереднього типу. Так само слід згадати, що будь-який об'єкт має в структурі даних OSM свій ідентифікатор (ID), унікальний в межах даного типу об'єктів. Саме по цьому ідентифікатору і відбувається посилання на сам об'єкт. Розглянемо структуру базових типів по порядку.

Перший тип: точка (node) – це мінімальний набір даних, який містить в собі інформацію про пару координат: широта, довгота (lat, lon) і є базовим в ієрархічній моделі. Це єдиний тип даних, який безпосередньо зберігає географічну інформацію – координати, у вигляді широти і довготи. Модель даних OSM оперує виключно двомірними даними в межах проекції WGS84. Надалі ми будемо вважати, що координати – це не інформаційна складова об'єкта точки, а невід'ємна частина його структури. У XML нотації osm-файлу, об'єкт даного типу буде виглядати так:

```
<Node id = '19' lat = '58.888047127548994' lon = '49.747870758186764' />
```

Одна точка з унікальним id рівним 19 і парою координат. Координати в OSM використовуються в десятковому записі, оскільки так їх набагато

простіше обробляти ніж формати координат з хвилинами і секундами. Сама по собі точка може бути самостійним об'єктом, що описує якийсь точковий об'єкт (геометричний примітив) або не мати зовсім власної інформаційної складової, а бути частиною іншого об'єкта (лінії або відношення). При цьому, забігаючи трохи вперед, слід зазначити, що точка одночасно може бути і самостійним об'єктом, що несе унікальну інформацію і бути частиною іншого об'єкта.

Другий тип даних: лінія (way) - це сукупність посилань на об'єкти типу точка (node). Як мінімум, лінія складається з однієї точки, тобто повинна містити як мінімум одне посилання на вже існуючий об'єкт типу точка. Лінія з однієї точки не суперечить структурі даних OSM, але суперечить поняттям елементарної геометрії і викликає проблеми при роботі з деякими алгоритмами обробки даних, тому правильна лінія завжди містить як мінімум посилання на два існуючих об'єкта типу точка.

Правильна XML нотація об'єкта типу лінія буде полягати в описі всіх необхідних точок, після чого слідує запис про саму лінію, в якій перераховуються всі її точки. У найпростішому варіанті це буде виглядати так:

```
<Node id = '23' lat = '58.875047918145675' lon = '49.785240674006126' />
<Node id = '22' lat = '58.86687448573524' lon = '49.737090974777324' />
<Way id = '24'>
    <Nd ref = '22' />
    <Nd ref = '23' />
</ Way>
```

Порядок перерахування точок в лінії важливий, він характеризує послідовність точок в лінії і напрямок самої лінії, тобто у лінії завжди є початок і кінець, навіть якщо вона замкнута (в цьому випадку вони просто збігаються). У даному прикладі ми спочатку описали дві точки, задавши їх координати, а потім описали лінію, пославшись на id цих точок. Одна точка може входити в будь-яку кількість об'єктів ліній, при цьому повинна

бути описана тільки один раз, тобто точка може бути загальною для двох ліній, в цьому випадку посилання на неї міститься в обох лініях. Таким чином будується цілісний граф об'єктів (найчастіше дорожній граф для розрахунків маршрутів), який представляє з себе сукупність об'єктів (ліній), що мають зв'язок через їх спільні члени (точки).

Якщо ми хочемо створити ще одну лінію з вже існуючих точок 19 і 23, то ми можемо описати її так:

```
<Way id = '48 '>
<Nd ref = '19 '/>
<Nd ref = '23 '/>
</ Way>
```

Дві вищенаведені лінії з ідентифікаторами 24 та 28 в проекції меркартора графічно представлені на рисунку 3.2.



Рисунок 2.2 – Дві лінії що перетинаються

Наступний тип даних: відношення (relation). По суті всі об'єкти крім точки – вже є відношеннями, проте лінії виділені в окремий тип даних як найбільш поширені, що описують основні геометричні примітиви: лінії, полілінії та полігони. Для всіх більш складних геометричних об'єктів, та для об'єктів, що не є чисто геометричними, а логічними (колекції, списки, ієархії взаємозв'язків) призначений універсальний тип даних – відношення.

В цілому, опис відношень відрізняється від лінії тим, що лінія – це завжди сукупність точок, а відношення – це сукупність будь-яких об'єктів, як точок і ліній, так і інших відношень. Отже у відношеннях вказується не тільки id об'єкта, але і його тип. У найпростішому випадку відношення може містити посилання тільки на один об'єкт. Використовуючи об'єкти описані в прикладах вище, можна описати наступне представлення:

```
<relation id = '31 '>
    <member type = 'way' ref = '24 '/>
    <member type = 'node' ref = '19 '/>
</ relation>
```

Це штучне абстрактне відношення, яке описує, що в нього входить два об'єкти (члени відношення) – точка 24 і лінія 19 та більше не несе жодної інформації. У реальному випадку у відношенні повинен бути вказанний тип, як тег (інформаційна складова) самого об'єкта відношення, а у членів відношення повинні бути зазначені ролі в посиланнях на об'єкти.

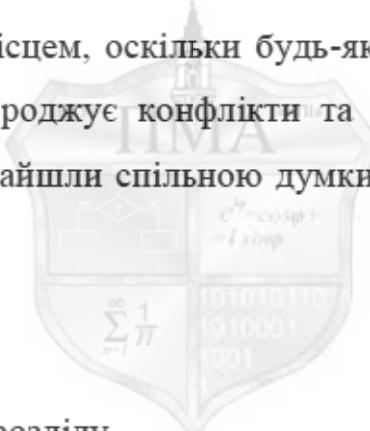
3.2 Інформаційна схема об'єктів

Тип об'єкта описує географічні (просторові) властивості об'єкта, але нічого не говорить про властивості самого об'єкта, його характеристики, призначення та інше. Для цього існує інформаційна частина структури даних OSM, заснована на принципах тегування об'єктів, тобто призначенні їм певних міток і зазначенням властивостей цих міток. Теги задаються у вигляді пари «ключ = значення», що в нотації XML для лінії 24, описаної в попередньому підрозділі виглядає так:

```
<Way id = '24 '>
    <Nd ref = '22 '/>
    <Nd ref = '23 '/>
```

```
<Tag k = 'highway' v = 'primary' />
</ Way>
```

