МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

кафедра «Інформаційні системи та мережі»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи на тему:

Інформаційна система «Місто у смартфоні»

Студента групи **ІТ-42 Бакум Павло Ігорович**

(шифр, прізвище та ініціали)

Керівник роботи _____ (Петро КРАВЕЦЬ)

Консультант _____ (_____)

_____ (_____)

Нормоконтроль _____ (Андрій ВАСИЛЮК)

Завідувач кафедри ІСМ _____(Василь ЛИТВИН)

«_09_» _червня_ **2023 р.**

ЛЬВІВ – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій
Кафедра «Інформаційні системи та мережі»
Спеціальність 126 "Інформаційні системи та технології"
Перший (бакалаврський) рівень вищої освіти
ОПП "Інтелектуальні інформаційні системи"

«ЗАТВЕРДЖУЮ»
Завідувач кафедри ІСМ _____
«_09_» __червня___ 2023 р.

ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу студента групи ІТ-42

_____Бакум Павло Ігорович_____

(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система «Місто у смартфоні»

_____

затверджена наказом по НУ «ЛП» від «_15_»__березня___2023р. №_778-4-08_____

2. Термін здачі студентом закінченої роботи___29.05.2023р._____

3. Вихідні дані для роботи: ресурси глобальної мережі Інтернет, літературні джерела.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): провести аналіз інформаційних систем та програмного забезпечення для побудови туристичних маршрутів на карті міста, побудувати концептуальну модель інформаційної системи побудови туристичних маршрутів, здійснити вибір методів та засобів для відображення на карті оптимальних маршрутів, реалізувати інформаційну систему.

5. Перелік графічного матеріалу: дерево цілей, діаграми IDEF0, схеми бази даних, скріншоти роботи програми.

6. Перелік програмних продуктів, які належить використати в процесі розроблення роботи (проекту): AllFusion Process Modeler, Visual Studio Code, Google Chrome/Opera, MS sql.

7. Консультування роботи, із зазначенням розділів роботи

Розділ	Консультанти	Підпис, дата	
		завдання видав	завдання отримав

8. Дата, коли видано завдання _____27.02.2023 р._____

Керівник _____
(підпис)

Завдання отримав до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Етапи кваліфікаційної роботи	Термін виконання етапів роботи	Примітки
1	Аналітичний огляд літературних та інших джерел	06.03.2023 – 18.03.2023	
2	Системний аналіз об’єкта дослідження	20.03.2023 – 07.04.2023	
3	Програмні засоби розв’язання задач	07.04.2023 – 15.04.2023	
4	Практична реалізація	15.04.2023 – 03.05.2023	
5	Оформлення бакалаврської кваліфікаційної роботи	04.05.2023 – 08.05.2023	

Студент _____
(підпис)

Керівник роботи _____
(підпис

ЗМІСТ

ВСТУП 5
РОЗДІЛ 1. Аналіз літературних джерел та предметної галузі 7
1.1. Аналіз міських мобільних програм 7
1.2. Аналіз методів пошуку оптимальних маршрутів 12
1.3. Аналіз відомих засобів вирішення проблеми 18
Висновок до 1-го розділу 22
РОЗДІЛ 2. Системний аналіз об’єкта дослідження 23
2.1. Дерево цілей 23
2.2. Конкретизація функціонування системи 26
2.3. Побудова ієрархії процесів 31
Висновок до 2-го розділу 32
РОЗДІЛ 3. Програмі засоби розв’язання задачі 33
3.1. Вибір та обґрунтування засобів розв’язання задачі 33
3.2. Технічні характеристики обраних програмних засобів 43
Висновок до 3-го розділу 45
РОЗДІЛ 4. Практична реалізація 46
4.1. Описання створеного програмного засобу 46
4.2. Інструкція користувача 58
4.3. Аналіз контрольного прикладу 59
Висновок до 4-го розділу 61
ВИСНОВКИ 62
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 63
АНОТАЦІЯ 66
ДОДАТКИ 70

ВСТУП

Використання міських мобільних програм – звичайна практика для жителів західноєвропейських країн, а також таких розвинених східних держав, як Японія та Південна Корея. Там люди встановлюють мобільні програми десятками – і це при тому, що за користування кожною доводиться платити. У великих містах активно впроваджують мобільний софт, заточений під міські реалії. Це стосується й міст України.

Основною проблемою на вирішення якої зорієнтовані такі програми є побудова оптимальних маршрутів, які би включали всі необхідні проміжні точки. Це дозволило, наприклад туристам зекономити час, та дало змогу оглянути більшу кількість визначних місць. Головною проблемою є побудова такого маршруту, який би включав всі точки лише один раз, за найкоротший шлях.

Ця задача відома з давніх часів, протягом років мінялись лише сфери її застосування. В період коли туризм не був розвинений вона була актуальною лише для логістичних та транспортних цілей. Та з підвищенням рівнем життя вона стала актуальною і у інших сферах, наприклад туристичній.

Актуальність задачі

Система побудови оптимальних туристичних маршрутів у смартфоні дала б змогу туристами відвідати більше місць. Також така система може бути використана не тільки для Львова а й для будь-яких інших міст.

Мета і задачі дослідження

Мета дослідження є розроблення інформаційної системи «Місто у смартфоні», яка б з набору точок на карті будувала оптимальний шлях.

Об’єкт дослідження – процес побудови інформаційної системи «Місто у смартфоні».

Предмет дослідження – методи та засоби побудови інформаційної системи «Місто у смартфоні» для розроблення оптимальних туристичних маршрутів.

Теоретичне дослідження одержаних результатів полягає в:

- Аналізі існуючих алгоритмів пошуку найкоротших відстаней та побудови шляхів;
- Проектуванні архітектури системи, що буде ефективно виконуватиме поставлені задачі.

Результатом дослідження буде програма яка буде будувати та відображати на карті оптимальний маршрут на основі, заданих користувачем, параметрів.

Практичне значення одержаних результатів полягає в розробці Android додатку, використання якого зможу збільшити туристичну привабливість міста.

РОЗДІЛ 1

Аналіз літературних джерел та предметної галузі

1.1. Аналіз міських мобільних програм

Використання міських мобільних програм – звичайна практика для жителів західноєвропейських країн, а також таких розвинених східних держав, як Японія та Південна Корея. Там люди встановлюють мобільні програми десятками – і це при тому, що за користування кожною доводиться платити. У великих містах активно впроваджують мобільний софт, заточений під міські реалії. Що стосується міст України, то для Києва міські мобільні додатки охоплюють далеко не всі можливі сфери життя, але їх вже є досить велика кількість. До того ж, майже всі вони безкоштовні. Проаналізуємо їх.

Для скаржників та заявників

Київська міська держадміністрація розробила свою програму «КМДА 1551» (КМДА 1551) для скарг та заяв до Контактного центру. Вона працює на базі операційних систем iOS , Android , Windows 8 та Windows Phone.

Оминаючи пробки

Це, мабуть, найпопулярніша сфера застосування міських програм. Дістатись з одного місця в інше в найкоротші терміни, минаючи «проблемні» ділянки – це актуальне завдання як для водія, так і для пішохода/пасажира.

«ВесьКиїв» від gorspravka.blogspot.com створено під iOS і містить різну корисну інформацію – про держустанови, культурний відпочинок та розваги, телефони таксі та довідкові, місця відпочинку, ресторани, бари, аптеки, банкомати та ін. По суті, це великий безкоштовний довідник по Києву.

Міжнародний додаток і Транспорт під iOS 3.1 може бути корисним при виборі схеми пересування містом, причому різними видами транспорту. Ця програма містить базу зупинок та маршрутів. Адресу можна вводити у вільній формі. В результаті вам запропонують кілька варіантів проїзду, з яких ви самі оберете

найприйнятніший. Програму можна, можливо завантажити на численних інтернет-ресурсах.

Додаток Metroshka для iPhone призначено для пасажирів метро З її допомогою можна дізнатися, на якій відстані від вас (не більше від 100 м до 10 км) знаходиться найближча станція метро і як до неї пройти. Завантажити можна на сайті компанії-розробника.

Для водіїв, які не хочуть стояти в пробках, доступний додаток, що є полегшеною мобільною версією сайту www . videoprobki . ua , що транслює відео з вуличних камер. Ви отримуєте перелік камер, з яких вибираєте потрібну. Знайти її можна як за номером, так і за адресою розташування, що можна побачити на карті. Також доступні нові новини про дорожні пригоди. Скористатися програмою можна, зайшовши зі смартфона на головну сторінку сайту, або відразу в розділ mobile.videoprobki.ua.

На сьогоднішній день у Києві розроблено програму, яка дозволяє визначати місце розташування транспорту і, таким чином, даремно не мерзнути, чекаючи на його зупинку. Весь мобільний софт і веб-сайт програми вже готові, але ще не запущені, т.к. залишилося отримати потрібну інформацію про маршрути від "Пастранса".

Ось наша добірка софту, що дозволяє зручно замовляти таксі в Києві та деяких інших містах.

Uklon під iOS / Android працює в Києві, Харкові, Дніпропетровську та Одесі та охоплює понад 100 служб таксі. Ось адреса полегшеної версії сайту для смартфонів uklon . com . ua / m . Можна задати різні параметри замовлення: крім часу і місця – тип кузова, тип вантажу, що перевозиться, та ін. Зареєстрованим користувачам пропонується знижка 5%. Після прийняття замовлення вам надходить SMS- ка з докладним описом авто, яке вас підвозитиме. Завантажити мобільну версію можна тут або тут.

IQTaxi під iOS / Android . Працює у Києві та Одесі. Серед переваг служби – можливість відстежувати переміщення авто містом, завдяки встановленим на них

маячкам. Крім того, програма визначає ваше поточне місцезнаходження, і навіть якщо ви не знаєте точної адреси, за якою знаходитесь, служба сама вас знайде. Вартість послуги розраховується автоматично. Можна задавати різні опції перевезення - кількість пасажирів, їх особливості (діти, тварини та ін.).

Taxioma під iOS / Android . Працює у Києві, Харкові та Одесі. Дозволяє зробити замовлення на найближчий або вказаний час. Можна задавати особливі умови перевезення (кондиціонер, водій, що не курить, тип кузова тощо). Програма зберігає вашу історію маршрутів. На рис. 1.1. наведено приклад інтерфейсу програми «Замовлення таксі».

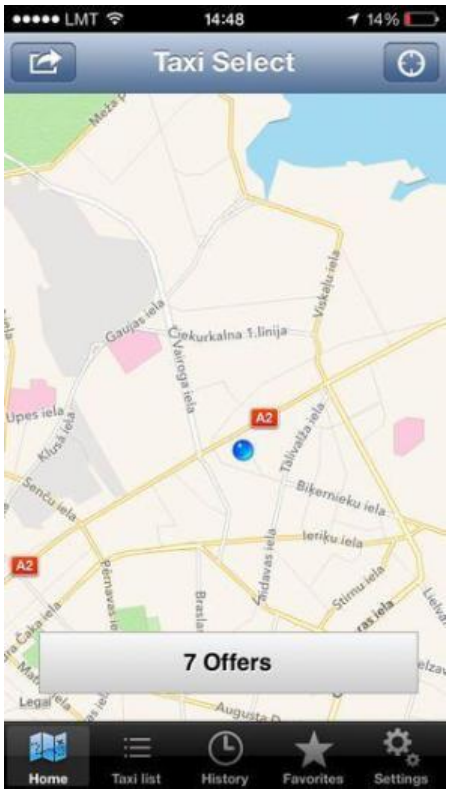


Рис. 1.1. Приклад інтерфейсу програми «Замовлення таксі»

Taxi Selet під iOS/Android. Міжнародна служба замовлення таксі, за допомогою якої можна вивчити пропозицію різних служб та вибрати відповідну послугу.

Сам собі детектив

Щоб даремно не хвилюватися за рідних та близьких, можна скористатися однією з мобільних програм, що відстежують місцезнаходження людини. У Києві доступні такі програми.

"Маячок" від Київстар . По суті це послуга для будь-яких мобільних пристроїв. Платна послуга вартістю 53 коп. на добу. Можна відстежувати становище людини на карті або отримувати SMS та MMS з потрібною інформацією, проте для цього потрібна згода того, ким ви цікавитеся. Активувати послугу можна, набравши 147*2*1*номер абонента через 380#. Вартість успішного запиту складає додатково 2 грн. До списку адресатів «Маячка» можна занести не більше десяти абонентів.

"Дитина під наглядом " від МТС. Крім звичайного мобільного сервісу, доступного через SMS , відправлене на номер 7005, можна встановити спеціальний додаток для Android , що дозволяє бачити розташування людини на карті. Вартість запитів, залежно від кількості, коливається від 0,5 до 2 грн. за один.

Sygic Family під iOS/Android. Версія для iPhone платна і коштує $4,99. Дозволяє відстежувати на карті розташування потрібних людей і навіть бачити рівень заряду батареї на їх смартфонах. Має також функцію надсилання безкоштовних SMS через Інтернет.

Android Device Manager під однойменну операційну систему дає можливість відстежувати розташування інших гаджетів, що працюють на Android, а також дозволяє вашій довіреній особі визначати ваше місцезнаходження за допомогою облікового запису в Google. Аналогічний сервіс існує і для власників " еппловських " гаджетів - це Find my iPhone. З його допомогою можна побачити на карті розташування всіх пристроїв на базі iOS, а також прокласти оптимальний маршрут до них.

І в ресторан, і в туалет

Замовлення піци та іншої їжі по телефону давно вже не новина. Для тих, хто не любить розмовляти, існують сервіси, що дозволяють все пояснити «на пальцях» – з їх допомогою ви вводите всю необхідну інформацію.

Ресурс restorania.com, який пропонує завантажити та встановити мобільні програми GidMenu для iPhone і для гаджетів, що працюють з операційною системою Android . Ця програма дозволяє робити замовлення не в одному, а в багатьох ресторанах, які співпрацюють із ресурсом. Ви бачите на екрані фото та опис страв та маєте можливість не тільки замовити доставку страв у зручне для вас місце, але й зарезервувати стіл в одному із закладів. Ви задаєте цікаві для вас параметри закладу, а потім із запропонованого програмою списку вибираєте те, що вам більше до вподоби - і робите замовлення. Інтерфейс англійська.

Ресурс ekipazh-service.com. ua аналогічний – він також приймає мобільні замовлення на доставку їжі. Але з його допомогою вам можуть бути доставлені інші товари, на кшталт ліків, квітів та ін. Має дві безкоштовні мобільні версії – для iOS і Android, з російським інтерфейсом. Але, судячи з відгуків споживачів, програма явно недопрацьована – в обох випадках вона не відображає нічого, окрім назв закладів.

Ресурс ukushuka . ua пропонує безкоштовний додаток для iPhone , iPad та iPod touch під операційну систему iOS версії 4.3 і вище, що дозволяє замовляти суші. Замовлення виконуються по Києву та області. Вам приходять PUSH -повідомлення про знижки та конкурси. На замовлення з iPhone робиться 5% знижка. Інтерфейс програми – англійська.

У Києві також є чудовий мобільний сервіс, що дозволяє знайти найближчий громадський туалет. Називається він WC – Київ і працює під iOS . Посилання на мобільну версію знаходиться на сайті проекту wc - kiev.org . Цікаво, що вказується міра комфортності туалетів. Причому нову інформацію можуть додавати й користувачі.

Поселитися, подивитися, закупитися

Забронювати номер у готелі Києва чи іншого міста України можна, встановивши Booking App під iOS, Android або Samsung Apps. Тут є детальні описи та фото номерів, можливість сортування пропозиції за ціною та іншими

параметрами, а також віртуальний гід. Під час замовлення зі смартфона обіцяють знижки до 20%.

Для туристів, які приїхали подивитися на краси Києва, існує мобільний додаток «Київ 100 чудес» під iOS . Тут список усіх визначних пам'яток Києва з коментарями історика Олександра Анісімова та чудовими ілюстраціями – сучасними та старими фото. GPS -навігація дозволить знайти будь-який об'єкт.

Ну, а якщо ви шопоголік, тоді встановіть додаток «Мегазнижки» під iOS / Android і отримуйте актуальну інформацію про знижки, акції та інші шопінгові радощі. Додаток робить інформаційну вичавку з таких сервісів, як Biglion («Бігліон»), Grupon (« Групон »), SuperDeal (« Супердіал »), Pokupon (« Покупон »), KupiKupon (« КупіКупон ») та ін.

1.2. Аналіз методів пошуку оптимальних маршрутів

Пошук шляху – термін в інформатиці інтелекті, який означає визначення найкращого, за певним критерієм маршруту між двома точками.

Для пошуку найкоротшого шляху серед декількох точок використовується теорія графів.

Теорія графів — розділ математики, що вивчає властивості графів. Наочно граф можна уявити як геометричну конфігурацію, яка складається з точок (вершини) сполучених лініями (ребрами). У строгому визначенні графом називається така пара множин G = (V, E), де V є підмножина будь-якої зліченної множини, а E — підмножина V × V [1]. Приклад графу наведено на рис. 1.2.

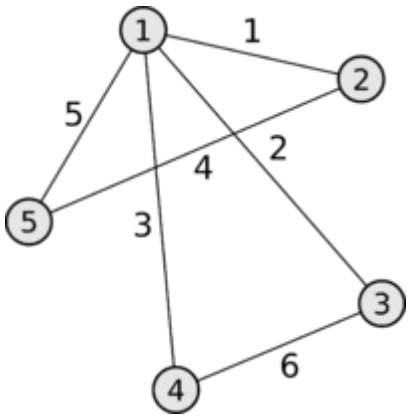


Рис. 1.2. Приклад графа

Родоночальником теорії графів вважається Ленард Ейлер, він запропонував рішення завдання про сім Кенінсберзьких мостів, що зараз є однією з класичних задач теорії графів.

Графи можуть бути використані для моделювання багатьох типів відносин і процесів фізичних, біологічних, соціальних та інформаційних систем. Багато практичних проблем можуть бути представлені графами [2-3].

