

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ОПТИМІЗАЦІЇ СИСТЕМИ ОБ'ЄКТІВ СОЦІАЛЬНОЇ ІНФРАСТРУКТУРИ

RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS TO

OPTIMIZE THE SYSTEM OF SOCIAL INFRASTRUCTURE FACILITIES

Виконав(ла): студент(ка) 2 курсу, групи КНТ-132м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЛІПІХІН С.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
« » 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЛПІХІНА Сергія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
оптимізації системи об'єктів соціальної інфраструктури
Research and Software Implementation of Methods to Optimize the System of Social
Infrastructure Facilities

керівник проєкту (роботи) к.т.н., доцент, ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «14» листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтролер	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-5 тижні	Розділ 2
4	Розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програми	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
8	Захист роботи	12 тиждень	

Студент(ка)

(підпис)

Сергій ЛІПІХІН

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Валерій ЛЬОВКІН

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
161 с., 46 рис., 21 табл., 3 дод., 41 джерело.

ПУНКТ НЕЗЛАМНОСТІ, ЗАДАЧА ПОКРИТТЯ МНОЖИНИ,
ГЕНЕТИЧНИЙ АЛГОРИТМ, ARCGIS, МОНОРЕПОЗИТОРІЙ.

Об'єкт дослідження – процес оптимізації розташування об'єктів соціальної інфраструктури. Предмет дослідження – програмне забезпечення для пошуку оптимального розташування об'єктів соціальної інфраструктури.

Мета роботи – створення програмного забезпечення геопросторової оптимізації об'єктів соціальної інфраструктури для знаходження найкращого варіанту їх розташування, сприяючи раціональному використанню ресурсів, зменшенню витрат та покращенню якості обслуговування.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мови програмування Python та TypeScript, середовища розробки Spyder та Visual Studio Code, геоінформаційна система ArcGIS, метаевристичні методи, генетичний алгоритм, шаблон проєктування архітектури MVC, монорепозиторий, бібліотека React.js, середовище Node.js, комп'ютер під керівництвом Windows 10 з відеокартою GeForce GTX 1060.

Результати. Наукова новизна роботи полягає в методі пошуку оптимального розташування об'єктів соціальної інфраструктури, що базується на використанні генетичного алгоритму, визначає етапи для формування фінального рішення, застосовує модифікований оператор відновлювання.

Висновки. Розроблено програмний продукт для проведення опису системи соціальної інфраструктури на реальній карті та оптимізації розташування об'єктів такої системи.

Галузь використання – державними установами та приватними організаціями для оптимізації суспільно значимих закладів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 161 pages, 46 figures, 21 tables, 3 appendixes, 41 sources.

POINT OF INVINCIBILITY, SET COVERING PROBLEM, GENETIC ALGORITHM, ARCGIS, MONOREPOSITORY.

Object of study is the process of optimizing the spatial arrangement of social infrastructure objects. Subject of the study is software for finding the optimal location of social infrastructure objects.

The purpose of the work is to create software for geospatial optimization of social infrastructure objects to find the best placement, promoting efficient resource utilization, cost reduction, and improvement in service quality.

Materials, methods and tools: object-oriented programming, functional programming, programming languages Python and TypeScript, development environments Spyder and Visual Studio Code, geoinformation system ArcGIS, set covering problem, metaheuristic methods, genetic algorithm, repair operator, MVC architectural design pattern, monorepository, React.js library, Node.js software environment, personal computer running Windows 10 OS with GeForce GTX 1060 graphics card.

Results: The optimization method for the placement of social infrastructure objects has been implemented, utilizing a genetic algorithm and a modified recovery operator.

Conclusions. Created a software solution designed to delineate social infrastructure systems on actual maps, enhancing the strategic positioning of components within this framework.

The field of use is government institutions and private organizations for optimizing socially significant establishments.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Оптимізація соціальної інфраструктури	11
1.1.1 Соціальна інфраструктура України	11
1.1.2 Пункти незламності	13
1.1.3 Задача покриття множини	17
1.2 Порівняння існуючих методів вирішення задачі покриття множини ...	22
1.4 Постановка задачі	27
1.5 Висновки за розділом 1	28
2 Матеріали і методи	29
2.1 Обробка вихідних даних	29
2.2 Реалізація ГА та модифікація оператора відновлення	31
2.3 Метод пошуку оптимального розташування об'єктів соціальної інфраструктури	35
2.4 Налаштування параметрів ГА	38
2.5 Висновки за розділом 2	42
3 Проєктування програмного забезпечення	44
3.1 Вибір інструментарію розробки програмного забезпечення	44
3.1.1 Аналіз технічного завдання та формування вимог до використовуваних програмних засобів	44
3.1.2 Вибір мови програмування	49
3.1.3 Вибір стратегії розробки	56
3.1.4 Вибір геоінформаційної системи	58
3.2 Структурна схема програми та її опис	60
3.3 Проєктування структури бази даних	65
3.4. Висновки за розділом 3	67
4 Основні рішення щодо реалізації програмного забезпечення	68

