Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
Інститут електроенергетики
(інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра
(бакалавра, спеціаліста, магістра)

студента Клепікова Данііла Руслановича
(ПІБ)
академічної групи 126-19-1
(шифр)
спеціальності 126 «Інформаційні системи та технології
(код і назва спеціальності)
за освітньо-професійною програмою _____
(за наявності)
«Інформаційні системи та технології»
(офіційна назва)

на тему Розробка інформаційної системи логістики станцій зарядки електромобілів у місті Дніпро на мові Python
(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	проф. Олевський В.І.			
розділів:				

Рецензент				

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
_____інформаційних технологій _____
_____та комп'ютерної інженерії_____
(повна назва)
_____ _Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«_____»_____2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____бакалавра _____
(бакалавра, спеціаліста, магістра)

студенту_____Клепікову Д.Р._____ **академічної групи** _____126-19-1____
(прізвище та ініціали) (шифр)
спеціальності _____126 «Інформаційні системи та технології»_____

за освітньою-професійною програмою _____

(за наявності)
_____«Інформаційні системи та технології»_____

на тему ___Розробка інформаційної системи логістики станцій зарядки електромобілів у місті Дніпро на мові Python_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 16.05.2023 №350-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз теми та постановка задачі	
Розділ 2	Огляд методів автоматизованного управління транспортними потоками	
Розділ 3	Експерементальна частина. Підготовка матеріалів для захисту роботи	

Завдання видано _____ ____В.І. Олевський___
(підпис керівника) (прізвище, ініціали)

Дата видачі _____

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ _____Клепіков Д.Р._____

(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 85 стор., 32 рис., 2 табл., 2 додатки, 21 джерело.

Об'єкт розробки: інформаційна система логістики станцій зарядки електромобілів у місті Дніпро.

Мета кваліфікаційної роботи: розробка зручної інформаційної системи логістики для пошуку найближчої станції електрозарядки.

У вступі наведено стан проблеми та обґрунтована її актуальність.

Перший розділ включає в себе постановку завдання і характеристику предметної області.

У другому розділі був проведений аналіз і вибір програмних засобів для реалізації поставленого завдання. Проведене моделювання транспортної логістики.

У третьому розділі створене програмне забезпечення, яке дозволяє розраховувати короткий маршрут до станцій зарядки електромобілів.

Практичне значення роботи полягає у створенні програмного забезпечення, яке б дозволило розраховувати короткий маршрут до станцій зарядки електромобілів та забезпечувати комфорт користувачів.

Список ключових слів: ЕЛЕКТРОМОБІЛЬ, ЗАРЯДНА СТАНЦІЯ, ЛОГІСТИКА, ІНФОРМАЦІЙНА СИТЕМА, HTML, CSS, MYSQL, PHP, UML

ABSTRACT

Explanatory note: 85 pages, 22 figures, 2 tables, 2 appendices, 21 sources.

The object of development: an information system for the logistics of electric vehicle charging stations in the city of Dnipro.

The purpose of the thesis: is to develop a convenient logistics information system for finding the nearest electric charging station.

The introduction describes the state of the problem and substantiates its relevance.

The first chapter includes the task statement and the characterization of the subject area.

In the second section, we analyze and select software tools for solving the problem. Modeling of transport logistics was carried out.

In the third section, software was created that allows calculating a short route to electric vehicle charging stations.

The practical significance of the work is to create software that would allow calculating a short route to electric vehicle charging stations and ensure user comfort.

Keywords: ELECTRIC CAR, CHARGING STATION, LOGISTICS, INFORMATION SYSTEM, HTML, CSS, MYSQL, PHP, UML

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ТЗ – транспортні засоби;

АТП – автотранспортного підприємства;

HTML - (HyperText Markup Language – мова розмітки гіпертексту) – стандартна мова розмітки документів у Всесвітній павутині;

JavaScript - об'єктно-орієнтована скриптова мова програмування;

MYSQL – вільна система управління базами даних;

CSS - Cascading Styling Sheet – це каскадна таблиця стилів;

PHP – Hypertext Preprocessor - «препроцесор гіпертексту», інструменти для створення персональних веб-сторінок;

БД – база даних;

СУБД – система управління базами даних;

IDEF – Information Modeling – методологія моделювання інформаційних потоків;

UML – Universal Modeling language – це у ніверсальна мова моделювання;

URL – Uniform Resource Locator – це у ніфікований покажчик інформаційного ресурс.

.

ВСТУП

Актуальність роботи. На даний момент, рівень розвитку електротранспорту в Україні можна оцінити як низький. У більшості міст України електротранспорт представлений в основному тролейбусами та трамваями, а електромобілі ще не є популярними серед населення.

У 2020 році було запущено перший електробус на маршруті в Києві, а також було запроваджено електропотяги на деяких залізничних маршрутах. Однак, загальна кількість електротранспорту в Україні є незначною порівняно з країнами Європи та світу.

Проте, уряд України оголосив про свій намір сприяти розвитку електротранспорту, в тому числі шляхом створення інфраструктури для зарядки електромобілів та підтримки виробництва електротранспорту в країні. Також запровадження спеціальних пільг та державних програм може допомогти зростанню популярності електротранспорту серед населення.

Об'єктом дослідження є методи створення інформаційної системи логістики станцій зарядки електромобілів.

Метою роботи є розробка інформаційної системи логістики станцій зарядки електромобілів у місті Дніпро на мові Python, для полегшення приняття рішення потенційним клієнтом.

1 АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1 Рівень розвитку і популяризації електромобілів в Україні

За даними Асоціації «Українська Асоціація електромобілів» (УкрАЕМ), статистика розвитку та популяризації електромобілів в Україні на початок 2023 року є наступною:

1. Загальна кількість електромобілів в Україні на 1 січня 2023 року становить 10882 одиниці. Найбільшою кількістю електромобілів володіє Київ (3209 одиниць), далі за ним йдуть Одеса (1344 одиниці) та Львів (1003 одиниці). У порівнянні з попереднім роком, загальна кількість електромобілів збільшилася на 62%.
2. За 2022 рік українці купили 5298 нових електромобілів, що є на 118% більше ніж у 2021 році. Найбільшу кількість нових електромобілів купили у Києві (2142 одиниці), далі за ним йдуть Одеса (795 одиниць) та Львів (443 одиниці). Найбільш популярними моделями електромобілів є Tesla Model 3, Hyundai Kona Electric та Volkswagen ID.4, [1].
3. За даними УкрАЕМ, на 1 січня 2023 року українські власники електромобілів проїхали більше 111 млн км, що в середньому складає більше 10 000 км на один електромобіль.
4. Найбільш популярним типом електромобілів в Україні є моделі з батареєю ємністю 60-90 кВтгод, що дозволяють проїхати близько 300-400 км на одному заряді. Також українці показали інтерес до моделей з більш великою ємністю батареї, що дозволяють проїхати понад 500 км на одному заряді.

На жаль, на сьогоднішній день вартість електромобілів в Україні є значною перепоною для їх широкого поширення. У 2022 році середня вартість нового електромобіля в Україні складала близько 37000 євро, що дуже високо порівняно зі середнім рівнем доходів населення.

У зв'язку з цим, уряд України продовжує працювати над стимулюванням попиту на електромобілі та підтримкою розвитку інфраструктури для зарядки. У рамках програми «Електромобільна Україна» передбачено встановлення до

2024 року не менше 10000 зарядних станцій та надання державної підтримки для покупки щонайменше 14000 електромобілів, [2].

На додаток до цього, уряд України розробив план дій щодо підвищення енергоефективності транспортних засобів, який передбачає створення сприятливих умов для розвитку електромобільності, зокрема, підвищення ефективності та використання альтернативних джерел енергії в транспорті.

Нарешті, національні та міжнародні екологічні та кліматичні вимоги та зобов'язання, які взяла на себе Україна, також стимулюють розвиток електромобільності в країні. Уряд України планує зменшити викиди вуглецю в атмосферу на 65% до 2030 року, що вимагає значних зусиль у розвитку екологічно чистих технологій, включаючи електромобільність.

Україна також активно співпрацює з міжнародними партнерами щодо розвитку електромобільності. Наприклад, українська компанія «Електрокар Україна» співпрацює з компанією «Байєріше Моторрад» з Німеччини, щоб випускати в Україні електромобілі BMW i3 та BMW iX3.

Зокрема, у серпні 2021 року компанії оголосили про підписання угоди про співпрацю на 10 років, в рамках якої «Електрокар Україна» буде забезпечувати продаж, сервіс та ремонт електромобілів BMW в Україні, а також займатись розробкою інфраструктури зарядки.

Крім того, українська компанія «KNESS Group» планує спільно з китайською компанією BYD створити в Україні виробництво електробусів та електротаксі. Передбачається, що перші автомобілі будуть випущені вже у 2023 році.

Водночас, на даний момент український ринок електромобілів все ще має деякі виклики, такі як недостатньо розвинена інфраструктура зарядки та висока вартість автомобілів. Однак, зусилля уряду та приватних компаній спрямовані на подолання цих перешкод та розвиток електромобільності в Україні.

У майбутньому можна очікувати подальшого зростання популярності електромобілів в Україні, зокрема, завдяки збільшенню кількості моделей на

ринку та зниженню їх вартості, покращенню інфраструктури зарядки, підвищенню екологічних та енергоефективних вимог.

1.2 Роль розташування та кількоті зарядних станцій в розвитку електромобільності України

Розташування та кількість зарядних станцій відіграють важливу роль у розвитку електромобільності в Україні, оскільки є вирішальними факторами для збільшення кількості електромобілів на дорогах. Доступність зарядних станцій забезпечує зручність користувачам електромобілів та зменшує їхню тривогу щодо досить обмеженого запасу ходу батареї. Крім того, розташування зарядних станцій може стимулювати попит на електромобілі в різних регіонах, що в свою чергу може прискорити розвиток електромобільності у всій країні. Також важливим є розвиток технологій зарядки, які дозволяють швидко та ефективно заряджати батареї електромобілів, забезпечуючи зручність та швидкість заправки.

На даний момент, кількість зарядних станцій для електромобілів в Україні значно менша, ніж у розвинених країнах. За даними порталу PlugShare, станом на квітень 2023 року, в Україні працює близько 1700 зарядних станцій для електромобілів, в той час як у найбільш розвинених країнах світу їх кількість може досягати десятків тисяч, [3].

Більшість зарядних станцій розташовані у крупних містах, таких як Київ, Львів, Одеса, Харків та Дніпро, а також на головних автомагістралях. Зокрема, у Дніпрі є кілька мереж зарядних станцій, таких як:

- Tesla Supercharger Network - мережа швидких зарядних станцій, призначених для власників електромобілів Tesla. У Дніпрі є одна така станція на території ТРЦ «Karavan».

- Green Fuel - це мережа зарядних станцій, яка налічує вже більше 40 станцій по всій Україні. У Дніпрі також є кілька станцій цієї мережі, в тому числі на території ТРЦ «Most City».

- Electro-City - мережа зарядних станцій, яка працює в Дніпрі з 2017 року. У мережі є більше 10 станцій, розташованих по всьому місту.

- ElectroUA - це мережа зарядних станцій, яка налічує більше 100 станцій по всій Україні, у тому числі декілька станцій у Дніпрі.

- Eneba - це мережа зарядних станцій, яка працює в Україні з 2018 року. У Дніпрі є одна станція мережі, розташована на вулиці Космічній.

Ці мережі зарядних станцій допомагають зробити електромобільність більш доступною для мешканців та гостей міста Дніпро.

Також на популярних автомагістралях встановлені зарядні станції від компаній «GreenFuel», «EcoFactor» та «Smart Energy».

Уряд України та приватні компанії активно займаються розвитком інфраструктури зарядки для електромобілів. Наприклад, уряд України розробляє національну стратегію розвитку електромобільності, яка передбачає підтримку створення мереж зарядних станцій та сприяння розвитку електромобільного транспорту в країні. Крім того, такі компанії, як «Укрзалізниця» та «Трансінвестхолдинг», також встановлюють зарядні станції для електромобілів на своїх територіях.

Загалом, розвиток інфраструктури зарядки є важливою складовою успішного впровадження електромобілів в Україні.

За останні кілька років кількість зарядних станцій у країні значно зросла, але все ще не відповідає потребам ринку. Для стимулювання розвитку інфраструктури зарядки, уряд України запровадив декілька програм та пільг, які надають фінансову підтримку для установки зарядних станцій.

Однією з таких програм є «Green Loan» від Фонду екологічних інвестицій, який надає фінансову підтримку для установки зарядних станцій для електромобілів, [4]. Крім того, згідно з податковим законодавством України, компанії, які встановлюють зарядні станції, можуть скористатися пільгами при сплаті податку на нерухомість.

Також, уряд України встановив ціль по розширенню мережі зарядних станцій у країні до 2030 року. Згідно з цією стратегією, уряд планує збільшити кількість зарядних станцій до 20 тисяч, а також встановити зарядні станції на кожній автозаправці в країні. Це допоможе забезпечити зручну

інфраструктуру для електромобілів, збільшити кількість їх користувачів і зменшити залежність від нафтопродуктів.

У цілому, розвиток електротранспорту та інфраструктури зарядки в Україні є перспективним напрямом розвитку транспортної системи країни, що дозволить зменшити забруднення довкілля, покращити якість повітря та знизити витрати на пальне.

Логістика і розташування зарядних станцій є важливими аспектами розвитку інфраструктури зарядки для електромобілів. Розташування зарядних станцій повинно враховувати рух транспорту та його маршрути, а також потреби транспортних підприємств та приватних власників електромобілів.

У цьому питанні рішення можуть бути засновані на різних критеріях. Наприклад, зарядні станції можуть бути розташовані на основних магістральних маршрутах, де їх буде найбільше використовувати транспортний потік. Також, зарядні станції можуть бути розташовані на парковках біля торгових центрів, готелів, ресторанів та інших об’єктів громадського харчування.

Крім того, розташування зарядних станцій повинно враховувати потреби транспортних підприємств. Залежно від специфіки транспортного бізнесу, зарядні станції можуть бути розташовані на автобазах або на територіях, де знаходяться транспортні компанії та логістичні центри.

Щодо логістики, то зарядні станції повинні бути взаємопов’язані і логічно розміщені на території країни. Для цього потрібно розробити мережу зарядних станцій, яка б враховувала розташування транспортної інфраструктури, дозволяла максимально оптимізувати маршрути зарядки та забезпечувала найефективніший доступ до зарядних станцій для користувачів електромобілів.

Важливою складовою логістики зарядних станц цій є також встановлення стандартів для зарядних станцій, щоб вони мали однакову фізичну і програмну сумісність і забезпечували безпечну та швидку зарядку для різних моделей електромобілів. Такі стандарти включають в себе, наприклад, стандарти для

підключення зарядних станцій до електромережі, типи розеток та кабелів, стандарти комунікації між зарядними станціями та електромобілями та інші.

З метою покращення доступу до зарядних станцій для користувачів електромобілів, в Україні було запроваджено кілька програм підтримки встановлення зарядних станцій. Наприклад, Програма «Зелений тариф» передбачає компенсацію вартості зарядних станцій для фізичних та юридичних осіб. Також, програма «Європейський фонд зеленої енергії для України» надає фінансову підтримку для встановлення зарядних станцій на території України.

Однак, на сьогоднішній день, кількість зарядних станцій в Україні все ще не достатня для повноцінного розвитку електромобільного транспорту. Наприклад, на початку 2021 року в Україні було всього близько 2000 зарядних станцій, що є значно меншою кількістю порівняно з Європейськими країнами.

Таким чином, розвиток інфраструктури зарядки є важливим завданням для подальшого розвитку електромобільного транспорту в Україні. Для цього необхідно продовжувати впроваджувати програми підтримки встановлення зарядних станцій, розробляти мережу зарядних станцій, встановлювати стандарти для забезпечення їх сумісності та визначати оптимальні місця розташування зарядних станцій на базі аналізу попиту та існуючої інфраструктури. Крім того, можна сприяти розвитку мережі зарядних станцій шляхом залучення приватних інвестицій та стимулювання використання сонячних панелей та інших джерел відновлювальної енергії для живлення зарядних станцій.

Розташування зарядних станцій відіграє важливу роль у популяризації електромобілів серед населення. Оптимальне розташування зарядних станцій має бути узгоджене з попитом на транспортні послуги в конкретній місцевості. Наприклад, в містах з великою кількістю жителів та високою щільністю населення необхідно розташовувати зарядні станції у більшості мікрорайонів, щоб забезпечити зручний доступ для користувачів електромобілів.

Крім того, зарядні станції повинні бути розташовані у місцях з високою інтенсивністю транспортного руху, таких як вузькі вулиці та торгові центри. Це забезпечить забезпечення зручної та швидкої зарядки для користувачів електромобілів, що перебувають у русі.

Розташування зарядних станцій може бути обумовлене наявністю існуючої інфраструктури, такої як електричні мережі та інші об'єкти. Наприклад, в будівлях офісів, торгових центрів та готелів необхідно встановлюват ити зарядні станції для забезпечення користувачів електромобілів, які відвідують ці об'єкти. Крім того, у віддалених від населених пунктів територіях можуть бути встановлені зарядні станції на автостоянках, на автодорогах або біля заправних станцій.

Розташування зарядних станцій повинно відповідати всім вимогам безпеки та надійності. Зарядні станції мають бути виконані згідно зі стандартами та міжнародними нормами безпеки, щоб уникнути пожеж та інших небезпек. Зарядні станції повинні мати механізми захисту від перенапруги та перевантаження, щоб забезпечити безпеку користувачів електромобілів.

Наразі в Україні існують різні види зарядних станцій: швидкі зарядні станції, повільні зарядні станції та зарядні станції для домашнього використання. Швидкі зарядні станції забезпечують зарядку електромобіля за короткий час, але вони вимагають більшої потужності інфраструктури та відповідного розташування. Повільні зарядні станції не вимагають великої потужності та можуть бути встановлені більш широко, але зарядка електромобіля може зайняти значно більше часу. Зарядні станції для домашнього використання забезпечують зручну зарядку електромобіля вдома, але не можуть задовольнити потреби користувачів у зарядці в місцях загального користування.