У інформатиці, графи використовуються для представлення мереж зв'язку, організації даних, потоків обчислень і т.д. Так, наприклад, структура посилання сайту може бути представлена у вигляді орієнтованого графа, в якому вершини представляють собою веб-сторінки і спрямуванням ребра є посилання з однієї сторінки на іншу. Методи дослідження на основі графів мають важливе застосування у лінгвістиці, оскільки мова є дискретною структурою. Теорія графів також використовується для вивчення молекул в хімії та фізиці.

Визначення графу є дуже загальним, тому цим терміном можна описати безліч подій та об’єктів. Завдяки високому рівню абстракції та узагальненню типові алгоритми теорії графів можна використовувати для зовнішньо несхожих задач у транспортних та комп’ютерних мережах, будівельному проектувані, молекулярному проектуванні та інших [4-6].

Інші означення графа випливають з різних концепцій. У ще одному узагальненому понятті V є множиною відношення інцидентності, яка зіставляє кожному ребру дві вершини. Інцидентність – це поняття, яке використовується для ребра, або дуги і вершини. Якщо ребро з’єднює дві вершини, то кожна з них є інцидентною до ребра. В іншому понятті E це мультимножина невпорядкованих пар вершин. V і E вважають скінченними, і багато з результатів є невірними для для нескінченних графів оскільки багато умов не справджуються в нескінченному випадку.

Графи поділяються на два види:

- Орієнтовані;

- Неорієнтовані.

Орієнтований граф (орграф) – граф, ребра якого мають певний напрямок. Орієнтовані ребра називаються дугами. Маршрутом в такому графі називають послідовність, яка складається з вершин та ребер, які чергуються. Орграф називається зв’язним, якщо всі його вершини взаємодосяжні. Приклад орієнтованого графа наведено на рис. 1.3.

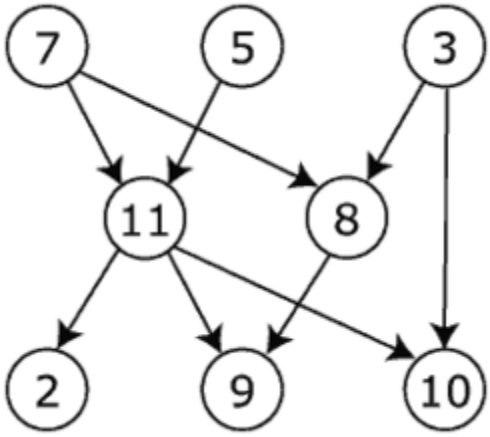


Рис. 1.3. Приклад орієнтованого графа

Граф у якому немає циклів, та існує шлях до будь якої вершини називається деревом. Приклад дерева наведено на рис. 1.4.

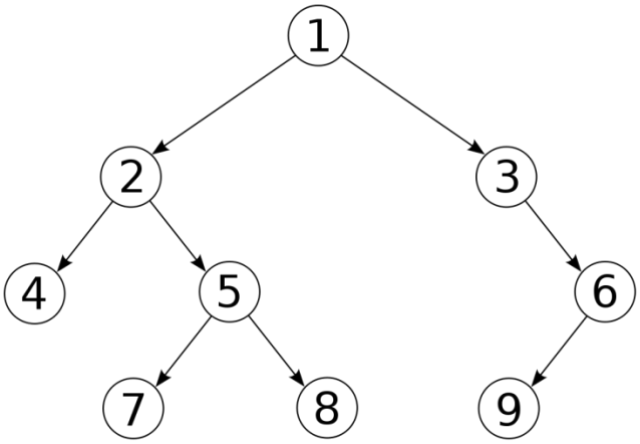


Рис. 1.4. Дерево

Також граф має ряд інших характеристик:

- зв’язний, якщо якщо існує шлях між будь-якими двома вершинами;
- повний, якщо будь-які дві вершини з’єднані ребрами;

- дводольний, якщо множину вершин можна розбити на дві підмножини, які не перетинаються, так що кожне ребро містить одну вершину з першої і одну вершину з другої підмножини.

Окрім графічного існують й інші способи представлення графів, а саме:

1. Матриця інцидентності – матриця у якої стовпці відповідають ребрам а рядки вершинам. Ненульове значення у клітинці матриці вказує на зв'язок між вершиною і ребром. У випадку орграфа використовують значення -1 для дуги яка виходить, та відповідно 1 для дуги яка входить у вершину. Приклад матриці інцидентності наведено на рис. 1.5.

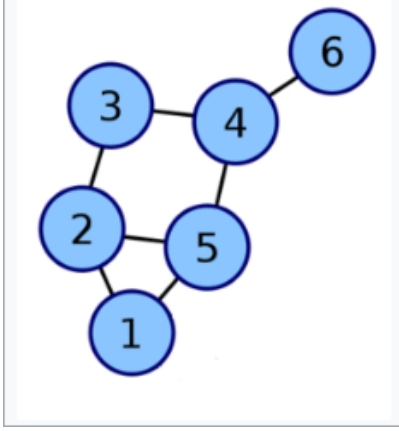
Неорієнтований граф	Матриця інцидентності
	$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

Рис. 1.5. Приклад матриці інцидентності

2. Матриця суміжності – це квадратна матриця розміру *n*, де n це кількість в якій значення елементу a_{ij} рівне числу ребер з *i*-ї вершини графа в *j*-у вершину. Іноді, у різі неорієнтованого графа петля рахується за два ребра. Приклад матриці суміжності наведено на рис. 1.6.

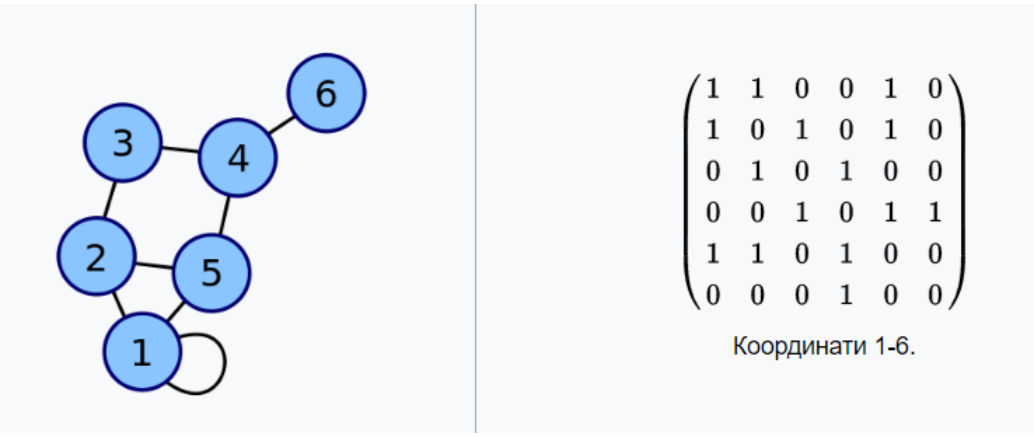


Рис. 1.6. Матриця суміжності

3. Список суміжності – список у якому кожній вершині графа відповідає рядок, у якому зберігається список суміжних вершин, така структура данних являється «списком списків».
4. Список ребер – список у якому кожній вершині графа відповідає рядок, у якому міститься дві інцидентні до цього ребра вершини.

У теорії графів існує задача про найкоротший шлях, яка полягає у знаходженні шляху, в якому сума ваг ребер буде найменшою.

Існує декілька варіацій задачі:

- Задача про найкоротші шляхи з одного входу, необхідно знайти найкоротші шляхи між вхідною вершиною та рештою.
- Задача про найкоротші шляхи з одним виходом, необхідно знайти найкоротші шляхи з усіх вершин до однієї вихідної.
- Задача про найкоротші шляхи для всіх пар, необхідно знайти найкоротші шляхи між кожної парою вершин у графі.

Існує декілька найбільш використовуваних алгоритмів для вирішення задач такого виду.

Алгоритм Дейкстри знаходить найкоротші шляхи від заданої вершини до всіх інших. Його недоліками є те що він не гарантує відвідування усіх вершин, а шукає строго шлях від точки А до точки Б; не допускаються з відємними вагами; працює лише з неорієнтованими графами.

Алгоритм Беллмана-Форда, як і алгоритм дейкстри шукає найкоротший шлях від заданої точки до всіх інших, та його перевагою є те що допускаються ребра з відємною вагою та можливість роботи з орієнтованим графом.

Алгоритм A^* – це евристичний алгоритм пошуку, використовується для пошуку найкоротшого шляху між двома вершинами неорієнтованого графу з додатніми вагами ребер. Алгоритм A^* знаходить оптимальний шлях між двома вершинами, оптимальність може означати найкоротший, найшвидший, чи навіть найпростіший шлях. Теоретично алгоритм дає змогу розв’язувати всі задачі, як можливо

представити у вигляді задачі пошуку оптимального шляху між двома вершинами у графі.

Основним недоліком даного алгоритму є необхідність зберігання усіх відомих та досліджених вершин, через це він придатний не для всіх задач.

Алгоритм Флойда-Воршелла у зваженому графі з додатніми або відємними вагами ребер знаходить найкоротший шлях між всіма парами вершин, але не видає інформації про самі шляхи. Алгоритм добре працює для щільних графів, в яких більшість вершин або всі пари вершин з'єднані ребрами. Для розріджених алгоритмів краще використовувати алгоритм Дейкстри.

Алгоритм Джонсона дозволяє знайти найкоротші шляхи між усіма парами вершин зваженого орієнтованого графа, він працює для графів у яких містяться негативні ваги дуг [2].

Також до задачі пошуку найкоротшого шляху можна віднести *задачу комівояжера*. Вона полягає у знаходженні найкоротшого маршруту через задані точки, з поверненням у початкову [3].

Незважаючи на простоту визначення та відносну нескладність знаходження хороших рішень завдання комівояжера відрізняється тим, що пошук дійсно оптимального шляху є досить складним завданням. З огляду на ці особливості, починаючи з другої половини XX століття дослідження задачі комівояжера має не стільки практичний сенс, скільки теоретичний, вона використовується в якості моделі для розробки нових алгоритмів оптимізації.

Вирішенням задачі комівояжера є гамільтонів шлях.

Гамільтонів шлях – це шлях що містить кожну вершину графа рівно один раз, гамільтонів шлях у якого збігаються початкова та кінцева точки називається гамільтоновим циклом.

Існує декілька простих алгоритмів для розв'язання цієї задачі: повний лексичний перебір, жадібні алгоритми (метод найближчого сусіда), метод включення найближчого міста, метод найдешевшого включення, метод мінімального кістяка дерева. На практиці ж застосовують модифікації більш

ефективних методів: метод гілок та меж, метод генетичних алгоритмів, алгоритм мурашиної колонії.

Для можливості застосування математичних методів задачу комівояжера представляють у вигляді графа. Для спрощення задачі та гарантії існування маршруту, вважається що граф є повністю зв’язним, тобто між довільною парою вершин існує ребро. Також задача має має три варіації симетрична, асиметрична та метрична. Асиметрична від симетричної відрізняється тим, що дуги можуть мути різну вагу залежно від напрямку, в такому випадку задача моделюється орієнтованим графом. Метрична задача комівояжера – це симетрична, для якої виконується нерівність трикутника. В таких задачах обхідні шляхи довші за прямі, тобто ребро від вершини *i* до вершини *j* не буде довшим за шлях через проміжні вершини.

Задача комівояжера може мати два варіанти: замкнений та незамкнений. У замкнутому варіанті завдання необхідно пройти через всі вершини графа, після чого повернутися у початкову вершину. Незамкнений варіант відрізняється від замкнутого тим, що в ньому непотрібно повертатися до стартової вершини.

Незамкнений варіант завдання зводиться до замкнутого плутаючи заміні ваг дуг, що входять у стартову вершину, на 0. Оптимальний замкнутий маршрут комівояжера в такому графі відповідає оптимальному незамкнутому маршруту в вихідному графі.

Відомі методи вирішення поділяють на дві групи, що можна поєднувати. Точні методи знаходять, маючи достатньо часу, гарантовано оптимальний шлях. Евристичні методи знаходять, зазвичай за коротший час, гарні розв'язки, що, в загальному випадку, можуть бути гіршими за оптимальні. Для метричної задачі існують евристики, що знаходять за поліноміальний час розв'язки гірші за оптимальні у 1.5—2 рази.

Евристичний алгоритм або евристика – це алгоритм, який видає прийнятний розвязок серед множини розвязків, але не може гарантувати що він буде найкращим.

Дана задача зазвичай використовується при плануванні кур'єрських доставок, прибирання великих територій, планування туристичних чи інших маршрутів.

1.3. Аналіз відомих засобів вирішення проблеми

На сьогоднішній день існує чимало додатків для планування звичайних та туристичних маршрутів. Це пов'язано зі стрімким розвитком геолокаційних технологій. Чи не кожен картографічний сервіс надає можливість прокладати маршрути та навігуватись між точками. Для покращення точності також використовуються дані про інші підключення смартфону: Wi-Fi, стільниковий зв'язкок [10-12].

Розглянемо детальніше застосунки які надають можливість навігації:

Google Maps – це набір додатків, які надають безкоштовний доступ до картографічних сервісів, та тісно пов'язані з іншими сервісами Google. Сервіс являє собою карту всього світу, туди також інтегрований бізнес-довідник, а також можливість пошуку маршрутів для автомобілів, пішоходів, а також громадським транспортом. Завдяки інтеграції з іншими сервісами Google Maps можна використовувати і як туристичний довідник.

Особливість даного сервісу є Google Streets View – це сервіс який надає можливість користувачам відчуття присутності, завдяки тривимірній проекції вулиць. Ця функція стала можлива за допомогою кругового фотографування місцевості в режимі реального часу. Також Google Maps надає можливість всім користувачам додавати власні фото до місць, та залишати відгуки.

Однак як туристичний сервіс Google Maps має ряд недоліків, а саме фото які додають користувачі не завжди відповідають дійсності. Також неможливість прокладання усіх маршрутів у режимі офлайн. Та не відсутність побудови складних маршрутів. Інтерфейс Android додатку Google Maps наведено на рис. 1.7.

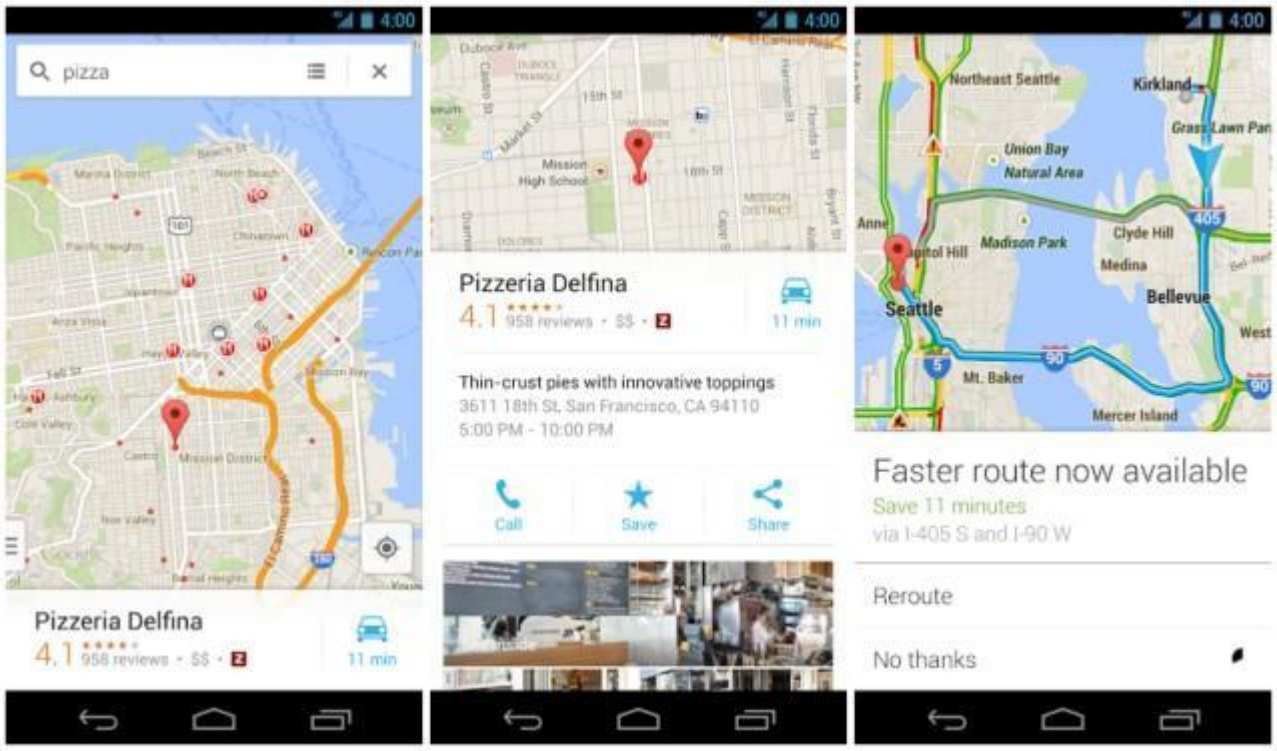


Рис. 1.7. Android додаток Google Maps

TripAdvisor – американським сервіс, який забезпечує огляд туристичного контенту, який додається користувачами сервісу.

TripAdvisor був одним з перших хто почав використовувати користувацький контент. Послуги сайту безкоштовні для користувачів, які забезпечують більшу частину контенту. Сервіс має додатки під усі сучасні мобільні платформи.

Додатки містять інформацію про популярні готелі, ресторани та визначних пам’яток. Надає можливість прокладати маршрути до тих які зацікавлять користувача, а також можливість роботи оффлайн.

Недоліками TripAdvisor є перевантаженість додатку, що створює незручності для користування, а також відсутня можливість прокладати складні маршрути. Також не завжди вчасна модерація, оскільки сервіс має велику користувацьку базу. Інтерфейс TripAdvisor наведено на рис. 1.8.