В даному випадку ми додали властивість нашої лінії, а саме вказали тег highway зі значенням primary, що в прийнятій схемі тегування позначає, що наша лінія є основною дорогою (дорога класу нижче магістралі, але вище другорядної). Тегів у будь-якого об'єкта може скільки завгодно багато, що дозволяє задати всі його основні властивості і описати всі другорядні параметри, а так само в довільній формі доповнити об'єкт будь-якою інформацією. Сама схема тегування в OSM є одночасно найголовнішою її архітектурною перевагою, оскільки дозволяє описати фактично будь-які властивості об'єкта, тому що реально ніхто не обмежує вас у виборі нових тегів для нових властивостей об'єктів, та одночасно є найпроблемнішим її місцем, оскільки будь-яка свобода у виборі способів позначення завжди породжує конфлікти та неоднозначності серед груп користувачів, які не знайшли спільною думки, як позначати той чи інший неоднозначний об'єкт.



3.3 Висновки до розділу

Дані з OpenStreetMap описуються в XML-нотації. Опис будь-якого елементу складається з трьох частин: тип об'єкт, інформаційна складова та службова інформація. Точка – мінімальний самостійний тип даних, що має свої координати. Будь-які інші об'єкти в OpenStreetMap описуються на основі точок. Таким чином при конвертації в БД деякого osm-файлу в першу чергу необхідно зберігати всі точки, що містяться в файлі.

4 МАТЕМАТИЧНІ МЕТОДИ РОЗВЯЗКУ ПОСТАВЛЕНОЇ ЗАДАЧІ

Транспортну мережу можна змоделювати зваженим орієнтованим графом, вершини якого – географічні об'єкти та перехрестя транспортних шляхів, а ребра – шляхи між ними. Кожному ребру (дузі) графу можна співставити сумарну вартість руху по відповідній ділянці маршруту.

Таким чином задачу пошуку оптимального маршруту можна інтерпретувати як проблему пошуку найкоротшого шляху у зваженому орієнтованому графі.

4.1 Опис математичної моделі

Граф – це впорядкована пара (V, E) , де V – непуста множина вершин або вузлів, E – множина пар вершин, котрі називають ребрами. Множини V та E вважаються скінченими.

Орієнтований граф – це граф у якого множина ребер E , тобто пар вершин, є впорядкованою. В орієнтованому графі ребра називають орієнтованими або дугами.

Вагова функція – це таке відображення $\omega : E \rightarrow \mathbb{R}$, яке ставить у відповідність ребру або дузі графу деяке дійсне число, яке називається вагою ребра або дуги.

Зважений граф – це граф, у якого кожна дуга або ребро має свою вагу.

Вершина графа a називається досяжною з вершини b , якщо існує така послідовність дуг, яка з'єднує a та b .

Суміжні вершини – це вершини, які з'єднані ребром або дугою [11].

Шлях – така послідовність ребер або дуг графу, що будь-яка пара має спільну вершину.

Шлях з вершини a у вершину b у зваженому графі називається найкоротшим, якщо сума значень вагових функцій дуг, що до нього входять є мінімальною.

Суміжні вершини – це вершини, які з'єднані ребром або дугою [11].

Оскільки математична модель транспортної мережі – зважений орієнтований граф, то необхідно визначити загальний вигляд вагової функції. Вагова функція дуги повинна відображати сукупну вартість руху по ній з урахуванням усіх чинників, що впливають на витрати палива доходи, заощадження від правильного планування поїздки.

Відповідно до постановки задачі система повинна бути розширюваною, мати можливість врахування різних критеріїв при побудові маршруту. Множину критеріїв можна представити у вигляді двійкового вектора \vec{x} , що визначається формулою (4.1).

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, x_i = \begin{cases} 1, & \text{якщо має місце } i - \text{й критерій} \\ 0, & i - \text{й не впливає на вартість руху} \end{cases} \quad (4.1)$$

Визначимо також вектор вагових коефіцієнтів, що характеризую величину впливу відповідного критерію (4.2).

$$\vec{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}, \alpha_i \in R \quad (4.2)$$

Якщо на всій протяжності ділянки маршруту величини α_i є сталими, то вагова функція відповідної дуги e буде мати наступний вигляд

$$\omega(e, \vec{x}, \vec{\alpha}) = \sum_{i=1}^n x_i \cdot \alpha_i. \quad (4.3)$$

Оскільки вектори $\vec{x}, \vec{\alpha}$ визначаються дугою графу, тобто кожна дуга має свою пару цих векторів, надалі вагову функцію будемо розглядати як функцію однією змінної e – дуги графу.

Введемо наступне позначення

$$\omega_i \equiv \omega(e_i), \quad (4.4)$$

тобто ω_i – значення вагової функції відповідної дуги.

Для врахування обмежень пов'язаних з забороною руху по деякій ділянці транспортної мережі та відповідних дугах графу значення вагової функції можна зробити як завгодно великим, тобто $\omega_i = \infty$. При програмній реалізації такому значенню можна спів ставити максимально допустиме числове значення відповідного типу, що підтримується в системі. Таким чином можна врахувати заборону руху по деякій ділянці не змінюючи структуру графу. Інші обмеження аналогічно можна виразити в числовому значенні, яке враховується при обчисленні вагової функції.

Під простим маршрутом з вершини a у вершину b будемо розуміти послідовність дуг графу з урахуванням їх напрямку, тобто деякий шлях, що з'єднує початкову та кінцеву вершину.

Не втрачаючи загальності, при побудові математичної моделі можна розглядати тільки прості маршрути. Якщо маршрут містить проміжні точки, його можна представити як послідовність простих маршрутів.