Усі види зарядних станцій можуть бути використані для розвитку і популяризації електромобілів в Україні. Однак, розташування та кількість

зарядних станцій є важливим фактором для забезпечення зручності та доступності зарядки електромобілів.

Для забезпечення ефективної логістики та розташування зарядних станцій в Україні, можна використовувати геоінформаційні системи та аналіз даних про рух електромобілів та їх власників. Це дозволить визначити найбільш популярні маршрути та місця, де потрібні зарядні станції, а також оцінити потужність та кількість зарядних станцій, які необхідні для забезпечення покриття даної території.

Крім того, важливим фактором є розумна інтеграція зарядних станцій з іншими елементами інфраструктури, такими як мережі громадського транспорту, автомагістралі та автостоянки. Це дозволить забезпечити зручний доступ до зарядних станцій та підтримувати ефективну роботу міського транспорту та автотранспорту загалом.

На сьогоднішній день в Україні існує декілька десятків компаній, які займаються встановленням та експлуатацією зарядних станцій для електромобілів. Деякі з цих компаній працюють на ринку зарядних станцій вже кілька років і мають розвинену мережу станцій по всій країні. Другі ж тільки починають розвиватись на цьому ринку.

Загалом, розвиток інфраструктури зарядних станцій є важливим фактором для розвитку електромобільності в Україні.

1.3 Поняття інформаційної системи для електростанцій

Інформаційна система (ІС) для електростанцій є набором програм, додатків та апаратних засобів, які використовуються для збору, зберігання, обробки та передачі даних про електростанцію. Основна мета ІС для електростанцій - забезпечення ефективного функціонування та підвищення ефективності управління електростанцією.

ІС для електростанцій може містити різноманітні модулі, такі як система моніторингу та діагностики, система управління енергопостачанням, система автоматичного регулювання, система управління ремонтами та

обслуговуванням, система контролю якості електроенергії, система управління взаємодією з електромережами тощо.

На рис. 1.1 подано спрощену схему потоків інформаційної логістики, що показує проходження основних інформаційних потоків, необхідних для функціонування підприємства, які мають зарядні станції.

Рис.1.1. Схема потоків інформаційної логістики

Для ефективного функціонування ІС для електростанцій використовуються різні методи та моделі, такі як системи підтримки прийняття рішень, математичне моделювання, оптимізаційні методи, теорія черг, аналіз відмов та інші. Використання таких методів дозволяє ефективно вирішувати завдання з оптимізації виробництва електроенергії, планування обслуговування та ремонту, виявлення та усунення несправностей тощо.

Одним з важливих аспектів ІС для електростанцій є забезпечення безпеки та захисту інформації, що збирається та обробляється. Для цього використовуються різноманітні технічні та організаційні заходи, такі як

шифрування даних, забезпечення контролю доступу, системи моніторингу та виявлення вторгнень тощо.

Для ІС дотримуються таких принципів їх побудови і функціонування [5, 6]:

1. Відповідність. ІС повинна забезпечувати функціонування об'єкта із заданою ефективністю. Критерій ефективності повинен бути кількісним.

2. Економічність. Витрати на обробку інформації в ІС повинні бути менші за економічний виграш на об'єкті при використанні цієї інформації.

3. Регламент. Велика частина інформації в ІС надходить і обробляється за розкладом, із строгою періодичністю.

4. Самоконтроль. Безперервна робота ІС з виявлення і виправлення помилок у даних і процесах їх обробки.

5. Інтегральність. Одноразове введення інформації в ІС і її багатократне, багатоцільове використання.

6. Адаптивність. Здатність ІС змінювати свою структуру і закон поведінки для досягнення оптимального результату за зовнішніх умов, що змінюються.

Серед інших особливостей ІС слід назвати обробку великих обсягів інформації за порівняно простими алгоритмами, високу питому вагу логічної обробки даних (сортування, групування, пошук, корегування) і подання переважної частини інформації у вигляді документів.

Для оцінки ефективності ІС служить набір критеріїв, які кількісно визначають ступінь відповідності системи цілям її створення. Критерій ефективності повинен бути наочним, безпосередньо залежати від роботи системи, допускати наближену оцінку за наслідками експериментів. Оцінюють як ІС в цілому, так і її компоненти. Одночасне досягнення всіх цілей неможливе, тому на практиці вибирають компромісне рішення: один з критеріїв оптимізується, а інші виступають як обмеження.

Інформаційні системи (ІС) мають дотримуватися певних принципів, які визначають їх будову і функціонування. Основні принципи включають [5]:

– відповідність;

– економічність;

– регламент;

– самоконтроль;

– інтегральність;

– адаптивність.

Критерій ефективності повинен бути кількісним, а витрати на обробку інформації в ІС мають бути менші за економічний виграш на об'єкті при використанні цієї інформації. Інформація в ІС надходить і обробляється за розкладом із строгою періодичністю, а ІС здатна здійснювати безперервну роботу з виявленням і виправленням помилок у даних та їх обробці. ІС забезпечує одноразове введення інформації та багатократне, багатоцільове використання. Інші особливості ІС включають обробку великих обсягів інформації за порівняно простими алгоритмами, високу питому вагу логічної обробки даних та подання переважної частини інформації у вигляді документів.

Для оцінки ефективності ІС використовуються кількісні критерії, які визначають ступінь відповідності системи її цілям. Критерій ефективності має бути наочним, допускати наближену оцінку за наслідками експериментів та залежати від роботи системи. ІС оцінюються в цілому, а також її компоненти. Одночасне досягнення всіх цілей неможливе, тому на практиці вибирають компромісне рішення.

1.4 Аналіз існуючих інформаційних систем логістики станцій зарядки електромобілів

Інформаційні системи логістики станцій зарядки електромобілів в Україні зазвичай забезпечують необхідну інформацію для водіїв електромобілів, щоб допомогти їм знайти та використовувати зарядні станції.

На сьогоднішній день в Україні існує кілька таких інформаційних систем, які допомагають водіям знайти зарядні станції та контролювати процес зарядки.

До найбільш поширених інформаційних систем логістики станцій зарядки електромобілів, що використовуються в Україні, належать:

1. PlugShare (рис.1.2)- це онлайн-карта зарядних станцій, яка охоплює всі зарядні станції у світі, включаючи ті, що знаходяться в Україні. PlugShare дозволяє користувачам знайти найближчу до них зарядну станцію, переглядати її рейтинг та відгуки інших користувачів, а також розраховувати вартість зарядки, [3].

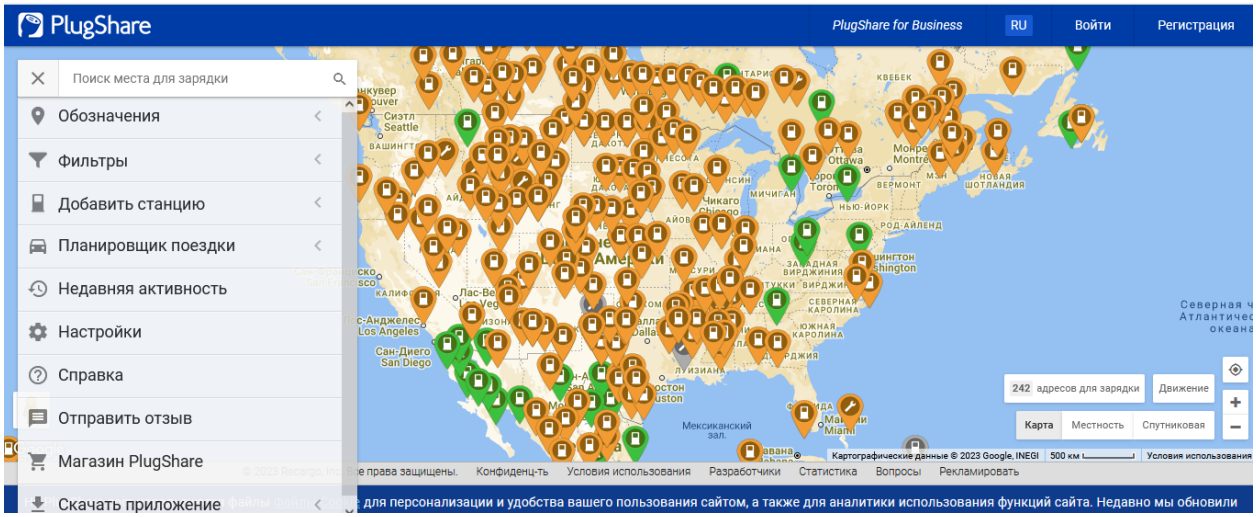

Рис.1.2. Інформаційна система логістики станцій зарядки електромобілів PlugShare

2. Tesla Supercharger (рис.1.3)- це мережа зарядних станцій, яка належить компанії Tesla і призначена для зарядки автомобілів цієї марки. Україна має кілька Tesla Supercharger-ів, розташованих в основному на заході та півдні країни, [7].

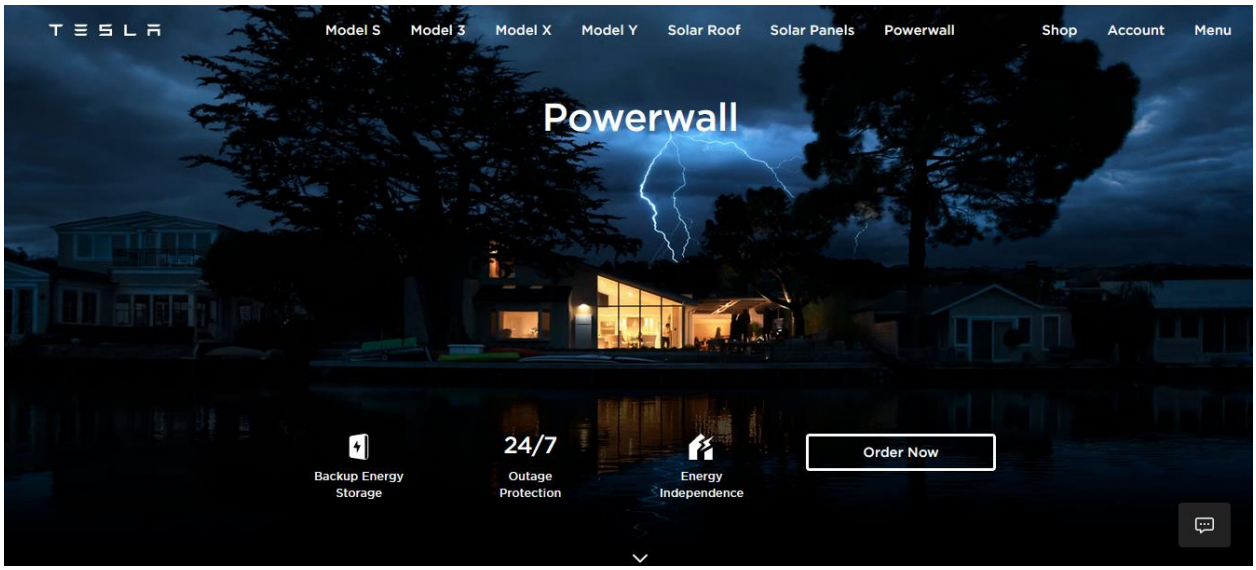

Рис.1.3. Інформаційна система логістики станцій зарядки електромобілів Tesla Supercharger

3. UGV Charge (рис.1.4) - це мережа зарядних станцій, що належить українській компанії "УкрГазВидобування". Мережа складається з понад 400 станцій, розташованих по всій Україні, і дозволяє заряджати як електромобілі, так і гібридні автомобілі, [8].

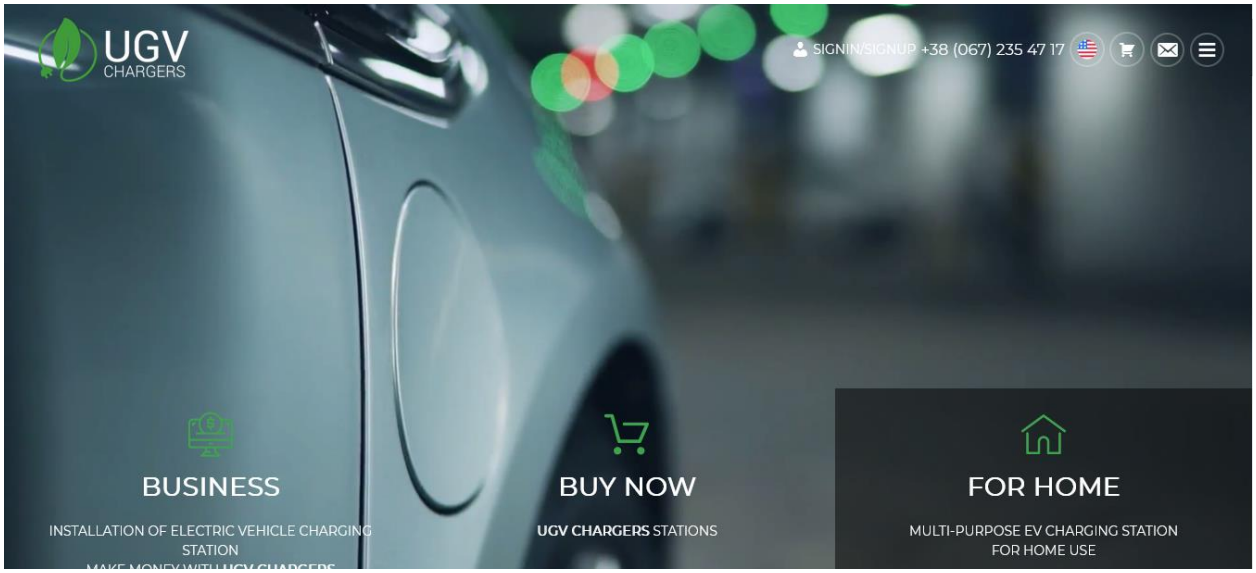

Рис. 1.4. Інформаційна система логістики станцій зарядки електромобілів UGV Charge

4. RODINA Energy Group (рис.1.5) - це українська компанія, яка займається виробництвом та продажем зарядних станцій для електромобілів. Крім того, компанія пропонує послуги з монтажу, підключення та обслуговування зарядних станцій, [9].

5. GreenFuel - це українська компанія, яка пропонує послуги з установки та обслуговування зарядних станцій для електромобілів. Крім того, GreenFuel має власну мережу зарядних станцій, що складається з більше, ніж 60 станцій, розташованих по всій Україні, [10].

Рис.1.5. Інформаційна система логістики станцій зарядки електромобілів GreenFuel

6. Ще однією з інформаційних систем логістики станцій зарядки електромобілів є система EVA+ (Electric Vehicle Aggregator Plus), яка була запущена в Україні у 2018 році компанією EVA (рис.1.6). Ця система є мобільним додатком, який дозволяє водіям електромобілів знайти найближчу зарядну станцію, зарезервувати місце на ній та оплатити послугу прямо через додаток. Крім того, EVA+ надає користувачам інформацію про рівень заряду батареї та час, необхідний для повного заряджання, [11].

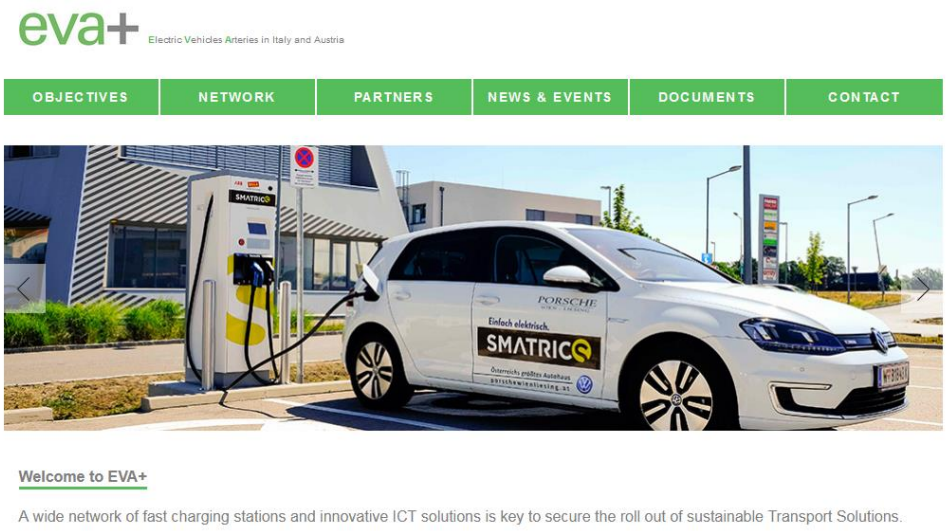

Рис.1.6. Інформаційна система логістики станцій зарядки електромобілів EVA+

Загалом, на сьогоднішній день в Україні існує декілька інформаційних систем логістики станцій зарядки електромобілів, які допомагають користувачам знайти найближчу зарядну станцію, зарезервувати місце на ній та оплатити послугу. Однак, важливо зазначити, що розвиток інфраструктури зарядних станцій та інформаційних систем логістики повинен продовжуватись, щоб забезпечити зручність і доступність заряджання електромобілів у будь-якій точці країни.

1.5 Висновки до розділу

Підводячи підсумки, можна зробити висновок, що електромобільність в Україні все ще знаходиться на ранній стадії розвитку, але з кожним роком зростає інтерес до електромобілів та підтримка їх використання на рівні держави. Державні пільги та підтримка від виробників автомобілів сприяють зростанню чисельності електромобілів на дорогах, а також розвитку інфраструктури зарядних станцій.

Проте, необхідно продовжувати працювати над збільшенням кількості зарядних станцій, розширенням мережі та вдосконаленням інформаційних

систем логістики станцій зарядки, щоб забезпечити зручне та швидке заряджання електромобілів та задовольнити потреби користувачів.

Крім того, досягнення стабільного розвитку електромобільності в Україні потребує змін в законодавстві, щодо оподаткування та підтримки електромобілів, а також удосконалення нормативної бази у сфері будівництва та експлуатації зарядних станцій.

