Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Системотехніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____
Розробка та аналіз інформаційної системи супроводження роботи маршрутно-кваліфікаційних комісій з планування гірських туристичних маршрутів
(тема)

Виконав:
здобувач _____2_____ року навчання,
групи _____ІТПм-24-2_____
_____Віктор Кравченко_____
(власне ім'я, прізвище)

Спеціальність _122 Комп'ютерні науки_____
_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма __Інформаційні технології проектування_____
(повна назва освітньої програми)

Керівник___доц. каф. СТ Віктор Решетнік___
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри __СТ____ _____ ___Ігор Гребеннік__
(підпис) (власне ім'я, прізвище)

2025 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«18» грудня 2025 р. Кравченко В. Д

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено «20» грудня 2025 р.

Керівник кваліфікаційної роботи  *доц. каф СТ Решетнік В. М*

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерних наук _____

Кафедра_____Системотехніки _____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність_____122 Комп'ютерні науки_____
(код і повна назва)

Тип програми _____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_____Інформаційні технології проектування_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Кравченку Віктору Дем'яновичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка та аналіз інформаційної системи супроводження роботи маршрутно-кваліфікаційних комісій з планування гірських туристичних маршрутів*
затверджена наказом університету від 24 листопада 2025 р. № 1058 СТ
2. Термін подання здобувачем роботи до екзаменаційної комісії 18 грудня 2025 р.
3. Вихідні дані до роботи *дослідити підходи для планування маршрутів на основі точок інтересу визначених користувачем, провести аналіз та визначити оптимальні алгоритми для використання у запропонованому підході. Перелік використовуваних програмних засобів: ОС Windows 10, середовище розробки Visual Studio Code*
4. Перелік питань, що потрібно опрацювати в роботі *4.1 Аналіз предметної області, яка визначає діяльність організації 4.2 Аналіз реалізованих систем-аналогів 4.3 Огляд існуючих підходів, методів, засобів, технологій, інструментарію для розв'язання подібних задач 4.4 Постановка задачі 4.5 Розробка системних вимог до системи, 4.6 Розробка функціональних вимог до розроблюваної системи 4.7 Розробка діаграми потоків даних системи 4.8 Розробка діаграми варіантів використання системи 4.9 Розробка діаграми класів системи 2.6 Розробка діаграми послідовності дій системи 4.10 Опис архітектури розроблюваної системи 4.11 Логічне та фізичне моделювання даних системи 4.12 Створення бази даних на платформі СУБДMySQL 4.13 Опис розробленої карти сайту 4.14 Розробка алгоритмів виконання бізнес-функцій 4.15 Розробка тригерів для серверної частини системи 4.16 Опис рішень прийнятих при розробці програмного коду системи 4.17 Порівняльний аналіз алгоритмів пошуку найкоротшого шляху на графі 4.18 Порівняльний аналіз алгоритмів розв'язання задачі орієнтування 4.19 Рекомендації стосовно вибору алгоритмів. 4.20 Рекомендації стосовно подальшого розвитку описаного підходу*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій *5.1. Реалізовані аналоги системи 5.2 Контекстна діаграма системи 5.3 Декомпозиція контекстної діаграми 5.4 Діаграма варіантів використання 5.5 Діаграма*

класів 5.6 Перший етап алгоритму розв'язання задачі 5.7 Другий етап алгоритму розв'язання задачі 5.8 Архітектура системи 5.9 Логічна модель даних 5.10 ERR-модель бази даних 5.11 Демонстрація роботи системи 5.12 Графік залежності часу виконання алгоритмів від кількості вершин у графів 5.13 модифікація запропонованого підходу для врахування кількох можливих точок старту та фінішу 5.14 модифікація запропонованого підходу для врахування часу на відвідування точки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	13.10.2025	Виконано
2	Аналіз предметної області, формулювання мети та задач дослідження	13.10.2025–20.10.2025	Виконано
3	Аналіз реалізованих систем-аналогів	20.10.2025–26.10.2025	Виконано
4	Дослідження існуючих підходів, методів, засобів, технологій для розв'язання задачі	26.10.2025–03.11.2025	Виконано
5	Визначення підходу для розв'язання поставленої задачі	03.11.2025–07.11.2025	Виконано
6	Проектування інформаційної системи	07.11.2025–13.11.2025	Виконано
7	Реалізація інформаційної системи	13.11.2025–22.11.2025	Виконано
8	Аналіз алгоритмів для розв'язання поставленої задачі	22.11.2025–15.12.2025	Виконано
9	Оформлення графічної частини та презентаційних матеріалів	15.12.2025	Виконано
10	Представлення на рецензування	15.12.2025	Виконано
11	Подання кваліфікаційної роботи до ЕК	18.12.2025	Виконано

Дата видачі завдання _13__ __жовтня_____2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _ _доц. каф СТ Віктор Решетнік_
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи: 72 ст., 29 рис., 2 додатки, 25 джерел інформації.

ЗАДАЧА ОРІЄНТУВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, МАРШРУТ, МАРШРУТНО-КФАЛІФІКАЦІЙНА КОМІСІЯ, МОДИФІКАЦІЯ, МУРАШИНИЙ АЛГОРИТМ, ТУРИЗМ

Об'єкт досліджень – алгоритми планування гірських туристичних маршрутів та інформаційні системи реєстрації туристичних походів за спланованими маршрутами в маршрутно-кваліфікаційних комісіях.

Предмет досліджень – алгоритми пошуку оптимальних маршрутів у графах з часовими обмеженнями та засоби їх застосування в інформаційних системах реєстрації туристичних походів.

Мета досліджень – визначення алгоритму побудови оптимального шляху на основі бажань користувача й часового обмеження та розробка серверної та клієнтської частини інформаційної системи для роботи маршрутно-кваліфікаційних комісій з функціоналом реєстрації походів

Методи дослідження – системний підхід, вивчення існуючих джерел, порівняльний аналіз часу виконання та критеріїв оптимальності, функціональне та об’єктно-орієнтоване проектування.

У роботі розглянуто підходи до процесу планування маршрутів походу, визначено підхід, який дозволить будувати оптимальний шлях на основі бажань користувача та часового обмеження, визначено алгоритми для виконання кожного етапу планування. Визначено вимоги та розроблено інформаційну систему, яка виконує планування маршрутів за визначеним підходом та надає функціонал реєстрації походів.

Сфера застосування – планування маршрутів походів та підтримка роботи маршрутно-кваліфікаційних комісій у сфері реєстрації походів.

ABSTRACT

Master’s thesis: 72 pages, 29 figures, 2 appendices, 25 references.

ORIENTEERING PROBLEM, INFORMATION SYSTEM, ROUTE, ROUTE-QUALIFICATION COMITEE, MODIFICATION, ANT COLONY ALGORITHM, TOURISM

Research object – algorithms for planning mountain tourist routes and information systems for registering tourist trips along planned routes in route qualification commissions.

Research subject – information technologies and software methods for creating the server and client parts of the information system for automating the accounting of regional sports events.

Aim of research – algorithms for searching for optimal routes in graphs with time constraints and means of their application in information systems for registering tourist routes.

Research methods – system approach, study of existing sources, comparative analysis of execution time and optimality criteria, functional and object-oriented design.

The paper considers approaches to the process of planning hiking routes, defines an approach that will allow building the optimal path based on user wishes and time constraints, defines algorithms for performing each planning stage. Requirements are defined and an information system is developed that performs route planning according to the defined approach and provides functionality for registering hikes.

Application domain – planning hiking routes and supporting the work of route qualification commissions in the field of registering hikes.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 7
ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1 Аналіз предметної області, яка визначає діяльність організації 9
1.2 Аналіз реалізованих систем-аналогів 12
1.3 Огляд існуючих підходів, методів, засобів, технологій, інструментарію для розв'язання подібних задач 14
1.4 Постановка задачі 18
2 РОЗРОБКА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ 20
2.1 Розробка системних вимог до інформаційної системи 20
2.2 Розробка функціональних вимог до інформаційної системи 20
2.3 Розробка діаграми потоків даних системи 28
2.4 Розробка діаграми варіантів використання інформаційної системи 30
2.5 Розробка діаграми класів системи 32
2.6 Розробка діаграми послідовності дій системи 34
2.7 Визначення алгоритму розв'язку задачі 37
2.8 Визначення та опис алгоритмів для розв'язання задачі 40
3 ОПИС ПРИЙНЯТИХ РІШЕНЬ ПРИ РОЗРОБЦІ СИСТЕМИ 44
3.1 Опис архітектури (структури) інформаційної системи 44
3.2 Логічне та фізичне моделювання даних системи 45
3.3 Створення бази даних на платформі СУБД MySQL 50
3.4 Опис розробленої карти сайту 53
3.5 Розробка алгоритмів виконання бізнес-функцій 54
3.6 Розробка тригерів для серверної частини системи 56
3.7 Опис рішень прийнятих при розробці програмного коду системи 58
4 ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ ВИРІШЕННЯ ЗАДАЧІ 63
4.1 Порівняльний аналіз алгоритмів пошуку найкоротшого шляху на графі 63
4.2 Порівняльний аналіз алгоритмів розв'язання задачі орієнтування 64

4.3 Рекомендації стосовно вибору алгоритмів 68
4.4 Рекомендації стосовно подальшого розвитку описаного підходу 69
ВИСНОВКИ 73
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 75
ДОДАТОК А Графічний матеріал кваліфікаційної роботи 78
ДОДАТОК Б Текст програми 93

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІС – інформаційна система

МКК – маршрутно-кваліфікаційна комісія

СУБД – система управління базами даних

SQL – structured query language

ВСТУП

Планування маршруту є основним етапом підготовки туристичного походу та є складним процесом, оскільки потребує визначення точок інтересу, які група встигне відвідати у заданий час, а також врахування послідовності їх відвідування. Ручне розв’язання цього завдання часто є трудомістким і часто призводить до неефективного використання доступного для походу часу. Використання алгоритмів оптимізації дає змогу автоматизувати побудову маршруту, обираючи оптимальну послідовність відвідування обраних точок з урахуванням часового обмеження, що дозволить більш ефективно використати наявний на похід час й відвідати більшу кількість запланованих місць. Система виконує дві основні функції: побудова маршруту на основі місць, які хоче відвідати користувач, і наявного в користувача часу на проходження походу та реєстрація походу в маршрутно-кваліфікаційних комісіях. Система дозволить автоматизувати облік проведених і запланованих походів, результатів походів та інформацію про складність, тривалість проходження та опис ділянки шляху, яка може бути присутня в маршруті. На основі заданих користувачем точок інтересу та часового обмеження система виконує побудову маршруту, який буде проходити через усі або частину заданих точок, не перевищуватиме заданий час та найбільше задовольнить користувача. Це дозволяє спростити процес планування маршруту, завдяки чому в походи може бути залучено більше туристів.

Задача кваліфікаційної роботи – визначення алгоритму побудови маршруту за обраними користувачем точками й заданим часовим обмеженням, аналіз існуючих та вибір оптимальних алгоритмів для виконання побудови маршруту, а також розробка компонентів інформаційної системи супроводження роботи маршрутно-кваліфікаційних комісій.

Цільова предметна область – маршрутно-кваліфікаційні комісії та організації, які займаються плануванням та проведенням походів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області, яка визначає діяльність організації

Гірський туризм є одним із видів активного відпочинку, який полягає у пішому пересуванні гірською місцевістю за визначеним маршрутом. Гірський туризм може мати різні цілі: огляд пам’яток природи, відпочинок від міського шум, випробування своїх власних сил в екстремальних умовах, тощо. Для того, щоб вирушити в похід, необхідно спланувати маршрут, мати спорядження, досвід, гарну фізичну форму й офіційну реєстрацію походу. Планування маршруту походу виконується туристом, який хоче піти в похід, на основі його вподобань та з урахуванням логістичних (можливість дістатися до місця старту й фінішу маршруту) та часових обмежень. Офіційною реєстрацією походів займаються маршрутно-кваліфікаційні комісії (МКК). МКК є органом туристично-спротивних організацій, що підпорядковується федерації спортивного туризму України й виконує обов’язки, що включають [1]:

– розгляд маршрутів походів та їх затвердження. МКК надає консультації з вибору маршруту для групи в залежності від досвіду її учасників і керівника та надає висновок стосовно категорії складності маршруту;

– офіційне реєстрування походів. МКК надає дозвіл на проходження маршруту й видає маршрутну книжку – документ, який містить детальну інформацію про маршрут походу, умови його проходження й склад та досвід учасників походу;

– розгляд звітів про пройдений похід. МКК перевіряє звіт про пройдений похід, й на основі результатів перевірки присвоює учасникам походу спортивні розряди у сфері спортивного туризму;

– облік проведених походів, паспортизація (опис на визначення складності) перешкод та ведення архіву маршрутних документів.

Основними нормативно правовими актами, які регулюють сферу

туризму і роботу МКК є Закон України «Про туризм» [2], який визначає загальні правові, організаційні та соціально-економічні засади реалізації державної політики України в галузі туризму та положення про туристські маршрутно-кваліфікаційні комісії [3] визначає мету, загальні принципи, організаційну структуру, а також порядок діяльності МКК як колегії суддів змагань туристських спортивних походів.

Реєстрація походу і сам похід неможливий без спланованого маршруту. Для планування маршруту необхідно визначити точки старту та фінішу маршруту, тривалість і основні ключові точки походу. Ключовими точками є місця, які турист хоче відвідати й через які має проходити маршрут. Такими місцями можуть бути гірські вершини, пам’ятки природи, історичні пам’ятки, тощо. Під час планування необхідно визначити такий маршрут, який буде проходити через усі або частину обраних ключових точок і буде вкладатися у встановлене часове обмеження. Основними проблемами під час побудови маршруту є визначення ключових точок, на відвідування яких вистачить часу, визначення послідовності відвідування цих точок і визначення маршруту між ними. Ці проблеми є актуальними як для туристів-початківців, так і для досвідчений туристів.

Після виконання планування маршруту турист має зареєструвати похід у МКК. Реєстрація походу покращує безпеку походу шляхом інформування контрольно-пошукової служби про маршрут та терміни проходження кожної його ділянки. У разі виникнення надзвичайних ситуацій рятувальники матимуть інформацію про приблизне місцеперебування групи, що дозволить їм швидше знайти туристів та надати їм необхідну допомогу. Реєстрація походу також надає можливість отримати дозволи на перебування в прикордонних зонах, через які проходить маршрут. Для реєстрації походу турист подає в МКК документи, серед яких є опис маршруту, мапа маршруту, інформація про учасників (якщо похід виконується групою туристів), графік руху та документи, які підтверджують туристичний досвід кожного учасника походу. Вся інформація про похід заноситься в маршрутну

книжку. МКК перевіряє маршрут й кваліфікацію групи й робить один з трьох висновків:

– затвердження походу, якщо у МКК не виникло жодних претензій до заявленого походу. Похід офіційно реєструється й видається маршрутна книжка;

– повернення заявки на реєстрацію походу на доопрацювання, якщо МКК виявила помилки в наданих документах або при побудові маршруту.

– відмова у реєстрації походу, якщо МКК виявила серйозні порушення, які не можуть бути виправлені.

Перед початком походу керівник групи має повідомити контрольно-рятувальну службу про початок походу, а також повідомляти цю службу про проходження контрольних точок маршруту та про його завершення. У разі, якщо група не дійде до контрольної точки впродовж часу, заявленого в маршрутній книжці, й не вийде на зв’язок зі службою, то буде розпочато пошукову операцію.

Після завершення походу керівник створює звіт про похід, який підтверджує проходження походу учасниками й виконує його захист в МКК. На основі захисту МКК робить висновки стосовно зарахування походу і надання спортивних звань учасникам. Створення звіту і його захист не є обов’язковими.

Хоча реєстрація походу дає велику кількість переваг, багато туристів ігнорують її через відсутність зручних механізмів виконання цього процесу. Для виконання цієї процедури необхідно прийти в МКК особисто або звернутися електронним листом, що є тривалим й незручним процесом. Також планування маршруту походу є складним процесом, що зменшує привабливість походів для туристів-новачків. Саме тому актуальність роботи полягає у спрощенні процесу реєстрації походів, що дозволить залучити до туризму більшу кількість людей, а також створенні інструментів планування маршруту, які дозволять будувати оптимальні маршрути для туристів на основі їх уподобань та наявного часу на похід.

1.2 Аналіз реалізованих систем-аналогів

На сьогодні вже існують інформаційні системи (ІС), які розв’язують частину проблем, описаних в попередньому розділі. Однією з таких ІС є вебсайт «В похід Карпатами» [4], інтерфейс якої подано на рисунку 1.1.

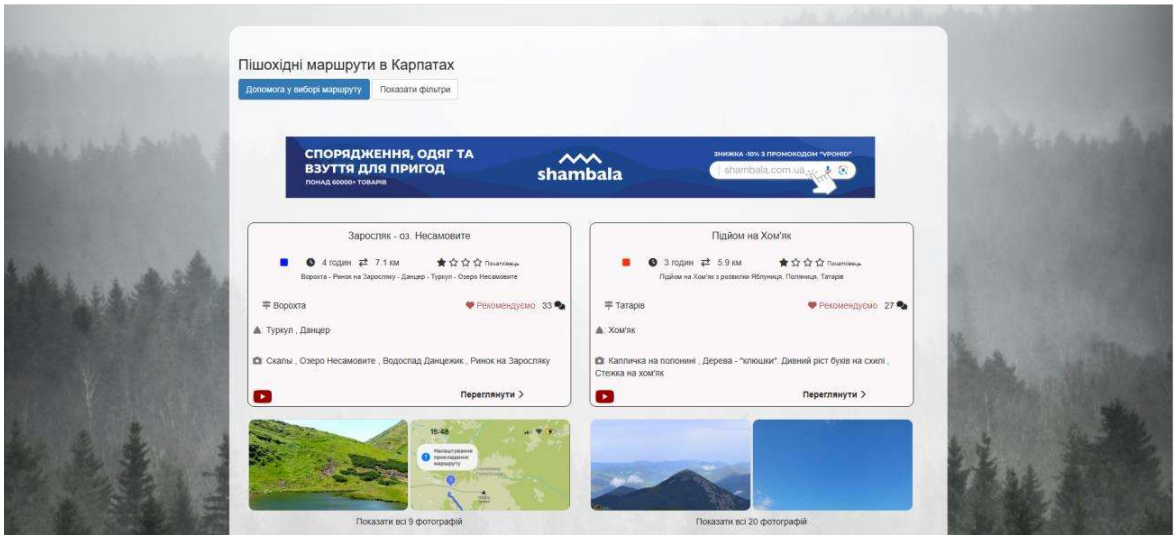


Рисунок 1.1 – Інтернет-ресурс «В похід Карпатами»

ІС надає користувачам можливість переглядати інформацію про різні цікаві точки, опис й фотографії маршрутів, карту із позначеними вершинами, джерелами, промаркованими маршрутами, покриттям мережами мобільного зв’язку, пунктів рятувальних служб, тощо. Також система дозволяє розміщувати оголошення про надання послуг гіда для проходження певних маршрутів. Після реєстрації у цій системі користувач отримує доступ до форуму, де він може знайти групу для походу чи задати питання більш досвідченим туристам. Недоліком системи є відсутність можливості створити свій маршрут на основі своїх точок інтересу: користувач може вибрати маршрут лише з тих, які присутні на сайті. Якщо маршруту, який містить усі цікаві для користувача точки, на сайті немає – сайт зможе надати лише детальну карту Карпат для допомоги в плануванні маршруту. Також ІС не пов’язана з МКК, через що виникає необхідність реєструвати похід з

використанням інших систем. Але оскільки в Україні не заборонено ходити в незареєстровані походи (якщо не проходить через прикордонні чи інші зони, для яких необхідний дозвіл на перебування в них), то користувач може знехтувати реєстрацією, тим самим наражаючи себе на більший ризик у разі надзвичайної ситуації.

Інша система, яка дозволяє виконувати схожі функції – вебсайт komoot.com [5]. Інтерфейс ІС подано на рисунку 1.2. ІС позиціонує себе як платформа для планування, навігації та обміну досвідом у походах, на велосипеді, трейлах, бігу та інших активностях. Вона надає можливість переглядати маршрути різних користувачів, а також підбирає користувачу маршрути відповідно до його уподобань (регіон, тип пересування, тощо). Для більш зручного перегляду система надає можливість фільтрувати маршрути за дальністю, складністю та початковою точкою. Ключовим функціоналом системи є інтерактивна карта, на якій є інформація про тип покриття (асфальт, ґрунт, каміння тощо), складність, нахил, висоту для кожного відрізку маршруту. Для кожного відрізку шляху можна переглянути фотографії, зроблені на ньому іншими користувачами.

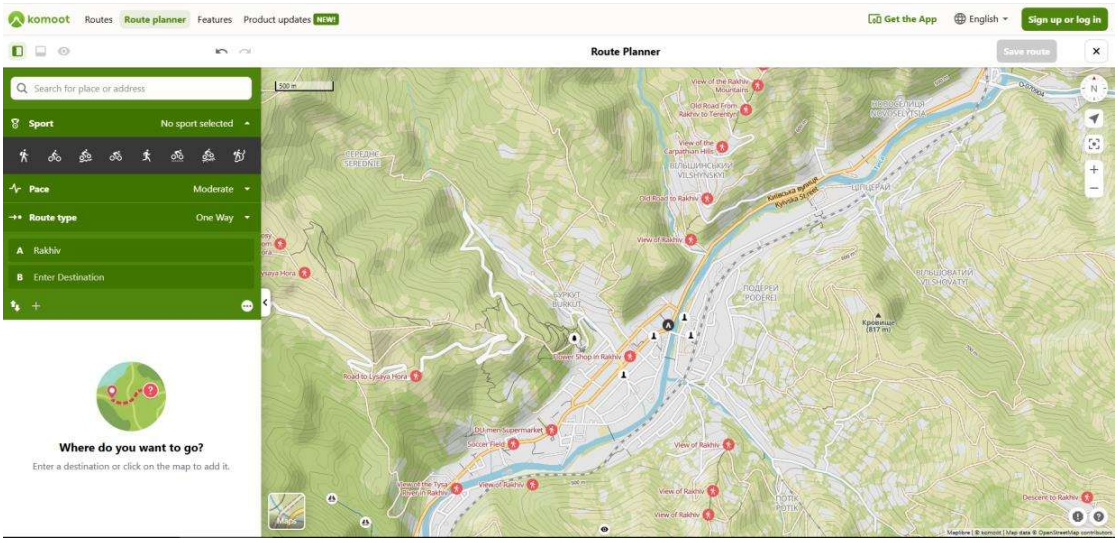


Рисунок 1.2 – Інтернет-ресурс komoot.com

За допомогою інструментів планування ІС надає можливість

спланувати власний маршрут, який буде проходити через задані точки. Недоліком системи є відсутність оптимізації послідовності обходу ключових точок, який буде проходити через усі вказані користувачем точки у визначеній користувачем послідовності. ІС не надає функціоналу для реєстрації походів.