Нехай

$$r_i = < v_1, v_2, v_3 \dots v_{p+1} > \quad (4.5)$$

деяка впорядкована множина вершин, що представляють шлях з v_1 у v_{p+1} вершину. Також даний шлях можна представити у вигляді впорядкованої множини дуг e_i , позначивши через e_i дугу між v_i та v_{i+1} вершиною.

Тоді r_i можна представити наступним чином

$$r_i = \langle e_1, e_2, e_3 \dots e_p \rangle \quad (4.6)$$

Введемо позначення $R(a,b)$ – множина усіх можливих шляхів з вершини a у вершину b .

Тоді задачу побудови оптимального маршруту з вершини a у вершину b , враховуючи визначення вагової функції дуги (4.3) можна представити наступним чином:

$$\min \sum_{j=1}^p \sum_{i=1}^n x_i \cdot \alpha_i \quad (4.7)$$

Таким чином для заданої початкової та кінцевої точки маршруту необхідно знайти такий шлях r , для якого вищеведений вираз буде приймати мінімальне значення..

4.2 Огляд алгоритмів пошуку на графах

Нехай заданий граф $G = (V,E)$, V – множина вершин, E – множина дуг (ребер). Введемо наступні позначення: n – кількість вершин, m – кількість ребер.

Пошук в ширину (breadth-first search) – один з базових алгоритмів для обходу графа і є основою для багатьох важливих алгоритмів для роботи з

графами. Наприклад, алгоритм Дейкстри використовує ідеї, які подібні до алгоритму пошуку в ширину.

Даний алгоритм може працювати з орієнтованими та неорієнтованими графами. Призначення алгоритму – обхід всіх вершин графу, знаходження мінімального кількості дуг (відстані) від заданої до будь-якої іншої вершини графа. У випадку графів, що мають постійну вагову функцію, знаходить мінімальний шлях від заданої до всіх інших вершин графа.

Оціночний час роботи – $O(n+m)$ [12].

Алгоритм Белмана-Форда. Даний алгоритм знаходить найкоротший шлях з однієї вершини графа до всіх інших у зваженому графі. Алгоритм працює для графів з дугами як додатної так і від'ємної ваги.

Оціночний час роботи – $O(nt)$ [12].

Алгоритм Дейкстри. Даний алгоритм вирішує задачу пошуку найкоротших шляхів із однієї вершини у зваженому орієнтованому графі. Алгоритм працює тільки для графів з дугами невід'ємної ваги.

Оціночний час роботи класичної реалізації даного алгоритму – $O(n^2+m)$ [12].

Для реалізації алгоритму Дейкстри використовується не спадаюча черга з пріоритетами. Існують різні варіанти зменшення часу роботи алгоритму за рахунок вдосконалення реалізації черги з пріоритетами. Наприклад, найкраща відома реалізація даного алгоритму використовує чергу з пріоритетами на основі піраміди Фібоначі, яка забезпечує роботу алгоритму за час $O(n \log n + m)$ [12].

4.3 Схеми збереження графів в пам'яті ЕОМ

Проблема правильного зберігання графа в пам'яті комп'ютера є дуже актуальною. Розглянемо основні методи зберігання графу в пам'яті комп'ютера [12].

Один з найпоширеніших способів зберігання графу – матриця суміжності. Вона представляє собою двовимірний масив. Якщо в елемент матриці з індексами i,j не дорівнює нулю, то це означає, що існує дуга, що виходить з i -ї вершини та закінчується в j -й.

Головний недолік даного методу – нераціональне використання пам'яті для розріджених графів, для збереження інформації про дуги необхідно використовувати додаткові структури даних.

Основна перевага – проста реалізація.

Показники часової складності основних операцій:

- перевірка суміжності вершин x та y – $O(1)$;
- перерахунок всіх вершин суміжних з x – $O(V)$;
- визначення ваги ребра (x,y) – $O(1)$.

Список дуг – спосіб зберігання графа, що базується на базі двовимірного масиву розмірністю $3E$. В першому рядку масиву зберігається інформація, з якої вершини починається дуга, у другому – в якій закінчується, а в третьому – вага дуги.

Переваги методу – порівняно з попереднім усуває недолік зберігання непотрібних нульових значень.

Часові оцінки виконання основних операцій:

- перевірка суміжності вершин x та y – $O(E)$;
- перерахунок всіх вершин суміжних з x – $O(E)$;
- визначення ваги ребра (x,y) – $O(E)$;
- пошук i -ї дуги – $O(1)$.

Списки суміжних вершин. Даний метод зберігання графа часто використовується при розв'язку задач великими обмеженнями по пам'яті та часу. Суть методу полягає в тому щоб для кожної вершини v зберігати номери вершин в які можна потрапити з v та вагу відповідних ребер (дуг).

Основна перевага методу – раціональне використання пам'яті при роботі з розрідженими графами.

Часові оцінки виконання основних операцій:

- перевірка суміжності вершин x та y – $O(E)$;
- перерахунок всіх вершин суміжних з x – $O(E)$;
- визначення ваги ребра (x,y) – $O(1)$;
- пошук i -ї дуги – $O(1)$.

Незважаючи на те, що асимптотичні оцінки часових показників виконання операцій даного методу дорівнюють оцінкам з попереднього, насправді більшість із них будуть виконуватися набагато швидше ніж у попередньому методі. Наприклад, перевірка суміжності вершин x та y та перерахунок всіх вершин суміжних з x в методі зі списком ребер буде гарантовано виконуватися $O(E)$ операцій, тому що необхідно обов'язково пройти увесь список, а у методі суміжних вершин необхідно пройти тільки по вершинам суміжних з x .

4.4 Залежність вибору алгоритму від типу задачі

Розглянемо кілька прикладів транспортних задач, які потрібно розв'язати найефективнішим для конкретної задачі алгоритмом.

Приклад 1. Задано два перехрестя у місті Києві, які знаходяться в межах одно району, також задано, що на даний момент витрати на рух між будь-якими перехрестями можна вважати одинаковими.

Для побудови оптимального маршруту можна скористатися алгоритмом пошуку в ширину, оскільки вагова функція даної транспортної мережі є постійною. Даний алгоритм дозволить вирішити задачу швидше за інші розглянуті алгоритми.