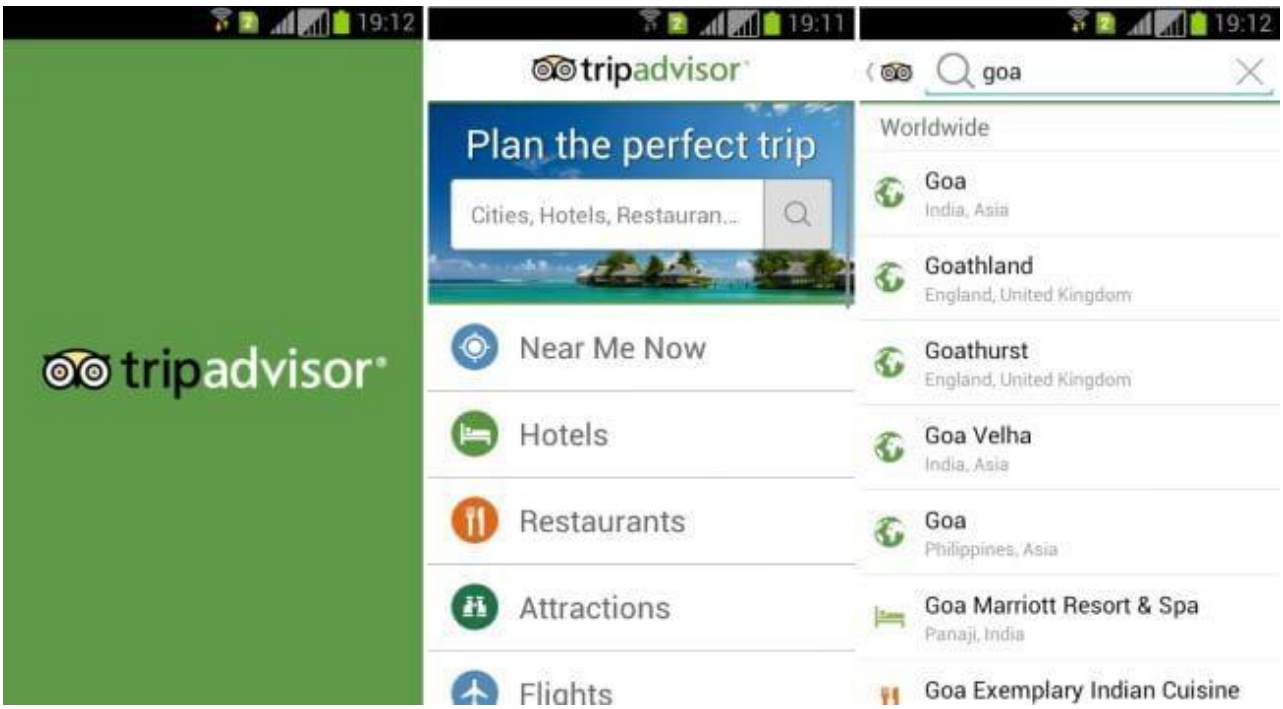


Рис. 1.8. Android додаток TripAdvisor

Maps.me – безкоштовний картографічний сервіс на основі вільної географічної карти OpenStreetMap. Сервіс має додатки для більшості сучасних мобільних платформ. Основною перевагою є велика деталізація карт для пішохідних маршрутів, також є можливість роботи без доступу до інтернету, але без детальної інформації.

Основними недоліками є невелика база популярних місць, а також обмеженість роботи у режимі офлайн, а також відсутність побудови складних маршрутів. Інтерфейс Maps.me наведено на рис. 1.9.

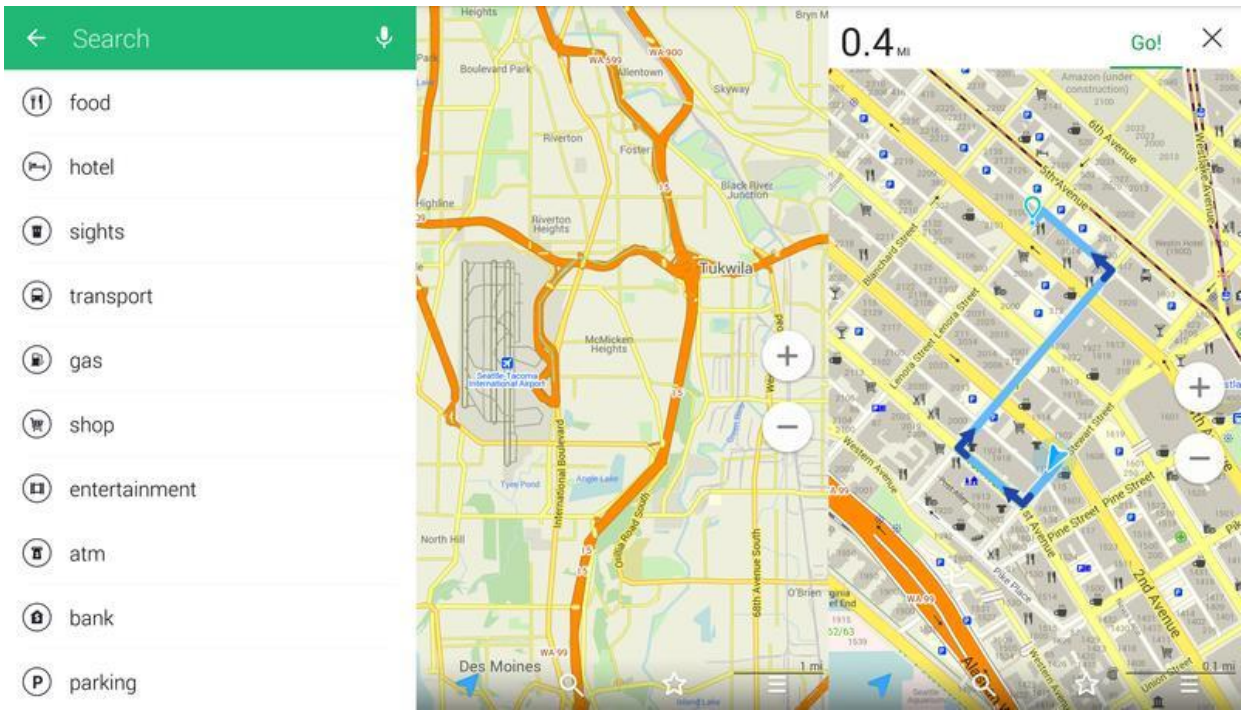


Рис. 1.9. Android додаток Maps.me

Here maps – картографічний сервіс створений компанією Nokia. Доступний у веб-браузері та смартфонах з різними платформами. Here maps налічує 196 країн, та навігацію з голосовими підказками у 96 країнах. Додаток дає інформацію про затори на дорогах в реальному часі, містить 3D-карти.

Основним недоліком даного сервісу є невелика кількість пам’яток культури, та обмежена інформація про них, відсутність інформації про пам’ятки культури в режимі офлайн, також відсутність складних пішохідних маршрутів, що є критично для туриста. Інтерфейс HERE WeGo наведено на рис. 1.10.

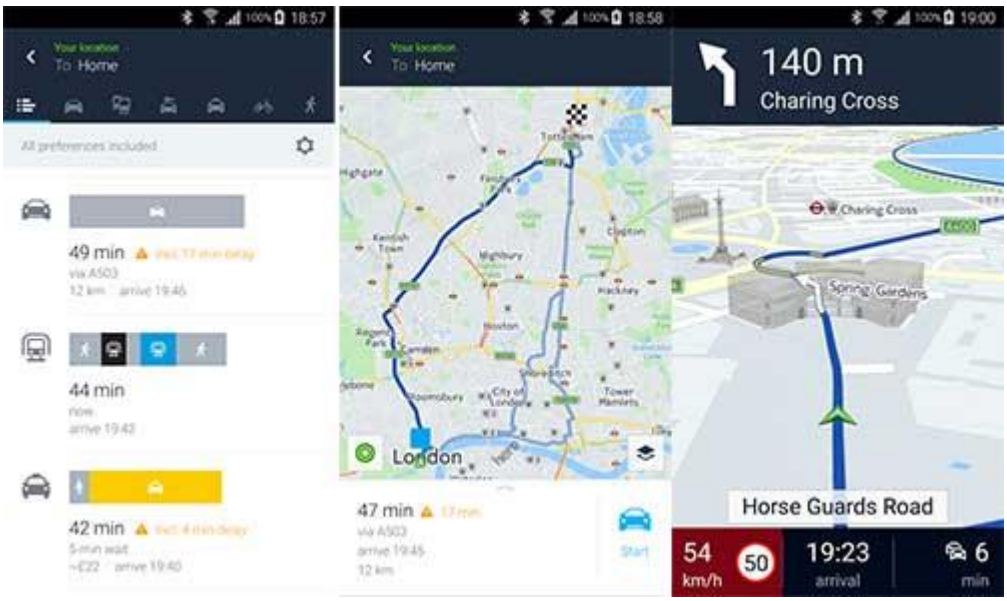


Рис. 1.10. Android додаток HERE WeGo

Висновок до 1-го розділу

Проведений аналіз алгоритмів пошуку оптимального шляху показав, що за довгу історію існування проблеми, було знайдено багато способів її вирішення. Оскільки проблема завжди була актуальною, для її вирішення залучалось немало ресурсів. Найбільший прорив було зроблено з залученням математичного апарату. Завдяки новій галузі математики, дискретній математиці з’явились вирішення не тільки задач пошуку оптимального маршруту і багатьох інших.

Також аналіз існуючих картографічних та туристичних сервісів показав, що всі вони добре справляються з прокладенням відстані між двома точками. Але для побудови складних маршрутів вони не підходять, також бракує функціональності в режимі офлайн.

РОЗДІЛ 2
Системний аналіз об’єкта дослідження

2.1. Дерево цілей

Діаграми дерева цілей допомагає зв'язати завдання габаритних цілей і малих цілей, і допомагають зробити складні завдання візуально більш керованими. Ці діаграми дозволяють зменшити ймовірність помилки в створені системи, оскільки кожна маленька ціль не займає великого обсягу часу [4].

Головним завдання є створення системи побудови маршрутів. Для реалізації цього завдання потрібно дослідити предметну область та розробити програмне забезпечення для системи. При дослідженні предметної області потрібно спроектувати архітектуру модулів системи та побудову архітектури бази даних. Також одним з основних завдань йде вибір алгоритму реалізації для пошуку найоптимальнішого маршруту. При розробці програмного забезпечення потрібно вибрати засоби для створення програмного забезпечення. Досягнення головної цілі можливе лише при виконанні всіх попередніх [5]. Дерево цілей системи наведено на рис 2.1.

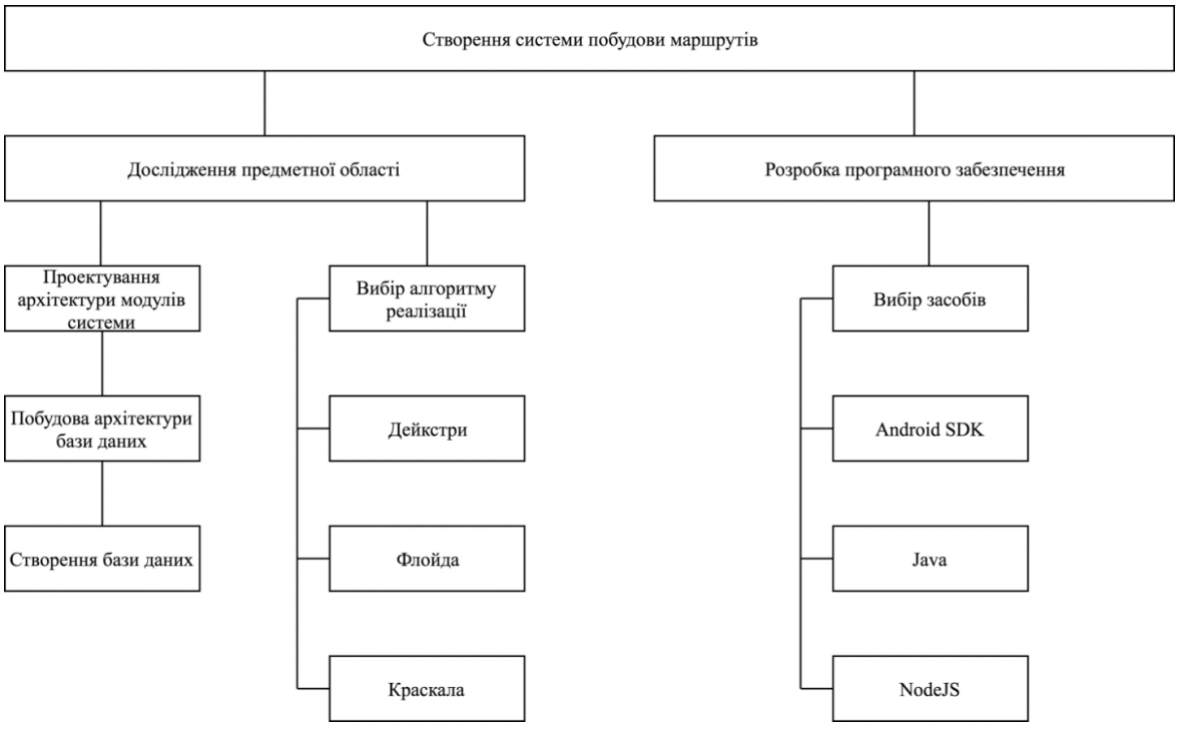


Рис. 2.1. Дерево цілей

Метод аналізу ієрархій використовується для визначення привабливості певних альтернатив при аналізі та проектуванні складних систем. Також метод аналізу ієрархій використовується в багатьох інших сферах. Метод аналізу ієрархій має набір оцінок критерії, а також набір альтернативних варіантів, серед яких краще рішення повинно бути прийнято. MAI генерує вага для кожного критерію оцінки відповідно до попарного порівняння. Важливо відзначити, що, оскільки деякі з критеріїв можуть бути кращі для однієї альтернативи, потрібно сконцентруватися над варіантом, який досягає найбільш підходящий компроміс між різними критеріями [7-9]. Ієрархію MAI наведено на рис. 2.2.

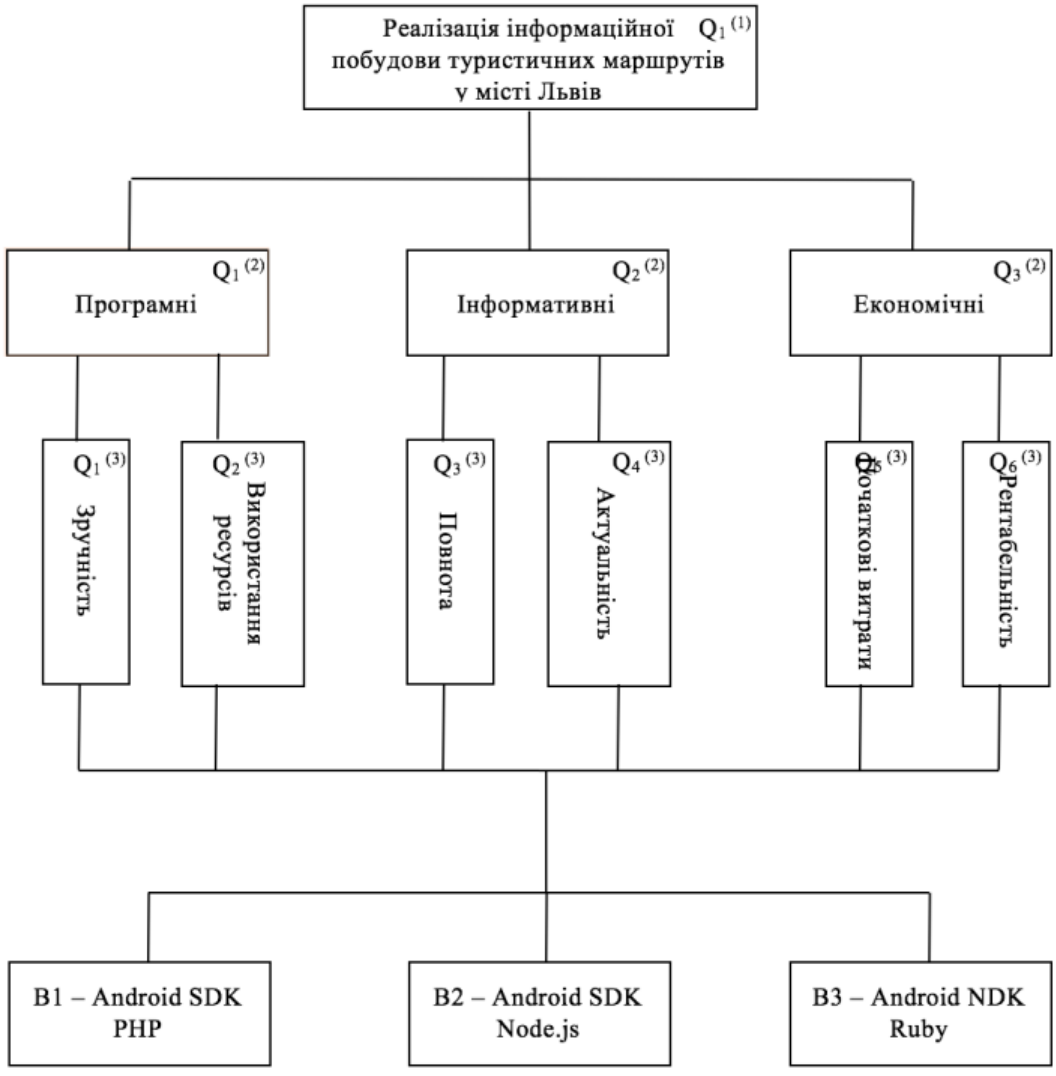


Рис. 2.2. Ієрархія альтернатив для MAI

У результаті аналізу обчислимо найкращі альтернативи:

Таблиця 2.1.

Важливості основних факторів

	$Q_1^{(2)}$	$Q_2^{(2)}$	$Q_3^{(2)}$	$x_1^{(1)}$
$Q_1^{(2)}$	1	7	4	0.451
$Q_2^{(2)}$	1/7	1	¼	0.512
$Q_3^{(2)}$	1/4	4	1	0.402
λ_{max}= 2.030; l_0= 0.10				

Далі обчислюємо критерії відносно факторів.