Ще одним реалізованим аналогом системи, що розробляється, є ІС RouteXL [6]. На відміну від попередніх ІС, дана система виконує оптимізацію маршруту за часом: користувач вводить стартову й фінішну адресу та адреси місць, які він має відвідати. На основі цих даних система визначає послідовність відвідувань заданих місць, а також найшвидший шлях між ними. Система створена для використання водіями для підвищення ефективності роботи за рахунок зменшення часу перебування в дорозі. Туристи не можуть використовувати цю ІС, оскільки вона дозволяє планування маршрутів лише за точками, що знаходяться в населених пунктах та мають адресу. Варто зазначити, що алгоритм побудови маршруту цієї ІС передбачає відвідування усіх точок за мінімальний час, а не відвідування усіх або частини точок без перевищення заданого часового ліміту. Тому побудову маршруту походу з використанням такого алгоритму можна було б лише у випадках, коли часу вистачить на відвідування усіх обраних точок.

1.3 Огляд існуючих підходів, методів, засобів, технологій, інструментарію для розв’язання подібних задач

Побудова маршруту має здійснюватися на основі місць, які хоче відвідати користувач, місця старту й фінішу маршруту та часу для його проходження. Кожне місце (точку) маршруту користувач самостійно оцінює певним ваговим коефіцієнтом (балом), що відповідає зацікавленості у його відвідуванні. Маршрут, який проходить через усі або частину заданих точок з ваговими коефіцієнтами, отримує оцінку, що дорівнює сумі балів вагових коефіцієнтів та відповідає зацікавленості користувача в маршруті. Система

має визначати оптимальний маршрут, що буде проходити через усі або частину заданих користувачем точок, тривалість якого не перевищує встановлений час та максимізує його оцінку. Маршрут має будуватися з використанням мапи, що зберігається у вигляді графу, де вершини – це перехрестя доріг, ребра – дороги між ними. Для кожного ребра зберігається інформація про час проходження та довжину.

Для визначення шляху між вершинами у графі використовують різні алгоритми пошуку шляху у графі. В роботі [7] для пошуку шляху у графі розглядаються алгоритм пошуку в ширину, алгоритм Дейкстри, алгоритм A-star (A*), та алгоритми Hierarchical Pathfinding A* (HPA*) і Lifelong Planning A* (LPA*). В роботі виконується порівняння виконання пошуку різними алгоритмами за часом виконання, використаною пам’яттю та кількістю відвіданих вершин на сітках розмірами від 64x64 до 1024x1024. За результатами дослідження алгоритм пошуку в ширину був найбільш повільним та відвідав найбільшу кількість вершин, найшвидшими алгоритмами були алгоритм A* та HPA*. В роботі [8] розглядається використання алгоритму Дейкстри та двоспрямованого алгоритму Дейкстри для розробки інформаційної системи визначення шляхів евакуації при пожежах, які охоплюють великі території. За результатами порівняння алгоритмів на 10 наборах даних алгоритм Дейкстри був повільнішим, ніж двоспрямований алгоритм Дейкстри у 4 з 10 випадках.

Алгоритми, які розглядаються в роботах вище, дозволяють визначити найкоротші шляхи між двома або декількома точками, але для їх роботи користувач повинен вказати точки, які мають бути присутні в маршруті, а також послідовність їх відвідування. Розглянуті алгоритми не передбачають можливості вибору точок, які будуть відвідані, а які – ні. Визначення точок, які мають бути відвідані у маршруті, а також послідовності їх відвідування визначається шляхом розв’язання задачі орієнтування. Задача орієнтування (англ. Orienteering Problem) – це NP-складна задача, яка є об’єднанням задачі комівояжера та задачі про рюкзак. Формулювання задачі наступне: нехай

задано повний граф G з n вершин, вершина 1 є стартовою точкою, вершина n – кінцевою. Для кожної вершини i визначено додатний прибуток S_i, а також бінарну змінну x_{ij}, яка дорівнює 1, якщо маршрут проходить через вершину i, і 0 – якщо ні. Для кожного ребра встановлено вагу, що позначає час маршруту за цим ребром. Ціль задачі – максимізувати прибуток маршруту, який складається з підмножини вершин графу G, починається з вершини 1 і закінчується у вершині n, й не перевищує обмеження за максимальним часом T_{max} [9]:

$$maximize \sum_{i=1}^{n}\sum_{j=1}^{n} S_i x_{ij}$$

За умов:

$$\sum_{i=2}^{n} x_{1i} = \sum_{j=1}^{n-1} x_{jn} = 1, \qquad k = 2, \dots, n-1 \tag{1.1}$$

$$\sum_{i=2}^{n} x_{ik} = \sum_{j=1}^{n-1} x_{kj} \ \leq 1, \qquad k = 2, \dots, n-1 \tag{1.2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij} x_{ij} \ \leq T_{max} \tag{1.3}$$

де:

i,j,k – вершини графу;

S_i – прибуток (ваговий коефіцієнт) вершини i;

x_{ij} – бінарна змінна, що позначає проходження маршруту через ребро ij;

T_{max} – максимальний час маршруту.

Обмеження (1.1) гарантує, що кількість ребер у маршруті, що з'єднують початкову вершину з іншими вершинами графу, а також кількість ребер, що з'єднують фінішну вершину з іншими вершинами графу

дорівнюватиме 1. Обмеження (1.2) гарантує неперервність маршруту а також гарантує відвідування кожної з *n* вершин не більше 1 разу. Обмеження (1.3) гарантує, що сумарна вага ребер (яка позначає загальний час маршруту) не буде перевищувати максимально дозволений час.

Оскільки задача орієнтування є NP-складною, точні методи можуть потребувати багато часу для знаходження розв'язку. Тому для розв'язання задачі також використовують наближені методи. В роботі [10] розглядаються виконується огляд різних алгоритмів для розв'язання задачі. Серед таких алгоритмів: алгоритми локального пошуку, генетичні алгоритми, табу-пошук, variable neighborhood search та алгоритми колективного інтелекту. В роботі детально розглянуто принцип розв'язання задачі з використанням генетичного алгоритму та алгоритмів локального пошуку й виконано порівняльний аналіз цих алгоритмів. За результатами порівняння роботи цих двох алгоритмів виявлено, що генетичні алгоритми знаходять краще рішення й виконуються швидше в порівнянні з алгоритмами локального пошуку. В роботі [11] більш детально розглянуто мурашиний алгоритм, його модифікації та застосування для розв'язання задачі орієнтування. Огляд алгоритму локального пошуку імітації відпалу (англ. Simulated annealing) для розв'язання задачі орієнтування виконано в роботі [12]. В роботі розглядається модифікація проблеми орієнтування – Orienteering Problem with Mandatory Visits. Особливість модифікації полягає в наявності множини вершин, що є підмножиною вершин графу G, які мають обов'язково бути присутніми в маршруті. В роботі досліджувалися алгоритми імітації відпалу (Simulated Annealing), змішаного цілочисельного лінійного програмування та variable neighborhood search. Для алгоритму імітації відпалу описано процес вибору параметрів алгоритму виконано порівняльний аналіз ефективності алгоритмів на основі часу пошуку рішення та якості знайденого рішення. За результатами дослідження алгоритм імітації відпалу знаходив рішення швидше, ніж алгоритм змішаного цілочисельного лінійного програмування.

1.4 Постановка задачі

На основі аналізу предметної області та дослідження існуючих підходів та методів для розв'язання подібних задач сформульовано постановку задачі: для зваженого графу G (V, E), який позначає карту місцевості, задано множину *I*, яка є підмножиною V і складається з *n* визначених користувачем вершин, та максимально допустимий час маршруту T_{max}. Для кожного ребра задано вагу, що дорівнює дорівнює часу, необхідним для його проходження. Для кожної вершини з множини *I* визначено додатній ваговий коефіцієнт (бал), що відповідає зацікавленості користувача у її відвідуванні. Для всіх вершин що не належать множині *I,* а також для стартової вершини x_i та фінішної вершини x_j (x_i, $x_j \in I$) ваговий коефіцієнт дорівнює 0. Необхідно визначити маршрут, який буде проходити через усі або частину вершин з множини *I,* починатися у вершині x_i, закінчуватися у вершині x_j, тривалість якого не перевищує встановлений час T_{max} та який максимізує оцінку маршруту R_{best} – сумарну кількість балів відвіданих вершин:

$$R_{best} = maximize \sum_{i=1}^{n} \sum_{j=1}^{n} S_i x_{ij} \quad i \in I$$

За умов:

$$\begin{gathered} S_i \geq 0, \qquad i \in I \\ \sum x_{ik} = \sum x_{kj}, \quad i,k,j \in E, k \neq i, k \neq j \\ \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij} \leq T_{max} \end{gathered} \tag{1.4}$$

де:

R_{best} – сума вагових коефіцієнтів (балів) вершин у маршруті;

I – множина вершин, які хоче відвідати користувач;

E – множина усіх вершин графу.

Оптимальний маршрут між двома вершинами з множини I може проходити через іншу вже відвідану вершину з цієї множини, тому обмеження (1.1) та (1.2) на заборону повторного відвідування вершин з множини I не може бути застосовано. Обмеження (1.4) є модифікацією обмеження (1.2) і гарантує неперервність шляху без заборони на повторне відвідування.

Розв’язання задачі дозволить виконувати побудову маршруту, що має бути однією із основних функцій розроблюваної інформаційної системи. Інформаційна система повинна мати тип E-BUSINESS, замовником системи є Федерація Спортивного Туризму України. Для роботи системи повинна бути розроблена база даних та інтерфейс доступу до неї у вигляді веб-сторінок. Користувачами системи є: незареєстрований користувач, турист (зареєстрований користувач) та представник МКК (адміністратор системи). ІС повинна реалізовувати бізнес-процеси:

– реєстрація та авторизація. Виконується незареєстрованим користувачем для отримання доступу до більшої кількості функцій системи. Бізнес-функції у бізнес-процесі: реєстрація користувача, авторизація користувача та функції особистого кабінету.

– реєстрація походу. Виконується зареєстрованим користувачем та адміністратором системи. Бізнес-процес реалізує наступні бізнес-функції: планування маршруту, подача заявки на реєстрацію походу та перевірка й затвердження заявки.

– адміністрування. Виконується адміністратором системи. Бізнес-процес включає наступні бізнес-функції: додавання нових ділянок шляху, які можуть бути використані у маршруті походу, редагування інформації про ділянки (зміна часу проходження, опису, вказаної складності), захист походу й присвоєння спортивних розрядів учасникам походу.

2 РОЗРОБКА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Розробка системних вимог до інформаційної системи

Система, що розроблюється, має складатися з двох компонентів: бази даних та інтерфейсу доступу до БД й бути реалізованою з використанням триланкової архітектури. Інтерфейс доступу до БД має бути у вигляді веб-сторінок на клієнтському шарі, які виконують бізнес-функції за допомогою надсилання запитів через мережу Інтернет на сервер з програмною логікою, яка взаємодіє з БД. Бізнес-функції мають виконуватися на стороні сервера, веб-сторінки мають відповідати лише за валідацію даних, введених користувачем. Система має підтримувати роботу трьох типів користувачів: незареєстрованого користувача, туриста (зареєстрованого користувача) та представника МКК (адміністратора) і реалізовувати наступні бізнес-процеси: реєстрація користувача, авторизація користувача, функції особистого кабінету, планування маршруту, реєстрації походу, захисту звіту про проведений похід, присвоєння спортивних розрядів та адміністрування. Незареєстрований користувач не повинен мати доступ до функцій системи, що виконують запис або зміну даних, окрім функції реєстрації. Для зареєстрованого користувача система має надавати інтерфейс для виконання функцій планування маршруту, реєстрації та захисту походу та функцій особистого кабінету, що включають у себе редагування персональних даних, перегляд попередніх походів та здобутих спортивних розрядів. Для адміністратора система повинна надавати веб-сторінки для виконання бізнес-функцій адміністрування, захисту походів й присвоєння спортивних розрядів.

2.2 Розробка функціональних вимог до інформаційної системи

Розробка функціональних вимог до системи виконувалась з використанням методології IDEF0 за допомогою CASE-засобу BPWin, який

дозволяє аналізувати та документувати бізнес-процеси й підтримує моделювання діаграм з використанням нотацій IDEF0, IDEF3 та DFD [13]. Контекстну діаграму, на якій подано усі вхідні та вихідні дані системи подано на рис. 2.1. Головною функцією системи є планування та реєстрація походів. Метою створення діаграми є визначення функціональних вимог до системи, що розроблюється. Діаграма була побудована з точки зору адміністратора системи.

Рисунок 2.1 – Контекстна діаграма системи

Вхідними даними для системи є логін та пароль користувача, персональні дані користувача, бажання користувача, що включає у себе інформацію про маршрут походу який користувач хоче зареєструвати, туристичний досвід користувача, запрошення до групи, звіт з проведеного походу та інформація про ділянку шляху. Вихідними даними для системи є результати реєстрації та зміни персональних даних, створений маршрут,

зареєстрований похід, відхилена заявка на проведення походу, прийняте або відхилене запрошення, результат захисту з рішенням про присвоєння спортивних розрядів, додана ділянка шляху й результат зміни інформації про ділянку шляху. Робота системи керується нормативною документацією, яка включає у себе закони та постанови, що описані у першому розділі, а також внутрішні положення та регламенти роботи МКК. Також робота системи керується керівництвом користувача, яке визначає алгоритм дій для виконання усіх функцій системи. Механізмами для системи є власне інформаційна система (ІС), а також усі три види користувачів: незареєстрований користувач (НК), зареєстрований користувач (ЗК) та адміністратор інформаційної системи (Адміністратор ІС).

Декомпозиція першого рівня подана на рис. 2.2. Інформаційна система виконує 3 бізнес-процеси: реєстрація та авторизація, реєстрація походу та адміністрування.

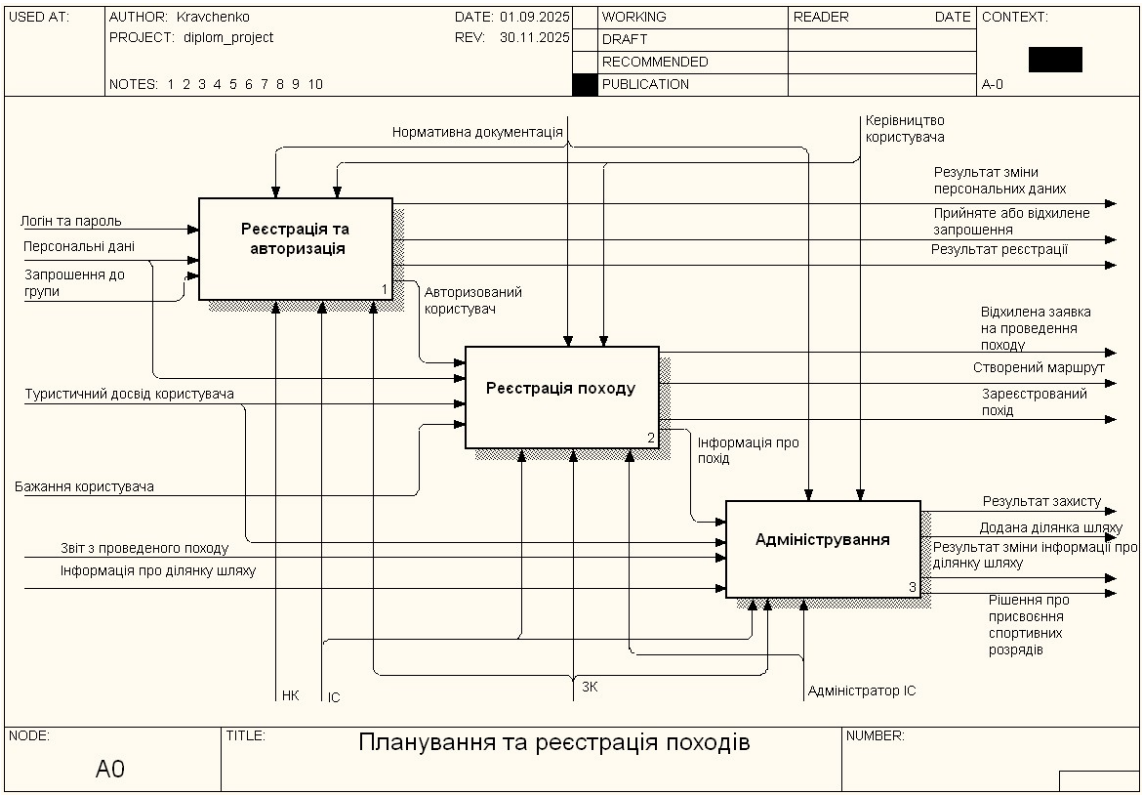


Рисунок 2.2 – Декомпозиція першого рівня

Для кожного бізнес-процесу виконано декомпозицію. Результат декомпозиції для бізнес-процесу реєстрації та авторизації подано на рис. 2.3.

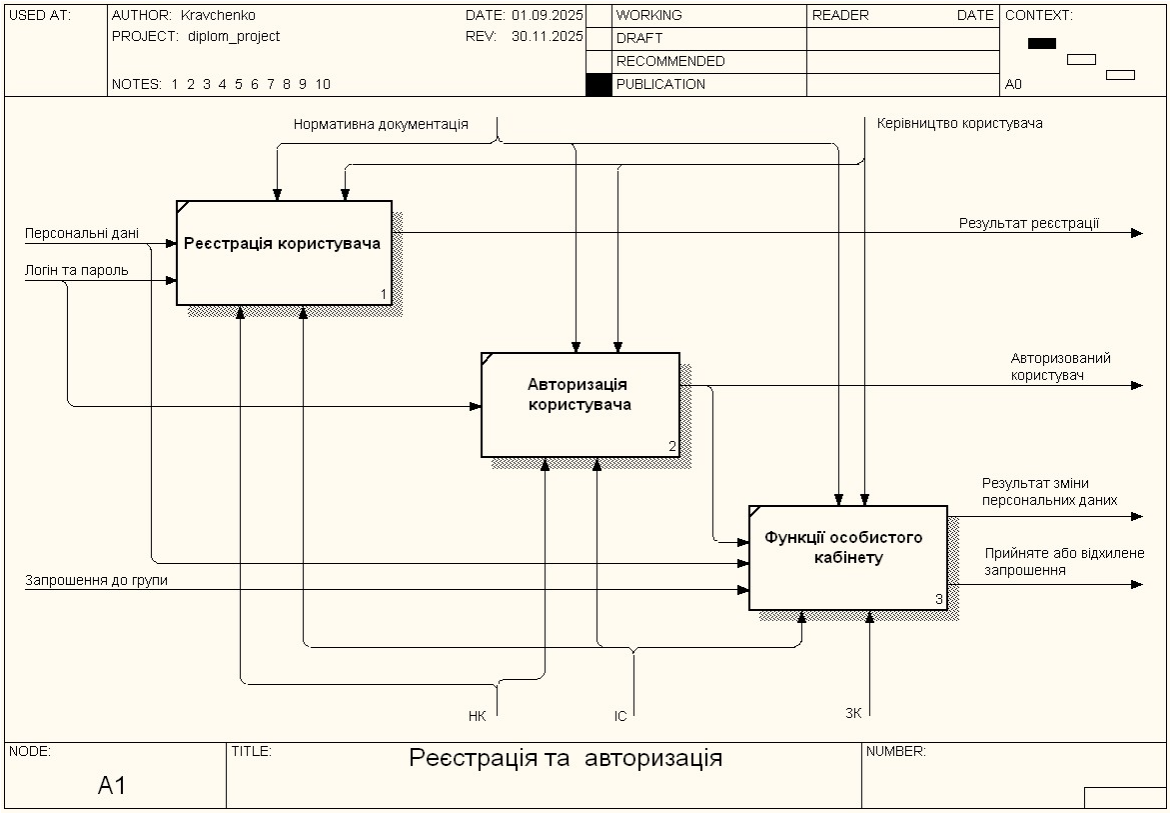


Рисунок 2.3 – Декомпозиція бізнес-процесу «реєстрація та авторизація»

Бізнес-процес складається з трьох бізнес-функцій: реєстрація користувача, авторизація користувача та функції особистого кабінету. У бізнес-функції реєстрації виконується створення аккаунту для користувача. Бізнес-функція у якості вхідних даних приймає персональні дані користувача, такі як ім’я, прізвище, номер телефону й логін з паролем для нового аккаунту. Вихідними даними є інформація про результат реєстрації, яка представляє собою повідомлення про успішну або невдалу реєстрацію. У разі успішної реєстрації користувач може виконати авторизацію у системі за допомогою логіна та пароля, що були передані у вхідних даних. Під час авторизації ІС приймає на вхід логін та пароль, що надаються користувачем, й у разі, якщо інформація про користувача з таким логіном та паролем

присутня у базі даних, виконує авторизацію й надає користувачу доступ до функцій системи, які недоступні неавторизованому користувачу, в залежності від виду користувача (звичайний зареєстрований користувач чи адміністратор системи). У разі, якщо користувача з наданим логіном та паролем не існує, то система запропонує користувачу зареєструватися. Функції особистого кабінету приймають у якості вхідних даних інформацію про авторизованого користувача та його персональні дані для внесення змін. Результатом бізнес-функції є результат зміни персональних даних. Також з використанням функцій особистого кабінету виконується прийняття та відхилення запрошень на вступ до групи походу, який організований іншим користувачем. Усі функції виконуються з використанням інтерфейсу інформаційної та контролюються керівництвом користувача та нормативною документацією.