Приклад 2. Побудова оптимального маршруту між двома перехрестями, одне з яких у Запоріжжі, а інше у Києві. Дляожної ділянки між перехрестями задано витрати на рух.

Для побудови оптимального маршруту можна використати алгоритм Дейкстри, оскільки вагова функція графу системи приймає не від'ємні значення. Вибір даного алгоритму обґрунтovується часом його роботи, що при покращеній реалізації можна оцінити величиною $O(n \log n + m)$ [13], де n, m – кількість вершин і дуг графа відповідно.

Приклад 3. Аналогічно до попереднього прикладу необхідно знайти оптимальний маршрут між перехрестями Запоріжжя та Києва. Транспортний засіб рухається з Запоріжжя, куди був відправлений для доставки певного вантажу. Дляожної ділянки між перехрестями задано витрати на рух, однак додатково задаються прибутки, які можна отримати, використавши певний маршрут та транспортувавши супутній вантаж.

Якщо інтерпретувати можливі прибутки від транспортування супутнього вантажу як від'ємні значення вагової функції графу системи для тих ділянок, де прибуток перевищує витрати на рух, можна скористатися алгоритмом Белмана-Форда.

Вибір даного алгоритму пояснюється тим, що тільки він серед розглянутих дозволяє виконати пошук мінімального маршруту з графом, вагова функція якого може приймати від'ємні значення.

4.5 Висновки до розділу

Математична модель транспортної мережі – зважений орієнтований граф. Для задачі пошуку найкоротшого маршруту, найоптимальнішим є використання алгоритму Дейкстри, оскільки вагова функція в даному випадку приймає не від'ємні значення.

Для збереження графу в пам'яті ЕОМ найраціональнішим є представлення графу за допомогою списку суміжних вершин. Даний підхід вимагає менше пам'яті порівняно з іншими способами.



5 ПРОГРАМНА РЕАЛІЗАЦІЯ

Програмна реалізація системи виконана на мові програмування Java восьмої версії. Для роботи з базою даних використовувалася Oracle 10g Express Edition. Данна СКБД представляє собою базову версію, яка є безкоштовною.

5.1 Архітектура системи

Транспортна мережа та її характеристики – основні дані з якими працює система. На основі цих даних створюється граф доріг. Для збереження даної інформації використовується БД. Також завдяки БД є можливість зберігати маршрути користувачів. Таким чином програмне забезпечення, що розроблювалося має клієнт-серверну архітектуру. Спрощена схема даної архітектури представлена на рисунку 5.1.

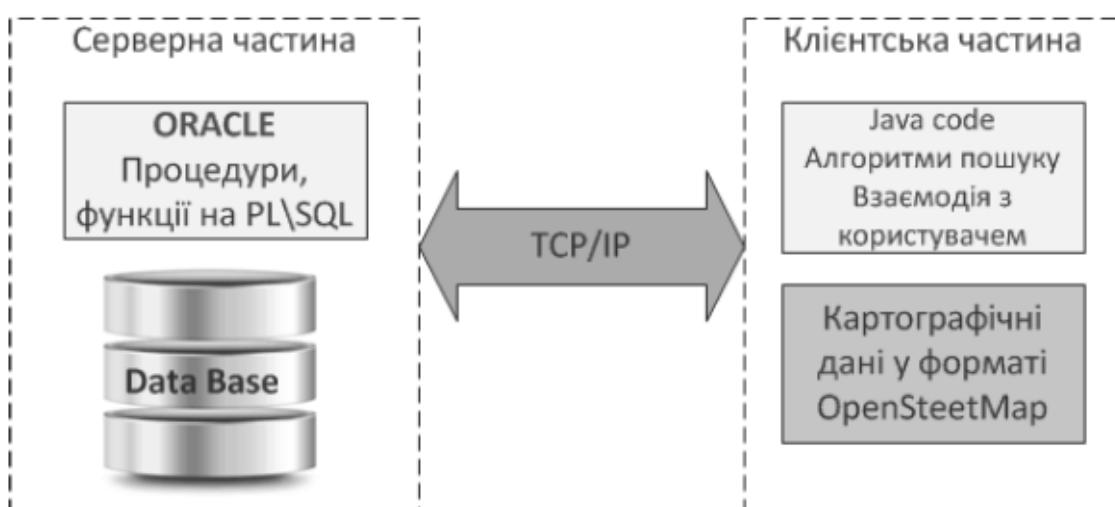


Рисунок 5.1 – Архітектура системи

5.2 Структура модулів програмного забезпечення

Розроблювана система є досить об'ємною та складною. Для спрощення розробки вся система була поділена на декілька модулів. На рисунку 5.2 зображена структурна схема модулів системи, стрілками показані модулі, що взаємодіють.