Таблиця 2.2.

Важливості факторів відповідно до критеріїв

Фактор $Q_1^{(2)}$				***Фактор*** $Q_2^{(2)}$				***Фактор*** $Q_3^{(2)}$			
	$Q_1^{(3)}$	$Q_2^{(3)}$			$Q_4^{(3)}$	$Q_5^{(3)}$	$x_2^{(2)}$		$Q_6^{(3)}$	$Q_7^{(3)}$	$x_3^{(2)}$
$Q_1^{(3)}$	1	1/5	0.412	$Q_4^{(3)}$	1	7	0.712	$Q_6^{(3)}$	1	5	0.641
$Q_2^{(3)}$	1/5	1	0.214	$Q_5^{(3)}$	1/7	1	0.387	$Q_7^{(3)}$	1/5	1	0.152
λ_{max}= 2. 123; l_0= 0.083				λ_{max}= 2. 423; l_0= 0.081				λ_{max}= 2. 241; l_0= 0.082			

Таблиця 2.3.

Матриці парних порівнянь відповідно до критеріїв

Критерій $Q_1^{(3)}$					***Критерій*** $Q_2^{(3)}$					***Критерій*** $Q_3^{(3)}$				
	B_1	B_2	B_3	$x_1^{(3)}$		B_1	B_2	B_3	$x_2^{(3)}$		B_1	B_2	B_3	$x_3^{(3)}$
B_1	1	3	6	0.281	B_1	1	3	5	0.241	B_1	1	3	4	0.321
B_2	1/3	1	4	0.491	B_2	1/3	1	4	0.415	B_2	1/3	1	2	0.52
B_3	1/6	1/4	1	0.345	B_3	1/5	1/4	1	0.291	B_3	1/4	1/2	1	0.15
λ_{max}= 2.0; l_0= 0.0					λ_{max}= 2.01; l_0= 0.009					λ_{max}= 2.02; l_0= 0.02				
Критерій $Q_4^{(3)}$					***Критерій*** $Q_5^{(3)}$					***Критерій*** $Q_6^{(3)}$				

	B_1	B_2	B_3	$x_4^{(3)}$		B_1	B_2	B_3	$x_5^{(3)}$		B_1	B_2	B_3	$x_6^{(3)}$
B_1	1	1/3	3	0.31	B_1	1	2	1	0.215	B_1	1	8	3	0.44
B_2	3	1	3	0.44	B_2	1/2	1	¼	0.331	B_2	1/8	1	1	0.55
B_3	1/3	1/3	1	0.1	B_3	1	4	1	0.296	B_3	1/3	1	1	0.12
λ_{max}= 2.05; l_0= 0.047					λ_{max}= 2.01; l_0= 0.009					λ_{max}= 2.02; l_0= 0.02				

Здійснюємо визначення альтернативи для кожного фактору ієрархії.

$$p_1^{(2)} = \begin{bmatrix} 0.281 & 0.241 \\ 0.491 & 0.415 \\ 0.345 & 0.291 \end{bmatrix} \times \begin{bmatrix} 0.412 \\ 0.214 \end{bmatrix} = \begin{bmatrix} 0.167 \\ 0.291 \\ 0.204 \end{bmatrix} \quad (2.1)$$

$$p_2^{(2)} = \begin{bmatrix} 0.321 & 0.31 \\ 0.52 & 0.44 \\ 0.15 & 0.1 \end{bmatrix} \times \begin{bmatrix} 0.712 \\ 0.387 \end{bmatrix} = \begin{bmatrix} 0.349 \\ 0.541 \\ 0.146 \end{bmatrix} \quad (2.2)$$

$$p_3^{(2)} = \begin{bmatrix} 0.215 & 0.44 \\ 0.331 & 0.55 \\ 0.296 & 0.12 \end{bmatrix} \times \begin{bmatrix} 0.641 \\ 0.152 \end{bmatrix} = \begin{bmatrix} 0.205 \\ 0.296 \\ 0.208 \end{bmatrix} \quad (2.3)$$

$$p_1^{(2)} = \begin{bmatrix} 0.167 & 0.349 & 0.205 \\ 0.291 & 0.541 & 0.296 \\ 0.204 & 0.146 & 0.208 \end{bmatrix} \times \begin{bmatrix} 0.451 \\ 0.512 \\ 0.402 \end{bmatrix} = \begin{bmatrix} 0.306 \\ 0.493 \\ 0.328 \end{bmatrix} \quad (2.4)$$

За результати обчислень альтернатив найкращим результатом буде використання Android SDK та NodeJS.

2.2. Конкретизація функціонування системи

Діаграма потоків даних (DFD) – це представлення потоку інформації для будь-якого процесу або системи. Для побудови діаграм використовуються певні символи як прямокутники, кола і стріли, також короткі текстові мітки, щоб показати вхідні та вихідні дані, пункти зберігання і маршрути між кожним призначенням. Дані блок-схеми можуть варіюватися від простих, навіть намальованою від руки, і

поглиблених, багаторівневий DFD, що поступово глибше деталізують як обробляються дані. Вони можуть бути використані для аналізу існуючої системи або для створення нової. Як і всі діаграми та графіки, DFD часто можуть візуально “сказати” те, що було б важко пояснити словами, і вони працюють як для технічної так і нетехнічної аудиторії, від розробника до генерального директора.[6] Ось чому DFD залишаються настільки популярними до сьогодні. Існують дві концепції при побудові діаграм потоків даних:

- Об’єктно-орієнтований аналіз та проектування, висунута Йорданом та Коедом для аналізу і розробки програм або систем;
- Структурний аналіз та проектування систем.

На рис. 2.3. наведено діаграму IDEF0.

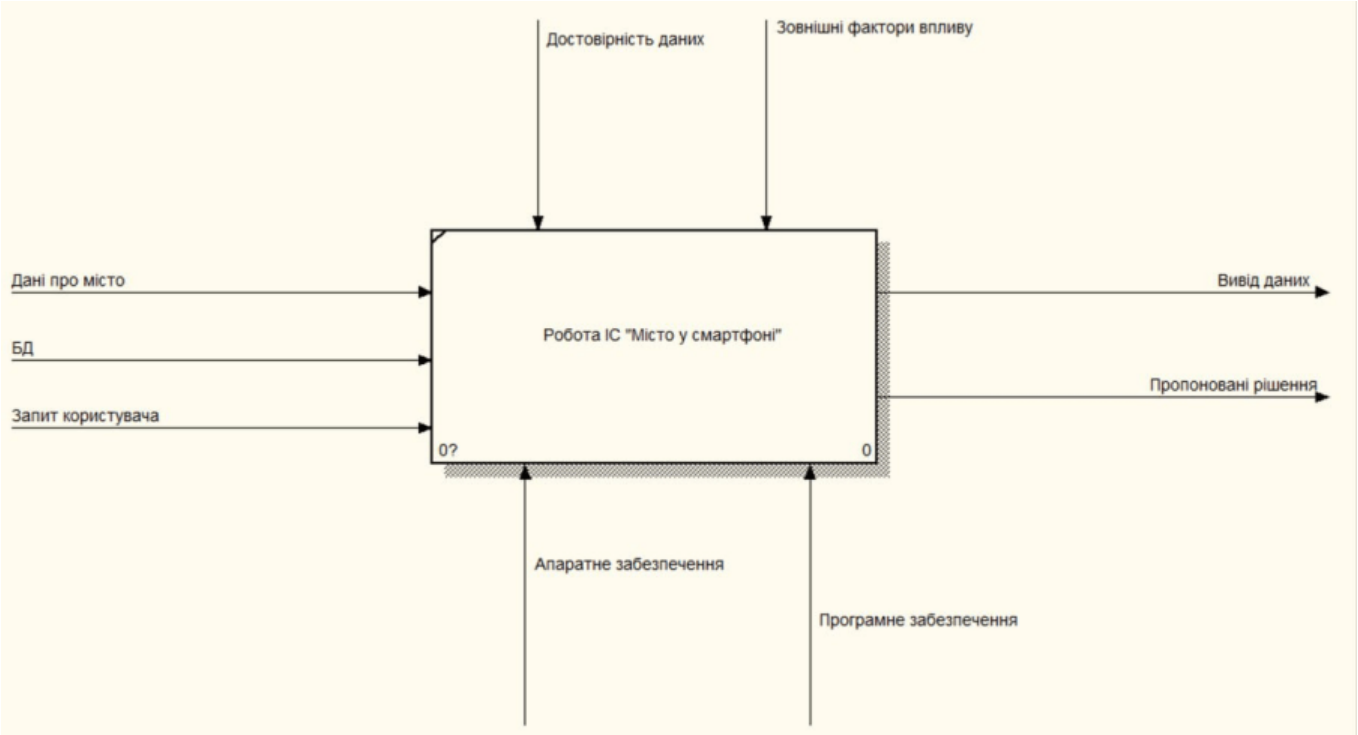


Рис. 2.3. Діаграма IDEF0

Контекстна діаграма (рис. 2.4) містить зв’язок системи з зовнішніми сутностями, які взаємодіють з нею за допомогою потоків. У проектованій системі присутні наступні сутності:

- Клієнт – це користувач, який буде користуватися системою, надавати системі інформацію про об’єкти, які він хоче відвідати, а також дані для автентифікації;
- Карти – це засіб для знаходження зв’язку між об’єктами та створення маршруту між ними;
- Сервер – це засіб, який використовується для взаємодії з віддаленими базами даних.

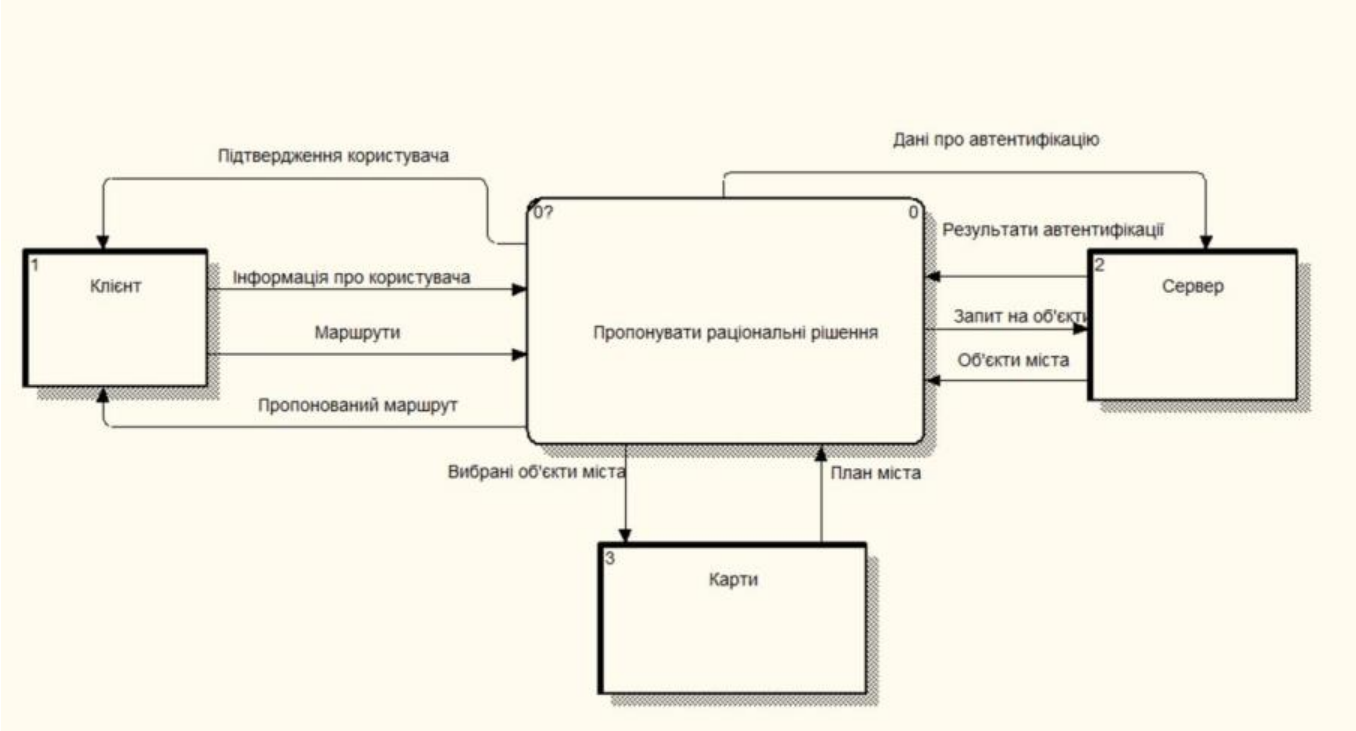


Рис. 2.4. Контекстна діаграма

Далі побудуємо діаграму потоків даних першого рівня (рис. 2.5). На цьому етапі ми вже зобразимо взаємодію між процесами, які відбуваються у системі. Кожен процес певним чином взаємодії з зовнішніми сутностями, але на деталізованій діаграмі потоків даних першого рівня ці зовнішні сутності не представлені, а стрілки, які ведуть до цих сутностей йдуть до різних сторін діаграми. Варто зазначити, що потоки даних, що зображені на контекстній діаграмі повинні відповідати тим, які на деталізованій. На діаграмі потоків першого рівня виділимо наступні процеси:

- Здійснити автентифікацію;

- Вибрати маршрут;
- Надати відгуки.

Процес “Здійснити автентифікацію” виконує автентифікацію користувача в системі. Процес “Вибрати маршрут” передбачає визначення об’єктів, що користувач хоче відвідати та створення для нього найоптимальнішого маршруту. Об’єкти надходять від зовнішньої сутності, “Клієнт”. Процес “Надіслати відгук” дозволяє надіслати відгуки про відвідані об’єкти, щоб допомогти наступним користувачам при плануванні маршруту. Також користувач має змогу поділитися своїм маршрутом у соціальних мережах.

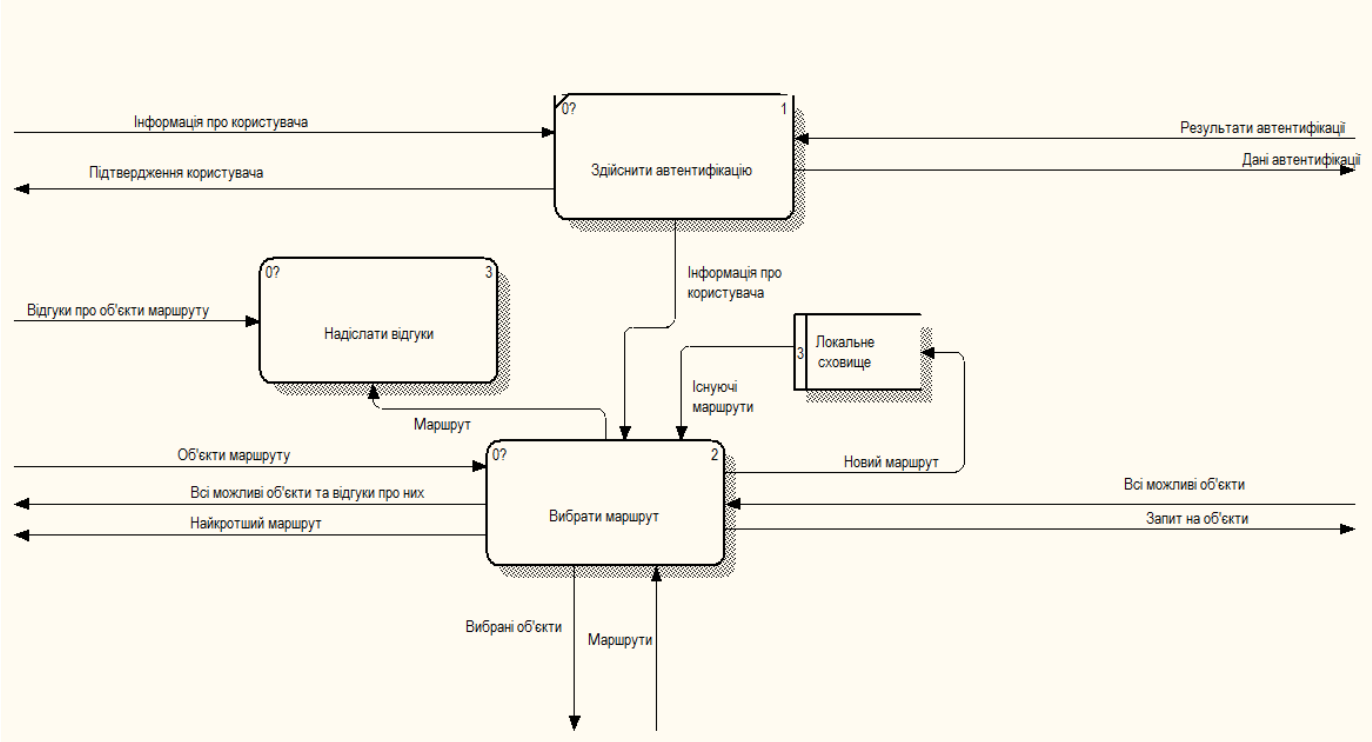


Рис. 2.5. Діаграма потоків даних першого рівня

Для другого рівня деталізації потоків даних виносимо процес “Вибрати маршрут” (Рис. 2.6) та “Надіслати відгук” (Рис. 2.7.). Процес “Надіслати відгуки” містить два підпроцеси: “Зберегти відгук” та “Пошири відгуки в соцмережі”. Підпроцес “Зберегти відгуки” передбачає збереження відгуків у сховищі даних, а “Пошири відгуки в соцмережі” поділитися маршрутом з друзями.