В цілому, розвиток електромобільності в Україні має потенціал для зниження шкідливого впливу на довкілля та покращення якості повітря, а також для створення нових робочих місць та розвитку вітчизняного автомобілебудування. Однак, для досягнення цих цілей, необхідно продовжувати розвивати електромобільність в Україні на всіх рівнях - від підтримки держави та виробників, до задоволення потреб користувачів.

2 МОДЕЛЮВАННЯ ТРАНСПОРТНОЇ ЛОГІСТИКИ

2.1 Підходи до моделювання систем процесів транспортування

Послідовність формування системи за системного підходу містить у собі кілька етапів [12]:

– визначаються і формулюються цілі функціонування системи;

– на підставі аналізу мети функціонування системи та обмежень зовнішнього середовища визначають вимоги, яким має задовольняти система;

– на базі цих вимог формуються, орієнтовно, деякі підсистеми;

– найскладніший етап синтезу системи: аналіз різних варіантів і вибір підсистем, організація їх у єдину систему.

У логістиці один з основних методів синтезу систем - моделювання.

Моделювання ґрунтується на подібності систем або процесів, яка може бути повною або частковою. Основна мета моделювання - прогноз поведінки процесу або системи. Ключове питання моделювання - "Що буде, якщо ...?".

Суттєвою характеристикою будь-якої моделі є ступінь повноти подібності моделі до модельованого об'єкта. За цією ознакою всі моделі можна розділити на ізоморфні та гомоморфні.

Ізоморфні - це моделі, що включають усі характеристики об'єкта-оригіналу, здатні, по суті, замінити його. Якщо можна створити його і спостерігати ізоморфну модель, то наші знання про реальний об'єкт будуть точними. У цьому разі ми зможемо точно передбачити поведінку об'єкта.

Гомоморфні моделі. У їхній основі лежить неповна, часткова подібність моделі до об'єкта, що вивчається. При цьому деякі сторони функціонування реального об'єкта не моделюються зовсім. У результаті спрощуються побудова моделі та інтерпретація результатів дослідження. Під час моделювання логістичних систем абсолютна подібність не має місця. Тому надалі ми розглядатимемо лише гомоморфні моделі, не забуваючи, однак, що ступінь подібності в них може бути різним.

Математичним моделюванням називається процес встановлення відповідності даному реальному об'єкту деякого математичного об'єкта, званого математичною моделлю.

Під час моделювання розвитку автотранспортного підприємства (АТП), як його стану, так і функціонування, намітилися два підходи: детерміністсько-оптимальний і ймовірнісно-адаптивний.

Детерміністсько-оптимальний підхід до прийняття управлінських рішень у більшості випадків забезпечує значний економічний ефект. За оптимального планування отримують не просто прийнятні або допустимі варіанти планів, а найкращі щодо прийнятого способу їхньої оцінки. При цьому широко використовують економіко-математичні моделі, що дають змогу вибирати варійовані показники плану з умов екстремуму прийнятої міри його ефективності (наприклад, максимізація прибутку, мінімізація витрат тощо).

У проаналізованих джерелах позитивним є те, що запропоновані моделі дають змогу розробляти рішення про розвиток АТП з урахуванням зміни як стану системи (АТП), так і її функціонування, тобто пов'язують різні сторони діяльності АТП. Оптимізація одночасно функціонування і стану системи - головна умова для досягнення її найвищої ефективності. Інший аспект полягає в тому, що вирішити завдання розвитку АТП за допомогою однієї моделі неможливо. Необхідно розчленувати це завдання на низку локальних, що входять у загальну систему завдань логістики транспорту [13].

Таким чином, до переваг детерміністського моделювання відносять більшу силу узагальнення і багаторазовість використання.

До числа недоліків детерміністсько-оптимального підходу до прийняття рішень належать такі як:

– практична відмова від розв'язання тих проблем ухвалення рішень, які нині не можуть бути математично формалізовані, тому що для цього необхідно знати явні залежності, що пов'язують шукані характеристики з початковими умовами, параметрами і змінними системами. Однак такі залежності вдається отримати тільки для порівняно простих систем;

– відмова від аналізу та вдосконалення організаційних структур;

– пасивна участь розробників електронних моделей у їх реалізації тощо.

До моделювання розвитку АТП як економічної системи в умовах ринкової економіки найприйнятніший імовірнісно-адаптивний підхід. Як основні характеристики цього підходу до моделювання завдань підприємства слід зазначити [12,13]:

– включення всіх переваг детерміністсько-оптимального підходу;

– створення людино-машинних систем планування, що дають змогу повніше й ефективніше використовувати в процесі планування досвід та інтуїцію фахівців-плановиків;

– врахування відомої частки невизначеності в наших знаннях про майбутнє, що зумовлює вибір найбільш адаптивних варіантів планів;

– персоніфікацію плану як системи взаємопов'язаних рішень;

– розгляд організаційних проблем.

Необхідність поєднання детерміністського та ймовірнісного підходу до вирішення завдань логістики транспорту зумовлена характерними особливостями завдань розвитку АТП. До них належать:

– значна невизначеність як майбутніх ситуацій, у яких, можливо, опиниться об'єкт під час своєї еволюції, так і кінцевих результатів ухвалених рішень;

– неповнота й істотно низька достовірність вихідної інформації, яка часом має занадто укрупнений, агрегований характер;

– труднощі методологічного та обчислювального характеру (врахування принципово неформалізованих елементів), що не дають змоги досягти повної адекватності моделей реальним процесам розвитку АТП.

Імітаційні моделі, створені на основі ймовірнісного моделювання, не розв'язують, а здійснюють прогін програми із заданими параметрами. Однак і вони мають низку суттєвих недоліків, які також необхідно враховувати. По-перше, дослідження за допомогою цього методу обходяться дорого. Причини [14]:

– для побудови моделі та експериментування на ній необхідний висококваліфікований фахівець - програміст;

– необхідна велика кількість машинного часу, оскільки метод ґрунтується на статистичних випробуваннях і вимагає численних прогонів програм;

– моделі розробляються для конкретних умов і, як правило, не тиражуються.

По-друге, велика ймовірність помилкової імітації. Процеси в логістичних системах мають імовірнісний характер і піддаються моделюванню тільки при введенні певного роду допущень.

Наприклад, розробляючи імітаційну модель товаропостачання району і приймаючи середню швидкість руху автомобіля на маршруті, що дорівнює 25 км/год, ми виходимо з припущення, що дорожні умови хороші. Насправді погода може зіпсуватися і, внаслідок ожеледиці, що настала, швидкість на маршруті впаде до /ч15 км. Реальний процес піде інакше.

Водночас процеси на транспорті, що включають елемент випадковості, не являють собою, однак, суто випадкові процеси: У них винятково високою є роль і організаційної складової - технологія ТО і ТР, графік режиму роботи тощо. Тому формули (моделі), розроблені тільки на основі ймовірнісного або детермінованого підходу до транспортних процесів, часто не відповідають існуючій системі транспорту.

Адаптаційна поведінка проявляється в різних тенденціях розвитку, які відображають еволюцію конкретної системи в ході її пристосування до впливів зовнішнього середовища. Функціонування автомобільного транспорту має переважно адаптивний характер.

Моделювання розвитку АТП може бути забезпечено поєднанням нормативних і дескриптивних моделей, що виробляють, з одного боку, рішення з активних впливів на розвиток АТП, а з іншого боку, описують процеси адаптації АТП в умовах невизначеності та неповної інформації.

Розроблення та впровадження ймовірнісно-адаптивного підходу можуть забезпечити реалізацію основних умов ефективного використання методів і

моделей у логістиці транспорту, а також методологічних принципів аналізу та синтезу логістичних систем, таких, як системність, надійність, адаптивність, стійкість тощо.

Оцінка рівня методичного забезпечення та підходів до моделювання завдань логістики дає змогу зробити такі висновки:

Існуючі методи і моделі розв'язання локальних задач здебільшого не забезпечували головної умови їхнього ефективного застосування, що проявляється у взаємозв'язку задач забезпечення, виробництва і збуту транспортних послуг.

Частина методів і моделей розв'язання задач логістики транспорту в частині розподілу матеріальних (транспортних) послуг створюють передумови порушення основоположних принципів логістики: системності та надійності логістичних систем.

Найприйнятнішим підходом до об'єднання методів і моделей розв'язання задач транспортної логістики є ймовірнісно-адаптивний. Об'єднання методів і моделей розв'язання задач логістики транспорту має базуватися на описі розвитку автопарку як послідовності явищ у часі з використанням апарату теорії випадкових процесів, тобто стохастичних моделей.

2.2 Функції логістики

В економіці логістика – сукупність наук управління матеріалопотоком і потоком продукції від джерела до споживача, що охоплює комбінування видів діяльності різних установ і служб, пов'язаних із розподілом, матеріальним забезпеченням, плануванням виробництва та управлінням ним, тобто логістика є системою, яка містить функціональні галузі.

Концепція (принцип) логістичної системи пов'язана з управлінням матеріалами та управлінням розподілом. Американські вчені вважають логістику - структурою планування, а не функцією підприємництва. Інакше кажучи, завдання управління в галузі логістики має справу не стільки з управлінням матеріальним потоком, скільки із забезпеченням механізму

розроблення завдань і стратегій, у рамках яких може здійснюватися повсякденна діяльність з управління розподілом.

Одна з особливостей принципу логістики полягає в тому, що вона приділяє увагу не тільки інтеграції видів діяльності, що традиційно належать до різних функцій підприємництва, а й при ухваленні рішення об'єднує їх. Наприклад, у багатьох компаніях, де відсутня логістична система, відповідальність за запаси і транспортування може бути відповідно функцією виробництва і розподілу, і рішення щодо перших можуть ухвалюватися без урахування останніх. У логістичній системі всі види діяльності взаємопов'язані; і під час ухвалення рішення мають бути враховані негативні та позитивні сторони різних функціональних областей.

Матеріальні потоки на стадії придбання засобів виробництва є об'єктом вивчення та управління закупівельної логістики, матеріальні потоки на стадії виробництва - об'єктом виробничої логістики. Об'єктом розподільчої логістики матеріальні потоки стають на стадії розподілу та реалізації готової продукції.

Розглянемо функціональні галузі логістики [15].

Запаси – виконують буферну роль між транспортом, виробництвом і реалізацією. Запаси дають змогу економічно й ефективно функціонувати всій системі. Продукція може бути зосереджена в запасах безпосередньо у виробника, або її зберігання може бути наближене до споживача. Величина виробничих запасів має бути оптимальною для всієї системи. Запаси продукції дають змогу даній системі швидко реагувати на зміну попиту і забезпечують рівномірність роботи транспорту.

Транспорт – включає в себе за логістичного підходу не тільки перевезення вантажу від постачальника до споживача, з підприємства на склад, зі складу на склад, а й також доставку зі складу споживачеві. Основними характеристиками транспорту є вартість і надійність.

Складське господарство – включає розміщення в складських приміщеннях для зберігання матеріалів, управління складською переробкою, пакування тощо.

Інформація – будь-яка логістична система управляється за допомогою інформаційної та контролюючої підсистем. Ці підсистеми передають замовлення, вимоги про відвантаження і транспортування продукції, підтримують рівень запасів.

Виробництво.

Інші функціональні області. Наприклад, виробниче планування і контроль над матеріалопотоком у процесі виробництва [16]:

– кадри. Важливий складовий елемент системи логістики. Їх підбору та підготовці надається велике значення;

– обслуговуюче виробництво. Підрозділи логістики, які обслуговують процес виробництва, мають не лише визначати його потреби, а й бути здатними згладжувати коливання попиту та пропозиції. Деякі економісти не розглядають виробничу одиницю як функціональну область у системі логістики. Проте зазначають, що виробничі потужності та економічна пристосованість підприємства мають важливе значення для функціонування логістичної системи. Основними проблемами для неї є визначення розміру та розміщення підприємства.

Специфіка логістики полягає не тільки в об'єднанні управління матеріальним потоком (включно з усіма функціональними галузями) в одних руках, а й взаємодією з управлінськими функціями (планування, організації, контролю).

Логістична система, яку використовує фірма для вироблення стратегії, у таких видах діяльності, як планування і виробництво, взаємодіє з функціональними галузями: виробництво і технологія, маркетинг, а також фінансування та адміністрування.

У плануванні логістика впливає на виробництво і технологію за допомогою визначення оптимальних розміщень фірми, планування складської

мережі, складського опрацювання вантажів, вибору обладнання, транспортної моделі; у сфері маркетингу - логістика визначає канали розподілу, цілі обслуговування споживачів; фінансування та адміністрування пов'язані з розробкою інформаційної системи, контролю за запасами і бюджетом.

Виробнича діяльність логістики пов'язана зі складанням виробничого календарного планування, прогнозом продажів, обробкою замовлення, диспетчеризацією, контролем за діяльністю, управлінням запасами готової продукції, зовнішнім і внутрішнім транспортом та іншими функціями.

Процес виробництва взаємодіє із системою логістики за двома напрямами. По-перше, виробництво має регулярно поповнювати запаси готової продукції в системі розподілу і, що особливо важливо, задовольняти спонтанні потреби, незалежно від того, чи є продукція стандартною, модифікованою або спеціальною. По-друге, виробництво залежить від системи матеріального забезпечення в частині сировини, матеріалів, комплектуючих частин у певній кількості та певної якості.

Управління виробничим процесом спрямоване насамперед на зниження витрат виробництва і, як правило, орієнтоване на ритмічну роботу з максимально можливим часом виробничого циклу і терміну виконання замовлення.

2.3 Основні елементи системи масового обслуговування

Система масового обслуговування характеризується структурою, яка визначається складом і функціональними зв'язками. Вона складається з таких елементів: вхідного потоку вимог, приладів (каналів) обслуговування, черги вимог, які очікують на обслуговування, і вихідного потоку вимог. Ця система може бути ускладнена, якщо вона складається не з однієї групи приладів, а з низки послідовних приладів, або з низки послідовно і паралельно пов'язаних приладів, або має ще складнішу мережеву структуру. Можливі системи, у яких відсутній такий елемент, як черга (системи з відмовами).

Вхідний потік являє собою сукупність вимог, які надходять у систему і потребують обслуговування. Саму вимогу можна розглядати як запит на задоволення якоїсь потреби. Часто вимогу ототожнюють із її носієм.

Прикладів вхідних потоків можна навести багато. Це потік інформації, що надходить на опрацювання в ЕОМ; потік клієнтів, які приходять у перукарню; хворих, які надходять у лікарню, поліклініку; судна, які надходять у порт; літаки супротивника, які налітають на об'єкт удару, і т. д.

У наведених прикладах як вимоги виступають відповідно клієнти, хворі, судна, бомбардувальники. Склад системи визначається ще й кількістю каналів (приладів, ліній тощо). За кількістю каналів (приладів, ліній) системи можна розділити на одноканальні та багатоканальні. Прикладом одноканальної системи може слугувати перукарня з одним майстром, одиночний пункт ВТК на потоці, бензозаправна колонка тощо. Природно, що в багатоканальних системах кількість приладів має дорівнювати двом і більше. Подібні системи зустрічаються набагато частіше. Як правило, кількість приладів у багатоканальних системах масового обслуговування обмежена. Однак можливі випадки, коли їх настільки багато, що вигідніше розглядати їх як системи з нескінченним числом приладів, каналів.

Зі свого боку, багатоканальні системи масового обслуговування можуть складатися з однакових, однотипних приладів або (найчастіше) різних, відрізняючись продуктивністю під час обслуговування. Справді, майстри з ремонту радіоапаратури в ательє, продавці в магазинах, касири тощо, незважаючи на практично однакові умови роботи, через різні причини (досвід, психічний стан, здібності тощо) працюють із різною продуктивністю.

Під час розв'язання багатьох перелічених завдань під терміном "обслуговування" розуміється задоволення потреб. Так, у наведених прикладах систем під обслуговуванням розуміють стрижку, гоління та інші операції, що виконуються над клієнтами в перукарні, приймання хворих у лікарні, розвантаження суден у порту, обстріл бомбардувальників системою ППО об'єкта тощо.

Під якістю роботи систем масового обслуговування розуміють не те, як добре виконано саме обслуговування (якість ремонту, навантаження суден тощо) - це оцінюють за іншими критеріями, - а як добре організовано обслуговування, наскільки повно завантажено обслуговувальні прилади, чи не створюють великої черги, чи не велике витікання із системи необслужених вимог.

Під час розв'язання практичних задач, пов'язаних із системами масового обслуговування, дуже важливо оцінити їхню повнодоступність, тобто можливість кожної вимоги надійти на обслуговування будь-якого приладу системи. Якщо ця умова не витримується, то система називається неповнодоступною. Прикладом може слугувати система протиповітряної оборони об'єкта, перед якою стоїть завдання відбиття нальоту літаків противника в широкій смузі. Якщо смуга нальоту досить широка і повністю не перекривається зонами дійсного вогню всіх вогневих засобів оборони, то, мабуть, вони не можуть обстріляти будь-який літак у нальоті, а "обслуговують" тільки ті, що пролітають через зони їхнього дійсного вогню.

Вихідний потік - це потік вимог, які залишають систему. Вимоги потоку можуть бути обслужені приладами системи і не обслужені.

Дослідження структури вихідного потоку має велике значення, оскільки він може бути вхідним потоком для іншої групи приладів. Розподіл вимог у вихідному потоці в часі залежить від щільності вхідного потоку і характеристик роботи приладів обслуговування системи. Наприклад, щільність потоку відремонтованих радіоприймачів з ательє залежить від продуктивності майстрів з ремонту та завантаження ательє несправною апаратурою, що надходить від населення.