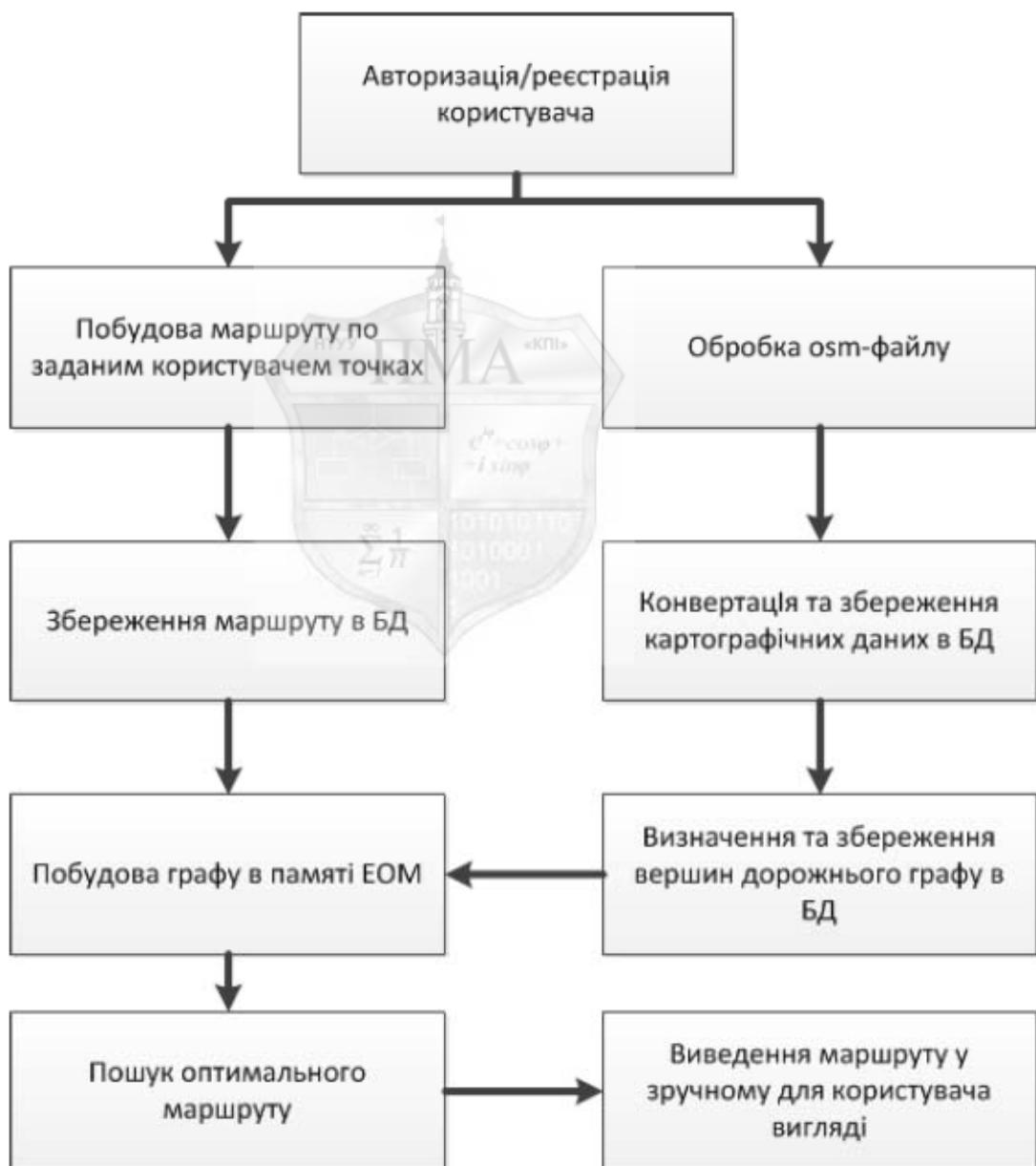


Рисунок 5.2 – Структурна схема модулів системи

5.3 Опис розроблених програмних засобів

Для представлення графу було розроблено наступні класи:

1. Node. Об'єкти даного класу представляють вершини графу. В даному класі інкапсульовано атрибути, які є необхідними для представлення вершини графу та роботи алгоритмів пошуку.
2. Edge. Об'єкти даного класу призначені для роботи з дугами графу.
3. Graph. Об'єкти даного класу інкапсулюють в собі об'єкти двох попередніх класів, та представляють зручний інтерфейс для роботи з графом. Основні методи інтерфейсу дозволяють додавати та видаляти вершини графу, додавати та видаляти дуги графу.

Алгоритми пошуку на графах, які були розглянуті у попередньому розділі реалізовані, як статичні методи класу ShortestMethod.

Для конвертації та збереження даних з osm файлу використовується наступна група класів:

1. Point – клас, об'єкти якого призначені для роботи з даними типу точка з osm файлу.
2. Way, Relation – класи, об'єкти яких призначені для роботи з відповідними даними osm файлу.
3. ParseOSMtoDB – клас, який використовуючи три вищезазначені класи, безпосередньо реалізує функціонал зчитування даних з osm файлу та збереження їх у базі даних.

GraphBuilder – клас, що призначений для зчитування даних з БД та побудови графу транспортної мережі у пам'яті ЕОМ.

GraphComponentCreatorInDB – клас, що призначений для роботи з базою даних, а саме – зчитування картографічних даних, попереднього

збережених БД з osm файлу, обробки цих даних, та збереження у відповідних таблицях БД графу транспортної мережі.

Користувач задає точки маршруту, які описує за допомогою повної адреси деякої будівлі. Дорожній граф будується на основі доріг та перехресть, що задаються послідовностями їх координат. На рисунку 5.3 наведено алгоритм пошуку ділянки дороги, що відповідає будівлі з заданою адресою.