Діаграма декомпозиції для бізнес-процесу реєстрації походу подана на рис. 2.4.

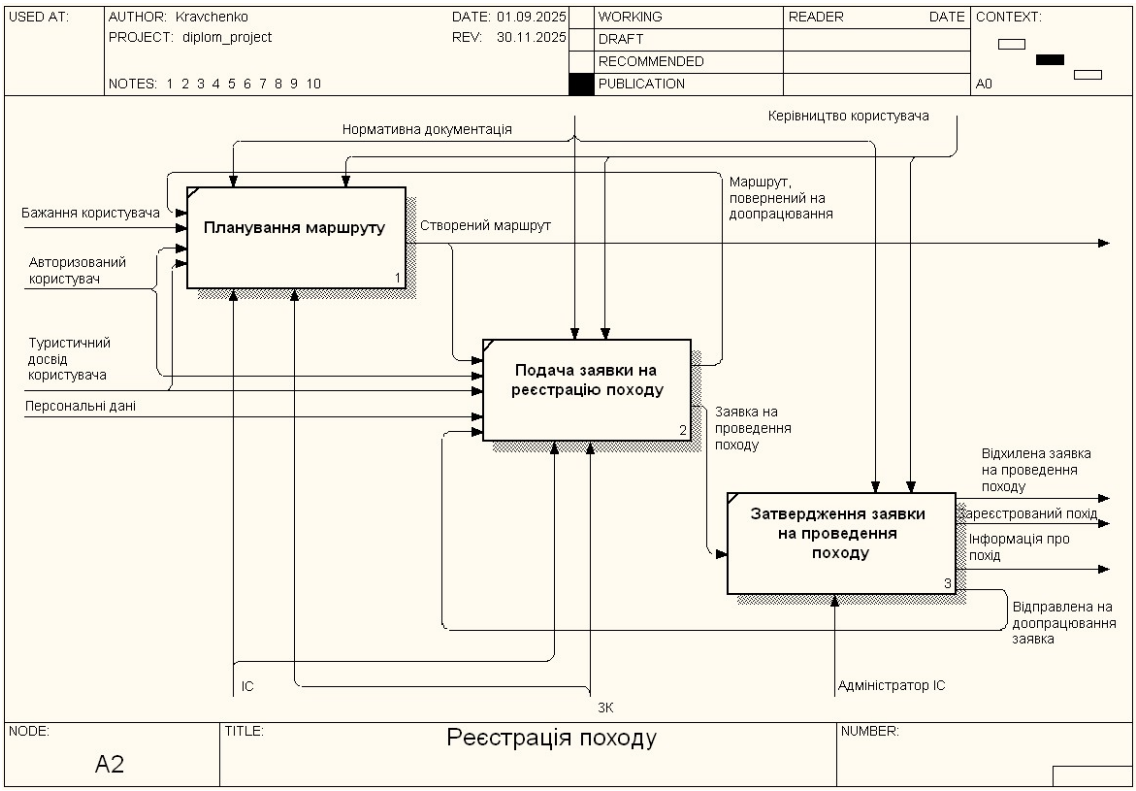


Рисунок 2.4 – Декомпозиція бізнес-процесу «Реєстрація походу»

Бізнес-процес складається з трьох функцій: планування маршруту, подача заявки на реєстрацію походу та затвердження заявки на проведення походу. Кожна з функцій виконується за допомогою інтерфейсу ІС. Планування маршруту виконується зареєстрованим користувачем на основі бажання користувача, що включає у себе місця, які він хоче відвідати й інші побажання стосовно маршруту та туристичний досвід користувача, який допомагає визначити ділянки маршруту, які будуть занадто складними для користувача. Вихідними даними бізнес-функції є створений маршрут. Подача заявки на проведення походу виконується зареєстрованим користувачем й на основі маршруту, досвіду та персональних даних створюється заявка на проведення походу. Ця заявка слугує вхідними даними для бізнес-функції затвердження заявки на проведення походу, яке виконується адміністратором ІС. У цій бізнес-функції адміністратор перевіряє заявку й робить рішення, кожне з яких є вихідними даними бізнес-функції: зареєстрований похід, відхилена заявка та заявка, що повернена на доопрацювання. Усі бізнес-функції контролюються керівництвом користувача та нормативною документацією.

Діаграма декомпозиції для бізнес-процесу адміністрування подана на рис. 2.5. Бізнес-процес складається з трьох функцій: захист походу, присвоєння спортивних розрядів та адміністрування маршрутів. Бізнес-функція захисту походу виконується зареєстрованим користувачем та адміністратором з використанням інтерфейсу ІС. Користувач надає інформацію про похід, звіт з проведеного походу й інформацію про свій досвід. На основі цих даних адміністратор робить рішення про зарахування чи не зарахування походу й рішення стосовно присвоєння спортивних розрядів. Бізнес-функція присвоєння спортивних розрядів приймає у якості вхідних даних результат захисту походу й туристичний досвід користувача й на їх основі адміністратор робить рішення стосовно присвоєння спортивних розрядів учаснику походу. Бізнес функція адміністрування маршрутів виконується адміністратором системи.

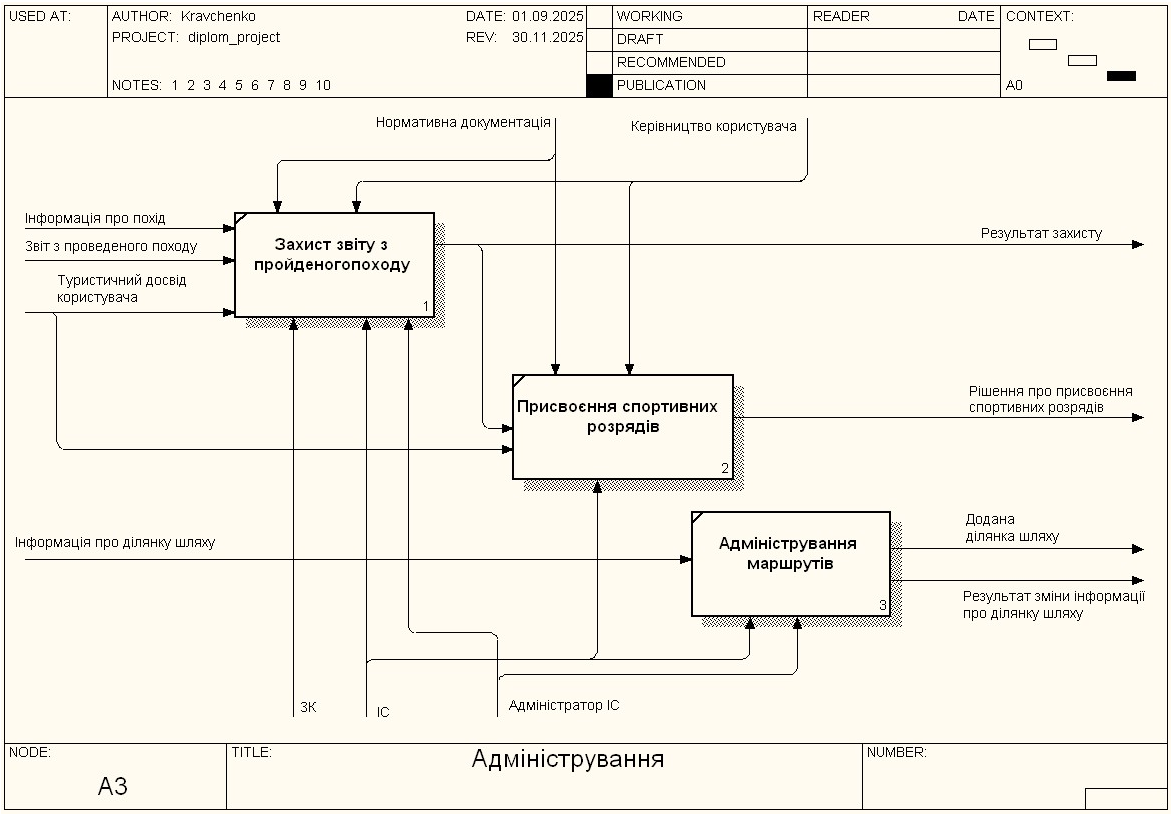


Рисунок 2.5 – Декомпозиція бізнес-процесу «Адміністрування»

На основі інформації про ділянку шляху адміністратор системи може додати нову ділянку шляху до мапи, яка використовується для планування маршрутів, або змінити інформацію про існуючу ділянку шляху (додати фото, змінити опис, встановити складність, тощо). Результатом бізнес-функції є додана ділянка шляху та результат зміни інформації про ділянку шляху. Усі бізнес-функції контролюються керівництвом користувача та нормативною документацією. Діаграму дерева вузлів подано на рис. 2.6. На діаграмі подана ієрархія усіх функцій у системі, що були визначені під час створення функціональних вимог. Основна функція системи складається з трьох бізнес-процесів, для яких визначені наступні вимоги: система повинна забезпечувати функції реєстрації й авторизації користувача, а також функції особистого кабінету для виконання бізнес-процесу реєстрації та авторизації. Для виконання бізнес-процесу реєстрації походу система має виконувати функції планування маршруту, подачі заявки на реєстрацію походу та

затвердження заявки на проведення походу. Для бізнес-процесу адміністрування система має надавати функції захисту походу, присвоєння спортивних розрядів та функцій адміністрування маршрутів.

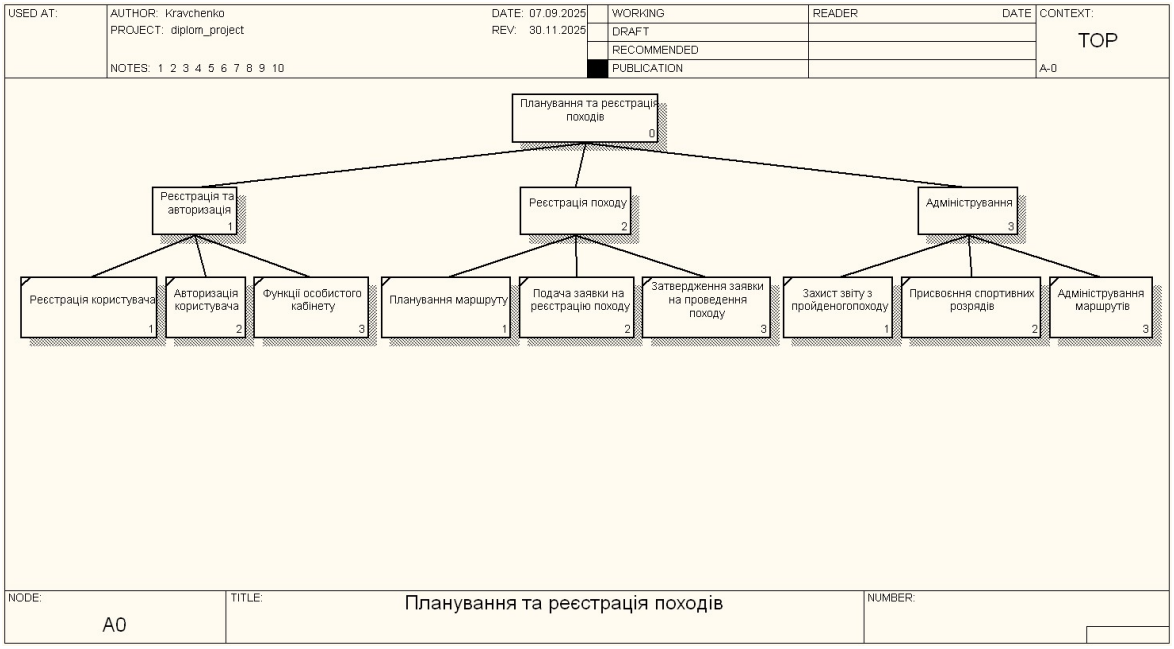


Рисунок 2.6 – Діаграма дерева вузлів

За результатами функціонального моделювання були уточнені функціональні вимоги, для кожного з видів користувачів визначено функції системи, які будуть йому доступні, визначено вхідні й вихідні дані системи а також документи, які регламентують її роботу. Система має виконувати 3 бізнес-процеси: реєстрація та авторизація, реєстрація походу й адміністрування. Реєстрація та авторизація виконується незареєстрованим користувачем для отримання доступу до більшої кількості функцій системи. Реєстрація походу виконується зареєстрованим користувачем та адміністратором системи. Для реєстрації виконується планування маршруту, подача заявки та затвердження заявки на проведення походу. Адміністрування виконується андімістратором системи. Бізнес-процес включає бізнес-функції адміністрування, яка полягає у додаванні нових ключових точок та ділянок шляху, що можуть бути використані під час

планування маршруту походу, захист звіту з пройденого походу та присвоєння спортивних розрядів учасникам походу.

2.3 Розробка діаграми потоків даних системи

Для визначення сутностей, інформація про яких має зберігатися у базі даних ІС розроблено діаграму потоків даних, яку подано на рисунку рис 2.7.

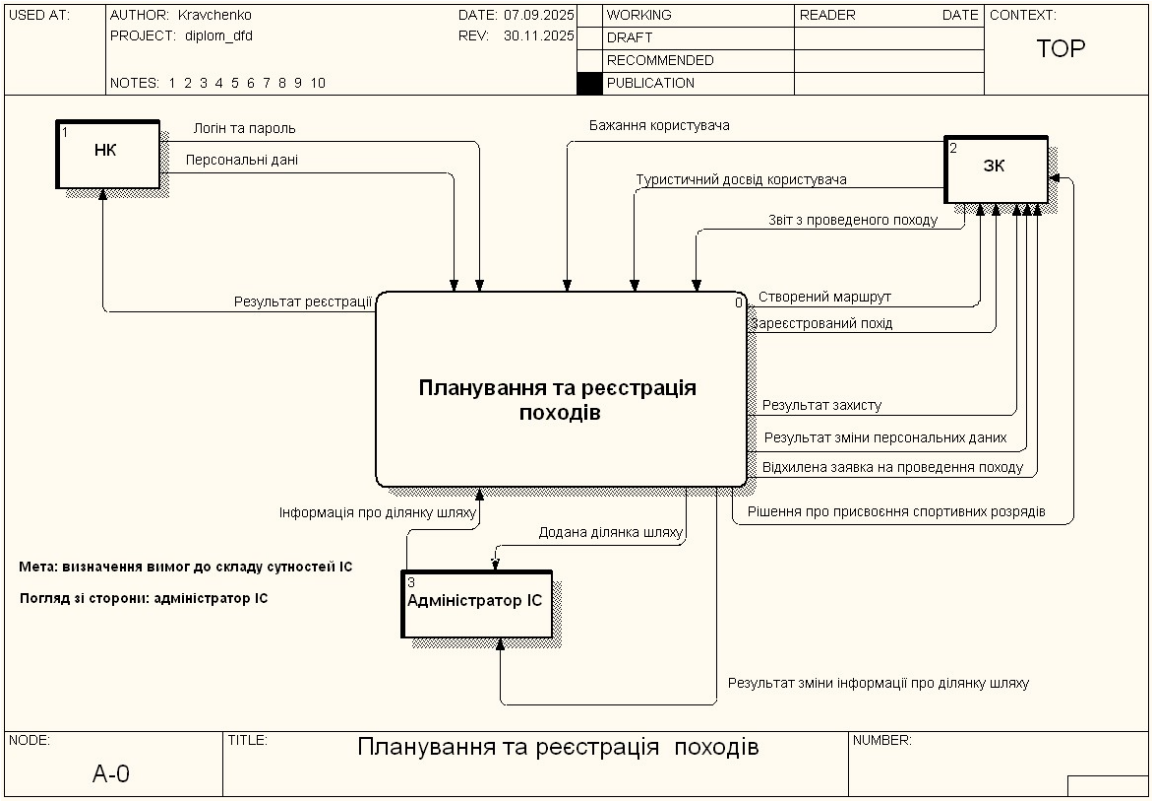


Рисунок 2.7 – Діаграма потоків даних

Діаграма створюється з точки зору адміністратора системи. Головною функцією системи є облік документів для проведення походів. Зовнішніми сутностями є незареєстрований користувач, зареєстрований користувач та адміністратор інформаційної системи. Незареєстрований користувач передає у систему логін, пароль та свої персональні дані для виконання реєстрації. Система надає незареєстрованому користувачу інформацію про результат реєстрації. Зареєстрований користувач передає у систему свої бажання

стосовно маршруту й походу та свій туристичний досвід. Також зареєстрований користувач передає у систему звіт про проведений похід. Система повертає користувачу інформацію про створений маршрут, зареєстрований похід або відхилену заявку на проведення походу, результат захисту походу й рішення про присвоєння спортивних розрядів. Адміністратор передає у систему інформацію про ділянку шляху й на її основі отримує інформацію про додану або змінену ділянку шляху.

Декомпозиція контекстної діаграми потоків даних подано на рис. 2.8. На діаграмі присутні три функції та чотири основні сутності для ІС. Функція «Реєстрація та авторизація» забезпечує роботу реєстрації, авторизації та функцій особистого кабінету. Функція використовує сутність User, що зберігає персональні дані клієнта: ім’я, прізвище, стать, вік, логін, пароль та адресу електронної пошти.

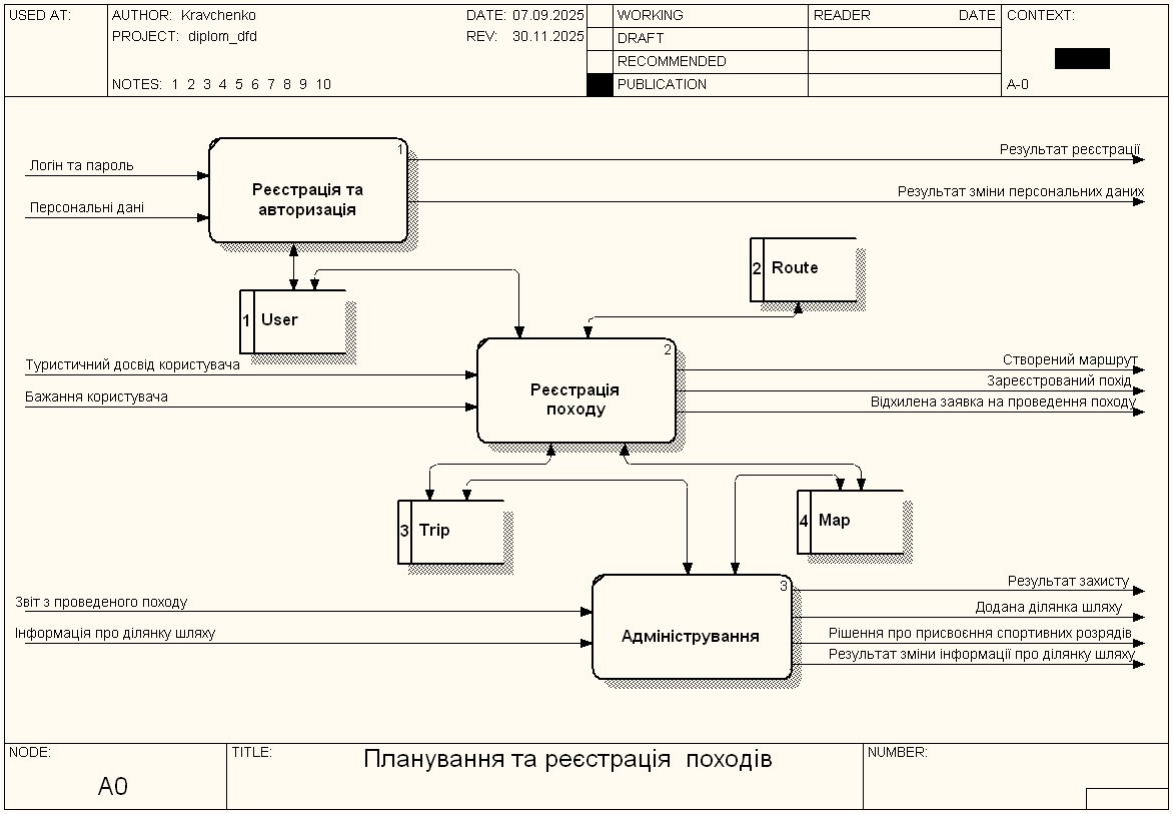


Рисунок 2.8 – Декомпозиція діаграми потоків даних

Функція «Реєстрація походу» використовує сутність User для отримання персональних даних учасників походу, сутність Route, що зберігає інформацію про маршрут походу, який складається з ключових точок походу та маршруту між ними, сутність Trip, в якій зберігається інформація про похід (склад групи, дата початку та завершення, тощо) та сутність Map для отримання інформації про можливі ключові точки походу та маршрутів між ними. Функція «Адміністрування» використовує сутності Trip для виконання захисту походу та Map для виконання адміністрування.

Проведене моделювання потоків даних дозволило визначити сутності, що мають бути у базі даних ІС та їх склад, а саме:

– User, у якій зберігається персональна інформація туриста;

– Route, у якій зберігається інформація про маршрут походу;

– Trip, у якій зберігається інформація про похід;

– Map, у якій зберігається інформація про ділянки шляху, що доступні для побудови маршруту.

2.4 Розробка діаграми варіантів використання інформаційної системи

Для визначення функцій, які ІС має надавати кожному з типів користувачів, розроблено діаграму варіантів використання, яку подано на рисунку 2.9. У попередніх розділах визначено три типи користувачів ІС: незареєстрований користувач, зареєстрований користувач (турист) та адміністратор ІС (представник МКК). Незареєстрований користувач може виконати реєстрацію та авторизацію у системі, а також переглянути маршрути, які були створені іншими користувачами й які були викладені у відкритий доступ. Користувач може переглядати усі створені маршрути, або лише ті, які проходять через обраний регіон. Зареєстрований користувач може виконувати усі дії, які доступні для незареєстрованого користувача. Зареєстрований користувач може виконувати зміну персональних даних, які були вказані під час реєстрації, й вихід з системи.