Основним завданням масового обслуговування є визначення кількісних показників функціонування систем масового обслуговування та їхньої залежності від параметрів вхідного потоку і структури власне системи (її складу та функціональних зв'язків). Розв'язання цього завдання дає змогу знайти в системі слабкі ланки, визначити їхній вплив на ефективність

обслуговування та знайти шляхи їхнього поліпшення або за заданих характеристик потоку вимог і критеріїв якості обслуговування дати пропозиції про структуру системи, яка забезпечить виконання поставленого перед нею завдання. Для вирішення цього можуть бути залучені різні методи оптимізації: лінійне або нелінійне програмування, динамічне програмування, теорія ігор тощо.

2.4 Практичний аналіз імовірнісної системи

Вибір оптимального числа машин М за детерміністською моделлю за бажання задовольнити всі заявки, що надійшли, можна робити на підставі формули (2.1) [17]:

$$M = [\frac{\overset{\max}{t}(T_{tf}(t)+T_{tr}(t))+T_l+T_u}{\bar{T}}]+1=[\frac{T_{0\min}}{\bar{T}}]+1, \quad (2.1)$$

де λ - середня інтенсивність пуасонівського потоку заявок;

$\bar{T}$ - середній період надходження ($\bar{T}=\frac{1}{\lambda}$);

T_0 - повний час циклу роботи транспортного засобу;

T_l , T_{tf} , T_u , T_{tr} - відповідно проміжки часу на завантаження вантажу, перевезення його, розвантаження і повернення на склад.

Тобто M дорівнює найближчому великому цілому числу до величини, що дорівнює відношенню мінімальної тривалості циклу роботи транспортного засобу $T_{0\min}$, (без очікування) до середнього періоду надходження заявки.

Недолік моделі - неможливість повного врахування ймовірнісного характеру розподілів заявок (навіть для найпростішого пуасонівського потоку) і часів руху транспортного засобу.

Облік характеру розподілу заявок у припущенні пуасонівського потоку можна здійснити таким чином. Середнє число заявок, що приходять за мінімальний цикл роботи транспортного засобу, можна розрахуват на формулі (2.2) [15]:

$$\alpha = \frac{T_{0\min}}{\bar{T}} = \lambda T_{0\min} \tag{2.2}$$

Імовірність того, що випадкова кількість заявок ξ за цей самий час не перевищить кількості наявних транспортних засобів M, а отже, всі заявки будуть задоволені згідно з розподілом Пуассона [14], на формулі (2.3):

$$P = P(\xi \le M) = \sum_{k=0}^{M} \frac{\alpha^k}{k!} e^{-\alpha} \tag{2.3}$$

Імовірність відмови у виконанні заявки, на формулі (2.4) [15]:

$$q = P(\xi > M) = 1 - P(\xi \le M) = 1 - \sum_{k=0}^{M} \frac{\alpha^k}{k!} e^{-\alpha} \tag{2.4}$$

Таким чином, мінімальне необхідне число машин, щоб імовірність виконання заявки була не меншою, ніж *p,* або ймовірність відмови не більшою, ніж *q,* визначається зі співвідношення на формулі (2.5) [14]:

$$M' = F^{-1}(p, \frac{T_{0\min}}{\bar{T}}) = F^{-1}(1-q, \frac{T_{0\min}}{\bar{T}}) \tag{2.5}$$

де $F^{-1}(P,\alpha)$ - інверсна кумулятивна функція розподілу Пуассона з коефіцієнтом α .

У середовищі GNU Octave було змодельовано таку ситуацію:

– параметр розподілу $= \lambda \frac{1}{30}$;

– тривалість циклу роботи транспортного засобу T_0 = 40 хв;

Під час підстановки цих значень у формулу (2.1) було отримано результат *M=2*, тобто за заданих умов детерміністська модель прогнозує, що 2 транспортні засоби забезпечать роботу без простоїв і без відмов.

Підставивши вихідні параметри у формули (2.3) і (2.4), отримано графік, який представлено на рис.2.1.

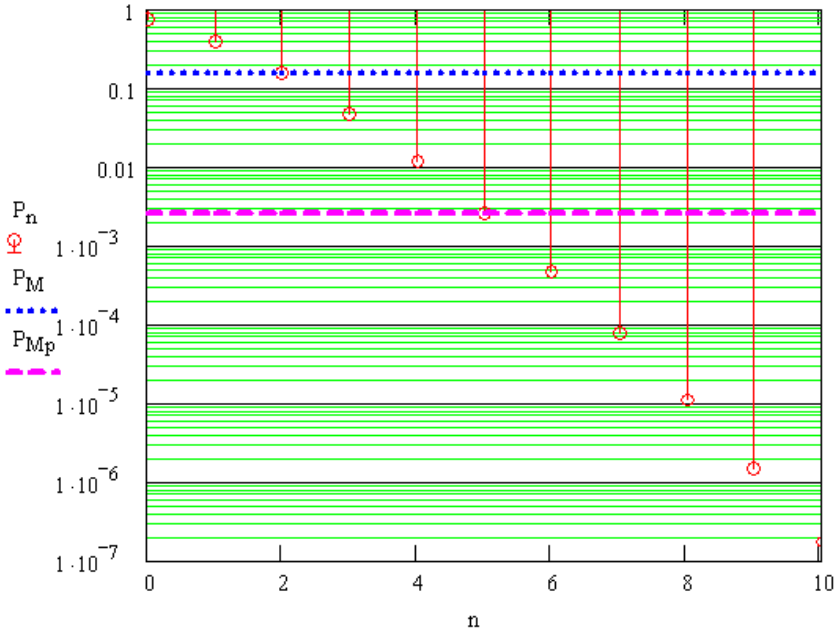

Рис.2.1. Графік залежності ймовірності відмови виконання заявки від числа транспортних засобів протягом циклу роботи транспортного засобу

Відповідно на осі абсцис відкладено граничну кількість заявок, а на осі ординат відкладено ймовірність того, що цю кількість буде перевищено (P)$_n$.

Значення P_M вказує, з якою ймовірністю за заданої кількості транспортних засобів ($M=2$) у системі настане відмова, у зв'язку з відсутністю вільного транспортного засобу ($P_M = 0,15$).

Якщо ми задамося якоюсь конкретною ймовірністю відмови і тим, скільки в цьому разі має бути транспортних одиниць, то необхідно звернутися до формули (2.3).

При заданні ймовірності відмови $q=0,01$ у системі з тими самими параметрами, було отримано число M_P =5, тобто необхідно 5 транспортних засобів. Цей випадок також відображено на графіку (P_{Mp}).

Побудуємо графік, зворотний графіку на рис. 2.1. Для цього використаємо інверсну кумулятивну функцію розподілу.

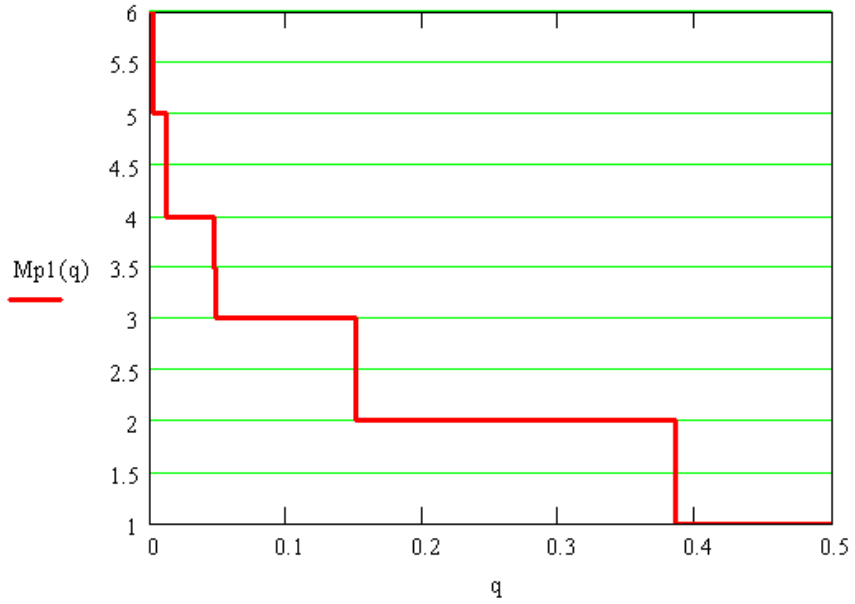

Рис.2.2. Графік залежності числа транспортних засобів від імовірності відмови виконання заявки протягом циклу роботи транспортних засобів

У цьому разі ситуація зворотна - на осі абсцис відкладено ймовірність того, що це станеться відмова, а на осі ординат відкладено кількість транспортних засобів, яка відповідає цій ймовірності.

Таким чином, за допомогою рівнянь, що включають пряму та інверсну функції розподілу Пуассона, можна аналізувати систему щодо ймовірності відмови за умови заданої кількості ресурсів і навпаки - обчислювати, яка кількість транспортних одиниць необхідна, щоб імовірність відмови була не більшою за задану.

2.5 Алгоритми пошуку найкоротших шляхів

Для розв'язання задачі пошуку найкоротшого маршруту, розглянемо декілька алгоритмів.

Алгоритм Дейкстри. Цей алгоритм шукає найкоротші відстані від однієї вершини до всіх інших у зваженому графі без ребер від'ємної ваги за час O(N2). Можлива модифікація до часу O(MlogN) при зберіганні графа списком ребер, її ми розглядати не будемо. У разі, коли цю задачу необхідно розв'язати

на графі з ребрами від'ємної ваги, більше підходить алгоритм Форда-Беллмана, але він працює вже за час O(N3) (або O(MN)).

Даний алгоритм простіше пишеться для графа, представленого на матриці суміжності, однак, якщо вам буде потрібно реалізувати його для списку ребер, це не складе великих труднощів, і підсумкова складність роботи залишиться тією ж.

Отже, перейдемо до опису роботи алгоритму. Крім матриці суміжності нам знадобиться масив відстаней D: *array[0..MaxN]* of Longint і масив міток Use: *array[0..MaxN]* of Boolean.

Алгоритм працює N ітерацій, до того ж на I-ій ітерації в елементі D[J] зберігається довжина найкоротшого шляху зі стартової вершини в J-ту, що складається не більше ніж з I ребер. На кожній ітерації з масиву D обирають найменший невикористаний елемент (нехай його номер дорівнює Min), і проводять релаксацію за всіма ребрами, які виходять із Min-ної вершини. Після цього позначаємо Min-ний елемент використаним і заходимо на наступну ітерацію. Таким чином, після відпрацювання N ітерацій у масиві D зберігатимуться найкоротші шляхи до всіх вершин із початкової, які складаються не більше ніж з N ребер, тобто всі найкоротші шляхи. Ось фрагмент програми, що виконує ці дії (тут *I, J:* Integer, а *X* - номер стартової вершини). На рисунку 2.3 наведено фрагмент лістингу [16].

```
1   For I := 1 to N do
2      D[I] := Inf;
3   D[X] := 0;
4   D[0] := Inf + 1;
5   For I := 1 to N do Begin
6      Min := 0;
7      For J := 1 to N do
8        If (D[J] < D[Min]) and not Use[J] Then Min := J;
9      Use[Min] := True;
10     For J := 1 to N do
11       If D[Min] + A[Min, J] < D[J] Then D[J] := D[Min] + A[Min, J];
12  End;
13
14
15
```

Рис.2.3. Фрагмент лістингу алгоритму Дейкстри

У рядках 1-2 ініціалізується масив D, а далі потрібно детально розглянути кілька моментів. По-перше, клітинка з номером початкової вершини забивається нулем, а не Inf-ом, оскільки на нульовій ітерації алгоритму до початкової вершини шлях існує (бо він складається з нуля ребер) і дорівнює, відповідно, нулю (рядок 3). По-друге, нам знадобиться нульовий осередок масиву D (незважаючи на те, що жодної нульової вершини в нас немає), який забивається числом Big + 1, щоб під час пошуку мінімального нам підійшов перший-ліпший невикористаний осередок (рядок 4), а заразом і нульовий осередок масиву Use, щоб не вилізти за межі масиву на першій ітерації циклу. Нульова комірка масиву D дорівнює Big + 1, а не просто Big для випадку з незв'язним графом, за якого в рядку 11 Min міг би дорівнювати нулю, що призвело б до звернення за межі матриці суміжності. А так нам підійде невикористаний елемент масиву D навіть якщо він дорівнює Inf.

У рядку 5 запускається цикл в N оновлень масиву D. У рядках 6-8 шукається мінімальний невикористаний елемент масиву D. У рядку 9 він позначається використаним. У рядках 10-11 проводиться релаксація за всіма ребрами, що виходять з обраної вершини (Min).

Якщо після закінчення роботи алгоритму в деякій комірці масиву D все ще зберігається значення Inf, це означає, що найкоротшого шляху з початкової вершини до відповідної цій комірці вершини не існує (граф незв'язний).

Варто зауважити, що цей самий алгоритм використовується для розв'язання більш поширеної задачі знаходження найкоротшого шляху між заданими двома вершинами в графі. Як початкова, відповідно, використовується одна з вершин, і виводиться значення комірки D з номером іншої вершини. Якщо це значення дорівнює Big, значить, між цими вершинами не існує шляху. Алгоритм, який шукає найкоротшу відстань тільки між двома вершинами, науці невідомий. Ба більше, доведено, що алгоритм знаходження найкоротшої відстані між двома вершинами графа асимптотично не може бути оптимальнішим за алгоритм, що знаходить найкоротші відстані від однієї вершини до всіх інших.

Алгоритм Форда-Беллмана іноді називається так само алгоритмом Беллмана-Форда [17].

Цей алгоритм шукає найкоротші відстані від однієї вершини до всіх інших у зваженому графі за час O(N3), якщо граф задано матрицею суміжності, або за O(MN) для графа на списку ребер.

Для роботи цього алгоритму нам додатково знадобиться лише масив відстаней D: *array[1..MaxN]* of Longint.

Алгоритм Форда-Беллмана дещо схожий на алгоритм Дейкстри: у нього теж на I-ій ітерації в елементі D[J] зберігається довжина найкоротшого шляху зі стартової вершини до J-ої, який складається не більше ніж з I ребер. Але на кожній ітерації відбуваються релаксації одразу по всіх ребрах. Саме через це, якщо не вдаватися в подробиці, він працює навіть для графів з від'ємними вагами ребер, хоча й асимптотично повільніше, ніж алгоритм Дейкстри.

Нижче наведено його реалізацію для графа, представленого списком ребер (тут *I, J*: Integer, а *X* - номер стартової вершини). На рисунку 2.4 наведено фрагмент лістингу.

```
1   For I := 1 to N do
2      D[I] := Inf;
3   D[X] := 0;
4   For I := 1 to N do
5      For J := 1 to M do
6        If D[R[J].B] > D[R[J].A] + R[J].C Then
7   D[R[J].B] := D[R[J].A] + R[J].C;
8
9
10
11
```

Рис. 2.4. Фрагмент лістингу алгоритму Форда-Беллмана

У рядках 1-3 аналогічно алгоритму Дейкстри ініціалізується масив D, щоправда тут нам не потрібен нульовий елемент цього масиву. У рядку 4 запускається N ітерацій алгоритму, у рядках 5-6 проводиться релаксація по всіх ребрах графа.

Оскільки алгоритм працює також на графах з від'ємними вагами ребер, клітинки, що відповідають вершинам, до яких немає шляху з початкової, після роботи алгоритму можуть містити не тільки значення Inf, а так само і менше число. Але в будь-якому разі, кінцеве значення такої комірки не може бути меншим за Inf - MinW * (MaxN - 1), де MinW - модуль мінімально можливої ваги ребра, а MaxN - максимальна кількість вершин. Якщо кінцеве значення комірки перевищує це значення, значить, шляху до відповідної вершини не існує. Однак не варто забувати, що якщо Inf <= (MinW + MaxW) * (MaxN - 1), то ми не можемо міркувати в такий спосіб - нескінченність обрано надто маленьку.

Алгоритм Форда-Беллмана має кілька чудових особливостей. По-перше, зверніть увагу на те, що під час розгляду ребер графа (рядок 5), ми розглядаємо одразу всі ребра, нас не цікавлять ті, що виходять із будь-якої конкретної вершини. Звідси випливає, що список ребер, представлений масивом R, зовсім не зобов'язаний бути впорядкований, як того вимагав, наприклад, алгоритм пошуку в глибину. Дійсно, ми ніде не використовуємо масив покажчиків G.

По-друге, припустімо, нам не відомо, чи містить цей граф цикли негативної ваги. Тоді ми можемо зробити хитру хитрість і визначити це. Нам потрібно всього лише після закінчення роботи алгоритму провести ще одну його ітерацію і подивитися, чи змінилися значення масиву D. Якщо так, то це означає, що ми змогли поліпшити отриманий остаточний результат, тобто алгоритм відпрацював неправильно, отже, у графі були цикли негативної ваги. Реалізація цієї перевірки може виглядати так (істинність наявності таких циклів повертається у змінній Found: Boolean). На рисунку 2.5 наведено фрагмент лістингу.

```
 8    Found := False;
 9    For J := 1 to M do
10       If D[R[J].B] > D[R[J].A] + R[J].C Then Begin
11         Found := True;
12         Break;
13       End;
14
15
16
17
```

Рис.2.5. Фрагмент лістингу алгоритму Форда-Беллмана

У цьому прикладі ми не зовсім робимо те, що було написано. Ми не робимо ще одну ітерацію і не перевіряємо зміни значень масиву D. Ми просто перебираємо всі ребра (рядок 9) і дивимося, чи може ще якась релаксація змінити значення масиву (рядок 10). Якщо це так, ми закінчуємо перевірку з позитивним результатом (рядки 11-12). Якщо жодна релаксація не змінює поточного стану масиву, то результат залишається від'ємним, тому що це значення було йому присвоєно перед циклом у рядку 8.

У даній роботі при створенні імітаційної моделі використовували алгоритм Дейкстри, який шукає найкоротші відстані від однієї вершини до всіх інших або до конкретної у зваженому графі без ребер від'ємної ваги. З його допомогою знаходиться маршрут до електро заправки.