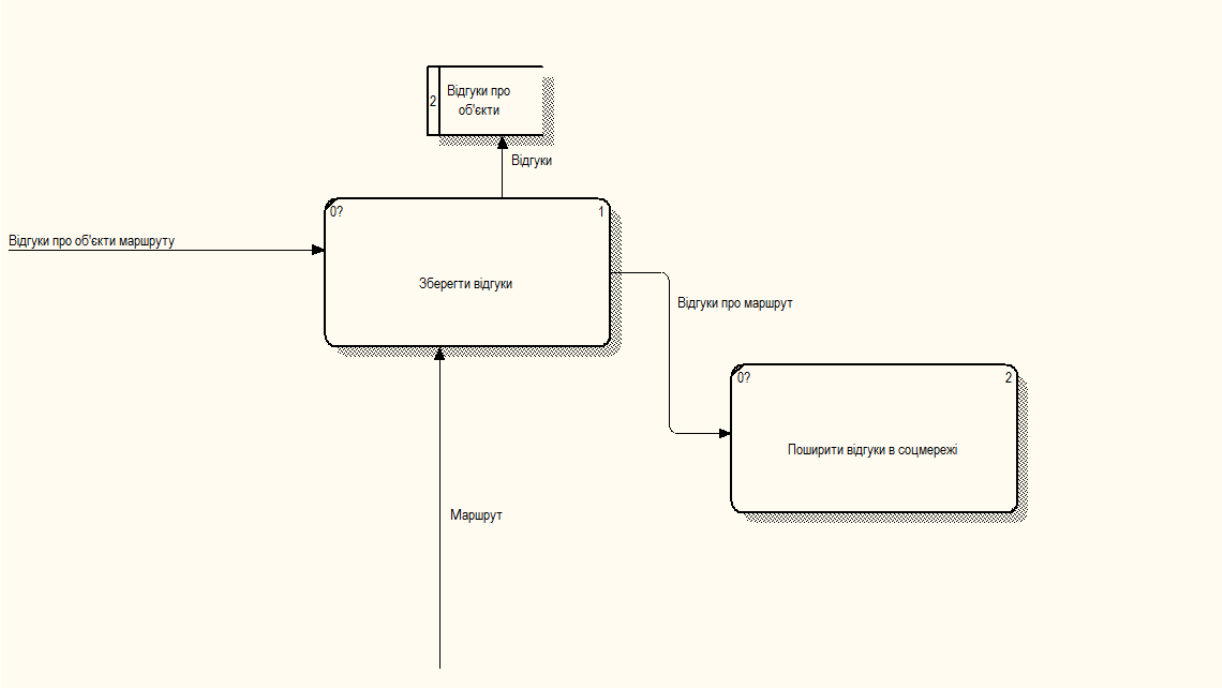


Рис. 2.6. Діаграма потоків даних другого рівня для процесу "Надіслати відгуки"

Процес "Вибрати маршрут" являє собою створення найоптимальнішого маршруту для проходження об'єктів, які вибрав користувач. Всі можливі об'єкти та відгуки про ці об'єкти надходять до користувача, далі користувач обирає об'єкти і система виконує побудову найоптимальнішого маршруту. Система перевіряє чи існує вже маршрут з обраних об'єктів і якщо існує то надає його користувачеві, а у випадку відсутності знайти найкоротший маршрут і зберегти його в сховищі даних [8].

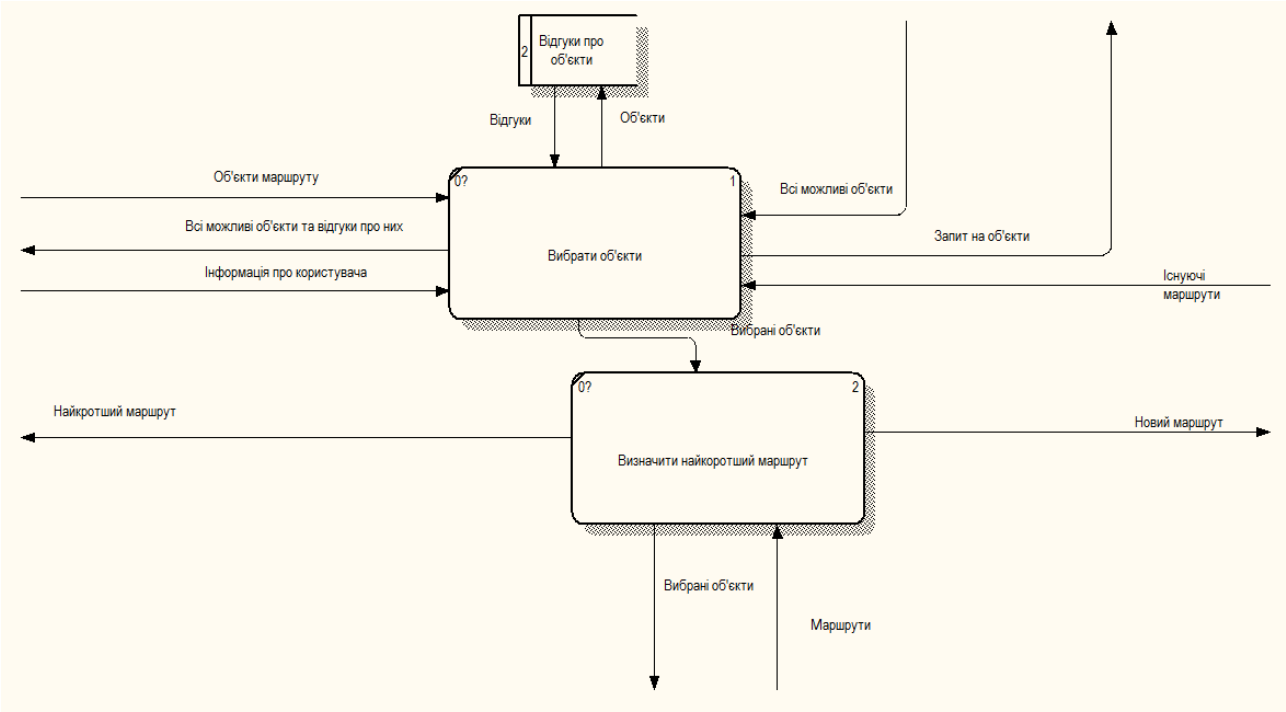


Рис. 2.7. Діаграма потоків даних другого рівня для процесу "Вибрати маршрут"

2.3. Побудова ієрархії задач

Ієрархія задач представляє задачі, які повинні бути виконані на різних етап при розробці системи. Основне завдання розробити систему. На першому етапі декомпозиції є наступні завдання: "здійснити автентифікацію", "вибрати маршрут" та "надіслати відгуки". Завдання "вибрати маршрут" та "надіслати відгуки" мають також другий рівень декомпозиції.

Ієрархію задач наведено на рис. 2.8.

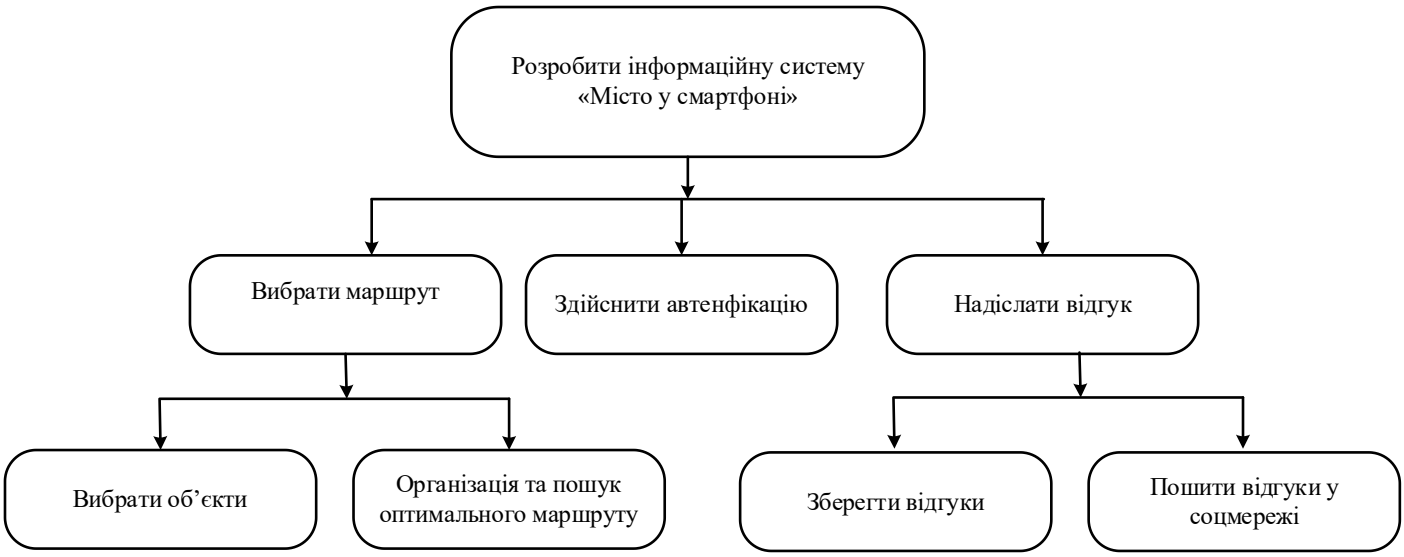


Рис. 2.8. Ієрархія задач системи

Висновок до 2-го розділу

Проаналізовано вимоги перед системою побудови маршрутів. Відповідно до проведеного аналізу, головним завданням системи є створення туристичного маршруту за вказаними об’єктами. Також цей маршрут повинен бути оптимальним. Для взаємодії з клієнтом потрібні такі зовнішні сутності як карти та сервер. Спроектовано діаграми потоків даних для системи. Також проведено аналіз ієрархій і розраховано альтернативи для кожного фактору ієрархії.

РОЗДІЛ 3

Програмні засоби розв’язання задачі

3.1. Вибір та обґрунтування засобів розв’язання задачі

Провівши аналіз проблем та способів її вирішення слід перейти до вибору технічних засобів реалізації обраного рішення. На початку буде розроблятися лише версія додатку для мобільної операційної системи Android, оскільки вони найпопулярніша на ринку, та охоплює найбільший відсоток цільової аудиторії.

Для цього слід проаналізувати методи розробки мобільних додатків, а саме мови програмування, архітектурні підходи, системи для керування базами даних.

Java – строго типізована об'єктно-орієнтована мова програмування, яка була розроблена компанією Sun Microsystems. Програми Java компілюється в спеціальний байт-код, тому вони можуть працювати на будь-якої комп'ютерної архітектурі, за допомогою віртуальної Java-машини (JVM – Java Virtual Machine). Станом на 2016 рік Java є однією з найпопулярніших мов програмування [10].

Синтаксис мови був запозичений з C/C++, за основу було взято об’єктну модель C++, яка була модифікована. Перш за все Java розроблялась як платформо-незалежна мова, як наслідок вона має менше низькорівневих засобів для роботи з апаратним забезпеченням, що в порівнянні, з C привело до зменшення швидкодії програм. Проте якщо виникає така необхідність Java дозволяє використовувати підпрограми написані на інших мовах.

На початку, було декілька основних цілі для створення Java:

- синтаксис мови повинен бути простим, та інтуїтивно зрозумілим;
- реалізація має бути надійною та «безпечною»;
- платформо незалежність, тобто повинна виконуватись на різних платформах;
- висока швидкодія;
- повинна бути інтерпретованою, багато потоковою та динамічною.

На даний момент Java є основною мовою для створення мобільних додатків для операційної системи Android. В даному варіанті код програми компілюється не в

стандартний байт-код, щоб вони виконувались віртуальною машиною Dalvik (починаючи з Android 5.0 Lollipop віртуальна машина була змінена на ART).

Основними перевагам Java є:

- автоматичне управління пам'яттю;
- розширені можливості обробки виняткових ситуацій (Exceptions);
- Java collections framework, це набір класів та інтерфейсів, які реалізують типові структури даних для зберігання та обробки данних;
- наявність класів, що дозволяють реалізувати HTTP-запити та обробляти відповіді;
- вбудовані засоби для створення багато поточних додатків;
- уніфікований доступ до баз даних:
- підтримка лямбда, замикань, вбудовані можливості функціонального програмування (починаючи з версії 1.8);

На даний момент останньою версією Java є версія 1.8, в якій було додано елементи функціонального програмування [11].

Kotlin – статично типізована мова програмування, яка працює на віртуальній машині Java (JVM), а також може бути скомпільована в вихідний код JavaScript [12]. Хоча синтаксис не сумісний з Java, Kotlin призначений для взаємодії з Java кодом і залежить від Java коду з існуючої Java Class Library, наприклад Java collections framework.

Автори хотіли створити лаконічнішу та типо-безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція.

Синтаксис мови схожий на Pascal, TypeScript, Haxe, PL / SQL, F #, Go і Scala, C ++, Java, C # і D. При оголошенні змінних та параметрів, типи даних вказуються після назви (роздільник двокрапка). Крапка з комою як роздільник операторів є не обов'язковою, як в Scala і Groovy, в більшості випадків переносу рядка досить щоб компілятор зрозумів, що вираз закінчився.

Як додаток до класів і методів (які називаються функціями-членам в Kotlin) об'єктно-орієнтованого програмування, Kotlin підтримує процедурне програмування з використанням функцій. Як і в C та C ++, точкою входу в програму на мові Kotlin є функція з ім'ям «main», яка приймає масив, що містить будь-які аргументи командного рядка.

Одним з найпопулярніших варіантів застосування Kotlin є розробка додатків для операційної системи Android. Kotlin має декілька переваг порівняно з Java, а саме захист від null значень, функції розширення та інфіксна нотація, що покликані покращити читабельність коду, спростити розширення класів Android SDK та прискорити розробку.

C++ – компільована, статично типізована мова програмування загального призначення. Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має велику стандартну бібліотеку, яка включає в себе найнеобхідніші контейнери і алгоритми, засоби введення-виведення, регулярні вирази, підтримку багато поточності та інші можливості. C ++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником – C, найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування [13].

Основними перевагами C++ є:

- сумісність з мовою C;
- можливість працювати на апаратному рівні;
- як наслідок попередніх пунктів, висока швидкодія;
- підтримка багатьох стилів програмування;
- автоматичний виклик деструктора в адекваному порядку;
- перевантаження операторів.

Також C++ має декілька достатньо вагомих недоліків:

- недосконала система модулів, що призводить до дуплікації частин коду, та збільшення часу компіляції програми;

- сама по собі мова є достатньо складною, що збільшує час її вивчення;
- наявність великої кількості можливостей, які порушують правила типобезпеки, що може призвести до помилок, які важко відслідкувати.

На сьогодні набирає популярності *PhoneGap*, це безкоштовний open-source фреймворк для створення мобільних додатків використовуючи JavaScript, HTML5 та CSS3, без необхідності знання «рідних» мов програмування під всі мобільні платформи (iOS, Android, Windows Phone 8, Bada, Symbian, Tizen). Готовий додаток компілюється у вигляді встановлюваних пакетів для кожної мобільної операційної системи.

JavaScript – мультипарадигмова мова програмування. Підтримує об'єктно-орієнтований, імперативний і функціональний стилі. Є реалізацією мови ECMAScript (стандарт ECMA-262).

JavaScript мова, використовується як вбудована мова для програмного доступу до об'єктів вже готових додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для створення інтерактивних веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

JavaScript виник під впливом багатьох мов, основною метою було розробити мову схожу на Java, але при цьому легку для використання непрограмістів. Мовою JavaScript не володіє жодна компанія чи організація, що відрізняє його від ряду мов програмування, які використовуються в веб-розробці.

Незважаючи на схожий з C синтаксис, JavaScript має кілька значних відмінностей:

- об'єкти з можливістю інтроспекції;
- функції як об'єкти першого класу;
- автоматичне приведення типів;
- автоматичне прибирання «сміття»;
- анонімні функції.

Node.js - програмна платформа, заснована на двигуні V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення. Node.js дозволяє JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C ++), підключати зовнішні бібліотеки, написані на різних мовах, даючи можливість викликати ці бібліотеки з JavaScript-коду. Node.js застосовується переважно для розробки серверного програмного забезпечення, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows і Mac OS). В основі Node.js лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням / виведенням.

Node.js розробив Райан Даль в 2009 році після двох років експериментування над створенням серверних веб-компонент. В ході своїх досліджень він прийшов до висновку, що замість традиційної моделі паралелізму на основі потоків слід звернутися до подієво-орієнтованих систем. Ця модель була обрана через простоту, низьких накладних витрат (в порівнянні з ідеологією «один потік на кожне з'єднання») і швидкодії. Метою Node є запропонувати «простий спосіб побудови масштабованих мережних серверів».

PHP – це скриптова мова загального призначення, яка активно використовується для розробки веб-додатків. На даний момент підтримується більшістю хостинг-провайдерів та є одним з лідерів серед мов, що застосовуються для створення серверної частини для динамічних веб-сайтів.

Популярність в області побудови веб-сайтів спричинена наявністю великої кількості набору вбудованих засобів для розробки веб-додатків. Основні з них:

- автоматичне вилучення POST і GET-параметрів, а також змінних оточення веб-сервера в зумовлені масиви;
- можливість взаємодії з великою кількістю різноманітних систем управління базами даних (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape і

Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB ++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB, Інтерфейс PDO);

- автоматичне додавання HTTP-заголовків;
- вбудована робота з HTTP-авторизацією;
- вбудована можливість працювати з cookies і сесіями;
- можливість роботи з локальними і віддаленими файлами та сокетами;
- обробка файлів, які завантажуються на віддалений сервер;
- робота з XForms.

Symfony – це фреймворк для створення веб-додатків PHP, який включає набір повторно використовуваних PHP-компонент та бібліотек. Symfony була опублікована як безкоштовне програмне забезпечення 18 жовтня 2005 року та випущена під ліцензією MIT.

Основною метою Symfony є прискорення створення і обслуговування веб-додатки та уникнути дуплікації коду. Недоліком даного фреймворку є низька продуктивність.

Symfony націлений на створення достатньо надійних додатків в корпоративному сегменті та створений для надання розробникам повний контроль над конфігурацією: від структури каталогів до зарубіжних бібліотек, майже все можна налаштувати. Відповідно до рекомендацій по розвитку підприємства Symfony поставляється з набором додаткових інструментів, які допомагають розробникам тестувати, налагоджувати і документувати проекти.