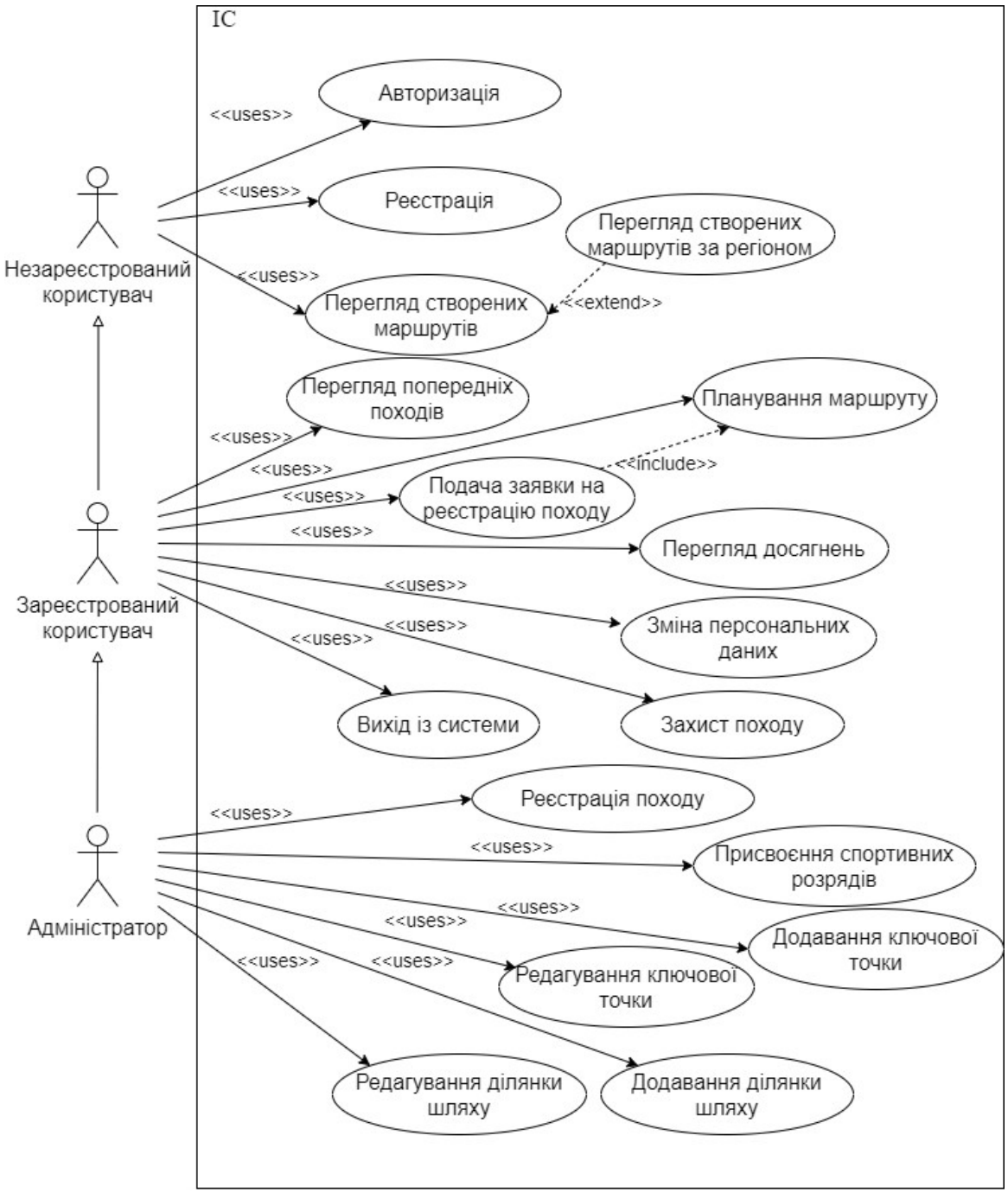


Рисунок 2.9 – Діаграма варіантів використання системи

Зареєстрований користувач також може виконувати функції перегляду походів, у яких він був учасником і які вже завершені, планування маршруту та подача заявки на реєстрацію походу, що включає у себе виконання планування маршруту. Після завершення походу зареєстрований користувач може виконати захист звіту з походу. Зареєстрований користувач може

переглядати свої спортивні досягнення, що включають отриманий досвід та отримані спортивні розряди за пройдені походи. Адміністратор системи може виконувати усі функції, що доступні зареєстрованому та незареєстрованому користувачу. Також адміністратор системи має доступ до функцій реєстрації походу на основі заявки користувача, присвоєння спортивних розрядів користувачам на основі інформації про туристичний досвід користувача і пройдені ним походи. Адміністратор системи може виконувати додавання нових та редагування існуючих ділянок шляху і ключових точок, які можна використати при плануванні маршруту.

2.5 Розробка діаграми класів системи

На основі діаграми використання системи виконано розробку діаграми класів системи. Діаграма класів системи подана на рисунку 2.10. Діаграма побудована з використанням шаблону MVC, що виконує розділення бізнес-логіки та її відображення, що дозволяє спростити обслуговування та пришвидшити розробку [14]. Система містить з чотири класи-контролери. Клас MainController забезпечує виконання переходу між сторінками й використовує вид MainView, який містить кнопки для виконання навігації. Навігація між сторінками виконується з використанням методів контролера GoToLogin (перехід на сторінку входу), GoToRegistration (перехід на сторінку реєстрації), GoToTripPlanner (перехід на сторінку планування маршруту), GoToAccount (перехід на сторінку особистого кабінету) та GoToMapEditing (перехід на сторінку редагування мапи). Також контролер містить метод Logout, який виконує вихід користувача із системи . Для виконання бізнес-функцій реєстрації та авторизації використовується контролер: LoginController, що містить функції для входу (Login) реєстрації (Register) та перевірки введених користувачем даних (ValidateUserData). Контролер використовує види LoginView для виконання входу та RegistrationView для виконання реєстрації.

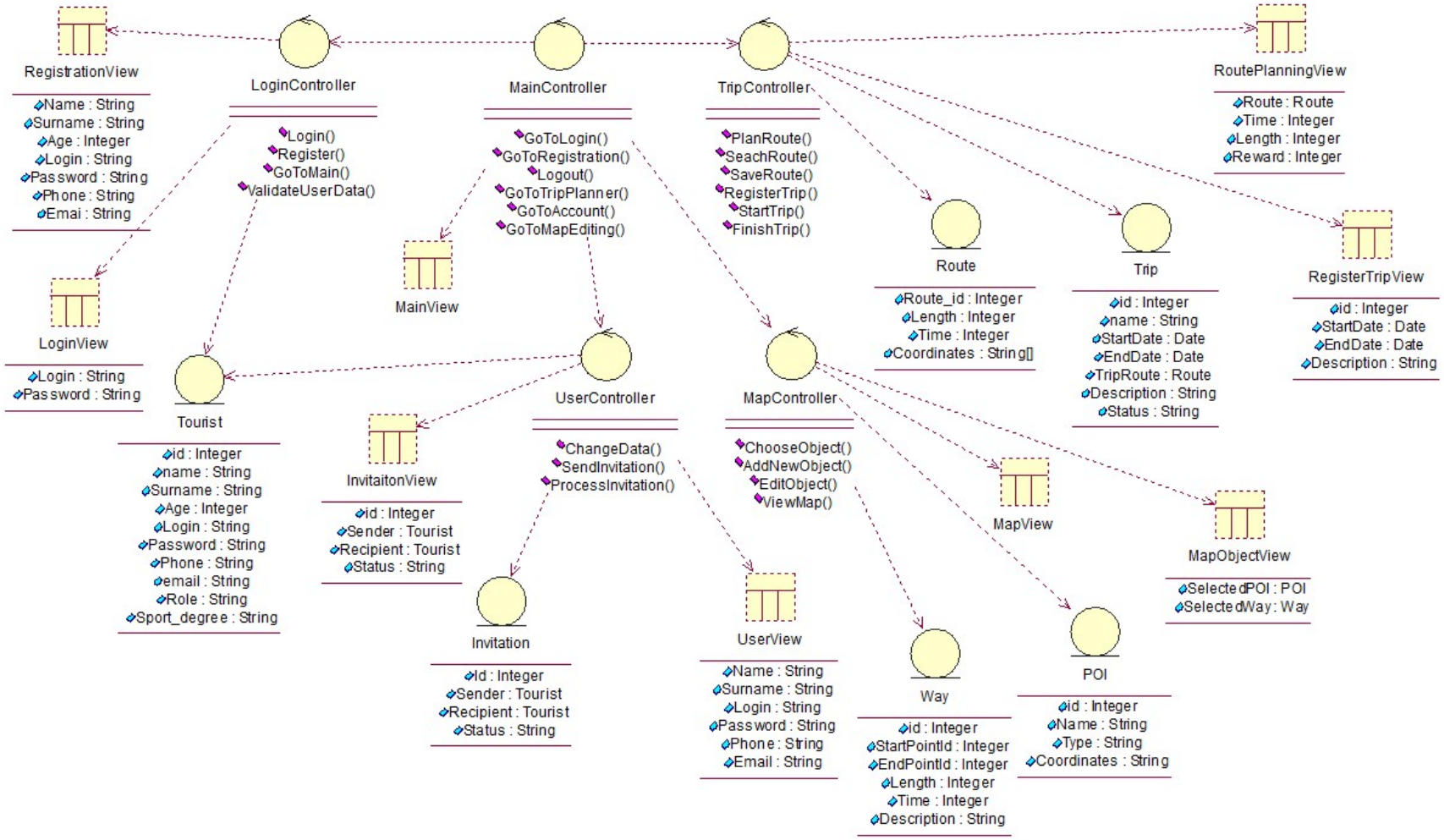


Рисунок 2.10 – Діаграма класів системи

Дані передаються з використанням сутності Tourist, яка зберігає усю інформацію про користувача. Функції особистого кабінету та надсилання запрошень виконуються з використанням контролеру UserController. Для зміни персональних даних використовується вид UserView та метод ChangeData. Для обробки запрошень використовуються методи SendInvitation і ProcessInvitation та вид InvitationView. Для передачі даних використовуються сутності Tourist та Invitation.

Контролер TripController виконує бізнес-функції планування маршруту та реєстрації походу. Планування маршруту виконується за допомогою методів PlanRoute (виконує планування маршруту) та SaveRoute (зберігає побудований маршрут) та виду RoutePlanningView. Реєстрація походу виконується з використанням методів RegisterTrip та виду RegisterTripView. Функції StartTrip та EndTrip виконують зміну статусу походу на «розпочатий» та «завершений» відповідно. Для передачі даних використовуються сутності Route, що зберігає інформацію про маршрут, та Trip, що зберігає інформацію про похід.

Контролер MapController виконує функції редагування мапи. Для перегляду мапи використовується функція ViewMap та вид MapView. Для редагування мапи використовуються функції ChooseObject, яка виконує вибір об’єкту (ключової точки або ділянки шляху) для редагування, AddObject, яка виконує додавання нового об’єкту на мапу та функція EditObject, яка виконує редагування обраного об’єкту. Для виконання додавання та редагування об’єктів використовується вид MapObjectView. Дані передаються з використанням сутностей POI для інформації про точки інтересу, та Way для інформації про ділянки шляху.

2.6 Розробка діаграми послідовності дій системи

Для прецедентів «Подача заявки на проведення походу» та «Редагування ділянки шляху» виконано розробку діаграм послідовності дій.

Діаграма послідовності для прецеденту «Подача заявки на проведення походу» подано на рисунку 2.11.

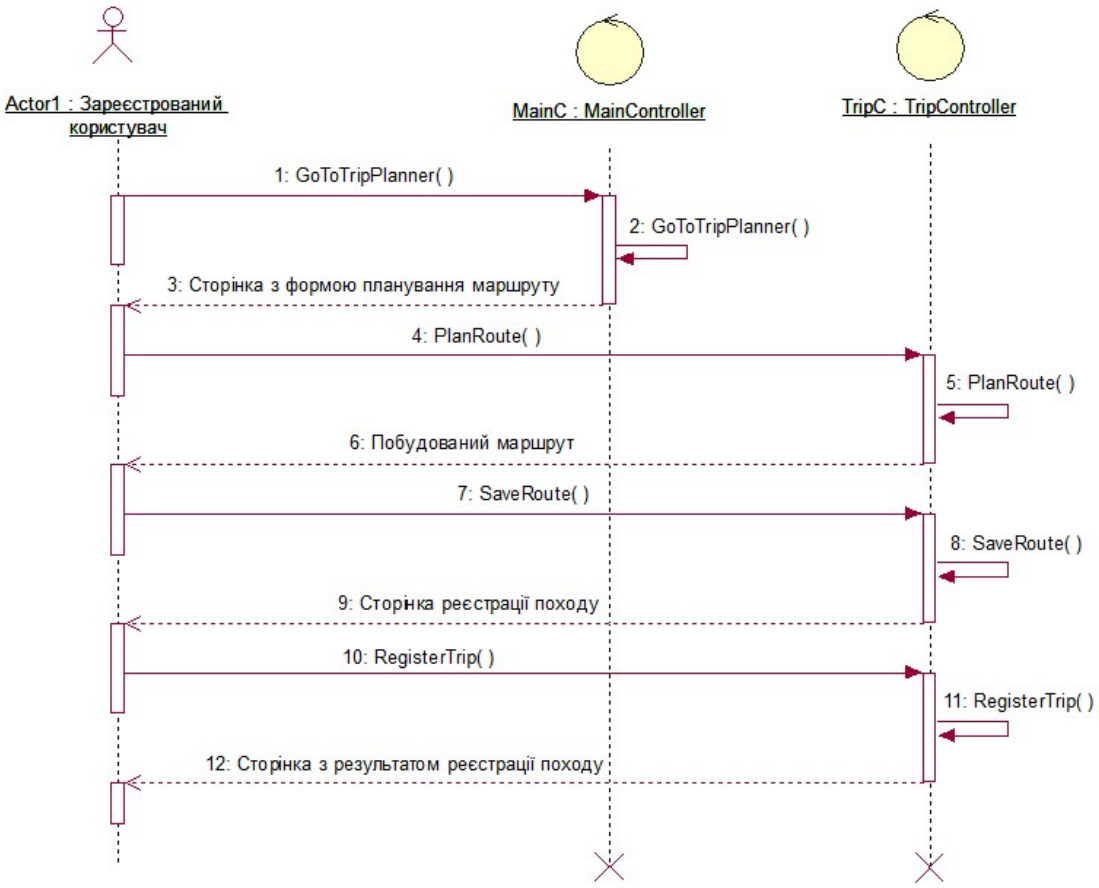


Рисунок 2.11 – Діаграма послідовності дій для прецеденту «Подача заявки на проведення походу»

На діаграмі подано актора з роллю зареєстрованого користувача. Користувач викликає метод GoToTripPlanner() об’єкта класу MainController, який повертає сторінку з формою планування маршруту. Користувач заповнює форму та викликає метод PlanRoute об’єкта класу TripController. Метод повертає побудований за критеріями користувача маршрут. Після перевірки маршруту користувач виконує змебеження маршруту шляхом виклику метода SaveRoute(). Метод виконує збереження маршруту та повертає стоінку для реєстрації походу за спланованим маршрутом. Для

реєстрації походу користувач викликає метод RegisterTrip() об’єкта класу TripController. Метод виконує реєстрацію походу та повертає користувачу інформацію про її результат.

Діаграма послідовності для прецеденту «Редагування ділянки шляху» подано на рисунку 2.12.

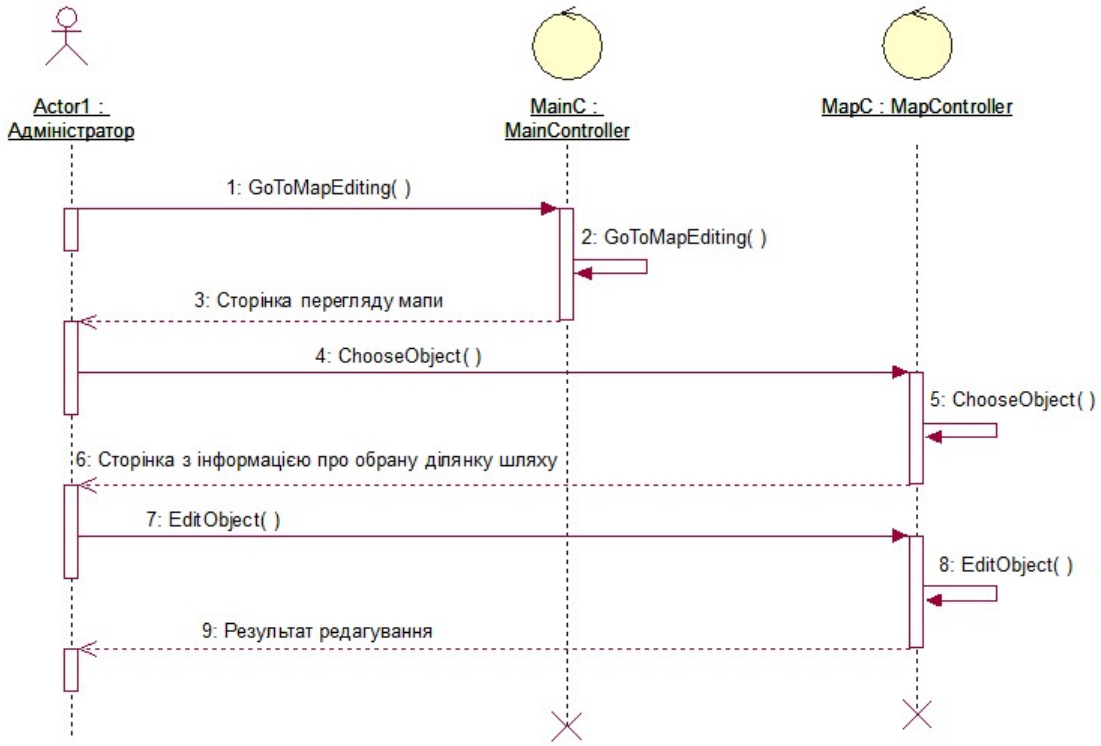


Рисунок 2.11 – Діаграма послідовності дій для прецеденту «Редагування ділянки шляху»

На діаграмі подано актора з роллю зареєстрованого користувача. Користувач викликає метод GoToMapEditing() об’єкта класу MainController, який повертає сторінку перегляду мапи. Для редагування ділянки шляху користувач обирає її з використанням функції ChooseObject() об’єкта класу MapController. Метод повертає сторінку з інформацією про обрану ділянку шляху. На отриманій сторінці користувач вносить зміни в інформацію про ділянку шляху й викликає метод EditObject() для збереження цих змін. Результатом методу є сторінка з інформацією про результат редагування

інформації.

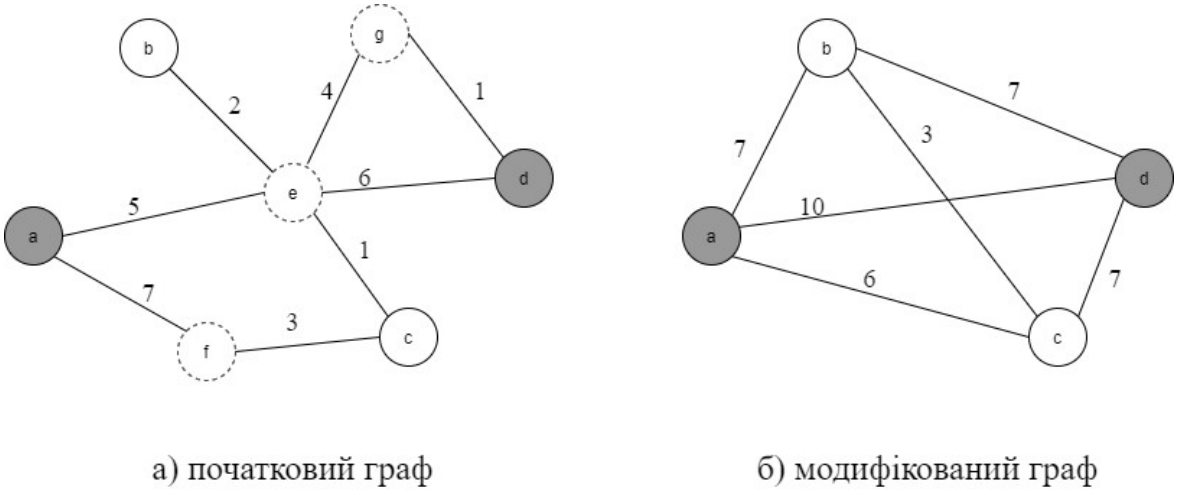
2.7 Визначення алгоритму розв’язку задачі

Як було визначено в попередніх розділах, однією з основних функцією системи є побудова маршруту на основі місць, які хоче відвідати користувач, місця старту та фінішу маршруту та доступного часу для його проходження. Кожна вершина графу, який позначає карту місцевості та за яким буде виконуватися побудова маршруту походу, може мати один з трьох типів: звичайне перехрестя, ключова точка та точка старту маршруту. Ключова точка – це місце, в якому розташований природний або рукотворний об’єкт, який користувач хоче відвідати. Прикладами таких об’єктів є: вершина гори, водоспад, старовинна церква, тощо. Для побудови маршруту система надає користувачу список ключових точок, і користувач має обрати ті, які він хоче відвідати. Точка старту маршруту – це точка з якої користувач може розпочати або завершити маршрут. До цих точок має бути простий доступ з використанням автомобільної техніки чи іншого сполучення. Такими точками є населені пункти. Для числового визначення задоволення користувача маршрутом введено систему балів. Користувач має надати кожній обраній точці ваговий коефіцієнт (кількість балів), який буде позначати наскільки сильно користувач хоче її відвідати. Діапазон балів обрано від 1 до 100. Оцінки в 0 балів немає, оскільки в такому випадку точка буде вважатися нецікавою користувачу й не буде враховуватися під час побудови маршруту. Оптимальним маршрутом вважається такий, сумарна кількість балів відвіданих точок якого є найбільшою. В даній роботі час маршруту не впливає на якість маршруту, але знайдений маршрут не буде розглядатися, якщо час його проходження перевищує встановлене користувачем часове обмеження.