3 ФУНКЦІОНАЛЬНИЙ ОПИС ІНФОРМАЦІЙНОЇ СИСТЕМИ ЛОГІСТИКИ СТАНЦІЙ ЕЛЕКТРОЗАРЯДКИ

3.1 Дані для створення інформаційної системи

Для реалізації інформаційної системи логістики станцій електрозярядки використовувались такі вхідні дані:

- схема-карта міських автотранспортних доріг із позначеним місцем розташування електро станцій та точки автомобіля;

- інформація про ймовірності розподілу надходжень замовлень від власників авто,

- дані про кількість транспортних засобів, швидкісний режим їхнього пересування певними шляхами залежно від часу доби.

На цьому етапі заявки у системі зберігаються у форматі такої таблиці (табл. 3.1).

Таблиця 3.1 – Формат зберігання замовлення

№ Електро станції	**Час**				**Маршрут**
	очікування		фактич.		
1	**2**	**3**	**4**	**5**	

Наступна таблиця 3.2 відображає залежність проміжків часу між надходженнями замовлень від населеності району, в якому розташована електростанція. Фрагмент районів нанесено на векторний шар та зображено на рис. 3.1.

Таблиця 3.2 – Дані про надходження замовлень від населеності району

№ електростанції	Пріоритет	Середній проміжок між замовленнями, година
1	2	6
2	1	4
3	2	6
4	3	8
5	1	4
6	3	8
7	3	8

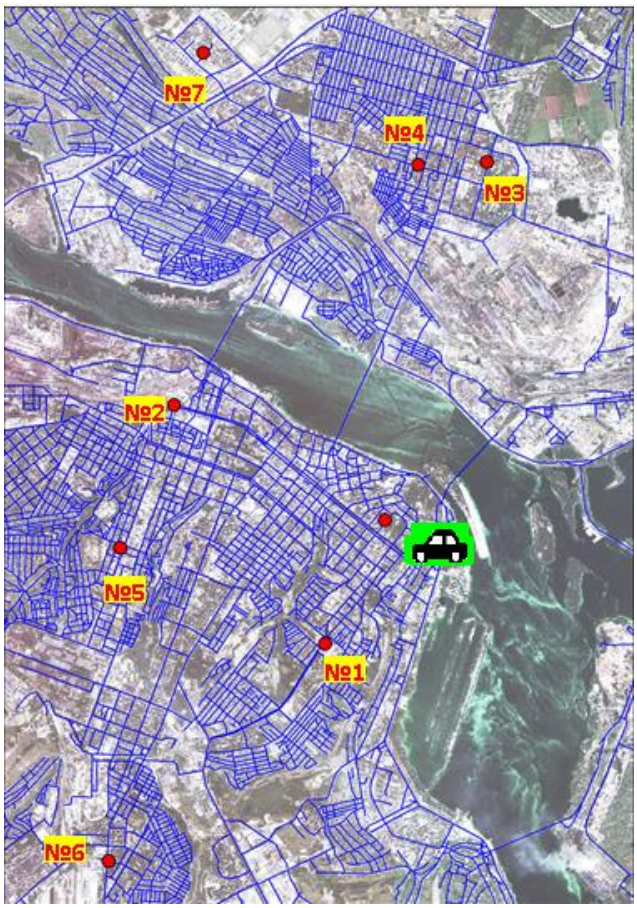

Рис.3.1. Схема розташування електро станцій по відношенню до міської транспортної мережі

3.2 Функціональна діаграма системи

Діаграма варіантів використання є важливим інструментом для аналізу та проектування системи логістики електростанцій. Вона дозволяє ідентифікувати основні варіанти використання системи та їх залежності.

Основними акторами в системі логістики електростанцій є адміністратор, оператор електростанції та користувачі системи. Для кожного з них існують певні варіанти використання.

Для адміністратора системи, основними варіантами використання є створення нових електростанцій, редагування існуючих даних про електростанції та призначення прав доступу для операторів.

Оператор електростанції має можливість додавати нові дані про електростанцію, відслідковувати стан обладнання та проводити технічне обслуговування. Також він може отримувати повідомлення про аварії та несправності на станції та реагувати на них.

Користувачі системи мають можливість здійснювати запити на пошук краткого маршруту до електростанції, отримувати інформацію про технічний стан станцій та проводити моніторинг стану електромережі.

Взаємодія між цими варіантами використання може бути проілюстрована на діаграмі варіантів використання, яка відображає взаємозв'язки між акторами та їх варіантами використання. Ця діаграма може бути корисною для проектування системи логістики електростанцій, оскільки вона дозволяє ідентифікувати потреби користувачів та забезпечити оптимальну взаємодію між різними варіантами використання.

На рисунку 3.2 наведено графічне представлення діаграми використання.

На діаграмі варіантів використання для системи логістики електростанцій необхідно додати варіант використання для знаходження короткого маршруту до електростанцій. Цей варіант використання можна позначити як "Знайти короткий маршрут до електростанції".

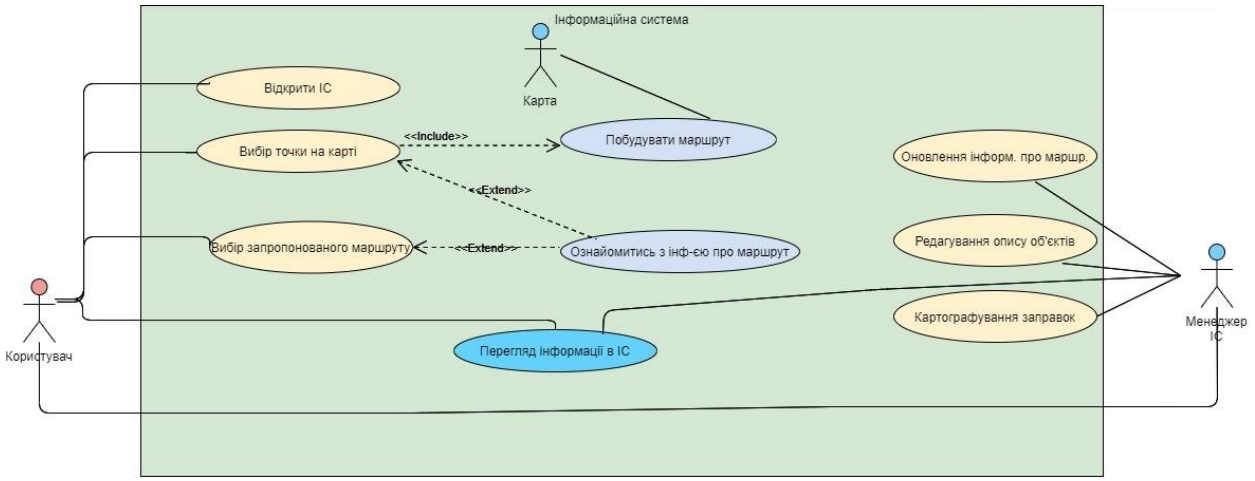

Рис.3.2. Діаграма варіантів використання для розробленої ІС

При використанні цього варіанту користувач повинен мати можливість ввести адресу, з якої потрібно знайти маршрут, та адресу електростанції, до якої потрібно дістатись. Система повинна підрахувати найкоротший маршрут та показати його користувачеві на карті. Крім того, система повинна підрахувати час, необхідний для проходження маршруту, та показати його користувачеві.

Для цього варіанту використання необхідно врахувати можливість введення адреси та пошук маршруту, показ маршруту на карті, розрахунок часу проходження маршруту та його відображення користувачеві. Також можна передбачити можливість збереження результатів пошуку для майбутнього використання та додаткових опцій, таких як зміна типу транспорту або врахування пробок на дорогах.

Таким чином, основні дії можна класифікувати на наступні варіанти використання:

1.Пошук короткого маршруту до заданої точки:

1.1 Задання точки призначення

1.2 Побудова маршруту

1.3 Відображення маршруту на мапі.

2.Перегляд інформації.

3. Додавання нової електростанції до бази даних та точки на мапі:

3.1 Заповнення форми з характеристиками нової електростанції

3.2 Додавання нової електростанції до бази даних

3.3 Підтвердження додавання нової електростанції

4.Редагування інформації про існуючі електростанції:

4.1 Пошук електростанції за назвою або місцезнаходженням

4.2 Редагування інформації про електростанцію (потужність, тип енергетичного джерела, витрати електроенергії тощо)

4.3 Підтвердження змін

5. Видалення електростанції з бази даних:

5.1 Пошук електростанції за назвою або місцезнаходження.

3.2.1 Бази даних

Схема бази даних - це логічна конфігурація всієї реляційної бази даних даних або її частини. Схема може існувати як у вигляді візуального представлення бази даних, так і у вигляді набору формул, що регулюють її пристрій. Ці формули виражаються з використанням мови опису даних MySQL. На діаграмі показано, як взаємопов'язані сутності, що складають базу даних. Взаємозв'язок баз даних для ІС буде представлено на рис.3.3.

Схема бази даних системи пошуку короткого маршруту електростанції містить наступні таблиці: карти, інформаційна система, маршрути, вибір маршруту, користувачі та повідомлення.

Важливим є таблиця "Маршрути", що містить6

– id код (уникальный идентификатор маршрута);

– електростанція_початку (електростанція, з якої починається маршрут);

– електростанція_кінця (електростанція, до якої веде маршрут);

– довжина_маршруту (довжина маршруту у кілометрах);

– тривалість (приблизний час проходження маршруту);

– витрати_електроенергії (витрати електроенергії на проходження маршруту).

Ця база даних забезпечує зберігання інформації про електростанції та їх характеристики, а також про розраховані маршрути та їх характеристики. Пошук короткого маршруту між електростанціями здійснюється за допомогою алгоритму Дейкстри.

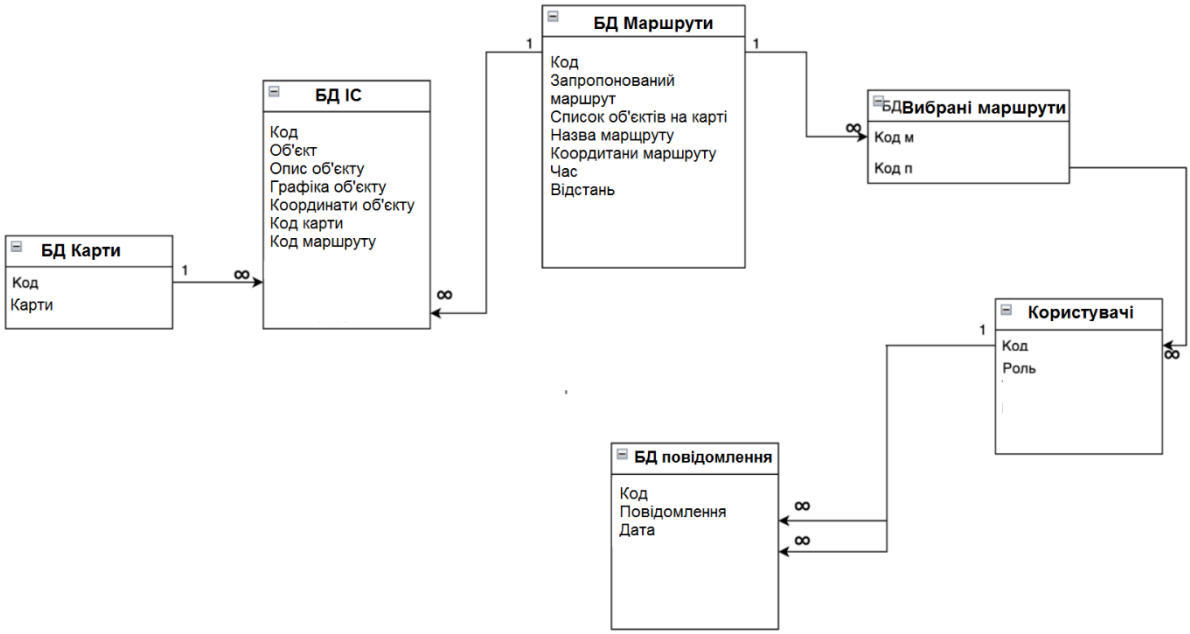

Рис.3.3. Схема бази даних розробленої ІС

3.3 Фактори впливу на розробку моделі інформаційної системи

При створенні моделі, що імітує роботу пошуку маршруту до електро станцій було враховано такі чинники:

- час пересування автотранспорту;

- час доби (тому що, наприклад, у вечірній час транспортні мережі менш завантажені і транспорт може пересуватися швидше);

- можливість відсутності транспортного засобу на нлектростанції в момент надходження замовлення;

- можливість обслуговування відразу декількох замовлень, якщо вони надійшли з малим розривом у часі;

- час відкриття і закриття заправок (під час генерації замовлень відсікаються ті, які виникають за півтори години до закриття заправок);

- "популярність" районів, у яких розташовані зарядні станції (у найбільш населених центральних районах заявки від користувачів електро мобілів надходять частіше).

3.4 Вибір програмного середовища

Python - це популярне високорівневе програмне середовище загального призначення, яке зручно використовувати для розробки різноманітних застосунків, в тому числі систем пошуку короткого маршруту.

Python має велику кількість сторонніх бібліотек, які дозволяють швидко та ефективно розробляти програмні продукти, зокрема для вирішення задач з графами. Наприклад, бібліотека NetworkX надає інструменти для роботи з графами, зокрема алгоритми пошуку найкоротшого шляху між вершинами графа, включаючи алгоритм Дейкстри.

Крім того, Python [18] має зрозумілий та лаконічний синтаксис, що робить код більш читабельним та легким для розуміння та редагування. Код Python також легко портативний, тобто може працювати на різних платформах, таких як Windows, MacOS та Linux.

Бібліотека Pygame є популярним інструментом для розробки ігор та графічних додатків на мові Python. Вона містить інструменти для створення графічних елементів, анімацій, звуків та інших візуальних ефектів. Pygame також має добре документовану API та велику кількість прикладів коду, що допомагає швидко розібратися з її можливостями.

Tkinter - це вбудована бібліотека Python для створення графічного інтерфейсу користувача (GUI). Вона містить набір віджетів, таких як кнопки, текстові поля, списки та інші, які дозволяють швидко та просто створювати інтерактивні програми. Tkinter також має простий та зрозумілий синтаксис, що робить його доступним для новачків у програмуванні.

Для системи пошуку короткого маршруту електростанції, бібліотека Pygame використовується для відображення графу, вершин та ребер у вигляді графічних елементів. Tkinter же використано для створення інтерфейсу користувача для введення параметрів системи та відображення результатів пошуку короткого маршруту.

Усі ці бібліотеки, Pygame, Tkinter та NetworkX [18-21], використані разом в системі пошуку короткого маршруту електростанції для створення повноцінного графічного додатку на основі алгоритму Дейкстри.

3.4.1 Опис логічної схема роботи програми

На рис. 3.4 зображено логічну схему програми. На ній чітко видно 3 блоки:

– завантаження та генерування даних, з якими надалі працюватиме програма;

– цикл обробки замовлень і розподілу транспортних засобів (на малюнку зліва);

– перевірка поточного транспортного засобу (на малюнку праворуч).

У першому блоці визначається кількість транспортних засобів, яка може обслуговувати заявки. Його задає користувач.

Далі відбувається зчитування даних про транспортну мережу (дороги), місце розташування електро станцій і складу з шейпфайлів. Структура шейпфайла наведена в додатку 3. Надалі ці дані приводяться в потрібну форму і організовуються у вигляді декількох масивів.

Блок завершується процесом генерації замовлень від власника ТЗ і впорядкування їх за часом надходження.

Другий блок полягає в тому, що при переході на нове замовлення. Якщо під час надходження замовлення за час навантаження ще одного замовлення відбувається оцінка ситуації в плані, чи можна їх обидва обслужити однією станцією так, щоб другий ТЗ, яка зробила замовлення, не чекала більше

допустимого часу. Залежно від вибору транспортний засіб вирушає або за маршрутом до двох електростанцій, або тільки до першої, яка подала заявку.

Якщо ж під час надходження замовлення вільних електростанцій немає, то замовлення отримує відмову.

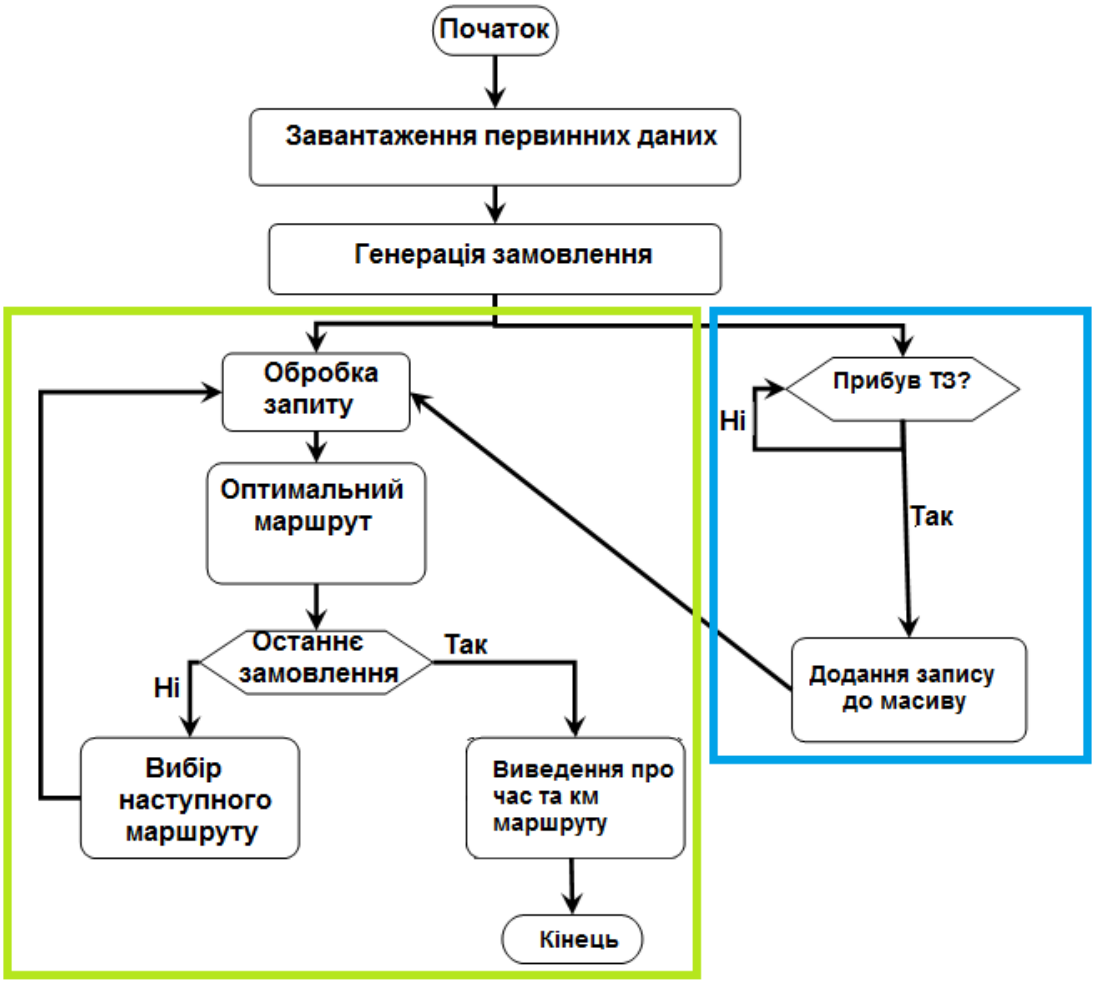

Рис.3.4. Логічна схема роботи ІС

Виконання третього блоку відбувається паралельно з другим. У ньому програма постійно перевіряє, чи не має запиту від власника ТЗ. У разі повернення ТЗ вона одразу ж вирушає і встановлюється в стан готовності.