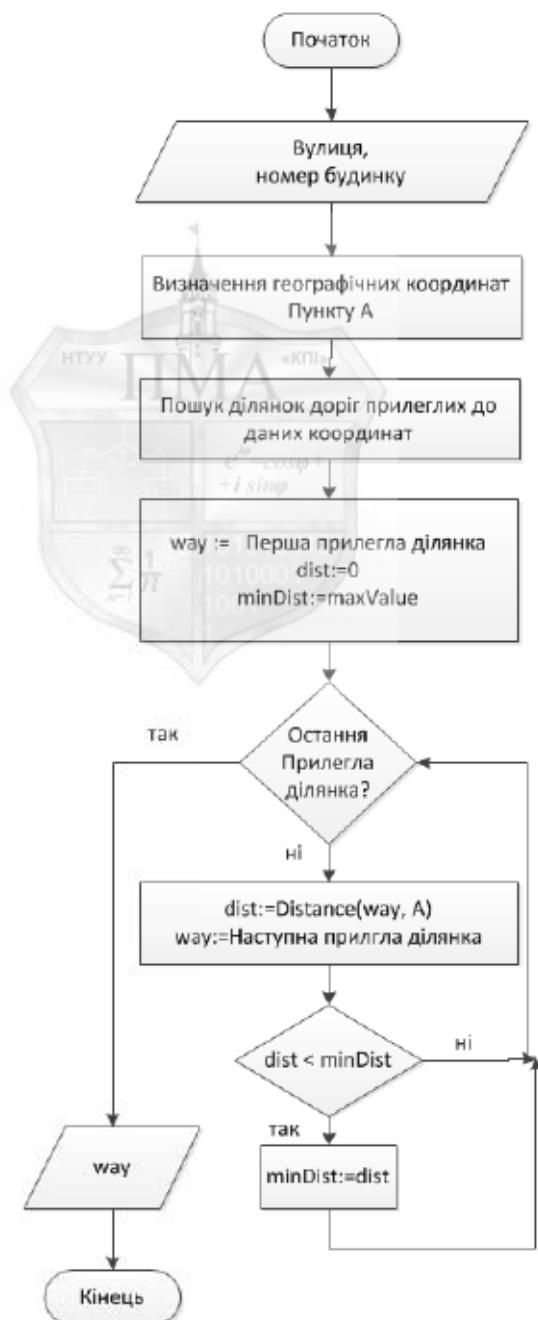


Рисунок 5.3 – Пошук ділянки дороги по адресі будівлі

OptimalPathBuilder – клас, що призначений для побудови оптимального маршруту, що задається послідовністю точок маршруту. В результаті отримується оптимальний маршрут, який задається у внутрішньому форматі системи, та не призначений для відображення користувачу.

PointOfOptimalWay – клас, для роботи з точками, що належать оптимальному маршруту.

WayDecoder – клас, що призначений для конвертації оптимального маршруту з внутрішнього представлення системи у список інструкцій та команд стосовно дотримання оптимального маршруту.

LogInUI, InputRouteUI – класи, що представляють екранні форми для реєстрації та авторизації, введення пошуку оптимального маршруту відповідно. Об'єкти даних класів призначені для взаємодії з користувачем.

LogInController – клас призначений для роботи з екранними формами авторизації та реєстрації в системі. Об'єкт даного класу обробляє дії користувача на екранних формах реєстрації та авторизації.

ComboListener – клас, екземпляри якого займаються обробкою дій користувача при виборі географічних назв при заданні точки маршруту. Основний функціонал – фільтрування географічних назв під час вводу користувачем.

5.4 Опис структури бази даних системи

У Додатку В наведено ERD-діаграму концептуальної моделі бази даних системи. Данна діаграма приведена до четвертої нормальної форми.

Account – сутність для зберігання логіну та паролю користувача.

User – сутність для збереження інформації про тип користувача.

Car – сутність для збереження інформації про автомобіль.

Petrol consumption – сутність для збереження даних про витрати палива автомобіля.

Driver – сутність для збереження інформації про водія.

Vertex – сутність для збереження вершин графу.

Point – сутність для збереження інформації про географічні точки.

Criterion – сутність для зберігання критеріїв, що можуть використовуватися.

Name of user route – сутність, що зберігає назви маршрутів користувачів.

Edge – сутність для збереження дуги графу.

Way structure – сутність, що зберігає структуру множини точок, що утворюють лінію.

Route – сутність для збереження маршрутів користувачів.

Way – сутність для збереження однайменного типу даних з файлу osm.

Info – сутність для збереження інформаційної частини типу даних Way з файлу osm.

Building – сутність для збереження інформації про будівлі.

Equivalent – сутність, яка слугує для побудови відповідності між географічними назвами та їх координатами. Фактично за допомогою даної сутності реалізовано механізм геокодування географічних координат

Country – сутність для зберігання назв країн.

City – сутність, що зберігає назви міст.

Street – сутність для зберігання назв вулиць.

Зв'язки між сутностями та перелік атрибутів сутностей наведені у Додатку В.

5.5 Графічний інтерфейс програми

Розроблене програмне забезпечення складається з чотирьох основних екранних форм:

- Вікно авторизації входу в систему, що зображено на рисунку 5.4;
- Вікно реєстрації нових користувачів, що зображене на рисунку 5.5;
- Основне вікно програми, що слугує для введення маршруту користувача, збереження його в БД, представлена на рисунку 5.6;
- Вікно для відображення оптимального маршруту представлена на рисунку 5.7.