Для пошуку оптимального маршруту з урахуванням встановлених обмежень пропонується використовувати двоетапний підхід. На першому

етапі визначаються маршрути з мінімальним часом проходження між кожною парою визначених користувачем точок. На другому етапі визначається порядок відвідування точок шляхом розв’язання задачі орієнтування на основі цих даних.

Розглянемо розв’язання задачі з використанням запропонованого підходу на графі з 7 вершинами, що поданий на рисунку 2.10, а.



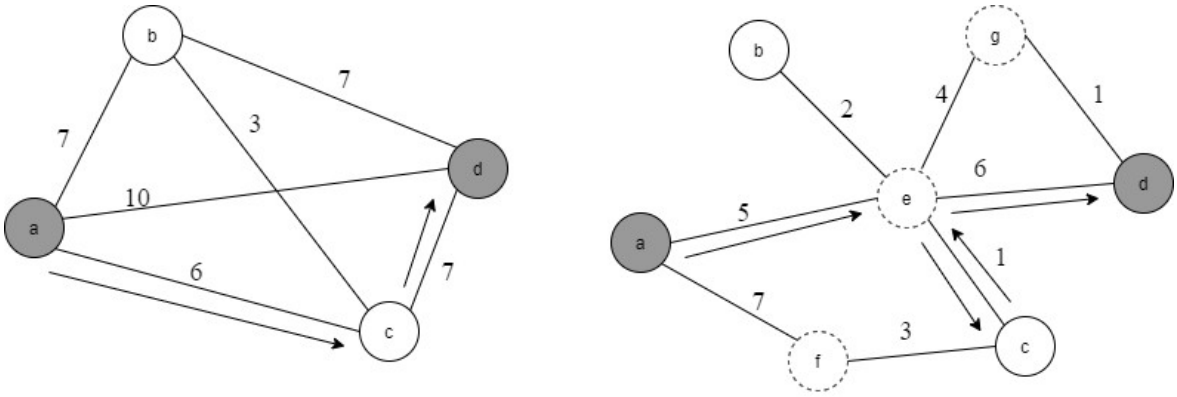
а) початковий граф б) модифікований граф

Рисунок 2.10 – Перший етап розв’язання задачі

Сірі вершини позначають вершини, в яких можна почати та завершити маршрут, вершини із суцільним кордоном – ключові точки, обрані користувачем, вершини з пунктирним кордоном – звичайні перехрестя доріг. Біля кожного ребра зазначено його вагу, яка позначає час його проходження. Для розв’язання задачі орієнтування використовується модифікований початковий граф, що подано на рисунку 2.10, б. Побудова модифікованого графу виконується шляхом видалення з початкового графу вершин, що не є цікавими користувачу, видалення всіх інцидентних ребер цих вершин й додавання нових ребер між вершинами, що залишилися, для формування повного графу. Вага нових ребер між вершинами модифікованого графу встановлюється у відповідності до найменшого часу переходу між відповідними вершинами у початковому графі. Для прикладу між вершинами *a* та *b* немає прямого шляху, але час найшвидшого шлях між ними дорівнює

7 одиниць часу, тому у модифікованому графі вага ребра a,b дорівнює 7. Для максимізації кількості відвіданих вершин, час на перехід між ними має бути мінімальним, тому виконується пошук найшвидшого, а не найкоротшого шляху.

Після знаходження оптимального маршруту на новому графі за допомогою розв'язання задачі орієнтування, необхідно замість кожного ребра маршруту підставити маршрут, знайдений під час створення графу. Розглянемо побудову маршруту на початковому графі зі стартом у вершині *a*, фінішем у вершині *d* й часовим обмеженням у 13 одиниць часу. На рисунку 2.11 подано розв'язання задачі орієнтування на модифікованому графі (рис. 2.11, а) та відповідний маршрут на початковому графі (рис 2.11, б), що був створений на основі розв'язку задачі орієнтування на модифікованому графі.



а) оптимальний маршрут на модифікованому графі

б) оптимальний маршрут на початковому графі

Рисунок 2.11 – Визначення остаточного маршруту

Розв'язком задачі орієнтування з встановленими обмеженнями для модифікованого графу є маршрут *a,c,d*. Найкоротшим шляхом між вершинами *a,c* та *c,d* на початковому графі є шляхи *a,e,d* та *c,e,d* відповідно. Тому остаточний маршрут: *a,e,c,e,d*.

2.8 Визначення та опис алгоритмів для розв’язання задачі

В попередньому підрозділі описано 2 основні етапи розв’язку задачі: знаходження найкоротшого шляху між кожною парою вершин та розв’язання задачі орієнтування. Задача пошуку найкоротшого шляху у графі полягає в наступному: нехай задано граф G (V, E), для кожного ребра з множини E задано ваговий коефіцієнт. Необхідно знайти шлях між двома обраними вершинами з множини V, для якого сума вагових коефіцієнтів ребер, з яких складається шлях, буде мінімальною [15]. Для виконання пошуку найшвидшого шляху розглядаються наступні алгоритми: алгоритм Дейксти, двоспрямований алгоритм Дейкстри та алгоритм A*, оскільки реалізація цих алгоритмів є в багатьох стандартних бібліотеках різних мов програмування.

Алгоритм Дейкстри є класичним алгоритмом для пошуку шляху між заданою вершиною й усіма іншими вершинами у зваженому графі без ребер з від’ємною вагою. Алгоритм обирає сусідню до вже оброблених вершин вершину з найменшою вагою ребра, розраховує до неї відстань, перевіряє чи є шлях через цю вершину до сусідніх оброблених вершин коротшим. Двоспрямований алгоритм дейкстри є модифікацією класичного алгоритму Дейкстри, в якому пошук починається з стартової та цільової вершини одночасно й завершується коли обидва пошуки обробили одну або кілька однакових вершин графу й коли новий шлях не може бути коротшим за вже знайдений. Алгоритм A* є модифікацією алгоритму Дейкстри, який робить вибір наступної вершини на основі відстані від стартової вершини до поточної та оціночної відстані від поточної вершини до фінішної, що дозволяє більш швидко знайти відстань між двома вершинами за рахунок відсутності дослідження безперспективних напрямків.

Для розв’язання задачі орієнтування розглядаються 1 точний та 2 наближені алгоритми з різною стратегією пошуку рішення. У якості точного алгоритму обрано алгоритм пошуку в глибину, оскільки він дозволяє просто врахувати обмеження за часом в порівнянні з іншими. У якості наближених

алгоритмів обрано мурашиний алгоритм та алгоритм імітації відпалу. Мурашиний алгоритм є алгоритмом на основі колективного інтелекту, який імітує поведінку мурах при пошуку найкращого шляху. Алгоритм обрано, оскільки він дозволяє досліджувати великі простори рішень, виконує пошук маршруту шляхом пересування агента графом, і може бути модифікований в залежності від задачі [11]. Алгоритм імітації відпалу є алгоритмом локального пошуку. Алгоритм обрано, оскільки він уникає локальних максимумів за рахунок можливості прийняття гіршого рішення, ніж поточне найкраще рішення, з певною ймовірністю, що поступово зменшується.

Алгоритм пошуку в глибину з обмеженням за часом виконує класичний пошук в глибину, але вершини, які можна відвідати з поточної вершини, визначає за обмеженням (2.1).

$$T_{curr} + T_{ij} + T_{jf} \leq T_{max}, \qquad j \notin V_{visited}, j \neq f \tag{2.1}$$

де:

T_{curr} – поточний час маршруту;

T_{ij} – час на перехід з поточної вершини у вершину j;

T_{jf} – час переходу з вершини j у фінішну вершину f;

T_{max} – максимально допустимий час маршруту;

$V_{visited}$ – множина відвіданих вершин та фінішна вершина.

Якщо не існує вершин, для яких виконується обмеження (2.1), тобто не існує таких вершин, після переходу в яких залишиться достатньо часу для переходу у фінішну вершину, алгоритм розраховує нагороду поточного шляху і виконує повернення на попередню вершину. У разі, якщо нагорода поточного шляху є вищою за нагороду найкращого знайденого шляху, алгоритм зберігає поточний шлях у якості найкращого. Алгоритм виконується поки усі вершини не будуть позначені як відвідані або як такі, з яких не вистачить часу дістатися до фінішної вершини.

Алгоритм мурашиної колонії – це евристичний алгоритм на основі

агентів, які рухаються по графу. Ймовірність переходу агента (мурахи) в кожну з сусідніх вершин визначається формулою (2.2). Для підвищення якості знайденого рішення агент не може перейти до фінішної вершини, доки існує хоча б одна вершина, що задовольняє обмеження (2.1), оскільки всі вершини, крім початкової та кінцевої, мають додатну кількість балів, і їх включення до маршруту підвищує сумарну нагороду:

$$P_{ij} = \begin{cases} \dfrac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{\sum_{l \notin V_{visited}} (\eta_{il})^{\alpha} \cdot (\tau_{il})^{\beta}}, l, j \notin V_{visited};\ T_{curr} + T_{ij} + T_{jf} \leq T_{max} \\ 0, \qquad j \in V_{visited} \text{ або } T_{curr} + T_{ij} + T_{jf} > T_{max} \\ 0, \qquad j = f;\ \exists j \neq f:\ T_{curr} + T_{ij} + T_{jf} \leq T_{max} \\ 1, \qquad j = f;\ \forall j \neq f:\ T_{curr} + T_{ij} + T_{jf} > T_{max} \end{cases} \quad (2.2)$$

де:

η_{ij} – евристична оцінка, що визначає ступінь привабливості вершини для переходу, формула якої визначається в залежності від задачі. В даній роботі $\eta_{ij} = \frac{1}{T_{ij}}$;

τ_{ij} – кількість феромону на ребрі ij;

l, j – вершини графу;

f – фінішна вершина маршруту у графі;

α, β – параметри, які визначають вплив евристичної оцінки та феромонів на ймовірність вибору вершини ;

$V_{visited}$ – множина відвіданих вершин.

Алгоритм імітаційного відпалу є алгоритмом локального пошуку, який повторює ітераційну процедуру генерації сусідніх рішень й уникає локальних максимумів за рахунок можливості прийняття гіршого рішення, ніж поточне найкраще рішення, з ймовірністю, яка розраховується за формулою [16]:

$$P = \begin{cases} 1, & \text{якщо } u_{[0,1]} \leq \exp(-\frac{E - E_{next}}{T_{curr}}) \\ 0 \end{cases}$$

де:

$u_{[0,1]}$ – неперервна рівномірна випадкова величина;

E – поточний кращий розв'язок;

E_{next} – знайдений гірший розв'язок;

T_{curr} – поточна температура.

Початкова температура, кінцева температура, розклад охолодження та кількість оброблених рішень на кожній температурі є параметрами алгоритму, від яких залежить якість рішення. Чим більша кількість ітерацій на кожну температуру й чим повільніше виконується зменшення температури – тим краще буде знайдене рішення завдяки більш детальному дослідженню області допустимих рішень, але час виконання також буде збільшуватись. Початкове рішення генерується шляхом випадкового переходу між вершинами, які не є стартовою та фінішною з використанням умови (2.1). Якщо ця умова не виконується для жодної з вершин – алгоритм переходить на фінішну вершину й закінчує побудову початкового розв'язку. Сусідні розв'язки будуються з використанням однієї з трьох дій: додавання вершини, видалення вершини та зміна порядку обходу двох випадкових вершин у маршруті. Ймовірність вибору кожної дії однакова. Рішення, які перевищують заданий час відкидаються.

3 ОПИС ПРИЙНЯТИХ РІШЕНЬ ПРИ РОЗРОБЦІ СИСТЕМИ

3.1 Опис архітектури (структури) інформаційної системи

Інформаційна система реалізовувалась з використанням триланкової архітектури. Триланкова архітектура розділяє серверну частину системи на 2 шари: шар бізнес-логіки та шар даних. Це дозволяє спростити розробку системи шляхом зменшення залежності між бізнес-логікою та функціями доступу до бази даних [17]. Зміна алгоритму в одному модулі не потребує зміни іншого модуля. Схема інформаційної системи подана на рис. 3.1.

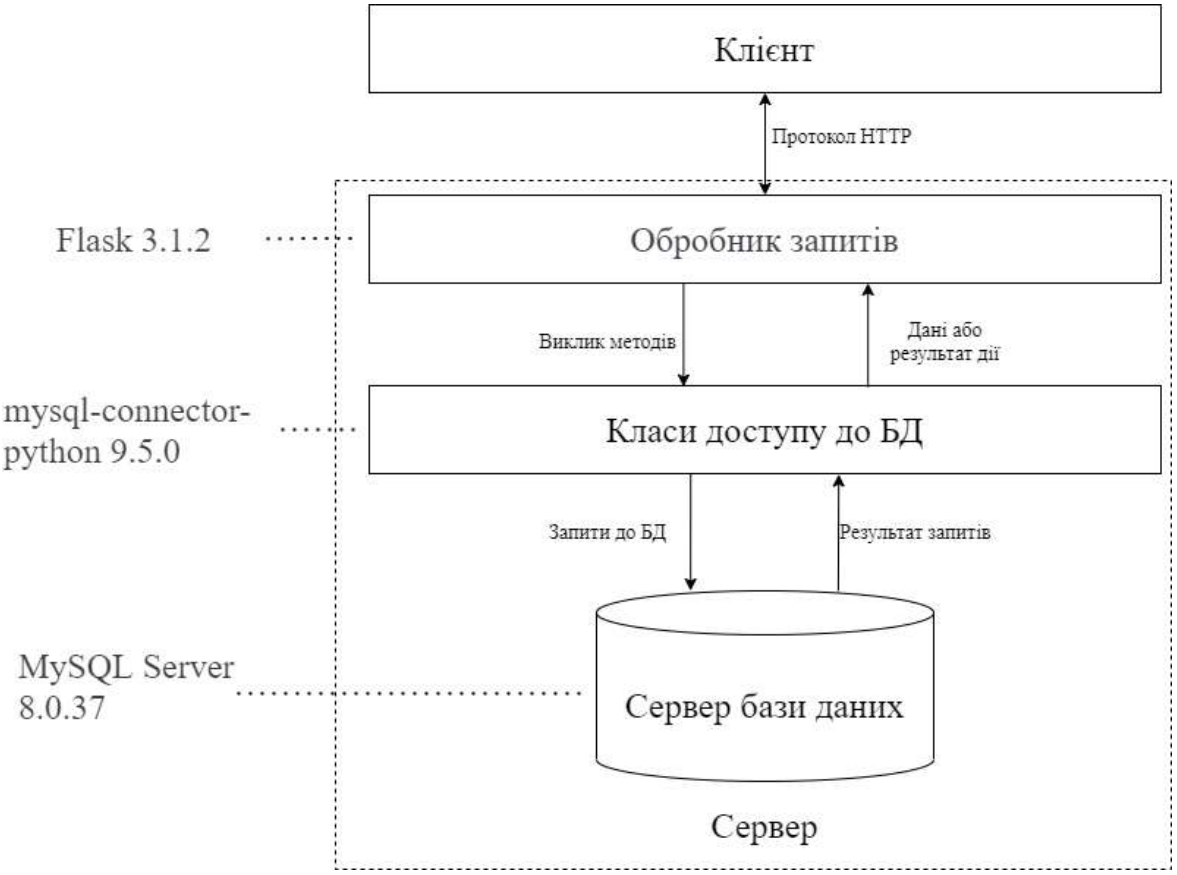


Рисунок 3.1 – Архітектура реалізованої системи

Взаємодія між клієнтом та сервером виконується за допомогою HTTP-запитів. Вся бізнес логіка, окрім валідації введених користувачем даних, використанням виконується на сервері. Сервер складається з обробника

запитів клієнта, який реалізує основну бізнес-логіку системи, класів доступу до БД, які виконують запити до бази даних для отримання або запису даних, що необхідно для виконання бізнес-логіки системи, та серверу бази даних, який виконує зберігання даних. У якості обробника запитів обрано WSGI фреймворк для створення веб-додатків Flask, оскільки він дозволяє більш швидко й просто виконати розробку веб-додатку в порівнянні з іншими фреймворками, та дозволяє масштабування до рівня складних веб-застосунків [18]. Робота з базою даних виконується з використанням mysql-connector-python, який надає інтерфейс для виконання та отримання результатів запиту до БД.

База даних реалізована з використанням СУБД MySQL, оскільки дана СУБД є простою у використанні, інтегрується з великою кількістю інструментів та мовами програмування та має високу швидкодію [19]. Розробка веб-сторінок виконувалась з використанням фреймворку bootstrap, оскільки він пришвидшує розробку веб-сторінок, дозволяє створювати адаптивний дизайн та є простим у використанні [20].

3.2 Логічне та фізичне моделювання даних системи

На основі діаграми потоків даних, що була розроблена в попередньому розділі, визначено наступні сутності, інформація про які має зберігатися у БД:

– User, у якій зберігається персональна інформація туриста;

– Route, у якій зберігається інформація про маршрут походу, його загальну довжину та час;

– Trip, у якій зберігається інформація про похід, дату його початку, завершення та статус заявки;

– Map, у якій зберігається інформація про ділянки шляху, що доступні для побудови маршруту.

На основі визначених сутностей виконано розробку ER-діаграми в

нотації IDEF1X за використанням CASE-засібу All Fusion Data Modeler. Логічну модель даних подано на рис. 3.2.

Під час створення діаграми з основних сутностей було виділено додаткові сутності: «роль» (Role), «Запрошення» (Invitation), «Користувач в поході» (User_in_trip), та «Шлях в маршруті» (Ways). Сутність Map було декомпозовано на 4 сутності: «точка інтересу» (POI), «тип точки інтересу» (POI_type), «дорога» (Way) та «Складність» (Difficulty).

У сутності User зберігається уся інформація про користувача: його ім'я, прізвище, вік, телефон, електрона пошта, логін, пароль та роль. Сутність пов'язана з сутністю Role, в якій зберігається інформація про ролі користувачів, зв'язком багато до одного, оскільки в декількох користувачів може бути однакова роль, але у користувача не може бути 2 ролі одночасно. Сутність також пов'язана із сутністю «Trip», в якій зберігається інформація про похід, зв'язком багато до багатьох, оскільки в одному поході може бути кілька учасників й один користувач може бути учасником кількох походів. Зв'язок утворюється з використанням проміжної сутності «User_in_trip».

Сутність «Invitation» зберігає інформацію про запрошення користувачів на приєднання до групи походу. Функція пов'язана двома зв'язками один до багатьох із сутністю «User», завдяки визначається користувач, який надіслав запрошення й користувач, якому надіслано запрошення. Також сутність пов'язана зв'язком багато до одного із сутністю «Trip», оскільки для одного походу може бути кілька запрошень для різних користувачів.

Сутність «Trip» зберігає загальну інформацію про похід і пов'язана із сутностями «Status» та «Route» зв'язками один до багатьох. Сутність «Status» є словником усіх можливих статусів походу й однаковий статус може бути у кількох походів. Сутність «Route» зберігає загальну інформацію про маршрут й в декількох походах може бути однаковий маршрут.

Маршрут зберігається у вигляді послідовності доріг, за яким він буде проходити. Сутність «Way» зберігає усі можливі дороги, які можуть бути використані в маршруті.

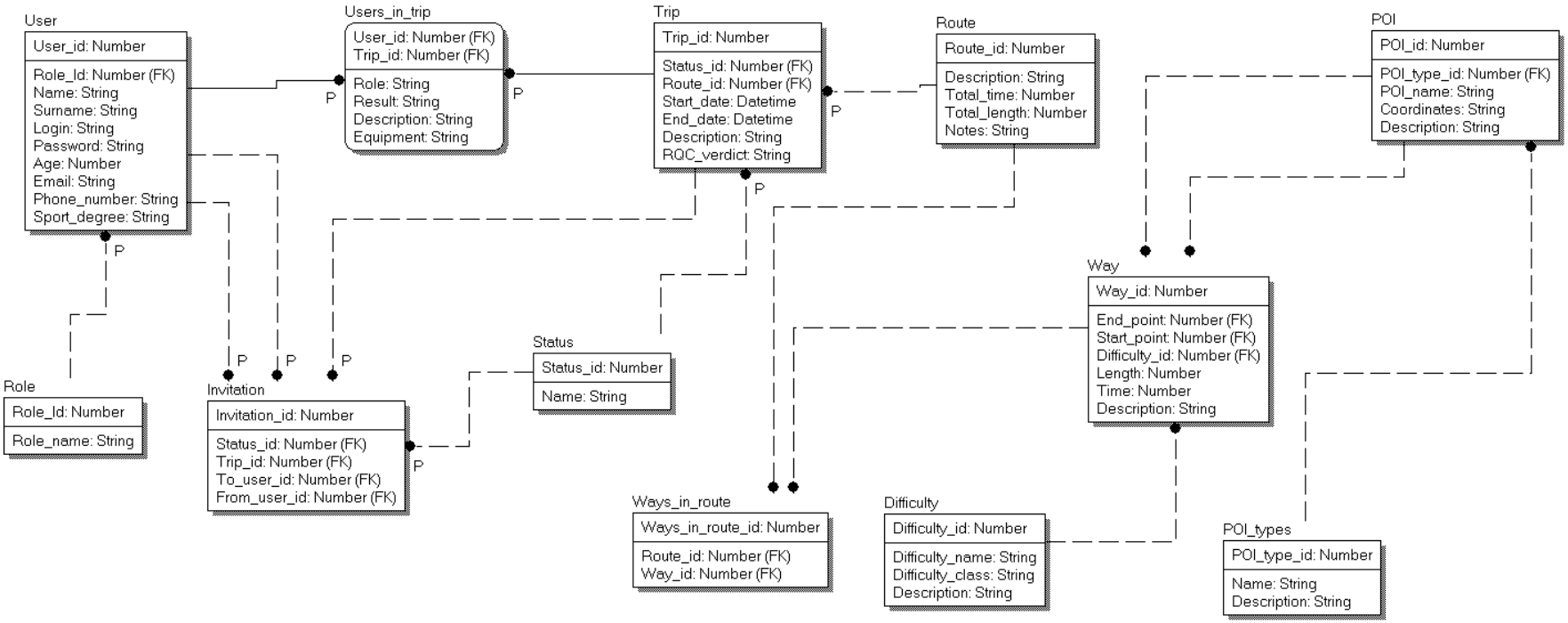


Рисунок 3.2 – Логічна модель даних

Сутність пов’язана із сутністю «Route» зв’язком багато до багатьох через проміжну сутність «Ways_in_route», в якій зберігається послідовність доріг в маршруті. Для кожної ділянки шляху визначається складність, інформація про яку зберігається в сутності «Difficulty», що пов’язана із сутністю «Way» зв’язком один до багатьох

Сутність «POI» зберігає інформацію про ключові точки, які використовуються при плануванні маршруту і які є стартовими та фінішними точками для доріг, що зберігаються у сутності «Way». Сутність пов’язана із сутністю «POI_types», яка визначає тип точки (звичайне перехрестя, точка, відвідування якої може зацікавити користувача, точка початку й завершення маршруту, тощо) зв’язком багато до одного.