Yii – об'єктно-орієнтований компонентний фреймворк, написаний на PHP і реалізує парадигму MVC.

Основні можливості:

- висока продуктивність;
- парадигма MVC;
- інтерфейси DAO для роботи з базами даних;

- підтримка інтернаціоналізації;
- кешування;
- перехоплення та обробка виняткових ситуацій;
- реалізована можливість аутентифікації та авторизації;
- велика кількість сторонніх бібліотек;
- вбудована можливість автоматизованого тестування.

Ruby – динамічна, рефлективна, інтерпретована високорівненва мова програмування. Мова має незалежні від операційної системи реалізації багато поточності, Також Ruby характеризується суворою динамічною типізацією, збиральником сміття та багатьма іншими можливостями. За особливостями синтаксису мова близька до Perl і Eiffel, та за об'єктно-орієнтованим підходом - до Smalltalk. Також деякі риси мови взяті з Python, Lisp, Dylan.

Однією з основних цілей розробки було зменшення рутинної роботи для програмістів, яку компютер може виконувати швидше та якісніше. Особлива увага, приділялася буденним рутинним заняттям (обробка текстів, адміністрування), і для них мова розроблена особливо добре. На противагу машинно-орієнтованим мовам, які працюють швидше, метою цієї розробки була мова, що стане найбільш близькою до людини.

Ruby on Rails, або просто Rails, являє собою платформу веб-додатків на стороні віддаленого сервера, написану на Ruby під ліцензією MIT. Rails - це модель-представлення-контролер (MVC), що надає структури за замовчуванням для бази даних, веб-служби та веб-сторінок. Платформа заохочує і полегшує використання веб-стандартів, таких як JSON або XML для передачі даних, а також HTML, CSS і JavaScript для відображення і взаємодії з користувачем. На додаток до MVC, Rails підкреслює використання інших добре відомих патернів і парадигм проектування програмного забезпечення, включаючи CoC (Convention over configuration – укр. угода головніша за конфігурацію), не повторюйте себе (DRY) і шаблон активних записів.

Gradle – система атоматизованого збирання, побудована на принципах Apache Ant і Apache Maven, але надає DSL на мові Groovy замість традиційної XML-подібної форми подання конфігурації проекту.

На відміну від Apache Maven, основою якого є життєвий циклк проекту, і Apache Ant, в якому порядок виконання завдань (targets) визначається відносинами залежності (depends-on), Gradle використовує спрямований ациклічний граф для визначення порядку виконання завдань.

Gradle був розроблений для розширюваних багатопроектних збірок, і підтримує інкрементальні збірки, визначаючи, які компоненти дерева збірки не змінилися і які завдання, залежні від цих частин, не вимагають перезапуску.

Основні плагіни призначені для розробки і розгортання Java, Groovy і Scala додатків, але готуються плагіни і для інших мов програмування.

MySQL – реляційна система управління базами даних, яка доступна у відкритому доступі. Розробкою та підтримкою MySQL займається корпорація Oracle, яка отримала права на торговельну марку разом з поглиненою Sun Microsystems, що раніше купила шведську компанію MySQL AB. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім того, розробники створюють функціонал за замовленнями ліцензійних користувачів. Саме завдяки одному з таких замовлень майже в найперших версіях з'явився механізм реплікації.

MySQL – це рішення для невеликих застосунків. Зазвичай MySQL використовують як сервер, до якого можуть звертатись як локальні так і віддалені клієнти, але дистрибутив містить і бібліотеку для внутрішнього сервера, що дозволяє включати MySQL в програми які працюють автономно [14].

SQLite – це система керування базами даних, що міститься в бібліотеці програмування C [15]. На відміну від багатьох інших систем управління базами даних, SQLite не є реалізує архітектуру клієнт-сервер. Зазвичай він вбудований в кінцеву програму

SQLite є ACID-сумісним і реалізує більшість стандартів SQL, використовуючи динамічно і слабо типізований синтаксис SQL, який не гарантує цілісність домену.

SQLite є популярним вибором в якості програмно-апаратних засобів баз даних для локального/клієнтського сховища в прикладному програмному забезпеченні, такому як веб-браузери, чи мобільні додатки. Це, мабуть, найбільш поширена система керування базами даних даних, оскільки сьогодні він використовується в кількох найбільш поширених браузерах, операційних системах і вбудованих системах (такий як мобільні телефони) і т.д. SQLite має прив'язки до багатьох мов програмування.

XML – це розширювана мова розмітки. Специфікація XML описує XML-документи та частково описує поведінку XML-процесорів (програм, які зчитують XML-документи, а також забезпечують доступ до їх вмісту). XML розроблялась як мова з простим формальним синтаксисом, який буде зручним для створення і обробки документів програмами та одночасно зручним для читання та створення документів людиною, з підкресленням націленості на використання в Інтернеті.

Мова називається розширюваною, оскільки нею не фіксується розмітка, яка може використовуватись в документах, тобто розробник сам може створити розмітку відповідно до його потреб, будучи обмеженим лише синтаксичними правилами мови. Розширення XML - це конкретна граматика, створена на базі XML і представлена словником тегів та його атрибутів, а також набором правил, що визначають які атрибути і елементи можуть входити до складу інших елементів.

Android SDK – універсальний засіб розробки мобільних додатків для операційної системи Android. Android SDK включає в себе повний набір засобів розробки. До них відносяться відладчик, бібліотеки, емулятор телефону на основі QEMU, документація, приклад коду та документація. На данний момент підтримуються всі популярні платформи, а саме: Linux під Linux (Macintosh Linux), Mac OS X 10.5.8 або пізнішої версії і Windows 7 або новіше. Станом на березень 2015 року SDK недоступний безпосередньо у операційній системі Android, але

розробка програмного забезпечення може проводитись, за допомогою спеціальних додатків.

Покращення Android SDK, йдуть рука об руку з загальним розвитком Android платформи. SDK також підтримує старі версії Android платформи в разі, якщо розробники хочуть підтримувати старі пристрої, а також для зворотної сумісності з вже існуючими додатками. Засоби розробок завантажуються компонентами, тому після того, як було скачано останню версію компоненти, старі платформи і інструменти, також можуть бути завантажені для тестування сумісності.

Android NDK – це набір бібліотек, написаних на C, C ++ та інших низькорівнених мовах, які можуть бути скомпільовані для ARM, MIPS або x86. Ці нативні бібліотеки можуть бути викликані з Java-коду, який працює під ART VM? використовуючи стандартні класи Java для Android.

Готові додатки можуть бути зібрані і встановлені з використанням традиційних інструментів розробки (Android SDK). Однак, відповідно до документації Android, NDK не повинен використовуватися тільки тому, що розробник вважає за краще програмувати на C / C ++, оскільки використання NDK збільшує складність написаного, в той час як більшість додатків не виграють від його використання.

Google рекомендує використовувати NDK тільки в рідкісних випадках. Наприклад:

- потрібно збільшити продуктивність (наприклад, сортування великого обсягу даних);
- необхідне використання стороньої бібліотеки. Наприклад, багато вже чого написано на C / C ++ мовами і потрібно просто використати існуючий матеріал. Приклад таких бібліотек, як, Ffmpeg, OpenCV;
- Програмування на низькому рівні (наприклад, все що виходить за рамки ART чи Dalvik).

Можна використовувати Android Studio з Gradle для розробки NDK проектів. Інші інструменти сторонніх виробників дозволяють інтегрувати NDK в Eclipse, та Visual Studio.

Обґрунтування. Провівши аналізу вимог до системи, та дослідження сучасних засобів для вирішення проблеми було поставлено задачі розробити серверну частину, яка буде зберігати загальні дані, та безпосередньо дані кожного користувача, та мобільний додаток для користувачів.

Після проведеного аналізу засобів для розробки серверної частини, було вибрано Node.js, оскільки він є більш швидким в порівнянні з аналогами, велика кількість написаних рішень, які можна перевикористати, також простота в навчанні, за рахунок великої кількості матеріалів у відкритому доступі.

Для збереження данних на стороні сервера буде використовуватись MySQL оскільки він має хорошу сумісність с Node.js та є зручним у використанні. Також має достатньо велику швидкість виконання команд, та підтримує багатопотоковість.

Для розробки мобільного додатку для ОС Android було вибрано Android SDK. Основними перевагами Android SDK над Android NDK є простота використання, більша кількість вже готових компонент. Також те що основною причиною використання Android NDK є важкі операції. Для вирішення поставленої задачі немає потреби використовувати потужності Android NDK.

Наступним етапом є вибір мови програмування, яка буде використовуватись для написання додатку. На Java написана більшість додатків для які можна знайти в Google Play Store, а також бібліотек для розробки.

Для збереження даних на стороні користувача буде використовуватись SQLite. Оскільки Android SDK має вже готові класи написані на Java для роботи з цією системою, що полегшить процес розробки, також вона повінсю задовільняє всі вимоги для вирішення поставленої задачі.

3.2. Технічні характеристики обраних програмних засобів

Android Studio, являє собою офіційне інтегроване середовище розробки (IDE) для Android платформи. Android Studio було представлено 16 травня 2013 року на конференції Google I / O.

Android Studio було розробленона основі Intellij IDEA від компанії JetBrains спеціально для розробки для ОС Android. Воно доступне на всіх популярних платформах а саме: Windows, MacOS і Linux.

Androd Studio має ряд корисних функцій а саме:

- розширений редактор макетів: WYSIWYG, здатність працювати з UI компонентами за допомогою Drag-i-Drop, функція попереднього перегляду макета на декількох конфігураціях екрану;
- збірка додатків, заснована на Gradle;
- різні види збірок і генерація кількох .apk файлів;
- «розумний» рефакторинг коду;
- статичний аналізатор коду (Lint), що дозволяє знаходити проблеми продуктивності, несумісності версій ще на етапі розробки;
- вбудований ProGuard і утиліта для підписування додатків;
- шаблони основних макетів та компонент Android.
- підтримка розробки додатків для Android Wear і Android TV;
- зручний вбудований емулятор.

IntelliJ IDEA – це інтегроване середовище розробки програмного забезпечення на багатьох мовах програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains [17].

Перша версія з'явилася в січні 2001 року і швидко стала популярною, як перше середовище для Java з широким набором інтегрованих інструментів для рефакторингу коду, що давало можливість програмістам швидко та безпечно реорганізовувати вихідний код додатів. Дизайн *IntelliJ IDEA* орієнтований на продуктивність роботи програмістів, дозволяючи сконцентруватися на функціональних завданнях, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.

Починаючи з шостої версії IntelliJ IDEA надає інтегрований набір інструменів для розробки графічного інтерфейсу користувача. Серед інших можливостей,

середовище досить добре сумісне з багатьма популярними інструментами розробників, які є у вільному доступі, такими як Android SDK, CVS, Subversion, Apache Ant, Maven і JUnit. У лютому 2007 року розробники IntelliJ анонсували ранню версію плагіна для підтримки програмування на мові Рубі.

Починаючи з версії 9.0, середовище доступне в двох редакціях, а саме: Community Edition і Ultimate Edition. Community Edition є повністю безкоштовною версією, доступною під ліцензією Apache 2.0, в ній реалізована повна підтримка Java SE, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. В версії Ultimate Edition реалізована підтримка Java EE, UML-діаграм, а також підтримка інших систем управління версіями, мов та фреймворків.

Eclipse являє собою, вільне інтегроване середовище розробки кроссплатформенних додатків. Розвивається і підтримується Eclipse Foundation. Головною особливістю Eclipse є те що він є платформою для розширень, що дає кожному розробнику розширити Eclipse своїми модулями.

Найбільш відомі програми на основі Eclipse Platform - різні «Eclipse IDE» для розробки ПЗ на багатьох мовах (наприклад, найбільш популярний «Java IDE», підтримувався спочатку, не покладається на будь-які закриті розширення, використовує стандартний відкритий API для доступу до Eclipse Platform).

Обґрунтування. Провівши аналіз існуючих засобів розробки, було обрано Android Studio для розробки додатку для платформи Android. Оскільки воно має зручний емулятор, всі необхідні засоби для відлагодження додатку, та зручні засоби для побудови інтерфейсу.

Для розробки серверної частини проекту ідеально підходить Intellij IDEA, оскільки вона має ряд переваг над Eclipse, а саме: можливість рефакторингу, хороша підтримка системи контролю версій.

Під час розробки Android додатку буде використовуватись смартфон LG Nexus 5X, а також емулятори для перевірки сумісності з іншими смартфонами на базі Android.

Висновок до 3-го розділу

У цьому розділі було проведену аналіз та пошук підходящих засобів для вирішення поставленої задачі. Було вибрано засоби для розробки серверної частини, та Android додатку.

Основними критеріями вибору були: зручність в користуванні для розробника, достатня документація по вибраній технології, можливість підтримувати проект у майбутньому. Все це дасть змогу спростити та пришвидшити процес розробки, а також створити якісний, та гнучкий для розвитку продукт.

РОЗДІЛ 4

Практична реалізація

4.1. Опис створеного програмного засобу

4.1.1. Опис структури бази даних

Мобільний додаток використовує систему керування базами данних SQLite. Оскільки вона має хорошу підтримку у операційній системі Android. Ця база даних міститиме інформацію про визначні місця, які ми отримаємо з сервера. У базі буде дві табиці, які будуть незалежними одна від одної. Перша для збереження точок, а інша для збереження користувачів (див. рис. 4.1).

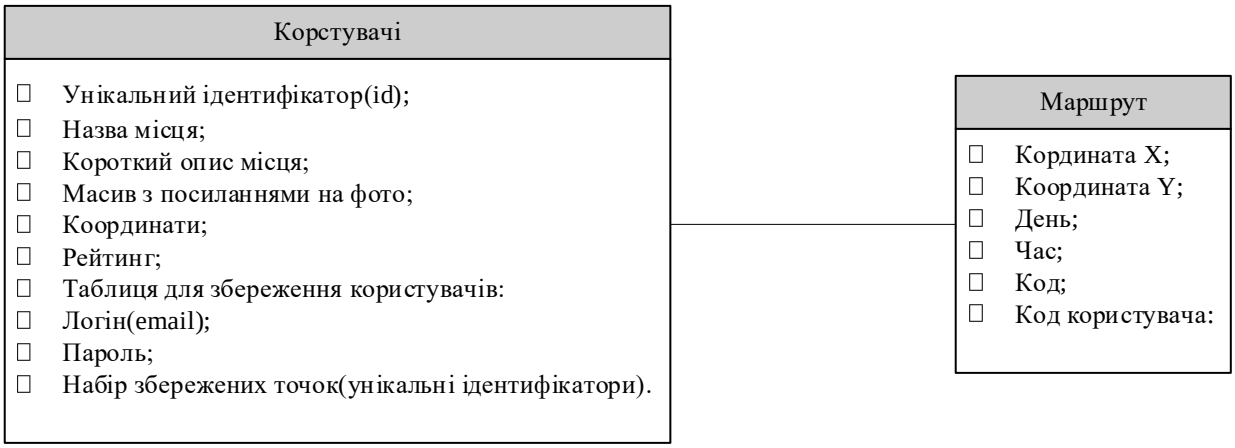


Рис. 4.1. База даних системи

Таблиця для користувачів міститиме наступні поля:

- Унікальний ідентифікатор(id);
- Назва місця;
- Короткий опис місця;
- Масив з посиланнями на фото;
- Координати;
- Рейтинг;
- Таблиця для збереження користувачів:
- Логін(email);
- Пароль;
- Набір збережених точок (унікальні ідентифікатори).

4.1.2. Загальні відомості

Програма має назву “Lviv Tourist”.

Клієнтська частина отримує всі данні з сервера, тому для першого запуску потрібен доступ до інтернету, що отримати дані про всі точки, та зберегти їх локально.

Для роботи серверної частини необхідно встановити її на хостингу, для доступу до неї через HTTP протокол[16]. Для коректної роботи потрібно встановити наступні засоби:

- Node.js;
- SQLite – для збереження данних на сервері;
- Npm – менеджер пакетів, який буде використовуватись, для встановлювання необхідних пакетів бібліотек;
- Для роботи клієнтської частини будуть використовуватись наступні засоби:
- Android SDK – для створення Android додатку;
- Google Maps Api – для відображення карти на стороні клієнта;
- SQLite – для збереження данних, для роботи додатку у режимі офлайн;
- Google Directions API – для побудови маршрутів.

4.1.3. Мови програмування на яких написана програма

Серверна частина реалізована на мові Node.js. Клієнтська частина написана на мові Java. Для створення розмітки буде використовуватись XML.

4.1.4. Функціональне призначення

Головним призначенням додатку є побудова туристичних маршрутів, у зручному для користувача вигляді. Також користувачеві буде надаватись коротка

інформація про обєкти які він матиме змогу відвідати. Головним призначенням є побудова маршрутів з декількома точками, та за допомгою відповідних алгоритмів оптимізація шляху. Це дасть змогу користувачам зекономити час оглянути більше визнчаних місць міста. Також перевагою додатку є те що він після одноразовго підключення до інтернету в подальшому він зможе працювати у режимі офлайн.

4.1.5. Опис логічної структури

Основою програми є обчислення оптимального маршруту між точками. Ці обчислення відбуватимуться на стороні клієнта, зроблено це для роботи без доступу до інтернету.

Архітектура серверної частини розділена на два рівні:

- Модуль взаємодії з базою даних (Persistence layer);
- Модуль зв’язок з мобільним додатком (API layer).

Серверна частина, являє собою віддалену базу данних, яка створена для можливості додавання нових туристичних атракцій, та редагування вже існуючих, без необхідності оновляти додаток.

Архітектура мобільного додатку розділена на наступні рівні:

- Головний модуль, який поєднює допоміжні модулі (Core layer);
- Модуль для обрахунків (Calculations layer);
- Модуль взаємодії з сервером (API layer);
- Модуль користувацького інтерфейсу (UI layer);
- Модуль взаємодії з локальною базою данних (Persistence layer).