Програма завершується, коли всі замовлення будуть зроблені.

3.4.2 Організація даних інформаційної системи

Повний код програми наведено в додатку А. Однак слід пояснити, у якій формі програма зберігає дані (опис типів записано в модулі uTypes_Proc.pas):

Спочатку дані про транспортну мережу (дороги), аптеки і склад задано у вигляді географічних координат, тобто в дійсних числах. Під час відображення на екрані цей формат незручний, тому координати приводять до типу TPoint, де X і Y представлені цілими числами.

Дані про аптеки зберігаються у вигляді масиву. Для того, щоб створити масив точок, необхідно спочатку створити клас для точок [19] як на рис.3.5.

```
class MyPoint:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

Рис. 3.5. Клас точок

Цей клас містить два поля - координати x та y.

Далі, щоб створити масив точок, можна використати стандартний тип списку як на рис.3.6.

```
point_array = [MyPoint(1, 2), MyPoint(3, 4), MyPoint(5, 6)]
```

Рис. 3.6. Масив точок

Цей код (рис.3.6) створить масив точок із трьома елементами, кожен з яких містить об'єкт класу MyPoint. Для доступу до координат x та y точки з масиву можна використати індексування списку та поля об'єкту як на рис.3.7.

```
x = point_array[0].x
y = point_array[0].y
```

Рис. 3.7. Об'єкт класу MyPoint

Таким чином, масив точок можна визначити як список об'єктів класу MyPoint.

Масив електростанцій представлено у вигляді списку об'єктів класу MyPoint_E зі словником (dict), який міститиме масив точок та індекс [20] як на рис.3.8.

```
class MyPoint_E:
    def __init__(self, point, index):
        self.point = point
        self.index = index
```

Рис.3.8. Об’єкт класу MyPoint_E

У цьому випадку конструктор класу приймає два аргументи: масив точок point та індекс index.

Далі створтуj масив електростанцій у вигляді списку об'єктів класу MyPoint_E [21] як на рис.3.9.

```
from typing import List
# Визначаємо тип точки (використовуємо кортеж для зберігання координат)
TPoint = tuple[float, float, float]
class MyPoint_E:
    def __init__(self, point: List[TPoint], index: int):
        self.point = point
        self.index = index
# Створюємо масив електростанцій
SkladN = [
    MyPoint([(1.0, 2.0, 3.0), (4.0, 5.0, 6.0), (7.0, 8.0, 9.0)], 0),
    MyPoint([(10.0, 11.0, 12.0), (13.0, 14.0, 15.0), (16.0, 17.0, 18.0)], 1),
    MyPoint([(19.0, 20.0, 21.0), (22.0, 23.0, 24.0), (25.0, 26.0, 27.0)], 2)
]
```

Рис. 3.9. MyPoint_E

Створтуj класи MyLine і LineSHP_norm, які містять поля зі вказаної структури даних, а також створили масив RoadN, який містить всі вулиці. Для додавання нової вулиці до масиву ми створюємо новий екземпляр класу LineSHP_norm і додаємо його до масиву RoadN [21] як на рис.3.10.

Структура даних для зберігання інформації про дороги

```
class MyLine:
    def __init__(self, parts, index):
        self.parts = parts
        self.index = index
class LineSHP_norm:
    def __init__(self, x_min, y_min, x_max, y_max, lines):
        self.x_min = x_min
        self.y_min = y_min
        self.x_max = x_max
        self.y_max = y_max
        self.lines = lines
# Масив вулиць
RoadN = []
# Додавання вулиці до масиву
road = LineSHP_norm(x_min, y_min, x_max, y_max, lines)
RoadN.append(road)
```

Рис. 3.10. LineSHP_norm

Масиви *RoadN*, *PharmaN* і запис *Estan* використовуються для візуалізації доріг і місця розташування електростанцій та автівок на екрані як на рис.3.11.

Масив доріг:

```
RoadN = []
```

Масив електростанцій:

```
PharmaN = []
```

Запис про станцію:

```
Estan = {}
```

Матриця вузлових точок доріг:

```
MasID = []
```

Список ребер транспортного графа:

MS = []

Масив маршрутів

Route = []

Запис ID точки:

class ID:

def init(self, point, smezn):

self.Point = point

self.Smezn = smezn

Запис для вулиць як:

class LineSHP_norm:

def init(self, x_min, y_min, x_max, y_max, lines):

self.XMin = x_min

self.YMin = y_min

self.XMax = x_max

self.YMax = y_max

self.Lines = lines

Запис для лінії:

class MyLine:

def init(self, parts, index):

self.Parts = parts

self.Index = index

Запис для ребра транспортного графа:

class Rebro:

def init(self, a, b):

self.A = a

self.B = b

Запис для маршруту:

class Marshr:

def init(self, length, mas):

self.Length = length

self.Mas = mas

 # Метод для додавання точки до маршруту

def add_point(self, point):

self.Mas.append(point)

Рис. 3.11. Масиви RoadN, PharmaN і запис Estan

Поле *Length* містить довжину цього маршруту, а поле *Mas* - список координат вершин, через які проходить маршрут (також з урахуванням масштабу відображення) (рис.3.12).

class Marshr:

def init(self, Length, Mas):

self.Length = Length

self.Mas = Mas

Дані про замовлення від власників автівок зберігаються у форматі

ZakazMas = []

class Zakaz:

def init(self, Time, Pharma_N):

self.Time = Time

self.Pharma_N = Pharma_N

ZakazMas.append(self)

І нарешті, масив транспортних засобів має такий вигляд

TrackMas = []

class Track:

def init(self, On_sklad, Time_return):

self.On_sklad = On_sklad

self.Time_return = Time_return

TrackMas.append(self)

Рис.3.12. Поле Length

3.5 Програмна реалізація алгоритму для пошуку найкоротшого шляху

Як уже згадувалося вище, під час побудови найкоротших маршрутів пересування транспортних засобів використовувався алгоритм Дейкстри. Його ідея полягає в тому, щоб розглядати вершини графу в порядку зростання довжини шляху від початкової вершини до них та оновлювати вагу найкоротшого шляху до кожної вершини за допомогою ребер, що йдуть з неї. Реалізовано в Python як [21] на рис.3.13.

```
def Marshroute(P1):
    Use = [False] * (GlobID+1)
    LngRoute = [float('inf')] * (GlobID+1)
    Previous = [0] * (GlobID+1)

    for i in range(GlobID+1):
        LngRoute[i] = float('inf')
        Use[i] = False
        Previous[i] = 0

    LngRoute[P1] = 0
    Cur = P1
    Use[P1] = True

    while True:
        for i in range(len(MasId[Cur].Smezn)):
            if LngRoute[MasId[Cur].Smezn[i]] > (LngRoute[Cur] +
                Dlina(MasId[Cur].Point, MasId[MasId[Cur].Smezn[i]].Point)):
                LngRoute[MasId[Cur].Smezn[i]] = LngRoute[Cur] + \
                    Dlina(MasId[Cur].Point, MasId[MasId[Cur].Smezn[i]].Point)
                Previous[i] = Cur
```

```
    Min = float('inf')
    k = -1
    for i in range(len(MasID)):
        if not Use[i] and Min > LngRoute[i]:
            k = i
            Min = LngRoute[i]

    if k != -1:
        Use[k] = True
        Cur = k

    flagALL = True
    for i in range(len(PharmaN)):
        if not Use[i]:
            flagALL = False
            break
    if flagALL:
        break

for i in range(len(PharmaN)):
    if P1 == i:
        Route[P1][i].Length = 0
    else:
        Route[P1][i].Length = LngRoute[i] * koef
        Route[P1][i].Mas[0] = [MasID[i].Point]

        j = i
        while j != P1:
            Route[P1][i].Mas[0].append(MasID[Previous[j]].Point)
```

j = Previous[j]

Route[P1][i].Mas[0].append(MasID[P1].Point)

Рис.3.13. алгоритм Дейкстри

Ця функція, Marshroute(P1), реалізує алгоритм Дейкстри для пошуку найкоротшого шляху від початкової точки P1 до всіх інших точок списку PharmaN.

Спочатку функція ініціалізує три списки: Use, LngRoute і Previous. Use - це логічний список для відстеження того, які вузли були відвідані, LngRoute - це список для відстеження найкоротшої відстані від P1 до кожного вузла, а Previous - це список для відстеження попереднього вузла на найкоротшому шляху.

Потім функція встановлює відстань від P1 до себе рівною 0 і позначає її як відвідану. Потім вона входить у цикл while, щоб продовжити ітерацію, поки не будуть відвідані всі вершини.

У циклі функція спочатку переглядає всі сусідні вершини до поточної вершини Cur і оновлює їх LngRoute, якщо відстань до них через Cur коротша, ніж їх поточний LngRoute. Він також оновлює попередню вершину до Cur.

Потім він знаходить невідвідану вершину з найкоротшим LngRoute і встановлює її як нову поточну вершину Cur. Якщо всі вузли було відвідано, функція виходить з циклу while.

Після завершення циклу функція обчислює фактичний маршрут для кожної вершини, проходячи по списку Previous від кінцевої до початкової вершини і записуючи шлях у масив Route.

Нарешті, функція повертає матрицю Route з найкоротшим шляхом і відстанню для кожного вузла.

Під час обчислення точок перетину ліній для подальшого складання з них матриці суміжності використовується метод Крамера.

Метод Крамера (Крамера правило) - спосіб розв'язання квадратних систем лінійних алгебраїчних рівнянь з ненульовим визначником основної матриці (причому для таких рівнянь розв'язок існує і єдиний).

Опис методу: Якщо система має n рівнянь з n невідомими, то метод Крамера полягає у знаходженні значення кожної невідомої шляхом обчислення визначників.

Для системи n лінійних рівнянь з n невідомими (над довільним полем) за формулоюю (3.1) [14]:

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n = b_1, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + \ldots + a_{nn}x_n = b_n \end{cases} \tag{3.1}$$

Містить визначник матриці системи Δ, відмінним від нуля, розв'язок записується у вигляді формули (3.2) [15]:

$$x_i = \frac{\Delta_i}{\Delta}, \tag{3.2}$$

де

$$\Delta_i = \begin{vmatrix} a_{11} \ldots a_{1,i-1} b_1 a_{1,i+1} \ldots a_{1n} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1} \ldots a_{n,i-1} b_n a_{n,i+1} \ldots a_{nn} \end{vmatrix}, \tag{3.3}$$

де і-й стовпець матриці системи замінюється стовпцем вільних членів.

Лістінг методу мовою Python наведено на рис.3.14.

```
for i in range(len(RoadN)):
for j in range(len(RoadN[i].Lines)):
for k in range(len(RoadN[i].Lines[j].Parts[scale])-1):
P1 = RoadN[i].Lines[j].Parts[scale][k]
P2 = RoadN[i].Lines[j].Parts[scale][k+1]
deltaX1 = P2.X - P1.X
deltaY1 = P2.Y - P1.Y
A1 = deltaY1
B1 = -deltaX1
C1 = P1.X * deltaY1 - P1.Y * deltaX1
```

```
for m in range(len(RoadN)):
        for n in range(len(RoadN[m].Lines)):
            for q in range(len(RoadN[m].Lines[n].Parts[scale])-1):
                if m == i and n == j and k == q:
                    continue

                P3 = RoadN[m].Lines[n].Parts[scale][q]
                P4 = RoadN[m].Lines[n].Parts[scale][q+1]
                deltaX2 = P4.X - P3.X
                deltaY2 = P4.Y - P3.Y
                A2 = deltaY2
                B2 = -deltaX2
                C2 = P3.X * deltaY2 - P3.Y * deltaX2

                _d1 = C1*B2 - C2*B1
                _d2 = A1*C2 - A2*C1
                _d = A1*B2 - A2*B1

                if _d != 0:
                    P.x = round(_d1/_d)
                    P.y = round(_d2/_d)
```

Рис. 3.14. Лістінг

Опис лістингу. Зовнішній цикл перебирає всі дороги у списку RoadN.

Другий цикл перебирає всі лінії на кожній дорозі.

Третій цикл перебирає всі частини (сегменти) в кожному рядку.

У третьому циклі код обчислює дві кінцеві точки поточного відрізка (позначені P1 і P2) і коефіцієнти рівняння лінії у вигляді Ax + By = C (позначені A1, B1 і C1). Потім код входить в інший набір циклів, які знову перебирають всі дороги, лінії та частини, цього разу для знаходження всіх

інших відрізків, які можуть перетинатися з поточним відрізком. Однак код пропускає сам поточний відрізок (використовуючи інструкцію continue), щоб уникнути пошуку самоперетинів.

Усередині цих циклів код обчислює кінцеві точки та лінійні коефіцієнти інших відрізків (позначені P3, P4, A2, B2 та C2).

Потім код використовує коефіцієнти обох прямих для розв'язання системи рівнянь за правилом Крамера, що дає точку перетину (позначено P). Якщо знаменник розв'язку дорівнює нулю, то прямі паралельні і точка перетину не обчислюється.

Нарешті, точка перетину P додається до списку точок перетину.

Таким чином, виконується обчислення всіх точок перетину різних відрізків доріг та зберігає їх у списку.

Для замовлень запрограмовано вираз:

*tmp = round(time_wait[i] + (-1)**random.randint(0, 1) * random.randint(0, time_wait[i]))*

Цей вираз означає, що час очікування на нове замовлення обчислюється як сума середнього проміжку часу між замовленнями *(time_wait[i])* та випадкового числа, яке може бути додатнім або від'ємним. Це випадкове число обчислюється з використанням функції *random.randint(0, 1)*, яка генерує випадкове ціле число 0 або 1, та функції *random.randint(0, time_wait[i])*, яка генерує випадкове ціле число в діапазоні від 0 до *time_wait[i]*.

3.6 Інтерфейсний модуль інформаційної системи

Розробка користувацького інтерфейсу для логістики станцій електрозарядки має декілька важливих переваг, які опишимо нижче.

Зручність користування. Користувацький інтерфейс може зробити пошук короткого маршруту більш зручним та інтуїтивно зрозумілим для користувачів. Наприклад, додавання кнопок та текстової інформації на екран може допомогти користувачам зрозуміти, які опції доступні для пошуку короткого маршруту та як їх використовувати.

Ефективність. Користувацький інтерфейс може допомогти зробити пошук короткого маршруту більш ефективним. Наприклад, використання віджетів та інших елементів користувацького інтерфейсу може дозволити користувачам швидко вибрати параметри пошуку та отримати результати без зайвих затримок.

Стабільність. Користувацький інтерфейс може допомогти зробити пошук короткого маршруту більш стабільним та надійним. Наприклад, використання елементів інтерфейсу, які надають інформацію про поточний стан пошуку, може допомогти користувачам зрозуміти, чому пошук займає певний час та які є можливі проблеми.

Надійність. Користувацький інтерфейс може допомогти зробити пошук короткого маршруту більш надійним та точним. Наприклад, використання елементів інтерфейсу, які надають інформацію про термін дії та точність результатів пошуку, може допомогти користувачам зрозуміти, наскільки надійними є результати пошуку та чи є необхідність у подальшому уточненні параметрів.

Отже, розробка користувацьк кого інтерфейсу для пошуку короткого маршруту може покращити якість та ефективність роботи з програмою. Також, він може допомогти залучити більше користувачів до використання програми, оскільки зручний та привабливий інтерфейс є важливим фактором при виборі програми для використання.

Крім того, користувацький інтерфейс може допомогти зменшити кількість помилок, які можуть виникати під час пошуку короткого маршруту. Наприклад, використання інтерактивних елементів, таких як кнопки та вікна з підказками, може допомогти користувачам зрозуміти, як правильно виконувати певні дії та уникнути помилок.

Користувацький інтерфейс також може бути корисним для покращення зворотного зв'язку між користувачами та розробниками програми. За допомогою елементів інтерфейсу, таких як форми зворотного зв'язку та опитування, користувачі можуть надати свої коментарі та пропозиції щодо

поліпшення програми. Це може допомогти розробникам зрозуміти, які аспекти програми потребують поліпшення та зробити програму ще кращою для користувачів.

Таким чином, розробка користувацького інтерфейсу для логістики станцій електрозарядки може покращити якість та ефективність програми, зробити її більш зручною та привабливою для користувачів, зменшити кількість помилок та забезпечити зворотний зв'язок між користувачами та розробниками.

Для розробки користувальницького ігтерйесу обрано бібліотеку Pygame на мові програмування Python. Щоб створити інтерфейс необхідно спочатку створити головне вікно програми (рис.3.15).