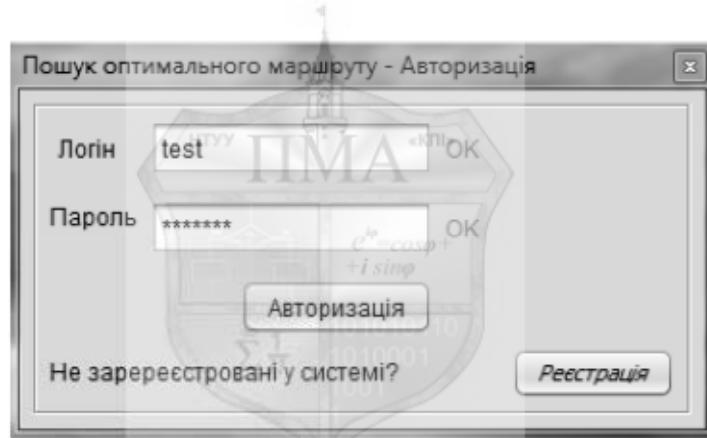


Рисунок 5.4 – Авторизація в системі

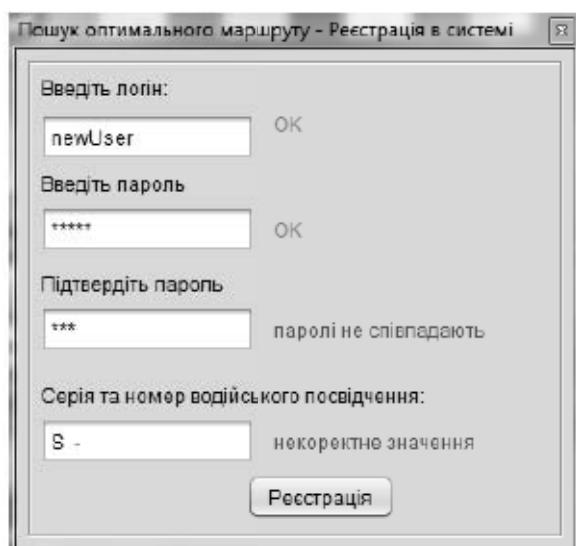


Рисунок 5.5 – Реєстрація нового користувача

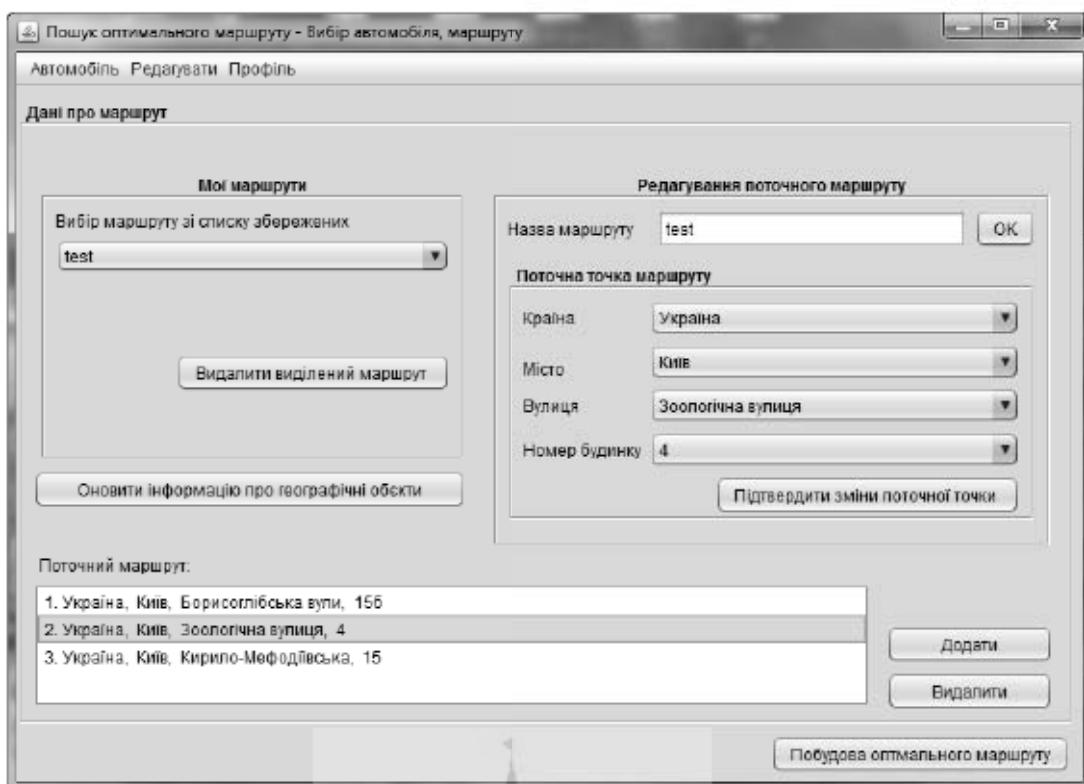


Рисунок 5.6 – Введення маршруту

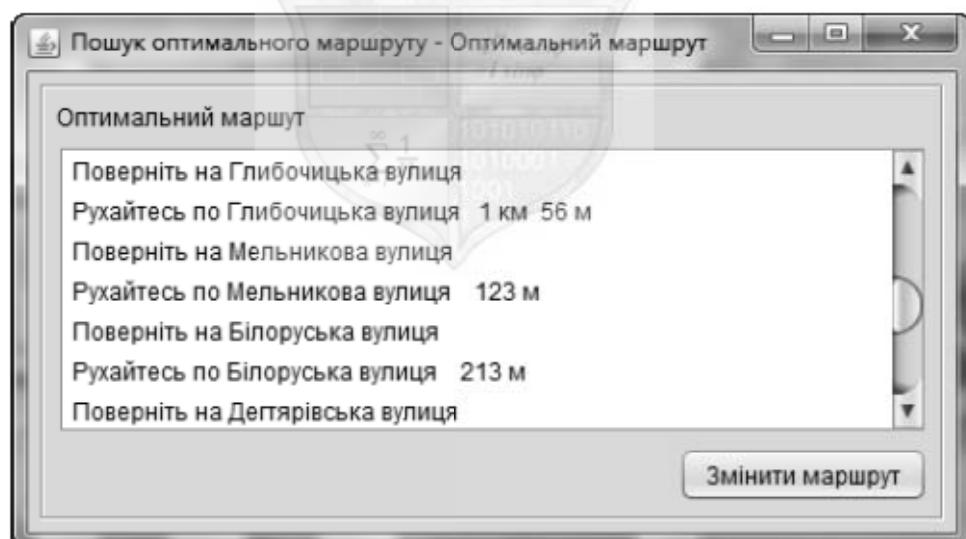


Рисунок 5.7 – Оптимальний маршрут

5.6 Тестування програмного забезпечення

Для перевірки правильності роботи програми розглянемо декілька контрольних прикладів, порівняємо маршрути отримані розробленою програмою з маршрутами отриманими в системі Google Maps.

Контрольний приклад 1.

Початкова точка маршруту – Україна, Київ, Костянтинівська вул, 66.

Кінцева точка маршруту – Україна, Київ, Печенізька вулиця, 1/7.

В результаті побудови оптимального маршруту було отримано наступні інструкції для руху:

Рухайтесь по Костянтинівська вул 474 м

Поверніть на Юрківська вулиця

Рухайтесь по Юрківська вулиця 221 м

Поверніть на Нижньоюрківська вул

Рухайтесь по Нижньоюрківська вул 1 км 6 м

Поверніть на Отто Шмідта вулиця

Рухайтесь по Отто Шмідта вулиця 305 м

Точка №2 досягнута! Поточна довжина маршруту: 2 км 6 м

Таким чином в результаті роботи програми було побудовано маршрут довжиною 2 км 6 м.

Маршрут побудований в системі Google Maps проходить через ті самі точки та має довжину 2,1 км.

Переглянути екранні форми даного контрольного прикладу можна в Додатку Г на рисунках 1-2. Порівняння роботи програми з роботою Google Maps знаходиться в Додатку Д на рисунках 1-2.

Контрольний приклад 2.

Початкова точка маршруту – Україна, Київ, Щекавицька вулиця, 53.