Така архітектура забезпечує незалежність всіх модулів. Тобто всі частини додатку працюють незалежно та не «комунікують» одне з одним напряму. Завдяки такому підходу в подальшому можна легко добавити нові модулі, без порушення архітектури, а також перевикористати вже існуючи для інших додатків, оскільки не привязані до певної реалізації, а лише мають свої інтерфейси доступу.

Діаграма пакетів, представлена на рис. 4.2., відображає залежності між модулями у мобільному додатку та серверної частини.

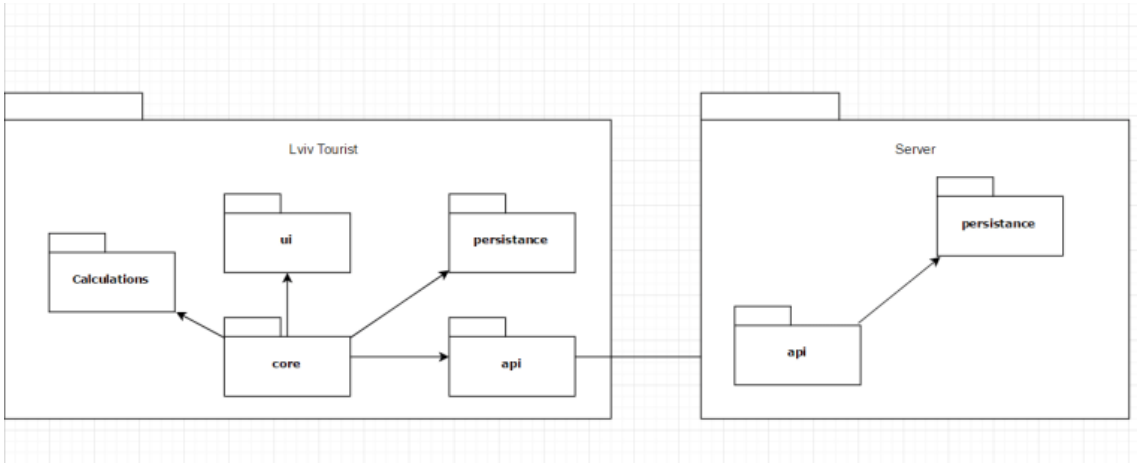


Рис. 4.2. Діаграма пакетів

Побудова маршруту складається з двох етапів. Обрахунок відстані між всіма парами точок(за допомою алгоритму Дейкстри). За допомогою одного з вирішень задачі про коміваояжера побудова оптимлаьного маршруту. Тобто користувач повинен відвідати кожну точку лише один раз та з найкоротшим шляхом.

Для створення головного фрагменту для вибору точок, що входитимуть до маршруту створимо фрагмент, який відображатиме усі точки.

Реалізація фрагменту коду наведено у додатку Б.

4.1.6. Опис алгоритму знаходження відстані між всіма токами

Найпростіша реалізація алгоритма Дейкстри потребує $O(V^2)$ дій [18]. У ній використовується масив відстаней та масив позначок. На початку алгоритму відстані заповнюються великим позитивним числом (більшим максимального можливого шляху в графі), а масив позначок заповнюється нулями. Потім відстань для початкової вершини вважається рівною нулю і запускається основний цикл.

На кожному кроці циклу ми шукаємо вершину з мінімальною відстанню і прапором рівним нулю. Потім ми встановлюємо в ній позначку 1 і перевіряємо всі сусідні з нею вершини. Якщо в ній відстань більша, ніж сума відстані до поточної вершини і довжини ребра, то зменшуємо його. Цикл завершується коли позначки всіх вершин стають рівними 1 [20].

Реалізація на мові Java наведено у додатку Б.

4.2. Інструкція користувача

4.2.1. Вступ

Програма призначена для туристів, відвідувачів міста Львова, для того щоб вони могли в умовах обмеженого часу оглянути більше туристичних атракцій.

4.2.2. Загальні відомості про програму

Для використання програми слід встановити її на свому смартфоні. Додаток сумісний з всіма смартфонами на базі операційної системи Android вище версії 5.0.

4.2.3. Опис основних характеристик і особливостей програми

Головною особливістю програми є можливість прокладання складних маршрутів. А також збереження їх на віддаленому сервері, для доступу з інших смартфонів.

4.2.4. Відомості про функціональні обмеження на застосування

Мобільному додатку необхідний інтернет лише при першому запуску програми, щоб отримати актуальні дані про туристичні місця міста Львова. Для роботи додатку необхідно надати дозвід для доступу в інтернет та визначення геолокації.

На першому екрані користувач повинен зареєструватись, або залогуватись, якщо акаунт вже був створений. Реєстрація вимагає лише e-mail та пароль, оскільки додаток не має потреби у інших данних користувача.

4.3. Аналіз контрольного прикладу

Для аналізу роботи додатку було додано декілька найбільш відомих туристичних точок міста львова на сервер. Для початку роботи користувач повинен ввести e-mail, у разі якщо користувач вже є на сервері, у додатку будуть відображені

збережені точки, якщо ж це новий користувач для нього буде створений новий акаунт (рис. 4.3).

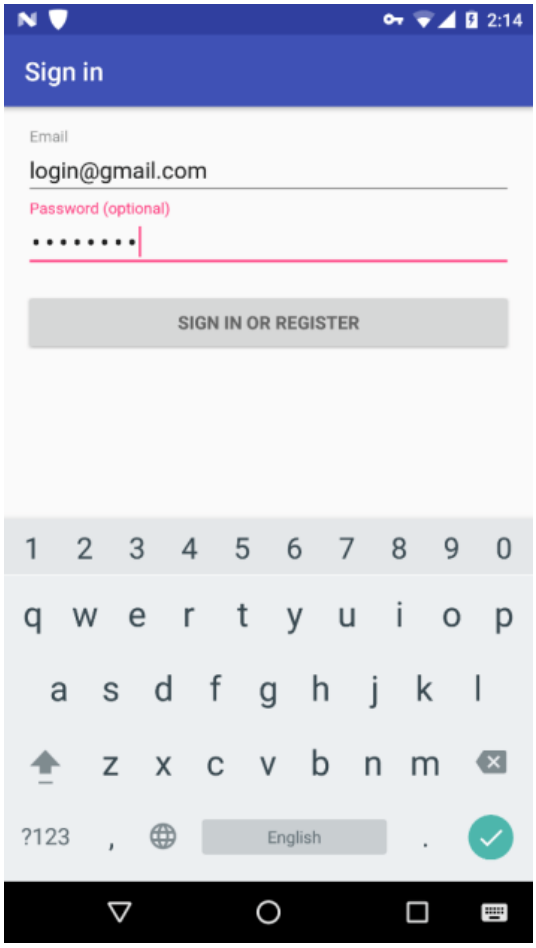


Рис. 4.3. Екран логування/реєстрації

На головному екрані відображено список з усіма можливими точками, з можливістю вибору, для побудови маршруту (рис. 4.4).

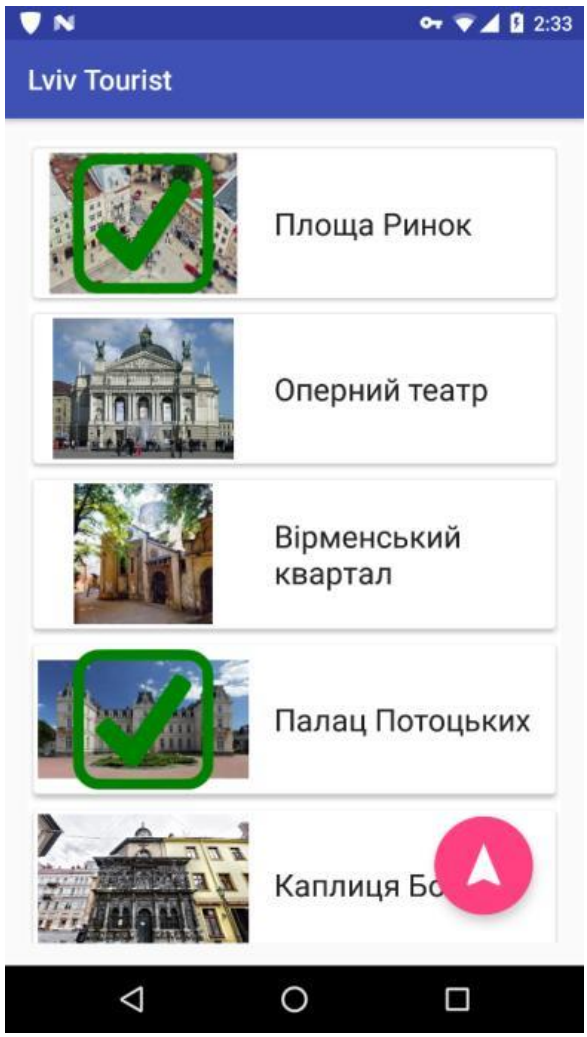


Рис. 4.4. Головний екран

Після вибору точок які входитимуть у маршртут, та підтвердження початкової точки, користувач бачить побудований маршрут на карті, з інформацією про найближчу, або початкову точку (рис. 4.5).

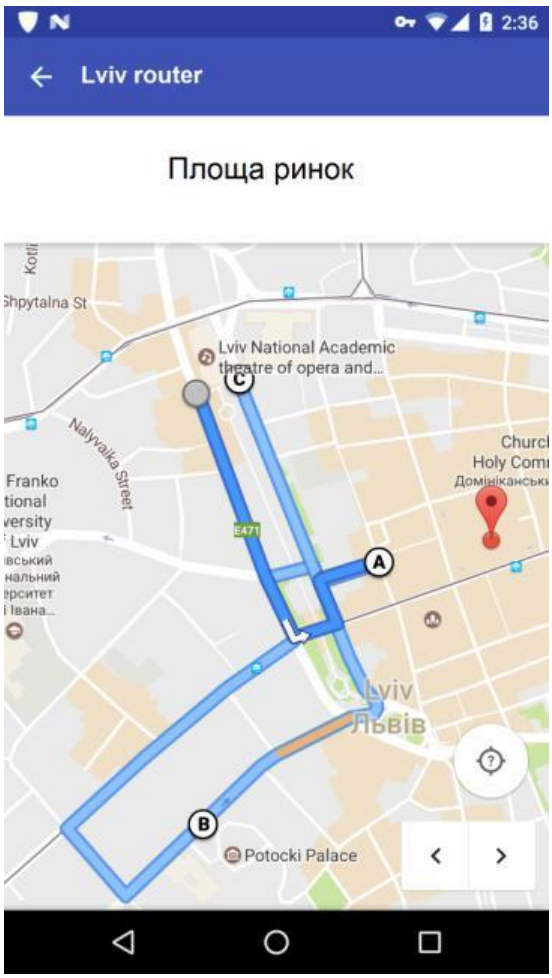


Рис. 4.5. Екран навігації

Висновок до 4-го розділу

Для вирішення поставленого завдання було розроблену серверну частину, яка відповідає за збереження данних на сервері, та Android додаток, для прокладання маршрутів. Всі обчислення відбувається на стороні клієнта, зроблено це було для можливості роботи в режимі офлайн. Також присутня простий процес реєстрації та логування, для збереження улюблених точок. Реалізований додаток дає можливість користувачеві прокладати складні машрути, які не матимуть повторів, та включати кожну точку лише один раз.

ВИСНОВКИ

Під час аналізу існуючих картографічних сервісів та туристичних сервісів, було досліджено що немає сервісу, який міг надати користувачеві можливість прокладати складні маршрути з оптимальним шляхом. Існує багато картографічних сервісів, які містять інформацію про видатні місця того чи іншого міста, але воне не дають можливості скаладати складні маршрути з оптимальною довжиною шляху. Також не всі сервіси коректно працюють в режимі офлайн, що є важливим критерієм при виборі сервісу такого типу. Ще однією перевагою розробленої системи, є можливість використовувати актуальні дані.

При роботі над проектом було проаналізовано завдання, які повинен вирішувати додаток. Побудовано діаграму потоків данних, дерево цілей та ієрархію задач. Осноновин завданням цих діаграм було створити оптимально систему як з точки зору користувача так і розробника, тобто можливість додавання нового функціоналу.

При виборі засобів для реалізації, було проведено аналіз існуючих технологій. Обрані засоби мають значну перевагу у швидкодії та зручності у використанні, що пришвидшить прокладання маршруту, та розробку додатку. При виборі засобів для реалізації Android додатку було виявлено що оптимальним варіантом є засоби від компанії Google, тобто Android SDK та мова Java, для розмітки буде використовуватись XML. Було обрано готові рішення для відображення карти та маршрутів, а також використано деякі стандартні компоненти з Android SDK.

Для збереження інформації було розроблено оптимальну структуру базу данних. Також наведено діаграму пакетів для компонент системи, сервера та клієнта. Що дасть змогу іншим розробникам швидко зрозуміти архітектуру системи, та при потребі розширити функціонал. Додано інструкцію для користувачів, яка містить короткий опис системи.

Було проведено дослідження та аналіз різноманітних алгоритмів пошуку найкоротшого шляху та обрано найбільш підходящі. Також було проаналізовано

різні підходи для вирішення поставленої задачі, за допомогою вже існуючих засобів та алгоритмів.

Таким чином, в ході виконання роботи були поставлені та виконані задачі для побудови туристичних маршрутів з оптимальним шляхом у містів Львів.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Граф (дискретна математика) // Дата оновлення: 04.05.2023. [Режим доступу]: https://en.wikipedia.org/wiki/Graph_(discrete_mathematics) (Дата звернення: 02.03.2023).
2. Introduction to Algorithms, 3rd Edition / Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein – MIT Press, 1989. – 300 с.
3. Задача комівояжера // Дата оновлення: 04.05.2023. [Режим доступу]: https://en.wikipedia.org/wiki/Travelling_salesman_problem (Дата звернення: 02.03.2023).
4. Саати Т. Аналитическое планирование. Организация систем / Т. Саати, К. Кернс . — М. : Радио и связь, 1991. — 224 с.
5. Згуровський М. З. Основи системного аналізу / М. З. Згуровський, Н. Д. Панкратова. – Київ: BHV, 2007. – 402 с.
6. Верес О.М. Технології підтримання прийняття рішень / О.М. Верес — Львів: Видавництво Львівська політехніка, 2010. — 203 с.
7. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Буч Г. — 2-е изд. ; [пер. с англ.]. — М. : Издательство Бином ; СПб. : Невский диалект, 1999. — 690 с.
8. Маклаков С. В. BPwin и ERwin. CASE-средства разработки информационных систем / С. В. Маклаков. – Москва : ДИАЛОГ-МИФИ, 1999. – 233 с.
9. Катренко А. В. Системний аналіз об’єктів та процесів комп’ютеризації: підручник з грифом МОН / А. В. Катренко. – Львів : «Новий світ 2000», 2003. – 225 с.
10. Charlie H. Java Performance / Charlie Hunt, Binu John – Addison-Wesley – 2012 – 150 с.
11. Шилдт Г. Искусство программирования на JAVA / Г. Шилдт, Д. Холмс. - М.: Изд. дом «Вильямс», 2005. – 228 с.

12. Заяць В. М. Логічне і функційне програмування : навч. посібник / В. М. Заяць, М. М. Заяць. — Львів : Бескид Біт, 2006. — 372 с.
13. Шилдт Г. Самоучитель C++ / Г. Шилдт. – СПб.: БХВ-Петербург, 2005. – 688 с.
14. Шаховська Н. Б. Сховища даних / Н. Б. Шаховська, В. В. Пасічник. – Львів: «Магнолія-2006», 2008. – 485 с.
15. Райордан Р. Основы реляционных баз данных / Райордан Р. ; [пер. с англ.]. – М. : Издательско-торговый дом «Русская Редакция», 2001. – 384 с. : ил.
16. Буров Є. В. Комп'ютерні мережі: підручник / Є. В. Буров. – Львів : «Магнолія- плюс», 2007. – 262 с.
17. Intellij IDEA // Дата оновлення: 04.05.2023. [Режим доступу]: http://uk.wikipedia. org/wiki/IntelliJ_IDEA (Дата звернення: 02.03.2023).
18. Algorithms + Data Structures = Programs / Niklaus Wirth – 432 с.
19. Фишберн П. Теория полезности для принятия решений / Фишберн П. — М. : Наука, 1978. — 352 с.
20. Дискретна математика: підручник / Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина.– Вид. 4-те, випр. і доп.- Львів: Магнолія -2006, 2016. – 432 с.
21. Берко А.Ю. Інформаційні технології бізнес-аналітики // А.Ю. Берко, Є.В. Буров, В.А. Висоцька : монографія – Львів: Видавництво «Новий Світ – 2000», 2022. – 520 с.
22. Григорович В. Г. Алгоритмізація та програмування. Частина 1: навчальний посібник / В. Г. Григорович. – Львів: Видавництво ПП «Магнолія 2006», Львів, 2022. – 312 с.

АНОТАЦІЯ

Бакум П.І., Кравець П.О. (керівник). Інформаційна система «Місто у смартфоні». – Національний університет “Львівська політехніка”, Львів, 2023.

Розширена анотація.

Використання міських мобільних програм – звичайна практика для жителів західноєвропейських країн, а також таких розвинених східних держав, як Японія та Південна Корея. Там люди встановлюють мобільні програми десятками – і це при тому, що за користування кожною доводиться платити. У великих містах активно впроваджують мобільний софт, заточений під міські реалії. Це стосується й міст України.

Основною проблемою на вирішення якої зорієнтовані такі програми є побудова оптимальних маршрутів, які би включали всі необхідні проміжні точки. Це дозволило, наприклад туристам зекономити час, та дало змогу оглянути більшу кількість визначних місць. Головною проблемою є побудова такого маршруту, який би включав всі точки лише один раз, за найкоротший шлях.

Ця задача відома з давніх часів, протягом років мінялись лише сфери її застосування. В період коли туризм не був розвинений вона була актуальною лише для логістичних та транспортних цілей. Та з підвищенням рівнем життя вона стала актуальною і у інших сферах, наприклад туристичній.