```
import pygame
# Розмір вікна
WINDOW_WIDTH = 944
WINDOW_HEIGHT = 545
# Ініціалізуємо pygame
pygame.init()
# Створюємо вікно
screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
# Головний цикл програми
while True:
    # Обробка подій
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
    # Оновлення екрану
    pygame.display.update()
```

Рис.3.15. Інтерфейс в Pygame

Далі, для створення карти з електрозарядками використовуємо файли форми (shapefiles) - це формат для зберігання географічної інформації, такої як лінії, полігональні області та точки. У контексті картографування маршруту shape файли використовуються для зберігання географічних об'єктів, таких як дороги.

Для використання shape файлів в картографічному додатку, такому як pygame, використано бібліотеку, таку як PyShp. Ця бібліотека дозволяє читати та записувати shape файли в коді Python, що дозволяє програмісту здійснювати різноманітні операції з географічною інформацією.

Картографування маршруту здійснено за допомогою, описаного вище алгоритму, таких як алгоритми пошуку найкоротшого шляху або маршрутів, що враховують різні обмеження, такі як наявність транспорту або обмеження на швидкість руху. Shape файли можуть бути використані для побудови цих маршрутів, дозволяючи вказати географічні обмеження та шляхи.

У додатку з pygame, використовуючи shape файли, можна зображати маршрут на карті з допомогою ліній або полігонів, які показують шляхи руху, а також зображати точки відстаней, які можуть бути використані для розрахунку маршруту. Файли форми можуть бути корисним інструментом для картографування маршруту та роботи з географічною інформацією в картографічних додатках.

Користувальницький інтерфейс (UI) в pygame може містить наступні компоненти:

– головне вікно – це вікно містить карту з електрозарядкою та модуль "Toolbar";

– модуль "Toolbar" – це модуль, що містить різні кнопки, за допомогою яких користувач може взаємодіяти з грою. Кнопки можуть містити зображення та текст, щоб допомогти користувачу зрозуміти їх призначення;

– кнопка "Додати точку автомобіля" – ця кнопка дозволяє користувачеві додавати нові точки на карту, що можуть бути використані як місце розташування автомобіля;

– кнопка "Додати електростанцію" – ця кнопка дозволяє користувачеві додавати нові електростанції на карту;

– кнопка "Переміщувати об'єкти" – ця кнопка дозволяє користувачеві переміщувати різні об'єкти на карті, такі як автомобілі та електростанції;

– кнопка "З'єднати електростанцію та автомобіль" – ця кнопка дозволяє користувачеві з'єднувати електростанцію та автомобіль, що дозволяє автомобілю отримувати електрозаряд;

– кнопка "Розрахувати короткий маршрут" – ця кнопка дозволяє користувачеві розрахувати короткий маршрут між двома точками на карті, що враховує наявність електростанцій та шляхи руху.

На рис.3.16 наведено інтерфейс програми після її запуску.

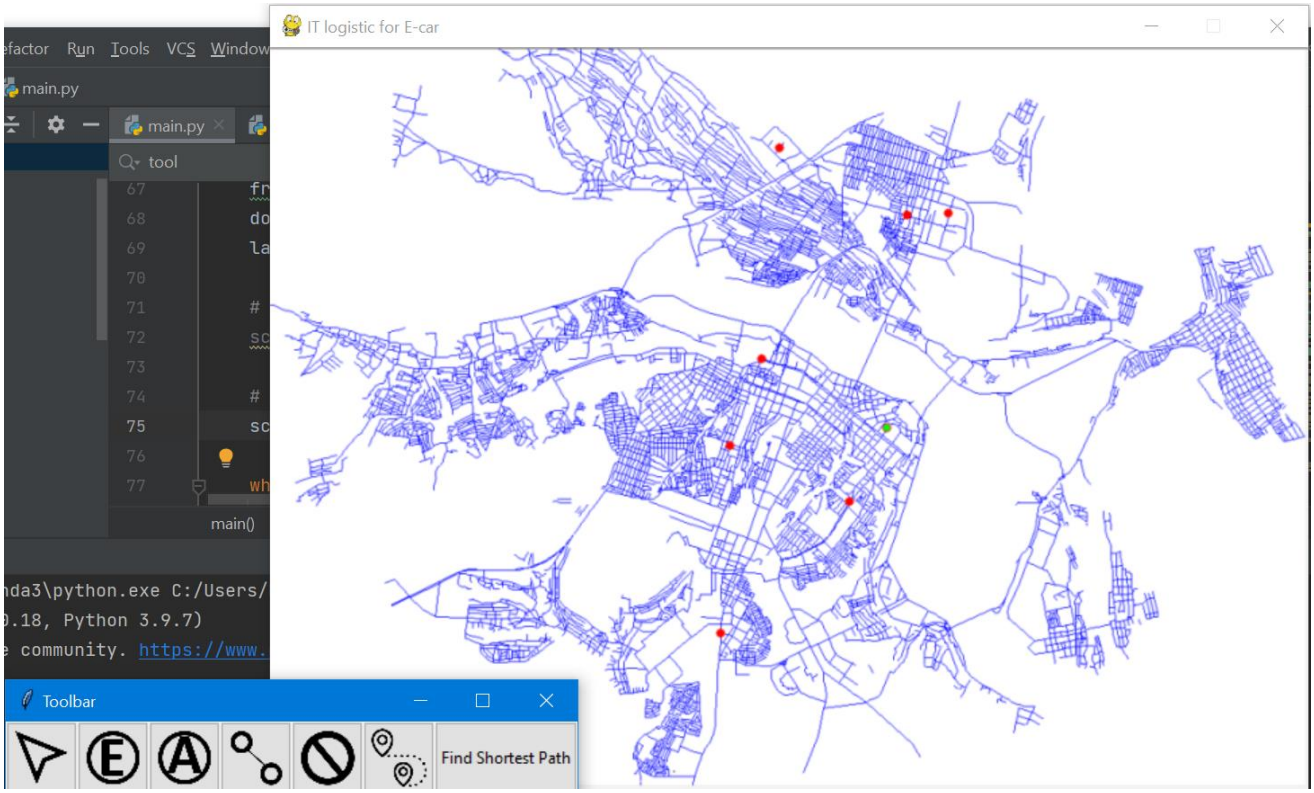

Рис.3.16. Головне вікно ІС

Як видно з рис.3.17 карта м. Дніпро завантажується відразу після запуску програми і представлена у вигляді дорожньої карти. Червоні маркери – це нанесені електро станції для зарядки автомобілів.

Для управління з програмною потрібно використати модуль "Toolbar". На тискаємо на графічну кнопку «*E*», шо позначає електростанції та проставляємо станції на карту. Має вид круглих маркерів і схоже на вершини графа. Далі обираємо графічну кнопку «*A*», що позначає автомобіль та ставимо на карті точку автомобіля з якої власник бажає доїхати до станції. Отримуємо результат як на рис.3.17.

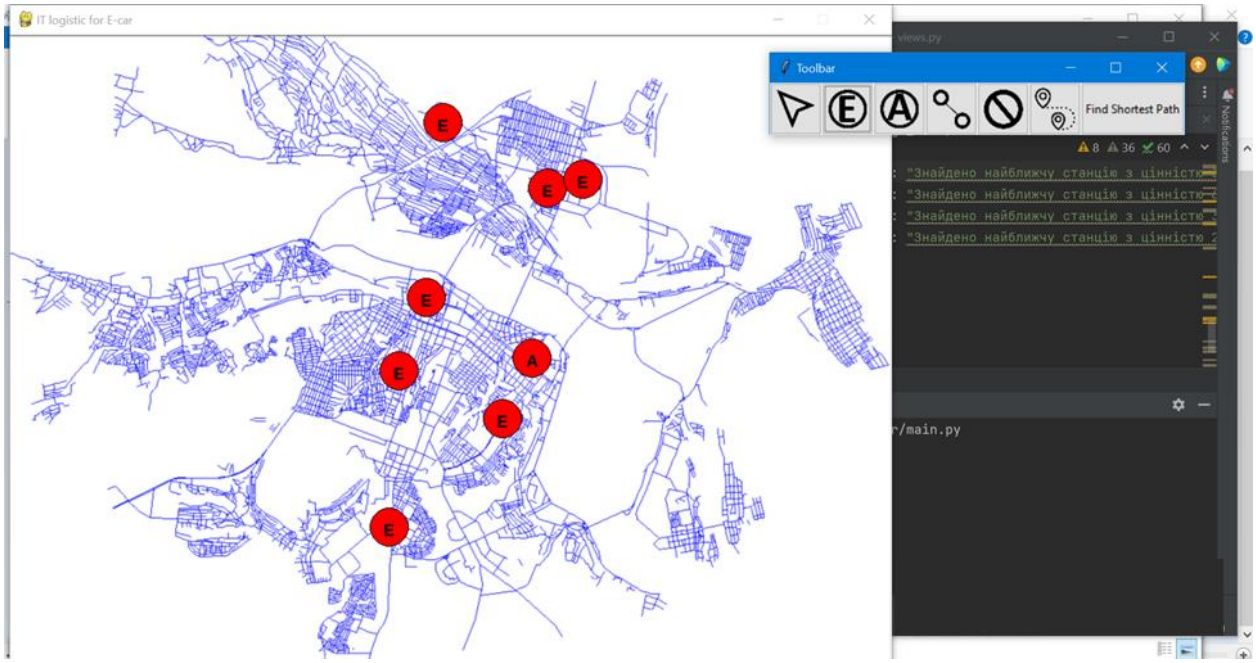

Рис.3.17 – Нанесення електро станцій та авто на карту

Наступним кроком потрібно з’єднати всі вершини графа, в нашому випадку це авто та електро станції. Для цього в модулі "Toolbar" обираємо піктограму у вигляді двох кол, що з’єднанні лінією, і починаємо будувати лінії. Результат наведено на рис.3.18.

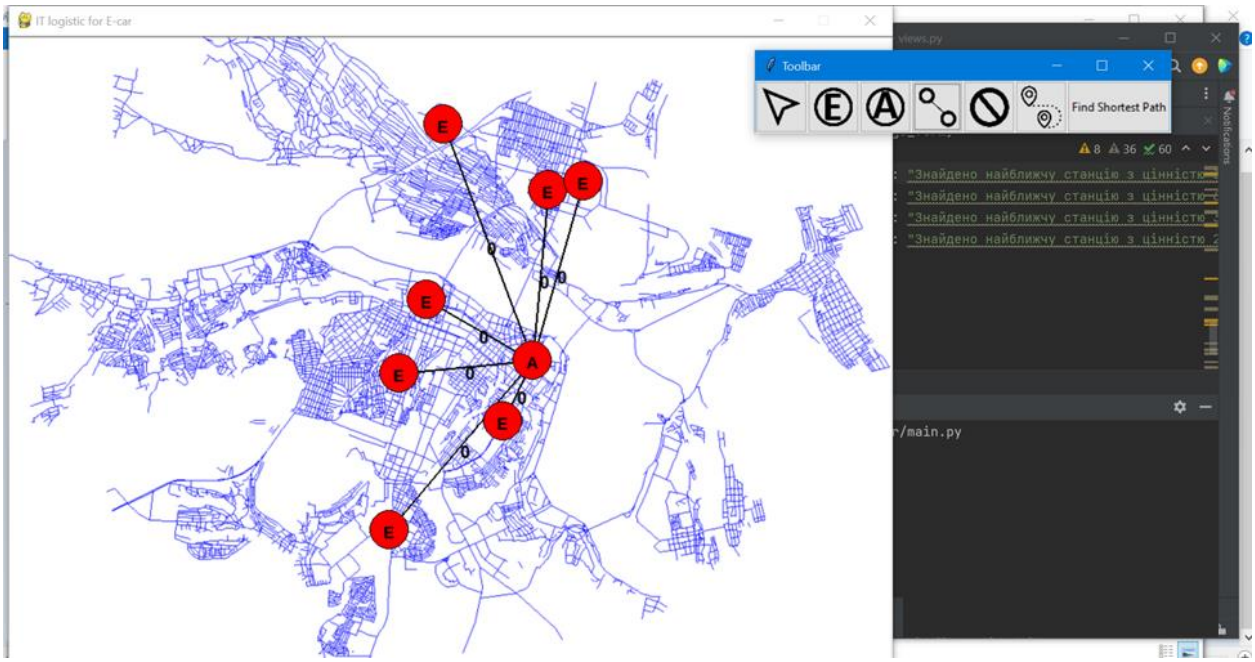

Рис. 3.18. Побудова з’єднань

Після того як підготовчий процес виконано, можна натиснути на кнопку «*Find Shortest Path*», що означає пошук короткого шляху. В результаті вікривається вікно, в якому потрібно обрати авто та електро станцію та натиснути на кнопку «Calculate». На головній формі виділяється червоним кольором короткий шлях (рис.3.19).

Рис. 3.19. Пошук короткого шляху

Результат рис.3.20 демонструє роботу алгоритму Дейкстри у вигляді графа. Але щодо власника авто не зовсім зрозуміло як рухатись до електро станції. Тому, для побудови маршруту потрібно обрати графічну піктограму у вигляді маршруту. Пісдя цього відкривається вікно з побудовою коротко шляху на основі shape файлів та виводить повідомлення щодо адреси, часу та км. Результат наведено на рис.3.20.

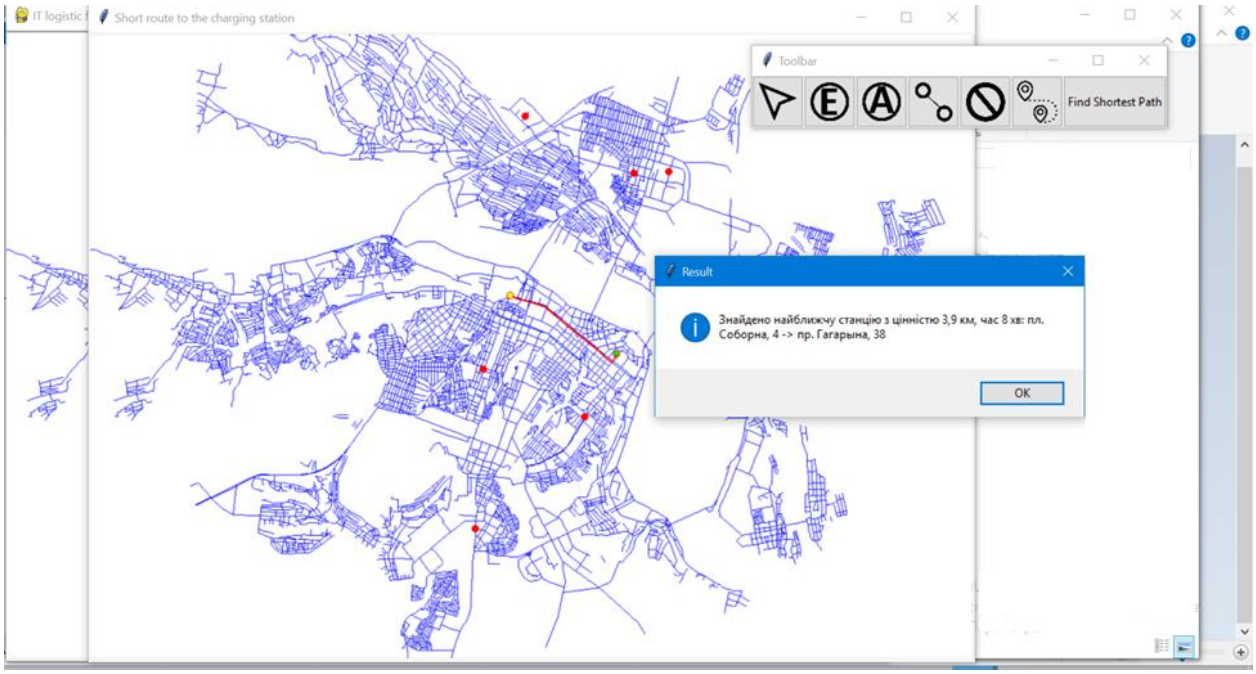

Рис. 3.20. Результат роботи ІС

Система може опрацювати максимально 5 запитів одночасно і побудувати маршрут до електростанцій як показано на рис.3.20. Але графічно у вигляді графа покаже найкоротший маршрут серед запитів із 5 ТЗ. Аналогічно після того як натискаємо на кнопку побудувати маршрут, система виводить повідомлення про один найкоротший маршрут із запропонованих запитів (рис.3.21). Результати тестування системи наведено в додатку Б.

Рис. 3.20. Нанесення на карту 5 ТЗ

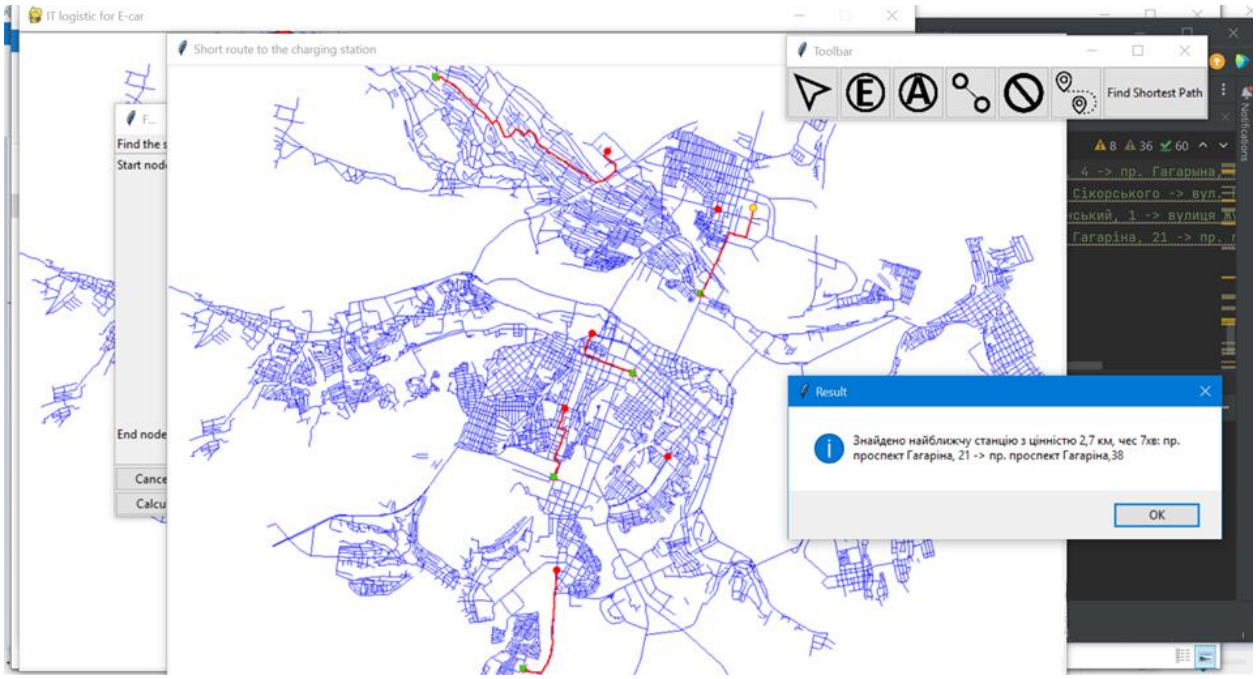

Рис. 3.21. Результат системи для 5 ТЗ

Для зручності роботи з ІС було створено exe-файл програми, з використанням інструменту pyinstaller.

Для створення exe-файлу за допомогою pyinstaller виконано наступні кроки:

– встановилии pyinstaller за допомогою pip.

– запустили pyinstaller, вказавши шлях до головного файлу програми;

– pyinstaller створює папку "dist" в кореневій папці проекту, в якій буде знаходитися exe-файл програми.

3.7 Результати виконання інформаційної системи

Результатом роботи програмного коду ІС пошуку короткого маршруту до електростанції є знайдений маршрут до електростанції з вказаної точки вхідного списку координат. Цей маршрут представлений у вигляді списку координат вершин, які потрібно відвідати, щоб дістатися до електростанції з найменшою загальною відстанню.

Вихідні дані програми включають:

– інформацію про знайдені найкоротші маршрути до електростанції з вказаної точки, які містять список координат вершин, через які проходить маршрут і довжину цього маршруту відстань;

– час, який потрібно для подолання маршруту з вказаної точки до електростанції;

– повний список координат вершин, що утворюють маршрут, включаючи початкову та кінцеву точки;

– список дій, які потрібно виконати, щоб дістатися до електростанції у вигляді картографування маршруту;

– рекомендації щодо раціональності використання знайденого маршруту з огляду на інші фактори (наприклад, , рівень заторів тощо).

ВИСНОВКИ

В даній кваліфікаційні роботі ОКР бакалавр розроблено інформаційну систему логістики станцій зарядки електромобілів у місті Дніпро на мові Python. Основними висновками роботи є:

1. Виконано огляд літератури з питання актуальності використання електростанцій в м. Дніпро.

2. Розроблена інформаційна система логістики станцій зарядки електромобілів є корисним інструментом для оптимізації роботи електрозаправок у місті Дніпро.

3. Використання бібліотеки Pygame дозволило створити інтуїтивно зрозумілий та простий у використанні інтерфейс користувача.

4. Використання алгоритму Дейкстри дозволило щвидко найти короткий маршрут до заправки з вказанням часу та нанесення результату на дорожню векторну карту м. Дніпро.

5. Результати тестування інформаційної системи показали її ефективність.

6. Інформаційна система логістики станцій зарядки електромобілів може бути використана в інших містах та країнах з метою підвищення ефективності роботи електрозаправок та покращення сервісу для користувачів електромобілів.

У майбутньому можна розглянути можливості розширення функціональності інформаційної системи, включаючи в неї функції моніторингу стану запасів електроенергії та додаткові аналітичні засоби.

ПЕРЕЛІК ПОСИЛАНЬ

1. Асоціація «Українська Асоціація електромобілів» (УкрАЕМ) [Електронний ресурс] – Режим доступу до ресурсу: https://ukrautoprom.com.ua/za-rik-ukrayinskyj-rynok-elektromobiliv-zris-u-pivtora-raza

2. Програма «Електромобільна Україна» [Електронний ресурс] – Режим доступу до ресурсу: https://www.kmu.gov.ua/news/ukravtodor-zatverdiv-plan-zahodiv-z-pidtrimki-elektromobilnosti

3. Інформаційна система логістики станцій зарядки електромобілів PlugShare [Електронний ресурс] – Режим доступу до ресурсу: https://www.plugshare.com/uk

4. Програма «Green Loan» [Електронний ресурс] – Режим доступу до ресурсу: https://www.worldbank.org/en/news/feature/2021/10/04/what-you-need-to-know-about-green-loans

5. T. D. Nguyen, N. T. Nguyen, M. D. Nguyen, and T. Q. Hoang, "Design and implementation of a web-based logistics management system for electric vehicle charging stations," International Journal of Engineering Business Management, vol. 10, pp. 1-13, 2018.

6. D. Yu, C. Zhang, Y. Shao, and Y. Zhang, "An optimization model for the design of electric vehicle charging stations based on the customer's charging behavior," IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 8, pp. 2701-2713, 2018.

7. Інформаційна система логістики станцій зарядки електромобілів Tesla Supercharger [Електронний ресурс] – Режим доступу до ресурсу: https://www.tesla.com/supercharger

8. Інформаційна система логістики станцій зарядки електромобілів UGV Charge [Електронний ресурс] – Режим доступу до ресурсу: https://ugv.ua/uk/

9. Інформаційна система логістики станцій зарядки електромобілів RODINA Energy Group [Електронний ресурс] – Режим доступу до ресурсу: https://rodinaeg.com/uk/golovna/

10. Інформаційна система логістики станцій зарядки електромобілів GreenFuel [Електронний ресурс] – Режим доступу до ресурсу: https://greenfuel.ua/

11. Інформаційна система логістики станцій зарядки електромобілів EVA+ [Електронний ресурс] – Режим доступу до ресурсу: https://www.evaplus.eu/

12. James H. Banks. Introduction to Transportation Engineering, 2015. McGraw-Hill Education, 640 pages.

13. Paul H. Wright and Norman Ashford. Transportation Engineering: Planning and Design, 2017. Wiley, 832 p.

14. Reinaldo Morabito. Modeling and Simulation of Transportation Networks: Applications for Intelligent Transportation Systems",2016. Springer, 266 p.

15. Robert B. Cooper, Bala Krishnamoorthy, and Philippe Nain. Introduction to Queueing Theory, 2019, CRC Press, 370 c.

16. Adriana Piazza. Deterministic Optimization Models in Finance, 2020, Springer, 170 p.

17. David Gao and Ning Ruan. Deterministic Global Optimization: Theory, Methods and Applications, 2018, Springer, 668 p.

18. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, 2009, MIT Press, 1312 p.

19. Art Lew and Holger Mauch. Dynamic Programming: A Computational Tool, 2011, Springer, 322 p.

20. Eric Matthes. Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 2019. No Starch Press. 544 p.

21. Wes McKinney. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2017. O'Reilly Media. 544 p.

ДОДАТОК А. Фагмент лістингу програми

Фрагмент лістингу **main. py**