Кінцева точка маршруту – Україна, Київ, Нижній Вал вулиця, 19.

В результаті побудови оптимального маршруту було отримано наступні інструкції для руху:

Рухайтесь по Щекавицька вулиця 173 м

Поверніть на Почайнинська вулиця

Рухайтесь по Почайнинська вулиця 296 м

Поверніть на Нижній Вал вулиця

Рухайтесь по Нижній Вал вулиця 748 м

Точка №2 досягнута! Поточна довжина маршруту: 1 км 21 м

В результаті роботи програми було побудовано маршрут загальною довжиною 1 км 21 м.

Маршрут побудований в системі Google Maps проходить через ті самі точки та має довжину 1,1 км.

Переглянути екранні форми даного контрольного прикладу можна в Додатку Г на рисунках 3-4. Порівняння роботи програми з роботою Google Maps знаходиться в Додатку Д на рисунку 3.

Контрольний приклад 3.

Початкова точка маршруту – Україна, Київ, Олени Теліги вулиця, 7.

Кінцева точка маршруту – Україна, Київ, Ванди Василевської, 10. В результаті побудови оптимального маршруту було отримано наступні інструкції для руху:

Рухайтесь по Олени Теліги вулиця 590 м

Поверніть на Олександра Довженка

Рухайтесь по Олександра Довженка 135 м

Поверніть на ділянка без назви

Рухайтесь по ділянка без назви 35 м

Поверніть на

Рухайтесь по 59 м

Поверніть на ділянка без назви

Рухайтесь по ділянка без назви 55 м

Поверніть на Дегтярівська вулиця
 Рухайтесь по Дегтярівська вулиця 1 км 95 м
 Поверніть на Довнар-Запольського
 Рухайтесь по Довнар-Запольського 682 м
 Поверніть на Ванди Василевської
 Рухайтесь по Ванди Василевської 572 м
 Точка №2 досягнута! Поточна довжина маршруту: 4 км 8 м.
 В результаті роботи програми було побудовано маршрут загальною довжиною 4 км 8 м.

Маршрут побудований в системі Google Maps проходить через ті самі точки та має довжину 4,0 км.

Переглянути екранні форми даного контрольного прикладу можна в Додатку Г на рисунках 5-6. Порівняння роботи програми з роботою Google Maps знаходиться в Додатку Д на рисунку 4.

У всіх розглянутих контрольних прикладах робота розробленого програмного забезпечення була коректною, такою що відповідає роботі сервісу Google Maps. Не значні розбіжності в довжині маршруту можна пояснити дещо різними координатами заданих початкової та кінцевої точки, оскільки розроблена система та сервіс Google Maps використовує картографічні дані з різних джерел.

5.7 Висновки до розділу

Розроблена система представляє має клієнт-серверну архітектуру. Програмне забезпечення поділено на декілька модулів.

Тестування системи показало, що вона працює коректно, відповідає поставленим вимогам та може використовуватися відповідними користувачами.

ВИСНОВКИ

В даній роботі було розглянуто існуючі картографічні системи, програмні інтерфейси цих систем. Порівнявши API найвідоміших он-лайн сервісів пошуку оптимального маршруту, визначено, що найбільш гнучкою та безкоштовною, без обмежень у використанні є система OpenStreetMap, яка дозволяє завантажувати карти будь-яких регіонів планети та використовувати їх дані.

Також в роботі було розроблено математичну модель транспортної мережі та досліджено алгоритми пошуку оптимального транспортного маршруту.

Використовуючи дані отримані з системи OpenStreetMap та розроблену математичну модель було спроектовано та побудовано програмне забезпечення для пошуку найкоротшого маршруту транспортного засобу. Для реалізації програмного забезпечення використовувалася мова Java. Розроблена система включає в себе роботу з базою даних, взаємодія з якою реалізована за допомогою СКБД ORACLE 10 g XE.

Розроблена система може використовуватися безкоштовно та не має обмежень на кількість запитів для побудови маршруту. Дано особливість системи є значною перевагою, оскільки є багато організацій, що під час своєї роботи потребують великої кількості запитів для побудови оптимального маршруту.

Розроблений програмний продукт тестувався на кількох контрольних прикладах. В результаті тестування було з'ясовано, що робота програмного продукту є коректною, тому він може використовуватися водіями та організаціями, які в своїй роботі використовують сервіси побудови оптимально маршруту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Офіційна документація Google Maps. Створення маршруту та планування поїздок [Електронний ресурс]

<https://support.google.com/maps/#topic=3292869>

2. Офіційна документація Google Maps. Веб-служби API Google Карт [Електронний ресурс]

<https://developers.google.com/maps/documentation/webservices>

3. Офіційна документація Google Maps. API маршрутів Google [Електронний ресурс]

<https://developers.google.com/maps/documentation/directions>

4. Офіційна документація Google Maps. API матрици розстояній Google [Електронний ресурс]

<https://developers.google.com/maps/documentation/distancematrix>

5. Офіційна документація Google Maps. API висотних даних Google [Електронний ресурс]

<https://developers.google.com/maps/documentation/elevation>

6. Офіційна документація Google Maps. API геокодування Google [Електронний ресурс]

<https://developers.google.com/maps/documentation/geocoding>

7. Офіційна документація Яндекс.Карти. API для розробників [Електронний ресурс] <https://tech.yandex.ru/maps/>

8. OpenStreetMap [Електронний ресурс]

<https://www.openstreetmap.org>

9. Project-OSRM. Server api [Електронний ресурс]

<https://github.com/Project-OSRM/osrm-backend/wiki/Server-api>

10. Структура даних проекта OpenStreetMap, 25 червня, 2012 [Електронний ресурс] <http://habrahabr.ru/post/146503/>

11. Таран Т.А. Основы дискретной математики – Киев: «Просвіта», 2003. – 288 с.
12. Кормен, Томас Х. и др. Алгоритмы: построение и анализ, 3-е изд.:Пер. с англ. – М. : ООО «И. Д. Вильямс», 2013. – 1328 с.: ил. – Парал. тит. англ.