Мета дослідження є розроблення інформаційної системи «Місто у смартфоні», яка б з набору точок на карті будувала оптимальний шлях.

Об’єкт дослідження – процес побудови інформаційної системи «Місто у смартфоні».

Предмет дослідження – методи та засоби побудови інформаційної системи «Місто у смартфоні» для розроблення оптимальних туристичних маршрутів.

Результатом дослідження є програма яка будує та відображає на карті оптимальний маршрут на основі, заданих користувачем, параметрів.

Практичне значення одержаних результатів полягає в розробці Android додатку, використання якого зможу збільшити туристичну привабливість міста.

При виборі засобів для реалізації Android додатку було виявлено що оптимальним варіантом є засоби від компанії Google, тобто Android SDK та мова Java, для розмітки буде використовуватись XML. Було обрано готові рішення для відображення карти та маршрутів, а також використано деякі стандартні компоненти з Android SDK.

Ключові слова – інформаційна система, інформаційні технології, структурне моделювання, розумне місто.

ANNOTATION

Bakum P., Kravets P. (supervisor). Information system "City by smartphone".

Extended abstract.

The use of urban mobile applications is a common practice for residents of Western European countries, as well as such developed Eastern countries as Japan and South Korea. There, people install dozens of mobile applications - and this despite the fact that you have to pay for the use of each one. In large cities, they are actively implementing mobile software , honed to urban realities. This also applies to the cities of Ukraine.

The main problem that such programs are aimed at solving is the construction of optimal routes that would include all the necessary intermediate points. This allowed, for example, tourists to save time, and made it possible to see more places of interest. The main problem is to construct such a route that would include all points only once, by the shortest path.

This task has been known since ancient times, only the spheres of its application have changed over the years. In the period when tourism was not developed, it was relevant only for logistical and transport purposes. But with the increase in the standard of living, it has become relevant in other areas as well, such as tourism.

The purpose of the research is to develop an information system "City in a smartphone", which would build an optimal route from a set of points on the map.

The object of the study is the process of building the information system "City in a smartphone".

The subject of the research is the methods and means of building the information system "City in a smartphone" for the development of optimal tourist routes. The result of the research is a program that builds and displays the optimal route on the map based on parameters specified by the user.

The practical significance of the obtained results lies in the development of an Android application, the use of which will increase the tourist attractiveness of the city.

When choosing tools for the implementation of the Android application, it was found that the best option is the tools from Google, that is, the Android SDK and the Java

language, XML will be used for markup. Ready-made solutions for displaying the map and routes were chosen, and some standard components from the Android SDK were used.

Keywords: information system, information technology, structural modeling, smart city.

ДОДАТКИ

Додаток А. Програмний код

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.romanprodeus.lvivtourist">


    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />


    <uses-permission
android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission
android:name="android.permission.READ_PROFILE" />
    <uses-permission
android:name="android.permission.READ_CONTACTS" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">

        </activity>

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />
```

```
        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps" />
        <activity
            android:name=".LoginActivity"
            android:label="@string/title_activity_login">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
 />

                <category
 android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".FakeMapActivity" />
        <activity android:name=".FakeMap2Activity"></activity>
    </application>

 </manifest>
```

GmapV2Direction.java

```
package com.example.romanprodeus.lvivtourist;


import java.io.InputStream;
import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
```

```
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import com.google.android.gms.maps.model.LatLng;

import android.content.Context;
import android.util.Log;

public class GMapV2Direction {
    public final static String MODE_DRIVING = "driving";
    public final static String MODE_WALKING = "walking";

    public GMapV2Direction() {
    }

    public Document getDocument(LatLng start, LatLng end, String
mode) {
        String url =
"http://maps.googleapis.com/maps/api/directions/xml?"
                + "origin=" + start.latitude + "," +
start.longitude
                + "&destination=" + end.latitude + "," +
end.longitude
                + "&sensor=false&units=metric&mode=driving"
                + "&key=AIzaSyBANtRA5Y-6WctTKpBrqUydLJ39vlN3UHo";
        Log.d("url", url);
        try {
            HttpClient httpClient = new DefaultHttpClient();
            HttpContext localContext = new BasicHttpContext();
            HttpPost httpPost = new HttpPost(url);
```

```
            HttpResponse response = httpClient.execute(httpPost,
localContext);
            InputStream in = response.getEntity().getContent();
            DocumentBuilder builder =
DocumentBuilderFactory.newInstance()
                    .newDocumentBuilder();
            Document doc = builder.parse(in);
            return doc;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public String getDurationText(Document doc) {
        try {

            NodeList nl1 = doc.getElementsByTagName("duration");
            Node node1 = nl1.item(0);
            NodeList nl2 = node1.getChildNodes();
            Node node2 = nl2.item(getNodeIndex(nl2, "text"));
            Log.i("DurationText", node2.getTextContent());
            return node2.getTextContent();
        } catch (Exception e) {
            return "0";
        }
    }

    public int getDurationValue(Document doc) {
        try {
            NodeList nl1 = doc.getElementsByTagName("duration");
            Node node1 = nl1.item(0);
            NodeList nl2 = node1.getChildNodes();
            Node node2 = nl2.item(getNodeIndex(nl2, "value"));
```

```
            Log.i("DurationValue", node2.getTextContent());
            return Integer.parseInt(node2.getTextContent());
        } catch (Exception e) {
            return -1;
        }
    }

    public String getDistanceText(Document doc) {
        /*
         * while (en.hasMoreElements()) { type type = (type)
en.nextElement();
         *
         * }
         */

        try {
            NodeList nl1;
            nl1 = doc.getElementsByTagName("distance");

            Node node1 = nl1.item(nl1.getLength() - 1);
            NodeList nl2 = null;
            nl2 = node1.getChildNodes();
            Node node2 = nl2.item(getNodeIndex(nl2, "value"));
            Log.d("DistanceText", node2.getTextContent());
            return node2.getTextContent();
        } catch (Exception e) {
            return "-1";
        }


    }

    public int getDistanceValue(Document doc) {
        try {
```

```
            NodeList nl1 = doc.getElementsByTagName("distance");
            Node node1 = null;
            node1 = nl1.item(nl1.getLength() - 1);
            NodeList nl2 = node1.getChildNodes();
            Node node2 = nl2.item(getNodeIndex(nl2, "value"));
            Log.i("DistanceValue", node2.getTextContent());
            return Integer.parseInt(node2.getTextContent());
        } catch (Exception e) {
            return -1;
        }

    }

    public String getStartAddress(Document doc) {
        try {
            NodeList nl1 =
doc.getElementsByTagName("start_address");
            Node node1 = nl1.item(0);
            Log.i("StartAddress", node1.getTextContent());
            return node1.getTextContent();
        } catch (Exception e) {
            return "-1";
        }

    }

    public String getEndAddress(Document doc) {
        try {
            NodeList nl1 =
doc.getElementsByTagName("end_address");
            Node node1 = nl1.item(0);
            Log.i("StartAddress", node1.getTextContent());
            return node1.getTextContent();
        } catch (Exception e) {
```

```
            return "-1";
        }
    }
    public String getCopyRights(Document doc) {
        try {
            NodeList nl1 = doc.getElementsByTagName("copyrights");
            Node node1 = nl1.item(0);
            Log.i("CopyRights", node1.getTextContent());
            return node1.getTextContent();
        } catch (Exception e) {
            return "-1";
        }

    }

    public ArrayList<LatLng> getDirection(Document doc) {
        NodeList nl1, nl2, nl3;
        ArrayList<LatLng> listGeopoints = new ArrayList<LatLng>();
        nl1 = doc.getElementsByTagName("step");
        if (nl1.getLength() > 0) {
            for (int i = 0; i < nl1.getLength(); i++) {
                Node node1 = nl1.item(i);
                nl2 = node1.getChildNodes();

                Node locationNode = nl2
                        .item(getNodeIndex(nl2,
"start_location"));
                nl3 = locationNode.getChildNodes();
                Node latNode = nl3.item(getNodeIndex(nl3, "lat"));
                double lat =
Double.parseDouble(latNode.getTextContent());
                Node lngNode = nl3.item(getNodeIndex(nl3, "lng"));
                double lng =
Double.parseDouble(lngNode.getTextContent());
```

```
                listGeopoints.add(new LatLng(lat, lng));

                locationNode = nl2.item(getNodeIndex(nl2,
"polyline"));
                nl3 = locationNode.getChildNodes();
                latNode = nl3.item(getNodeIndex(nl3, "points"));
                ArrayList<LatLng> arr =
decodePoly(latNode.getTextContent());
                for (int j = 0; j < arr.size(); j++) {
                    listGeopoints.add(new
LatLng(arr.get(j).latitude, arr
                            .get(j).longitude));
                }

                locationNode = nl2.item(getNodeIndex(nl2,
"end_location"));
                nl3 = locationNode.getChildNodes();
                latNode = nl3.item(getNodeIndex(nl3, "lat"));
                lat =
Double.parseDouble(latNode.getTextContent());
                lngNode = nl3.item(getNodeIndex(nl3, "lng"));
                lng =
Double.parseDouble(lngNode.getTextContent());
                listGeopoints.add(new LatLng(lat, lng));
            }
        }

        return listGeopoints;
    }

    private int getNodeIndex(NodeList nl, String nodename) {
        for (int i = 0; i < nl.getLength(); i++) {
            if (nl.item(i).getNodeName().equals(nodename))
                return i;
```

```
        }
        return -1;
    }

    private ArrayList<LatLng> decodePoly(String encoded) {
        ArrayList<LatLng> poly = new ArrayList<LatLng>();
        int index = 0, len = encoded.length();
        int lat = 0, lng = 0;
        while (index < len) {
            int b, shift = 0, result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlat = ((result & 1) != 0 ? ~(result >> 1) :
(result >> 1));
            lat += dlat;
            shift = 0;
            result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;
                shift += 5;
            } while (b >= 0x20);
            int dlng = ((result & 1) != 0 ? ~(result >> 1) :
(result >> 1));
            lng += dlng;

            LatLng position = new LatLng((double) lat / 1E5,
(double) lng / 1E5);
            poly.add(position);
        }
        return poly;
```

```
        }
    }
```

Додаток Б. Програмний код класів

```
public class MainFragment extends Fragment {

    private OnFragmentInteractionListener mListener;
    private RecyclerView mRecyclerView;
    private Adapter mAdapter;

    public MainFragment() {

    }

    public static MainFragment newInstance() {
        MainFragment fragment = new MainFragment();
        Bundle args = new Bundle();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
        }
    }

    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {

        View view =  inflater.inflate(R.layout.fragment_main,
container, false);

        mRecyclerView = (RecyclerView)
view.findViewById(R.id.recycler_view);


        mRecyclerView.setHasFixedSize(true);

        LinearLayoutManager LayoutManager = new
LinearLayoutManager(getContext());
        mRecyclerView.setLayoutManager(LayoutManager);

        mAdapter = new Adapter(getPoints());
        mRecyclerView.setAdapter(mAdapter);

        FloatingActionButton floatingActionButton =
```

```
(FloatingActionButton) view.findViewById(R.id.next);

floatingActionButton.setImageDrawable(getContext().getDrawable(R.d
rawable.ic_navigation_white_48dp));
        floatingActionButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

mListener.onNextClicked(mAdapter.getSelectedItems());
            }
        });

        return view;
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof OnFragmentInteractionListener) {
            mListener = (OnFragmentInteractionListener) context;
        } else {
            throw new RuntimeException(context.toString()
                    + " must implement
OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }


    public interface OnFragmentInteractionListener {
        void onNextClicked(List<Point> points);
    }
```

Код для розміщення елементів у створеному фрагменті:

```
   <?xml version="1.0" encoding="utf-8"?>
     <FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"

    tools:context="com.example.romanprodeus.lvivtourist.MainActivit
    y">
```

```
        <android.support.v7.widget.RecyclerView
            android:id="@+id/recycler_view"
            android:scrollbars="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent"/>

        <android.support.design.widget.FloatingActionButton
            android:id="@+id/next"
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:layout_margin="20dp"
            android:clickable="true"
            android:layout_gravity="bottom|right"                    />

    </FrameLayout>



public class DijkstraAlgorithm {

    private final List<Vertex> nodes;
    private final List<Edge> edges;
    private Set<Vertex> settledNodes;
    private Set<Vertex> unMarkedNodesList;
    private Map<Vertex, Vertex> previousList;
    private Map<Vertex, Integer> distance;

    public DijkstraAlgorithm(Graph graph) {
        this.nodes = new ArrayList<Vertex>(graph.getVertexes());
        this.edges = new ArrayList<Edge>(graph.getEdges());
    }

    public void execute(Vertex source) {
        markedNodesList = new HashSet<Vertex>();
        unMarkedNodesList = new HashSet<Vertex>();
        distance = new HashMap<Vertex, Integer>();
        previousList = new HashMap<Vertex, Vertex>();
        distance.put(source, 0);
        unMarkedNodesList.add(source);
        while (unMarkedNodesList.size() > 0) {
            Vertex node = getMinimum(unMarkedNodesList);
            markedNodesList.add(node);
            unMarkedNodesList.remove(node);
            findMinimalDistances(node);
        }
    }

    private void findMinimalDistances(Vertex node) {
        List<Vertex> adjacentNodes = getNeighbors(node);
        for (Vertex target : adjacentNodes) {
            if (getShortestDistance(target) >
```

```
getShortestDistance(node)
                    + getDistance(node, target)) {
                distance.put(target, getShortestDistance(node)
                        + getDistance(node, target));
                previousList.put(target, node);
                unMarkedNodesList.add(target);
            }
        }

    }

    private int getDistance(Vertex node, Vertex target) {
        for (Edge edge : edges) {
            if (edge.getSource().equals(node)
                    && edge.getDestination().equals(target)) {
                return edge.getWeight();
            }
        }
        throw new RuntimeException("Should not happen");
    }

    private List<Vertex> getNeighbors(Vertex node) {
        List<Vertex> neighbors = new ArrayList<Vertex>();
        for (Edge edge : edges) {
            if (edge.getSource().equals(node)
                    && !isSettled(edge.getDestination())) {
                neighbors.add(edge.getDestination());
            }
        }
        return neighbors;
    }

    private Vertex getMinimum(Set<Vertex> vertexes) {
        Vertex minimum = null;
        for (Vertex vertex : vertexes) {
            if (minimum == null) {
                minimum = vertex;
            } else {
                if (getShortestDistance(vertex) <
getShortestDistance(minimum)) {
                    minimum = vertex;
                }
            }
        }
        return minimum;
    }

    private boolean isSettled(Vertex vertex) {
        return markedNodesList.contains(vertex);
    }
```

```
    private int getShortestDistance(Vertex destination) {
        Integer d = distance.get(destination);
        if (d == null) {
            return Integer.MAX_VALUE;
        } else {
            return d;
        }
    }


    public LinkedList<Vertex> getPath(Vertex target) {
        LinkedList<Vertex> path = new LinkedList<Vertex>();
        Vertex step = target;

        if (previousList.get(step) == null) {
            return null;
        }
        path.add(step);
        while (previousList.get(step) != null) {
            step = previousList.get(step);
            path.add(step);
        }

        Collections.reverse(path);
        return path;
    }

}
```

4.1.7. Опис реалізації алгоритму пошуку оптимального шляху

Задача комівояжера має може бути розвязана з допомогою декількох алгоритмів. Оптимальним для реалізації було вибрано метод k-найближчих сусідів.

Реалізація розвязу задачі комівояжера на мові Java:

```
public class OptimalPath
{
    private int numberOfNodes;
    private Stack<Integer> stack;


    public OptimalPath()
    {
        stack = new Stack<Integer>();
    }
```

```
    public void tsp(int adjacencyMatrix[][])
    {
        numberOfNodes = adjacencyMatrix[1].length - 1;
        int[] visited = new int[numberOfNodes + 1];
        visited[1] = 1;
        stack.push(1);
        int element, dst = 0, i;
        int min = Integer.MAX_VALUE;
        boolean minFlag = false;
        System.out.print(1 + "\t");

        while (!stack.isEmpty())
        {
            element = stack.peek();
            i = 1;
            min = Integer.MAX_VALUE;
            while (i <= numberOfNodes)
            {
                if (adjacencyMatrix[element][i] > 1 && visited[i]
== 0)
                {
                    if (min > adjacencyMatrix[element][i])
                    {
                        min = adjacencyMatrix[element][i];
                        dst = i;
                        minFlag = true;
                    }
                }
                i++;
            }
            if (minFlag)
            {
                visited[dst] = 1;
                stack.push(dst);
```

```
                System.out.print(dst + "\t");
                minFlag = false;
                continue;
            }
            stack.pop();
        }
    }

    public static void main(String... arg)
    {
        int number_of_nodes;
        Scanner scanner = null;
        try
        {
            System.out.println("Enter the number of nodes in the
graph");
            scanner = new Scanner(System.in);
            number_of_nodes = scanner.nextInt();
            int adjacency_matrix[][] = new int[number_of_nodes +
1][number_of_nodes + 1];
            System.out.println("Enter the adjacency matrix");
            for (int i = 1; i <= number_of_nodes; i++)
            {
                for (int j = 1; j <= number_of_nodes; j++)
                {
                    adjacency_matrix[i][j] = scanner.nextInt();
                }
            }
            for (int i = 1; i <= number_of_nodes; i++)
            {
                for (int j = 1; j <= number_of_nodes; j++)
                {
                    if (adjacency_matrix[i][j] == 1 &&
adjacency_matrix[j][i] == 0)
```

```
                    {
                        adjacency_matrix[j][i] = 1;
                    }
                }
            }
            System.out.println("the citys are visited as
follows");
            OptimalPath OptimalPath = new OptimalPath();
            OptimalPath.tsp(adjacency_matrix);
        } catch (InputMismatchException inputMismatch)
        {
            System.out.println("Wrong Input format");
        }
        scanner.close();
    }
}
```