```
import pygame

import sys


def exit_program():

pygame.quit()

sys.exit()


def close_window():

pygame.display.quit()

pygame.quit()


def create_window():

pygame.init()

pygame.display.set_caption("Опис інтерфейсної частини інформаційної системи")

screen = pygame.display.set_mode((800, 600))

return screen


def main():

screen = create_window()

clock = pygame.time.Clock()

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:
```

```
close_window()
elif event.type == pygame.KEYDOWN:
if event.key == pygame.K_ESCAPE:
exit_program()
elif event.type == pygame.MOUSEBUTTONDOWN:
if event.button == 1:
# дії по кліку лівою кнопкою миші
pass
screen.fill((255, 255, 255)) # колір фону
# рисування інтерфейсу
# ...
pygame.display.update()
clock.tick(60)


if name == "main":
main()
import pygame
import math
def on_node_hover(graph):
    """
    This method marks the nodes that are hovered onto by the mouse.
    """
    for node in graph.nodes:
        node.hovered = False
        if node.pos[0] <= pygame.mouse.get_pos()[0]+node.radius and \
            node.pos[0] >= pygame.mouse.get_pos()[0]-node.radius and \
```

```
                node.pos[1] <= pygame.mouse.get_pos()[1]+node.radius
and \
                    node.pos[1] >= pygame.mouse.get_pos()[1]-
node.radius:
            node.hovered = True


def main():
    """
    This is the main method of the program
    """



    global RUNNING


    # sets the window position
    os.environ['SDL_VIDEO_WINDOW_POS'] = f"{100},{100}"


    # initialize pygame
    pygame.init()
    pygame.font.init()


    # initialize tkinter
    graph = Graph()
    toolbar = ToolBar(graph)
    toolbar.geometry(
        f'{toolbar.winfo_width()}x{toolbar.winfo_height()}+100+600')
    toolbar.protocol("WM_DELETE_WINDOW", quit_callback)
```

```
    # start pygame clock
    clock = pygame.time.Clock()

    # initialize variables
    font = pygame.font.SysFont(None, 25)
    framerate = 30
    double_click_duration = 150  # ms
    last_click = 0

    # sets the window title
    screen = pygame.display.set_caption('IT logistic for E-car')

    # sets the window size
    screen = pygame.display.set_mode((902, 644))

    while RUNNING:
        for event in pygame.event.get():
            if event.type == QUIT:
                RUNNING = False
            elif event.type == MOUSEMOTION:
                toolbar.tool.handle_mouse_move(event)

                on_node_hover(graph)
            elif event.type == MOUSEBUTTONDOWN:
                now = pygame.time.get_ticks()
                double_click = (now - last_click) <=
double_click_duration
```

```
                last_click = pygame.time.get_ticks()

                toolbar.tool.handle_mouse_down(event, double_click)

            elif event.type == MOUSEBUTTONUP:
                toolbar.tool.handle_mouse_up(event)


        # Rendering
        clock.tick(framerate)
        #screen.fill(BACKGROUND_COLOR)
        screen.blit(BACKGROUND_COLOR, (0,0))
# Функція для визначення точок
def Dlina(p1, p2):
    dx = p1[0] - p2[0]
    dy = p1[1] - p2[1]
    return math.sqrt(dx**2 + dy**2)

# Функція алгоритмом Дейкстри
def Marshroute(P1):
    global Route, MasID, PharmaN, koef, Inf, GlobID

    Use = [False] * (GlobID + 1)
    LngRoute = [Inf] * (GlobID + 1)
    Previous = [0] * (GlobID + 1)

    for i in range(GlobID + 1):
```

```
        LngRoute[i] = Inf
        Use[i] = False
        Previous[i] = 0

    LngRoute[P1] = 0
    Cur = P1
    Use[P1] = True

    flagALL = False
    while not flagALL:
        for i in range(len(MasID[Cur].Smezn)):
            if LngRoute[MasID[Cur].Smezn[i]] > LngRoute[Cur] +
Dlina(MasID[Cur].Point, MasID[MasID[Cur].Smezn[i]].Point):
                LngRoute[MasID[Cur].Smezn[i]] = LngRoute[Cur] +
Dlina(MasID[Cur].Point, MasID[MasID[Cur].Smezn[i]].Point)
                Previous[i] = Cur

        Min = Inf
        k = -1
        for i in range(len(MasID)):
            if not Use[i] and Min > LngRoute[i]:
                k = i
                Min = LngRoute[i]

        if k != -1:
            Use[k] = True
            Cur = k
```

```
        flagALL = True
        for i in range(len(PharmaN)):
            if not Use[i]:
                flagALL = False


    for i in range(len(PharmaN)):
        if P1 == i:
            Route[P1][i].Length = 0
        else:
            Route[P1][i].Length = LngRoute[i] * koef
            Route[P1][i].Mas = [[] for j in range(3)]
            Route[P1][i].Mas[0].append(MasID[i].Point)
            Route[P1][i].Mas[1].append((round(MasID[i].Point[0]/2),
round(MasID[i].Point[1]/2)))
            Route[P1][i].Mas[2].append((round(MasID[i].Point[0]/4),
round(MasID[i].Point[1]/4)))
            j = i
            while j != P1:
                Route[P1][i].Mas[0].append(MasID[Previous[j]].Point)

Route[P1][i].Mas[1].append((round(MasID[Previous[j]].Point[0]/2),
round(MasID[Previous[j]].Point[1
```

ДОДАТОК Б. Тестування програми

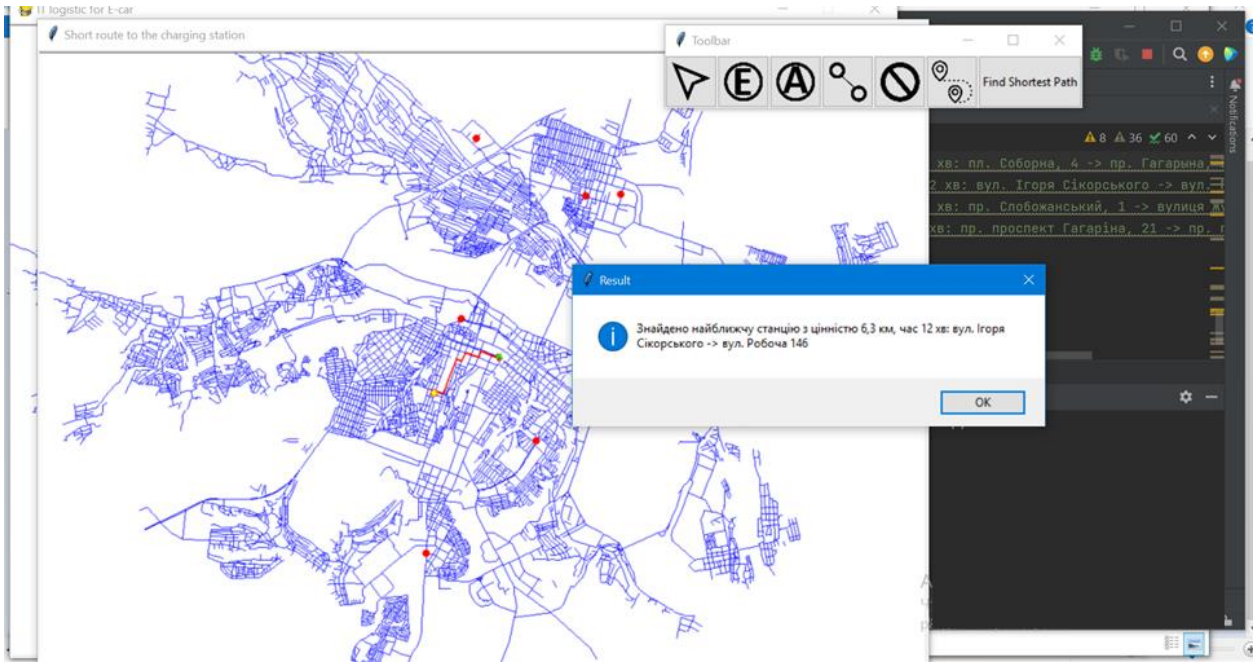

Рис. Б.1 – Результат пошуку короткого шляху

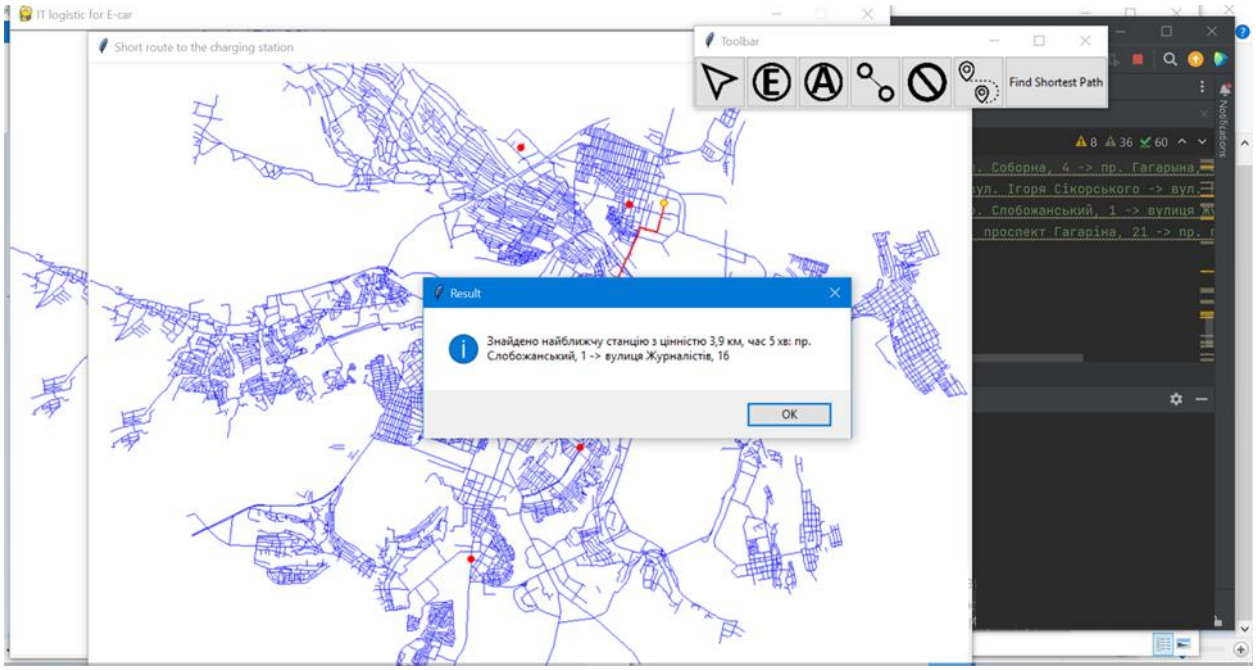

Рис. Б.2 – Результат пошуку короткого шляху