Для атрибутів сутностей задано домени. Для числових атрибутів сутностей використовується домен Number, який позначає ціле число в діапазоні від -2^{31} до 2^{31}. Для збереження текстової інформації використовувався домен string, який позначає текст довільної довжини. Для атрибутів, які зберігають інформацію про дату, використовується домен datetime, який дозволяє зберігати дату та час з 1 січня 1000 року до 31 грудня 9999 року.

Фізичне моделювання даних виконувалось для СУБД MySQL. Результат фізичного моделювання даних подано на рисунку рис. 3.3. Для кожного домену вибрано відповідний тип даних обраної СУБД. Для полів з доменом Number обрано тип даних INT, для домену datetime обрано тип даних Date. Для домену String обрано типи даних VARCHAR та TEXT. Розмір змінної з типом VARCHAR визначався в залежності від даних, які будуть зберігатися у цій змінній. Стандартним розміром було встановлено 100, оскільки такий розмір точно дозволить зберегти короткі рядкові змінні (такі як ім’я, прізвище, логін, тощо). Для паролю встановлено довжину у 255, оскільки в цьому полі буде зберігатися хеш пароля.

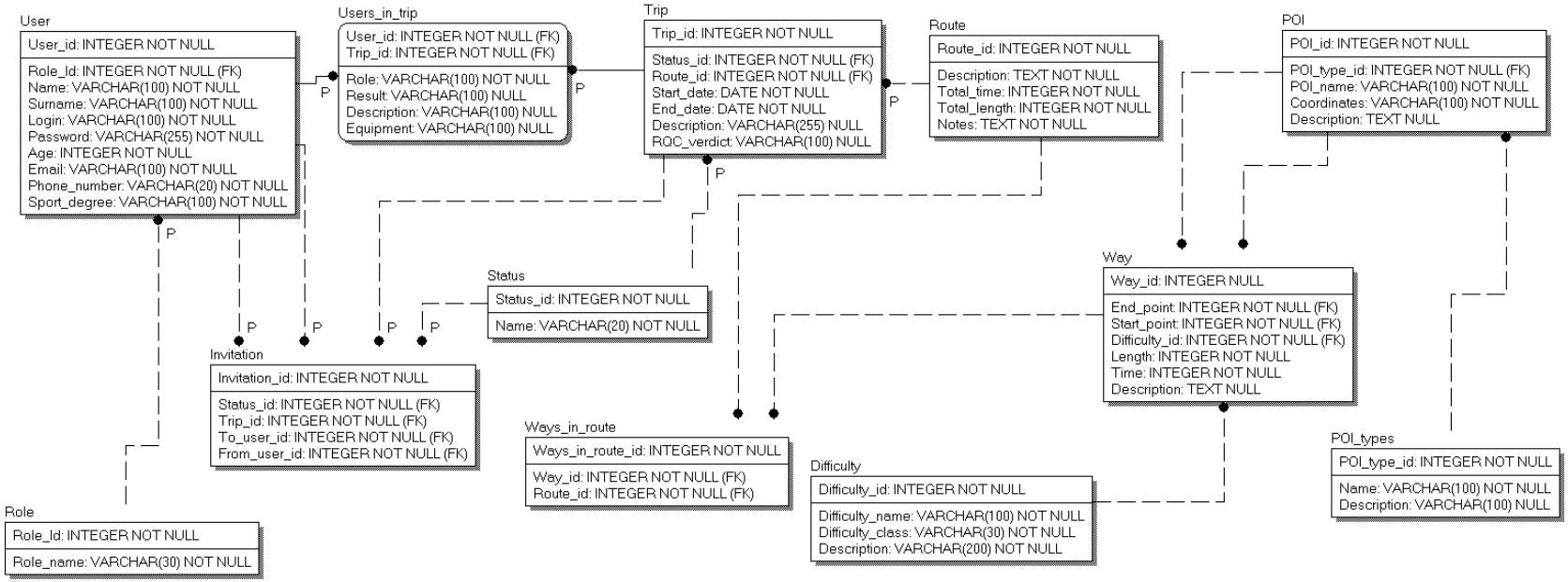


Рисунок 3.3 – Фізична модель даних

3.3 Створення бази даних на платформі СУБД MySQL

Створення бази даних з використанням СУБД MySQL виконувалось з використанням середовища розробки MySQL Workbench. На основі фізичної моделі, створеної в попередньому розділі, побудовано ERR-модель, яку подано на рисунку 3.4. Усі таблиці мають тип InnoDB, оскільки в порівнннні з типом MyIsam цей тип таблиць дозволяє використовувати зовнішні ключі для забезпечення цілісності даних, має вищу продуктивність при навантаженні, в якому використовуються різні типи запитів, дозволяє виконувати відновлення даних у разі виникнення збою, а також має підтримку транзакцій [21]

На ERR-моделі зображено 15 зв’язків типу один до багатьох між таблицями. Для кожного зв’язку виконано встановлення тригерів ON UPDATE та ON DELETE для забезпечення цілісності даних. Ці тригери визначають дії при оновленні поля, на яке посилається зовнішній ключ. Тригер може мати одне з трьох значень: RESTRICT, яке забороняє оновлення поля, на яке посилається зовнішній ключ, CASCADE, яке виконує оновлення зовнішнього ключа при оновленні поля, на яке посилається ключ, та SET NULL, яке встановлює значення зовнішнього ключа у значення NULL при видаленні запису, на який він посилається. Для усіх зовнішніх ключів таблиць встановлено значення тригеру ON DELETE RESTRICT для заборони видалення записів з таблиць, на які посилаються зовнішні ключі записів з інших таблиць

В таблиці User міститься 1 зовнішній ключ RoleId. Для забезпечення можливості зміни id ролі тригер встановлено у значення ON UPDATE CASCADE. Таблиця User_in_trip містить 2 зовнішні ключі: User_id та Trip_id. Ці два значення не повинні оновлюватися в системі, тому тригер встановлено у значення ON UPDATE RESTRICT.

Таблиця Invitation містить 4 зовнішні ключі, які посилаються на таблиці User, Trip та Status.

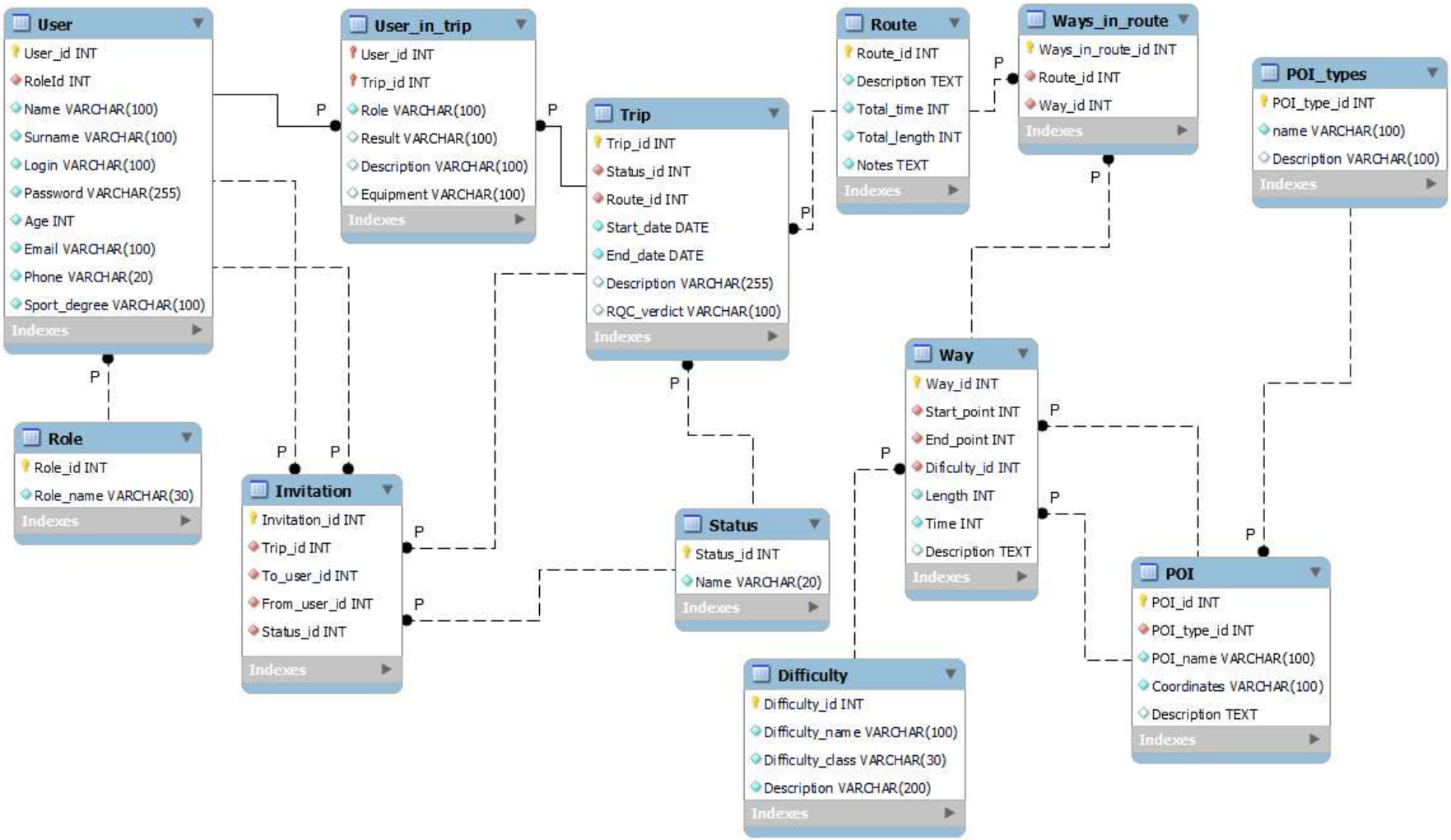


Рисунок 3.4 – ERR-модель даних

Для зовнішніх ключів, які посилаються на таблиці User та Trip встановлено тригер ON UPDATE RESTRICT для заборони зміни значень User_id та Trip_id. Для зовнішнього ключа Status_шв встановлено ON UPDATE CASCADE, оскільки значення ідентифікаторів статусів можуть змінюватися в процесі розробки

Таблиця Trip містить 2 зовнішні ключі: Status_id та Route_id які визначають статус та маршрут походу відповідно. Оскільки таблиця Status виконує функції словника статусів й ідентифікатори статусів можуть бути змінені в процесі розробки, для зовнішнього ключа Status_id встановлено ON UPDATE CASCADE. Для зовнішнього ключа Route_id встановлено ON UPDATE RESTRICT, оскільки маршрут не може змінювати свій унікальний ідентифікатор.

Таблиця Ways_in_route містить 2 зовнішні ключі: Way_id, який вказує на ділянку шляху, Route_id, який визначає маршрут, в який входить вказана ділянка шляху. Для обох зовнішніх ключів встановлено ON UPDATE RESTRICT, оскільки унікальні ідентифікатори ділянок шляху та маршрутів не мають змінюватися.

В таблиці POI є один зовнішній ключ, який вказує на таблицю з можливими значеннями типу точки, тому для цього зовнішнього ключа встановлено ON UPDATE CASCADE.

Таблиця Way містить три зовнішніх ключі: Start_point, End_point Difficulty_id. Ключі Start_point та End_point вказують на початкову та кінцеву точку ділянки шляху. Ідентифікатори точки не мають змінюватися, тому для цих зовнішніх ключів встановлено ON UPDATE RESTRICT. Ключ Difficulty_id посилається на таблицю Difficulty, яка є словником усіх можливих значень складності ділянки шляху, тому дял цього ключа встановлено ON UPDATE CASCADE.

3.4 Опис розробленої карти сайту

На основі діаграми варіантів використання виконано розробку карти сайту. Карта сайту складається з 12 сторінок і зображена на рисунку 3.5. На карті сайту подано сторінки для трьох типів користувачів: незареєстрований користувач, зареєстрований користувач та адміністратор системи. Незареєстрований користувач маж доступ до 4 сторінок: Головної сторінки, сторінки авторизації, сторінки реєстрації та сторінки з опублікованими маршрутами інших користувачів. Для отримання доступу до інших сторінок користувач повинен авторизуватися в системі. Авторизуватися у системі можна шляхом вводу свого логіну та пароля на сторінці авторизації або шляхом реєстрації у системі.

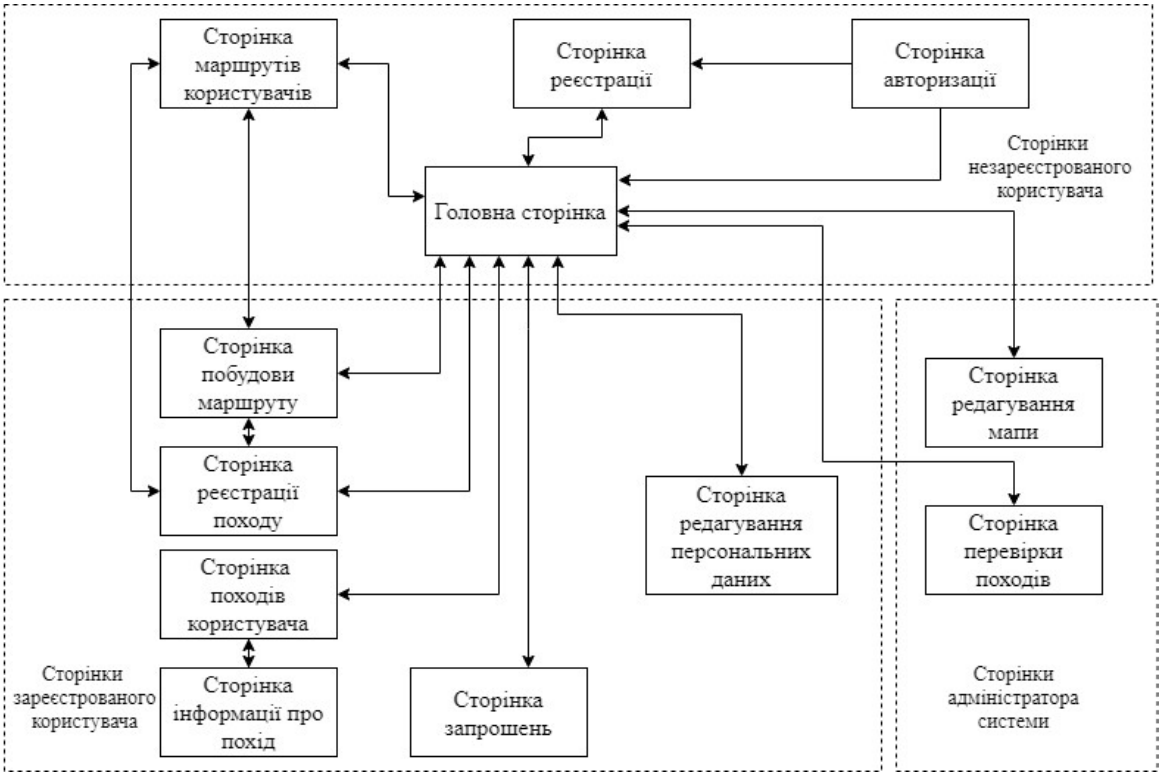


Рисунок 3.5 – Карта сайту

Зареєстрований користувач має доступ до усіх сторінок незареєстрованого користувача, а також до сторінок побудови маршруту,

реєстрації походу, сторінки з інформацією про усі свої походи та конкретний похід, а також до сторінки запрошень. На сторінці побудови маршруту користувач може виконати планування маршруту походу за ключовими точками, після чого перейти до реєстрації походу. На сторінку реєстрації походу можна перейти зі сторінки усіх маршрутів користувачів обравши маршрут із запропонованих на сторінці, за яким буде виконано реєстрацію походу. З головної сторінки користувач може перейти на сторінку з усіма своїми походами, з якої він може перейти на сторінку з повною інформацією про похід, яка включає у себе статус розгляду заявки на реєстрацію походу. На сторінці запрошень користувач може відправити запрошення на приєднання до групи іншим користувачам, а також переглядати, приймати та відхиляти запрошення інших користувачів. Зареєсрований також має доступ до сторінки зміни персональних даних

Адміністратор системи має доступ до сторінок незареєстрованого та зареєстрованого користувача, а також до сторінок редагування мапи й перевірки походів. На сторінці перевірки походів адміністратор може переглядати усю інформацію про похід, додати коментар, зареєструвати його або відправити на доопрацювання. На сторінці редагування мапи адміністратор може переглядати усі доступні для побудови маршрутів точки та ділянки шляху, змінювати їх час проходження, складність, а також додавати нові точки й ділянки шляху..

3.5 Розробка алгоритмів виконання бізнес-функцій

Однією з основних функцій системи є подача заявки на реєстрацію походу. Для виконання цієї бізнес-функції користувач має бути авторизованим у системі.. Для реєстрації походу користувач має визначити маршрут походу. Визначення маршруту походу може виконуватися шляхом вибору маршруту походу, який був створений іншим користувачем або виконанням створення власного маршруту на основі обраних точок інтересу.

Якщо користувач обирає перший варіант, то система надає йому список маршрутів інших користувачів, які позначені як доступні для перегляду усіма користувачами.

Користувач обирає маршрут, який йому найбільше подобається та переходить на сторінку реєстрації походу. Якщо користувач обирає варіант «Побудувати маршрут самостійно», система перенаправляє його на сторінку побудови маршруту. Для побудови маршруту користувач має вказати стартову й фінішну точку, ключові точки, які він хотів би відвідати, вагові коефіцієнти, які визначають наскільки сильно користувач хоче відвідати конкретну точку, та максимальний час походу. Система визначає оптимальний маршрут за вказаними точками й пропонує його користувачу. Якщо користувач незадоволений маршрутом, він може повернутися до етапу вибору ключових точок та визначення вагових коефіцієнтів. У разі, якщо створений маршрут задовольняє бажання користувача, система переходить на сторінку реєстрації походу. Для реєстрації походу користувач має вказати дату початку й завершення походу. Після введення цих даних система вносить інформацію про заявку у базу даних та надає їй статус «Очікує розгляду».

Бізнес-функція редагування мапи доступна лише для адміністраторів інформаційної системи. Першим кроком адміністратор обирає дію: виконати додавання нової ключової точки або ділянки шляху або відредагувати інформацію про вже існуючі. При додаванні нового об’єкта (ключової тачки або ділянки шляху) адміністратор вибирає тип об’єкта та вносить інформацію про нього. Якщо користувач вибрав опцію редагування існуючих ключових точок або ділянок шляху, система відображає йому усі точки та ділянки, інформація про які зберігається у системі. Після внесення змін до даних система зберігає оновлені дані в базі даних. Якщо адміністратор змінив час проходження або довжину якоїсь ділянки шляху – інформація про це має відобразитися в усіх маршрутах, які проходять через цю ділянку. Загальний час або довжина маршруту може збільшитись або зменшитись в

залежності від того зменшився чи збільшився час або довжина ділянки шляху після виконання оновлення.

3.6 Розробка тригерів для серверної частини системи

Як було визначено під час аналізу алгоритмів виконання бізнес-функцій, при зміні часу або довжини ділянки шляху, в усіх маршрутах, що проходять через цю ділянку шляху, мають змінитися загальний час та довжина відповідно до внесених змін. Також при додаванні або видаленні ділянки шляху з маршруту його довжина й час проходження мають збільшуватись (при видаленні – зменшуватись) на значення довжини й часу доданої або видаленої ділянки шляху. Для виконання цих функцій ватро застосовувати тригери. Тригер – це відкомпільована SQL-процедура, виконання якої обумовлене настанням певних подій усередині реляційної бази даних [22]. Для виконання зміни часу й довжини маршруту при додаванні або видаленні запису в таблицю Ways_in_route, що зберігає ділянки шляху маршруту використовувалися 2 тригери: `added_new_way` та `way_deleted`. Код першого тригеру подано нижче. Тригер виконується після виконання додавання запису в таблицю Way, щоб у разі помилки вставки не була порушена цілісність даних

```
CREATE TRIGGER added_new_way AFTER INSERT ON Ways_in_route
FOR EACH ROW BEGIN
    DECLARE new_time INT;
    DECLARE new_length INT;
    SELECT Time, Length INTO new_time, new_length FROM Way WHERE
Way_id = NEW.Way_id;
    UPDATE Route SET Total_time = Total_time + new_time,
Total_length = Total_length + new_length WHERE Route_id =
NEW.Route_id;
END
```

Тригер виконує 2 запити: отримання часу та довжини нової ділянки шляху в маршруті й записує їх у змінні `new_time` та `new_length`, й виконує

додавання часу й довжини доданої ділянки шляху до загальної довжини й тривалості маршруту, яка зберігається в таблиці Route. Код тригеру way_deleted подано нижче.

```
CREATE TRIGGER way_deleted AFTER DELETE ON Ways_in_route
FOR EACH ROW BEGIN
    DECLARE new_time INT;
    DECLARE new_length INT;
    SELECT Time, Length INTO new_time, new_length FROM Way WHERE
Way_id = NEW.Way_id;
    UPDATE Route SET Total_time = Total_time - new_time,
Total_length = Total_length - new_length WHERE Route_id =
NEW.Route_id;
END
```

Тригер виконується після видалення запису з таблиці Ways_in_route й виконує такі само дії, як і тригер added_new_way, але час та довжина видаленої ділянки шляху віднімаються від загальної довжини й тривалості маршруту.

Для виконання оновлення інформації про час та довжину маршруту після виконання оновлення інформації про ділянку шляху створено тригер way_updated, код якого подано нижче. Тригер виконується після виконання оновлення таблиці Way

```
CREATE TRIGGER way_updated AFTER UPDATE ON Way FOR EACH ROW BEGIN
    DECLARE time_diff INT;
    DECLARE length_diff INT;
    SET time_diff = NEW.Time - OLD.Time;
    SET diff_length = NEW.Length - OLD.Length;
    UPDATE Route as r JOIN Ways_in_route as wir ON wir.Route_id =
r.Route_id SET r.Total_time = r.Total_time + time_diff, r.Total_length
= r.Total_length + length_diff WHERE wir.Way_id = NEW.Way_id;
END
```

Першим кроком тригер розраховує різницю між старими та новими значеннями довжини й часу, які записується у змінні length_diff та time_diff відповідно. Наступним кроком тригер виконує оновлення маршрутів, що зберігаються в таблиці Route. Для цього виконується

визначення маршрутів, в яких присутня ділянка маршруту, яка була оновлена, з використанням конструкції JOIN для таблиці `Ways_in_route`. Для часу проходження та довжини всіх знайдених маршрутів додається різниця у часі та відстані між старим та новим значенням.

3.7 Опис рішень прийнятих при розробці програмного коду системи

Для створення підключення до бази даних використовувався метод get_contection. Метод створює та повертає об’єкт підключення до бази даних, який буде використовуватись іншими класами та методами для роботи з базою даних.

Для побудови маршруту програма отримує усі точки та ділянки шляху з БД. Інформація про точки з типом «Ключова точка» й «Точка старту або фінішу» надається користувачу. Після визначення користувачем точок старту, фінішу й ключових точок маршруту система виконує планування маршруту з використанням підходу, описаного в розділі 2. Розрахунок найшвидших шляхів виконується з використанням реалізації алгоритму Дейкстри бібліотеки NetworkX, яка надає функціонал для виконання операцій на графах [23].

Для визначення точок, які будуть в маршруті, та послідовності їх відвідування використовуються 3 алгоритми, які визначені під час проектування системи:: алгоритм пошуку в глибину з обмеженням, мурашиний алгоритм та алгоритм імітації відпалу. Для роботи алгоритми використовують матрицю відстаней – квадратну матрицю зі стороною n, елементи якої показують відстань від вершини *i* до вершини *j*. Значення n дорівнює кількості обраних користувачем ключових точок. Оскільки ідентифікатори обраних користувачем вершин не завжди починаються з 0 і є послідовними, для кожної зі вказаних вершин необхідно визначити індекс від 0 до n для побудови матриці відстаней. Для виконання переходу від поточних індексів вершин до індексів у проміжку від 0 до n використовуються словник

orig_to_new, в якому ключ – значення id вершини, що збережене у БД, значення – індекс від 0 до n. Код функції, яка виконує формування словника, подано нижче:

```
    def remap_vertices (routes):
        vertices = set()
        for (A, B) in routes.keys()
            vertices.add(A)
            vertices.add(B)
        orig_to_new = {orig_id: i for i, orig_id in
enumerate(vertices)}
        new_to_orig = {i: orig_id for i, orig_id in
enumerate(vertices)}
        return orig_to_new, new_to_orig
```

Функція приймає у якості параметру словник маршрутів, отриманих в результаті роботи алгоритму пошуку найкоротшого шляху між вершинами у графі. Ключем словника є множина вершин (A, B), де A – стартова вершина, B – фінішна вершина. У якості значення словник містить послідовність вершин маршруту та його час. Функція також повертає словник new_to_orig, в якому ключі – індекс від 0 до n, значення – id вершини, що зберігається у базі даних. Словник new_to_orig використовується після визначення послідовності вершин в маршруті. Для формування маршруту, який буде відвідувати ключові точки в зазначеній послідовності використовується функція reconstruct_full_path. Функція приймає у якості параметрів список ключових точок маршруту та словник маршрутів. Результатом роботи функції є послідовність вершин маршруту, яка включає у себе не ключові точки (звичайні перехрестя).

Усі ключові точки, що доступні для планування походу, показуються на мапі. Мапа використовується із сервісу OpenStreetMaps – проекту, що спрямований на створення детальної та безкоштовної карти місцевост і який дозволяє вільно використовувати створену проектом карту для будь-яких цілей [24] за умови наявності посилання на проект та його учасників. Для роботи з картою на сторінці використовується бібліотека leaflet [25], яка

дозволяє відображати мапу на сторінці, додавати мітки, маршрути та виконувати інші дії з мапою. Інтерфейс сторінки для вибору ключових точок маршруту подано на рисунку 3.6.

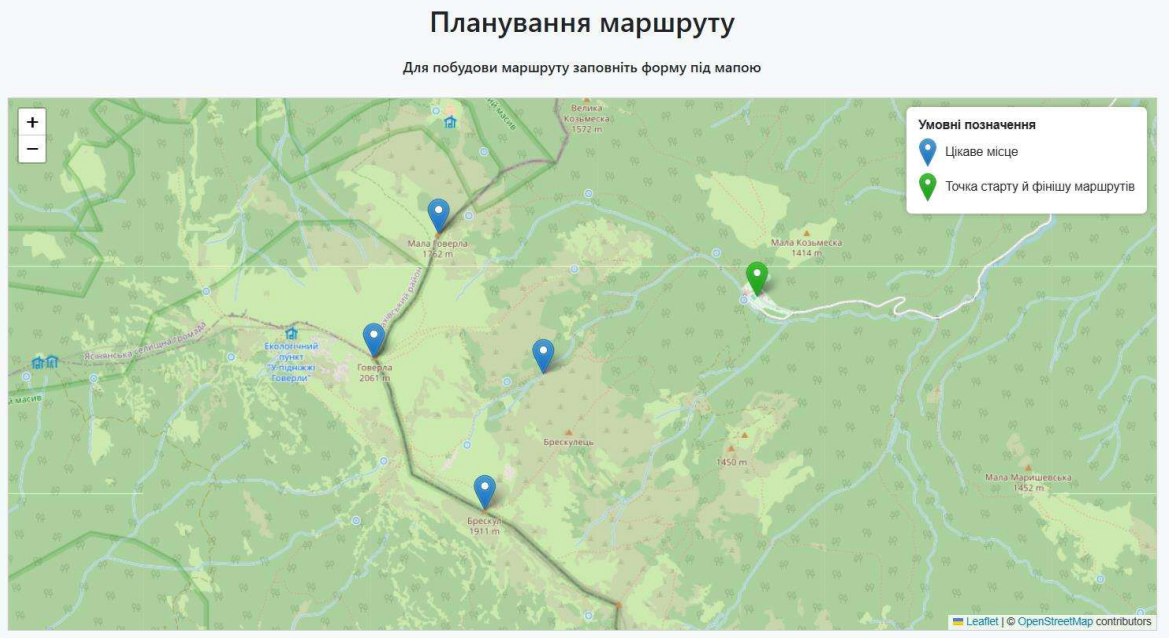


Рисунок 3.6 – Мапа для планування маршруту

На мапі позначено 5 ключових точок, які доступні для планування маршрутів: навчально-спортивна база Заросляк, гора Говерла, гора Мала Говерла, Прутський (Говерлянський) водоспад та гора Брескул. В подальшому в систему буде вноситись інформація про інші ключові точки. Для побудови маршруту користувач має обрати стартову та фінішну точку, вказати наявний на похід час, а також обрати одну чи декілька точок, що позначені як цікаві, та надати їм оцінку. Форма для введення цих даних подана на рисунку 3.7. Після введення необхідних даних для планування маршруту, система розраховує оптимальний маршрут, який буде проходити через усі або частину обраних користувачем точок й не перевищуватиме заданий час. Маршрут показується у вигляді ліній між точками. Для відображення послідовності обходу точок біля кожної вказується її номер у маршруті (рис 3.8).

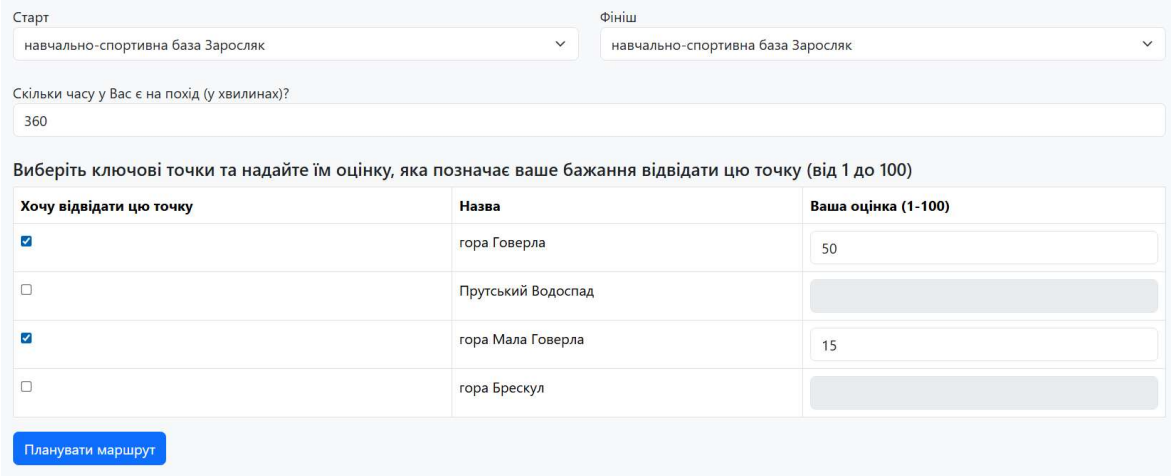


Рисунок 3.7 – Форма для планування маршруту

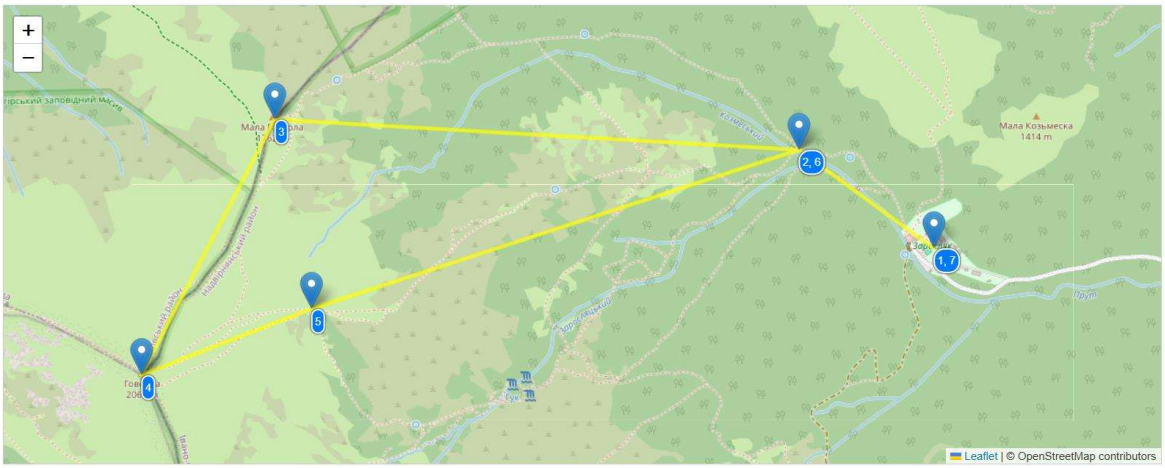


Рисунок 3.8 – Побудований маршрут

Для більш зручного розуміння маршруту під мапою показується послідовність обходу точок у вигляді таблиці, а також загальна інформація про маршрут – довжина, час та сумарна оцінка (рис. 3.9). Користувач може зберегти побудований маршрут й перейти до реєстрації походу або повернутися до визначення ключових точок для побудови іншого маршруту без збереження поточного. Оскільки маршрути зберігається у якості послідовності ділянок шляху, а не у якості послідовності точок, для збереження маршруту послідовність вершин необхідно перетворити на послідовність ребер.

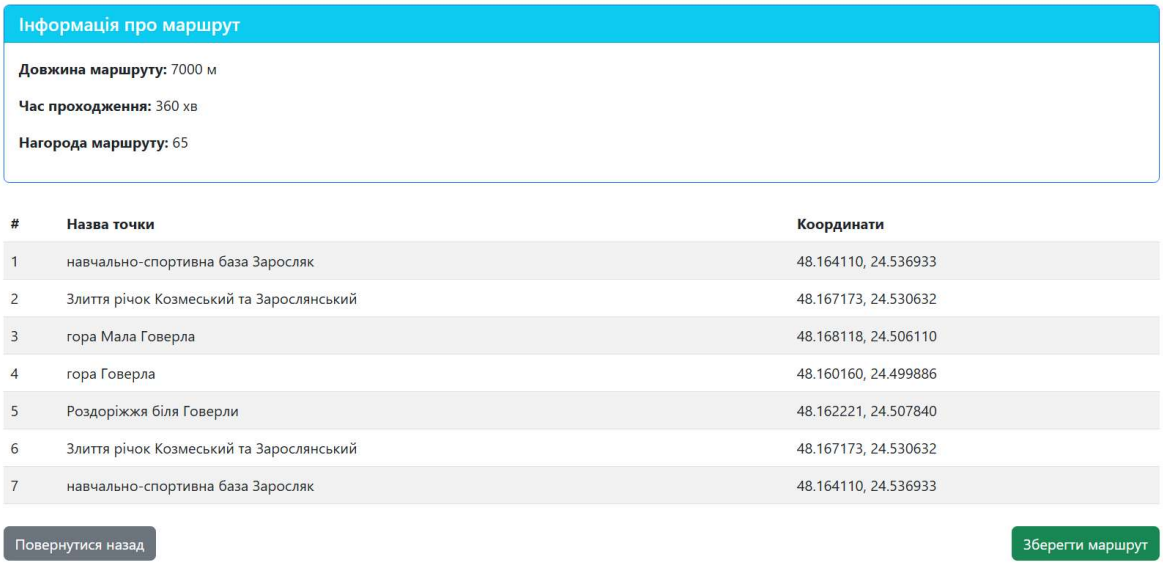


Рисунок 3.9 – інформація про маршрут

Для цього використовується функція `convert_route`, яка отримує на вхід послідовність вершин й повертає список ребер маршруту. Після виконання цієї функції відбувається збереження інформації про маршрут шляхом додавання записів про ділянки шляху в маршруті у таблицю Ways_in_route.

4 ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ ВИРІШЕННЯ ЗАДАЧІ

4.1 Порівняльний аналіз алгоритмів пошуку найкоротшого шляху на графі

Для порівняння алгоритмів пошуку найкоротшого шляху у графі виконувався їх запуск на трьох графах з 1000, 5000 та 10000 вершинами. Граф створювався шляхом випадкового розміщення точок на координатній площині розміром 1000x1000, яка необхідна для визначення оціночної відстані від поточної вершини до цільової для алгоритму A*. Кожна вершина з'єднувалася з 3 найближчими вершинами. За допомогою алгоритму мінімального стягуючого дерева визначалися ребра, які необхідно додати для забезпечення зв'язності графу. Вага кожного ребра встановлювалась у значення його довжини. Оскільки алгоритмам неважливо, що позначає вага ребра, алгоритми пошуку найкоротшого шляху на графі можна застосовувати для пошуку найшвидшого шляху, якщо вага ребер позначає час. Для аналізу використовувалась стандартна реалізація алгоритмів бібліотеки NetworkX мови програмування python, яка буде використана для розробки ІС. Для кожного графу створено набір з 5 пар вершин (початкова та фінішна), які використовувались у якості вхідних даних алгоритмів та евклідова відстань між якими перевищує 1000 одиниць. Це обмеження дозволяє збільшити середній час виконання алгоритмів. Ефективність алгоритмів оцінювалась за часом виконання пошуку. Вимірювання часу роботи алгоритму виконувалось за допомогою бібліотеки datetime, Для зменшення похибки алгоритм запускався 3 рази для кожної пари вершин, фінальний час пошуку розраховувався як середнє значення часу пошуку на усіх трьох запусках. Середній час пошуку найшвидшого шляху кожним алгоритмом для кожного зі створених графів подано у таблиці 4.1

Таблиця 4.1 – Час виконання алгоритмів в залежності від кількості вершин у графі

	1000 вершин	5000 вершин	10000 вершин
алгоритм Дейкстри	0,0044 с	0,03 с	0,075 с
двоспрямований алгоритм Дейкстри	0,0035 с	0,023 с	0,054 с
алгоритм A*	0,0033 с	0,022 с	0,058 с

Аналіз часу виконання алгоритмів доводить, що час виконання пошуку найкоротшого шляху з використанням алгоритму Дейкстри є на 20-30% більшим за час пошуку найкоротшого шляху з використанням двоспрямованого алгоритму Дейкстри та алгоритму A* (табл. 4.1). Різниця в часі роботи двоспрямованого алгоритму Дейкстри та алгоритму A* є незначною.

4.2 Порівняльний аналіз алгоритмів розв’язання задачі орієнтування

Порівняння алгоритмів для розв’язання задачі орієнтування здійснювалось за сумарною вагою знайденого маршруту R_{best} й часом виконання пошуку. Час виконання пошуку не повинен перевищувати 10 секунд на будь-яких наборах даних, оскільки це дозволить не створювати вузьке місце роботи функцій ІС, які будуть використовувати ці алгоритми. Аналіз виконувався на 8 графах , які мають від 5 до 40 вершин (з кроком у 5 вершин) без урахування стартової та фінішної вершини. Графи генерувалися випадковим чином на площині 100x100, вага ребра дорівнює евклідовій відстані між вершинами. Кожній вершині, окрім стартової та фінішної, надавалась випадковий ваговий коефіцієнт зі значенням від 1 до 100 балів. Дослідження проводилося в трьох сценаріях, які відрізняються кількістю

вершин в найкращому рішенні:

– сценарій №1 – мало часу (5–25% вершин). Імітує сценарій, в якому користувач обрав дуже багато місць, на обхід більшої частини яких не вистачить часу. Це може статися у випадку, коли користувач не знає точно куди хоче піти або помилився в оцінці часу на шлях для обходу усіх точок;

– сценарій №2 – середня кількість часу (40–60% вершин). Імітує середній сценарій між попереднім та наступним сценарієм;

– сценарій №3 – багато часу (75–95% вершин). Імітує сценарій, в якому користувач доволі точно оцінив час на маршрут між точками, або він має достатньо багато часу, щоб обійти більшість вершин графу.

Сценарій, в якому часу вистачить на відвідування усіх вершин, не розглядався, оскільки в такому випадку спрацює умова ранньої зупинки виконання алгоритму. Оскільки алгоритми виконують лише максимізацію нагороди без мінімізації витраченого часу, якщо маршрут міститиме усі вершини, що вказані користувачем, він буде одразу запропонований користувачу. Для підбору початкових даних (стартової й фінішної вершини та ліміту часу) використовувався пошук в глибину, оскільки він завжди знаходить найкращий маршрут. У випадках, коли час виконання цього алгоритму ставав занадто довгим, підбір початкових даних виконувався за допомогою мурашиного алгоритму.

На рисунку 4.1 подано результати для першого сценарію. Час виконання пошуку в глибину був однаковим для графів із меншою, ніж 25, але далі час виконання зростає. Час виконання алгоритму імітації відпалу більший за час виконання мурашиного алгоритму на 1,5 секунди. Сумарна оцінка найкращого знайденого шляху R_{best} для кожного алгоритму на кожному графі подана у таблиці 4.2. Мурашиний алгоритм майже завжди знаходив найкраще рішення, алгоритм імітації відпалу у 6 з 8 експериментах не знаходив найкраще рішення.

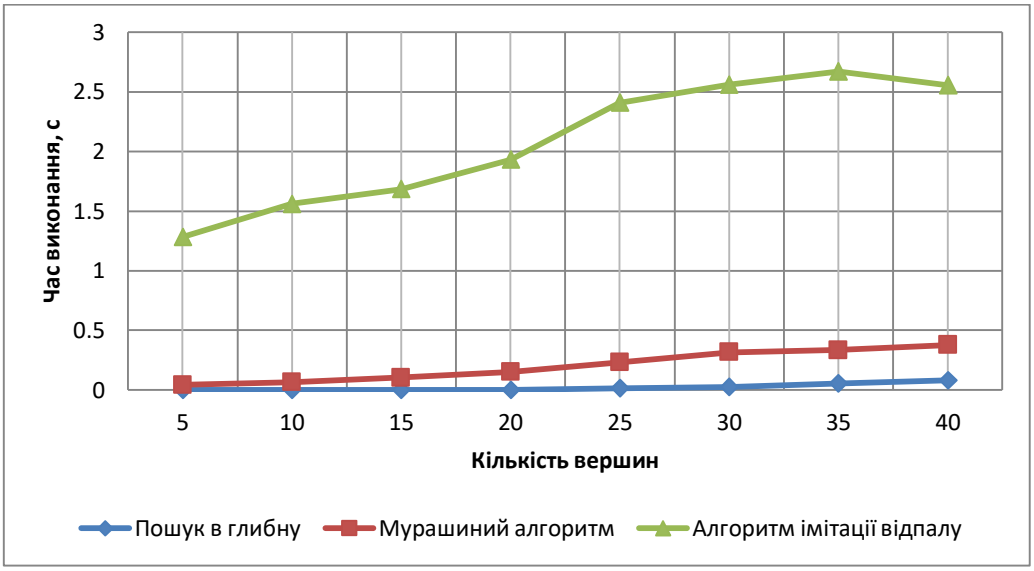


Рисунок 4.1 – Залежність часу виконання від кількості вершин для сценарію №1

Таблиця 4.2 – Залежність R_{best} від кількості вершин у графі для сценарію №1

	5	10	15	20	25	30	35	40
Пошук в глибину	87	73	162	268	355	413	455	460
Мурашиний алгоритм	87	73	162	268	351	413	455	460
Імітований відпал	69	66	138	268	351	407	423	440

На рисунку 4.2 подано результати для другого сценарію. Час виконання пошуку в глибину різко зростав для графів з більше ніж 15 вершинами, час виконання мурашиного алгоритму та алгоритму імітації відпалу збільшувався повільно при збільшенні кількості вершин у графі, але імітований відпал в середньому працював на 2 секунди довше. Сумарна оцінка найкращого знайденого шляху R_{best} для кожного алгоритму на кожному графі подано у таблиці 4.3. Пошук в глибину завжди дає точний розв'язок. Інші знаходили найкращий розв'язок для графів при кількості вершин менше за 15, далі знайдені розв'язки не є найкращими, але нагорода

шляху, отриманого за допомогою мурашиного алгоритму, була не меншою за нагороду найкращого шляху знайденого алгоритмом імітації відпалу для усіх графів, що розглядалися.

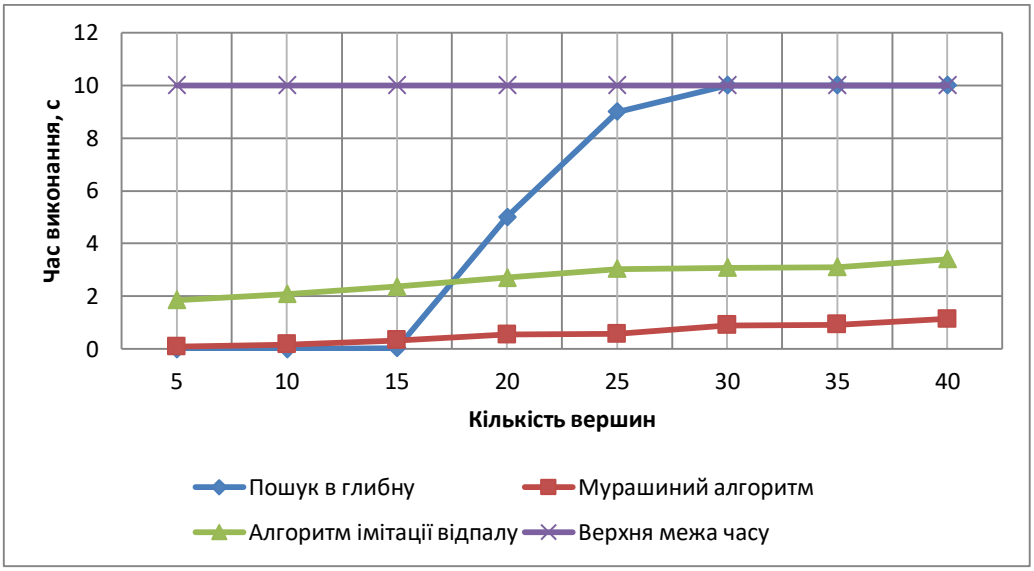


Рисунок 4.2 – Залежність часу виконання від кількості вершин для сценарію №2

Таблиця 4.3 – Залежність R_{best} від кількості вершин у графі для сценарію №2

	5	10	15	20	25	30	35	40
Пошук в глибину	214	271	526	748	832	-	-	-
Мурашиний алгоритм	214	271	526	735	808	988	876	1139
Імітований відпал	214	271	526	733	773	913	867	1130

На рисунку 4.3 подано результати для третього сценарію. Час виконання пошуку в глибину різко зростав для графів з більше ніж 10 вершинами, час виконання мурашиного алгоритму та алгоритму імітованого відпалу при збільшенні кількості вершин у графі зростав більш повільно, але імітований відпал в середньому працював на 2,5 секунди довше.

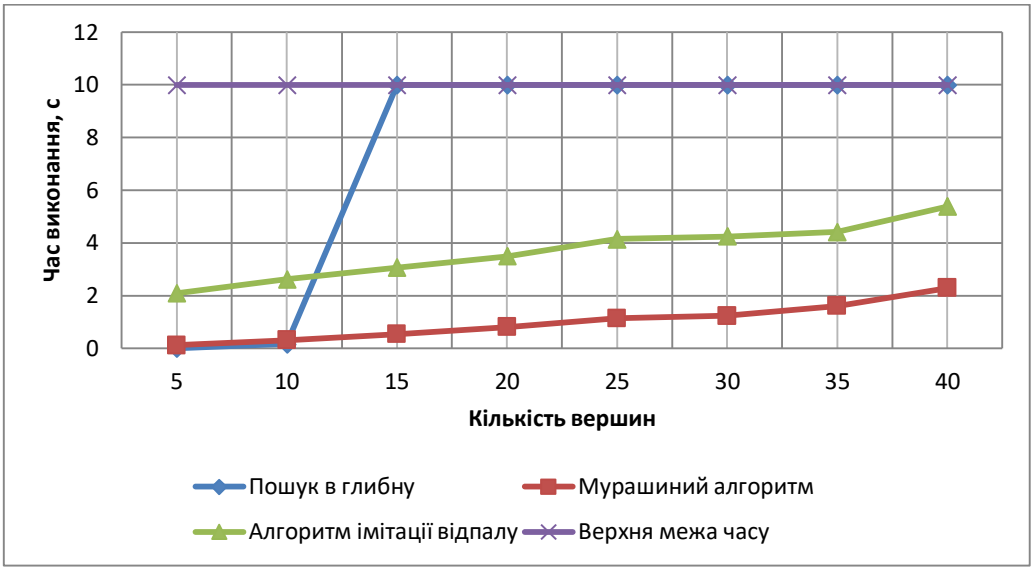


Рисунок 4.3 – Залежність часу виконання від кількості вершин для сценарію №3

Сумарна оцінка оптимального маршруту, знайденого з використанням кожного алгоритму на кожному графі подана у таблиці 4.3. На відміну від попередніх сценаріїв, алгоритм імітованого відпалу на графах з більш ніж 10 вершинами знаходив рішення, що мають на 5-10% більшу оцінку, ніж мурашиний алгоритм

Таблиця 4.3 – Залежність R_{best} від кількості вершин у графі для сценарію №3

	5	10	15	20	25	30	35	40
Пошук в глибину	304	551	-	-	-	-	-	-
Мурашиний алгоритм	304	551	857	1070	1416	1436	1654	2247
Імітований відпал	304	551	850	1072	1450	1519	1734	2218

4.3 Рекомендації стосовно вибору алгоритмів

На основі аналізу алгоритмів, виконаного в попередніх розділах,

надаються наступні рекомендації з використання алгоритмів для розв'язання задачі при розробці інформаційної системи. Для виконання пошуку найшвидшого шляху у графі рекомендовано використовувати двоспрямований алгоритм Дейкстри, оскільки різниця у часі пошуку найшвидшого шляху в порівнянні з алгоритмом A* складає менше ніж 0.004 секунди, але цей алгоритм не потребує використання евристичних функцій для виконання пошуку. Для розв'язання задачі орієнтування пропонується використовувати алгоритм пошуку в глибину, якщо кількість ключових точок вказаних користувачем не перевищує 10. Якщо кількість таких точок перевищує 10, цей алгоритм використовувати не можна, оскільки час виконання значно зросте, що виявлено під час дослідження сценарію №3. Для розв'язання задачі орієнтування із більше ніж 10 вершинами рекомендовано використовувати мурашиний алгоритм, оскільки час його виконання більш ніж у 2 рази менший, ніж час виконання алгоритму імітованого відпалу, й оцінка шляху R_{best} для цього алгоритму була вищою для сценаріїв №1 та №2. Під час аналізу сценарію №3 R_{best} для алгоритму імітованого відпалу була вищою для деяких випадків, але ця різниця не перевищує 5-10% при збільшенні часу розв'язання більш ніж у 2 рази в порівнянні з мурашиним алгоритмом.

4.4 Рекомендації стосовно подальшого розвитку описаного підходу

Підхід, описаний в підрозділі 2.1, дозволяє розв'язувати поставлену задачу, але в нього можна внести модифікації, які дозволять врахувати додаткові умови:

– додатковий критерій оптимальності (наприклад довжина чи час проходження маршруту);

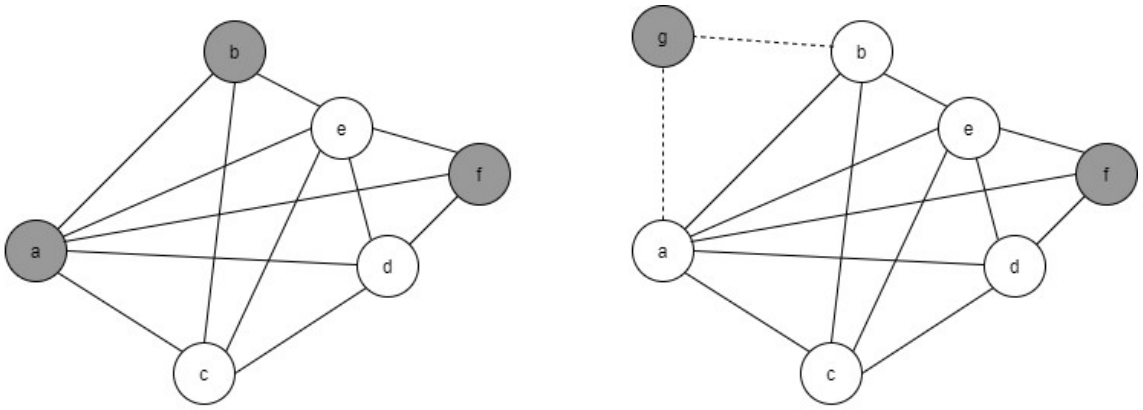
– вибір декількох стартових та фінішних точок;

– врахування часу на відвідування точок.

Для додавання додаткового критерію оптимальності необхідно змінити

алгоритм порівняння нового та поточного найкращого маршрутів. В поточній версії алгоритмів, які виконують ро'зв'язок задачі орієнтування, порівняння виконується лише за сумарною оцінкою шляху, новий шлях стає поточним найкращим тільки якщо його сумарна оцінка є більшою, в інших випадках знайдений шлях відкидається. При додаванні додаткових критеріїв оптимальності у випадку рівної нагороди нового й найкращого маршрутів вони будуть порівнюватися з використанням додаткових критеріїв оптимальності.

В другій модифікації користувач задає дві непорожні множини точок з яких може починатися і в яких може закінчуватися маршрут, система будує маршрут зі стартом в одній точці з першої множини й фінішем – у точці з другої множини. Для виконання побудови маршруту з декількома можливими точками старту пропонується ввести додаткову вершину, як показано на рисунку 2.6, б. Додавання нової вершини виконується для графу, на якому буде виконуватися розв'язок задачі орієнтування (рис 2.6, а).



а) Початковий граф б) Граф з доданою вершиною

Рисунок 2.6 – модифікація підходу з кількома стартовими вершинами

Для початкового графу вершини *a* та *b* є стартовими, вершина *f* – фінішна. Суцільними лініями позначені ребра з додатньо вагою, пунктирною – ребра із вагою, що дорівнює 0. На другому графі додано вершину g, яка є

новою стартовою вершиною. Вершина поєднана з усіма стартовими вершинами, які визначив користувач, вербрами з вагою 0, щоб вибір стартової вершини не впливав на загальний час маршруту. Такий підхід дозволяє не виконувати модифікацію програмного коду алгоритмів. Для випадків з декількома фінішними вершинами пропонується змінити обмеження (2.1) наступним чином:

$$T_{curr} + T_{ij} + T_{jk} \leq T_{max}, \qquad \exists k: j \notin V_{visited}, j \neq k, k \in V_{finish} \qquad (4.1)$$

де V_{finish} – множина фінішних вершин. Якщо існує така вершина, після відвідування якої вистачить часу на перехід хоча б в одну з фінішних вершин, то маршрут не буде переходити у фінішну вершину. Підхід з додатковою фінішною вершиною, яка буде поєднана ребрами з нульовою вагою з іншими фінішними вершинами, буде більш важким для реалізації за рахунок більш складного обчислення обмеження (4.1).

Для врахування часу відвідування обраних вершин користувач для кожної обраної ключової точки задає час, який він планує витратити на перебування у цій точці. Система враховує цей час під час побудови маршруту: сума часу, витраченого на перехід за ребрами маршруту, а також загальний час на відвідування ключових точок, через які проходить маршрут, не перевищує встановлене користувачем обмеження. Обмеження (1.3) для задачі буде мати наступний вигляд:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n} S_i t_i x_{ij} \leq T_{max}$$

Де t_i – час, який користувач планує витратити у вершині S_i. Для реалізації цієї модифікації необхідно внести зміни у матрицю відстаней, яка використовується алгоритмами і позначає час, який потрібен на перехід з вершини i у вершину j. До елементів кожного стовпця i матриці, окрім

стовпців, які позначають стартову й фінішну вершину, додається час t_i, значення елементів головної діагоналі матриці не змінюється. Час на відвідування стартової й фінішної вершини не враховується, оскільки маршрут вважається розпочатим після виходу зі стартової вершини й завершеним у момент заходу до фінішної вершини.

Як приклад розглянемо граф з 6 вершинами, вершина з індексом 1 є стартовою вершиною, вершина з індексом 6 – фінішною. Матриця відстаней графу подана нижче:

$$\begin{bmatrix} 0 & 1 & 5 & 6 & 4 & 5 \\ 1 & 0 & 2 & 8 & 4 & 5 \\ 5 & 2 & 0 & 4 & 2 & 7 \\ 6 & 8 & 4 & 0 & 1 & 2 \\ 4 & 4 & 2 & 1 & 0 & 8 \\ 5 & 5 & 7 & 2 & 8 & 0 \end{bmatrix}$$

Нехай користувач хоче витратити в кожній вершині (окрім стартової та фінішної) час, який дорівнює індексу вершини. Значення елементів кожного стовпця, окрім першого та останнього, збільшуються на значення індексів цих стовпців, елементи головної діагоналі залишаються без змін. Оновлена матриця відстаней подана нижче:

$$\begin{bmatrix} 0 & 3 & 8 & 10 & 9 & 5 \\ 1 & 0 & 5 & 12 & 9 & 5 \\ 5 & 4 & 0 & 8 & 7 & 7 \\ 6 & 10 & 7 & 0 & 6 & 2 \\ 4 & 6 & 5 & 5 & 0 & 8 \\ 5 & 7 & 10 & 6 & 13 & 0 \end{bmatrix}$$

Оскільки час на відвідування вершини враховується лише на етапі розв’язання задачі орієнтування, час на відвідування вершини *a* не буде впливати на загальний час маршруту між двома іншими вершинами, який проходить через вершину *a*. На практиці це описує ситуацію, в якій користувач пройшов повз ключову точку без зупинки в ній.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи проаналізовано предметну область проведення походів, виявлено основні проблеми під час планування та проведення походів. Виконано проектування системи, створення бази даних та реалізацію інформаційної системи, яка дозволить виконувати планування та реєстрацію походів.

Для виконання планування походу на основі місць, які хоче відвідати користувач, та наявного в нього часу запропоновано двоетапний підхід, який дозволить визначити точки, які будуть в маршруті, порядок їх відвідування й маршрут між ними. На першому етапі виконується пошук найкоротших шляхів між усіма вершинами графу, що визначені користувачем. На другому етапі виконується розв’язання задачі орієнтування для визначення точок, що будуть у маршруті та послідовності їх відвідування. Для кожного з етапів підходу визначено алгоритми, які можуть бути використані для їх виконання, та виконано їх порівняльний аналіз. Для виконання першого етапу розглядалися алгоритм Дейкстри, двоспрямований алгоритм Дейкстри та алгоритм A*. Для виконання першого етапу рекомендовано використовувати двоспрямований алгоритм Дейкстри, оскільки він є простим для використання в програмі та є одним з найшвидших алгоритмів, що розглядалися.

Для виконання другого етапу розглядалися три алгоритми з різною стратегією пошуку рішення: алгоритм пошуку в глибину, мурашиний алгоритм та алгоритм імітації відпалу. Алгоритми порівнювались за часом роботи та сумарною оцінкою на графах з різною кількістю вершин. На кожному з графів порівняння алгоритмів виконувалося за трьома сценаріями, які відрізняються кількістю вершин в оптимальному шляху. За результатами дослідження визначено, що алгоритм пошуку в глибину є найбільш ефективним, якщо кількість вершин у графі не перевищує 10. В інших випадках найбільш ефективним алгоритмом є мурашиний алгоритм.

Алгоритм імітації відпалу не рекомендовано використовувати для розв’язання задачі, оскільки час його виконання в середньому на 2 секунди більший за час виконання мурашиного алгоритму, й сумарна оцінка найкращого шляху не перевищує оцінку шляху, знайденого з використанням мурашиного алгоритму.

Для розробленого підходу описані модифікації, які дозволять враховувати додаткові умови при вирішенні задачі, такі як врахування часу на відвідування точки та розв’язання задачі з кількома точками старту та фінішу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про туристські маршрутно-кваліфікаційні комісії. *Турклуб КПІ "Глобус"*. URL: https://www.tkg.org.ua/node/18617 (дата звернення: 13.10.2025).

2. Про туризм: Закон України від 15.09.1995 № 324/95-ВР// Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/324/95-%D0%B2%D1%80#Text (дата звернення: 14.10.2024).

3. Положення про туристські маршрутно-кваліфікаційні комісії навчальних закладів: Наказ Міністерства освіти і науки України №1124 від 14.10.2014. URL: https://zakon.rada.gov.ua/laws/show/z1342-14#Text Text (дата звернення: 14.10.2024).

4. В Похід Карпатами: вебсайт. URL: https://vpohid.com.ua/ (дата звернення: 16.10.2025).

5. Komoot: вебсайт. URL: https://www.komoot.com/ (дата звернення: 17.10.2025).

6. RouteXL: веб-сайт. URL: https://www.routexl.com/ (дата звернення: 18.10.2025).

7. Zarembo I., Kodors S. Pathfinding Algorithm Efficiency Analysis in 2D Grid. *Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference*. 2015. Vol. 2. P. 46. URL: https://doi.org/10.17770/etr2013vol2.868 (дата звернення: 21.10.2025).

8. Rahayuda I. G. S., Santiari N. P. L. Dijkstra and Bidirectional Dijkstra on Determining Evacuation Routes. *Journal of Physics: Conference Series*. 2021. Vol. 1803, no. 1. P. 012018. URL: https://doi.org/10.1088/1742-6596/1803/1/012018 (дата звернення: 22.10.2025).

9. Asymmetric orienteering problem with profitable penalty / D. Mocková та ін. *Neural Network World*. 2024. Т. 34, № 3. С. 169–188. URL: https://doi.org/10.14311/nnw.2024.34.009 (дата звернення: 25.10.2025).

10. Karbowska-Chilinska J., Zabielski P. A Genetic Algorithm vs. Local

Search Methods for Solving the Orienteering Problem in Large Networks. *Lecture Notes In Computer Science*. 2012. No. 7828.

11. Liang Y.-C., Smith A. E. An ant colony approach to the orienteering problem. *Journal of the Chinese Institute of Industrial Engineers*. 2006. Vol. 23, no. 5. P. 403–414. URL: https://doi.org/10.1080/10170660609509336 (дата звернення: 01.11.2025).

12. Lin S.-W., Guo S.-R., Wu W.-J. Applying the Simulated Annealing Algorithm to the Set Orienteering Problem with Mandatory Visits. Mathematics. 2024. Vol. 12, no. 19. P. 3089. URL: https://doi.org/10.3390/math12193089 (дата звернення: 02.11.2025).

13. BPWin Software. *BPM Microsystems*. URL: https://www.bpmmicro.com/support/software/ (дата звернення: 06.11.2025).

14. MVC glossary. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата звернення: 12.11.2025).

15. A Performance Comparison of Shortest Path Algorithms in Directed Graphs / F. Sapundzhi та ін. *International Scientific Conference TechSys* 2025. Basel Switzerland, 2025. С. 31. URL: https://doi.org/10.3390/engproc2025100031 (дата звернення: 16.11.2025).

16. Simulated Annealing - Algorithms for Competitive Programming. URL: https://cp-algorithms.com/num_methods/simulated_annealing.html (дата звернення: 17.11.2025).

17. IBM. What Is Three-Tier Architecture. *IBM*. URL: https://www.ibm.com/think/topics/three-tier-architecture (дата звернення: 20.11.2025).

18. Welcome to Flask. *Flask Documentation*. URL: https://flask.palletsprojects.com/en/stable/ (дата звернення: 21.11.2025).

19. Порівняння СУБД My SQL, PostgreSQL та MS SQL Server. DataBI. URL:https://data-b-i.com/uk/article/porivnyannya-subd-mysql-postgresql-mssqlserver.html (дата звернення: 23.11.2025).

20. Bootstrap - Introduction. *Bootstrap*. URL: https://getbootstrap.com/docs

/5.0/getting-started/introduction/ (дата звернення: 23.11.2025).

21. MyISAM vs InnoDB: differences & performance comparison. *Devart Blog*. URL: https://blog.devart.com/myisam-vs-innodb.html (дата звернення: 24.11.2025).

22. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань, книга 2: Системи управління базами даних та знань. Навчальний посібник (рек. МОН України), 2021, 584 с.

23. NetworkX documentation. NetworkX. URL: https://networkx.org/en/ (дата звернення: 25.11.2025).

24. OpenStreetMap Україна: веб-сайт. URL: https://openstreetmap.org.ua/ (дата звернення: 27.11.2025).

25. Leaflet – an open-source JavaScript library for interactive maps: вебсайт URL: https://leafletjs.com/ (дата звернення: 27.11.2025).