	7
4.1 Алгоритми функціонування програмного забезпечення	68
4.2 Опис модулів реалізованої програми	71
4.2.1 Опис клієнтського застосунку	71
4.2.2 Опис модулів для проведення еволюційного пошуку	79
4.3 Висновки за розділом 3	84
5 Експлуатація, тестування та експериментальне дослідження програми	85
5.1 Призначення програми.....	85
5.2 Умови виконання програми	86
5.2.1 Файлова структура.....	87
5.2.2 Наявність зовнішніх залежностей	90
5.2.3 Мінімальні апаратні вимоги.....	94
5.3 Виконання програми	95
5.4 Тестування програми	101
5.4.1 Тестування функціональності програми	101
5.4.2 Тестування моделі на реальному прикладі	103
5.5 Висновки за розділом 5	106
Висновок.....	107
Перелік джерел посилання	108
Додаток А Технічне завдання.....	113
Додаток Б Текст програми.....	132
Додаток В Слайди презентації	155

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CRO	– Chemical Reaction Optimization;
GA	– Genetic Algorithm;
GIL	– Global Interpreter Lock;
GSEM	– Global Simple Evolutionary Multi-Objective Optimizer;
JPSO	– Jumping Particle Swarm Optimization;
JS	– JavaScript;
MVC	– Model View Controller;
SA	– Simulated Annealing;
SEIP	– Stochastic Evolutionary Integer Programming;
SPA	– Single Page Application;
TS	– TypeScript;
БД	– база даних;
ГА	– генетичний алгоритм;
ДСНС	– Державна служба України з надзвичайних ситуацій;
ЗПМ	– задача покриття множини;
ПЗ	– програмний засіб;
ПН	– пункт незламності;

ВСТУП

Соціальна інфраструктура є фундаментальним елементом для країни, оскільки вона включає важливі установи та послуги, необхідні для задоволення основних соціальних, освітніх та медичних потреб населення. Ефективність та доступність соціальної інфраструктури безпосередньо впливають на якість життя громадян та економічний розвиток країни.

Актуальність оптимізації об'єктів соціальної інфраструктури полягає у необхідності максимізації їх ефективності та доступності. Ці дві категорії виражаються через радіус обслуговування, що визначає, які клієнти можуть отримати послуги від певного об'єкта, а які ні. Оптимізація в кінцевому рахунку дозволить краще використовувати ресурси, покращувати якість послуг та забезпечувати більш повний доступ до них.

У контексті України, ця актуальність набуває особливої ваги через триваючу війну з Росією. Військовий конфлікт впливає на структуру та розподіл соціальної інфраструктури у цілому, створюючи потребу в покращенні її ефективності та зниженні коштів на її утримання, з метою вивільнення ресурсів для перемоги у війні. Однак, особливу увагу для вивчення привертають пункти незламності, які були впроваджені восени 2022 року та покликані забезпечити безперервний доступ громадян до основних, життєво необхідних соціальних послуг навіть у кризових умовах.

Об'єктом дослідження є оптимізація розташування об'єктів соціальної інфраструктури. Особливістю задачі є врахування специфічних умов, зокрема інтеграція критично важливих об'єктів, які обов'язково мають бути включені в рішення через їх значення або політичні директиви. Також важливо забезпечити, щоб кожен користувач мав доступ до принаймні одного сервісного центру.

Задача оптимізації розташування вже добре відома в академічній літературі, і вона зазвичай розглядається як задача покриття множинами. Хоча існує багато методів для знаходження оптимального рішення, важливо обрати

метод, який враховує специфіку даної задачі, особливо з урахуванням вимог до пунктів незламності. При цьому основною метою є розробка ефективної функціональності для об'єктів інфраструктури загалом.

Важливою умовою задачі також є розробка такого програмного застосунку, що не просто дозволить автоматизувати проведення оптимізації об'єктів інфраструктури, а також забезпечить можливість проводити це систематично, на прикладі реальних проблем та на основі реальної географічної карти світу.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

В даному розділі розглянуто поняття соціальної інфраструктури України; описано призначення та функції пунктів незламності, а також висвітлено основні проблеми, пов'язані з їхньою оптимізацією в сучасних умовах; розглянуто задачу покриття множини як математичне представлення задачі оптимізації закладів та проаналізовано методи її вирішення. Згідно з актуальними проблемами системи пунктів незламності України була сформульована конкретна задача по їх оптимізації за допомогою генетичного алгоритму.

1.1 Оптимізація соціальної інфраструктури

1.1.1 Соціальна інфраструктура України

Соціальна інфраструктура є важливою складовою життя суспільства. Це система, яка допомагає людям отримувати основні послуги, взаємодіяти між собою та розвиватися як особистість.

Загалом, соціальна інфраструктура розглядається як сукупність об'єктів та служб, які забезпечують функціонування суспільства та задоволення його соціальних потреб. Це включає освітні установи, медичні заклади, культурні та спортивні установи, місця відпочинку та інші об'єкти, що впливають на якість життя громадян [1].

У контексті України соціальна інфраструктура поділяється на два основні рівні [2]:

- I рівень включає галузі, чиї послуги гарантовані громадянам України згідно з Конституцією та які в основному фінансуються за рахунок державного бюджету;

- II рівень об'єднує галузі, які надають свої послуги переважно на комерційній основі. До них відносяться торгівля, громадське харчування, житлово-комунальне господарство та побутове обслуговування [2].

Схематично класифікацію зображено на рисунку 1.1.

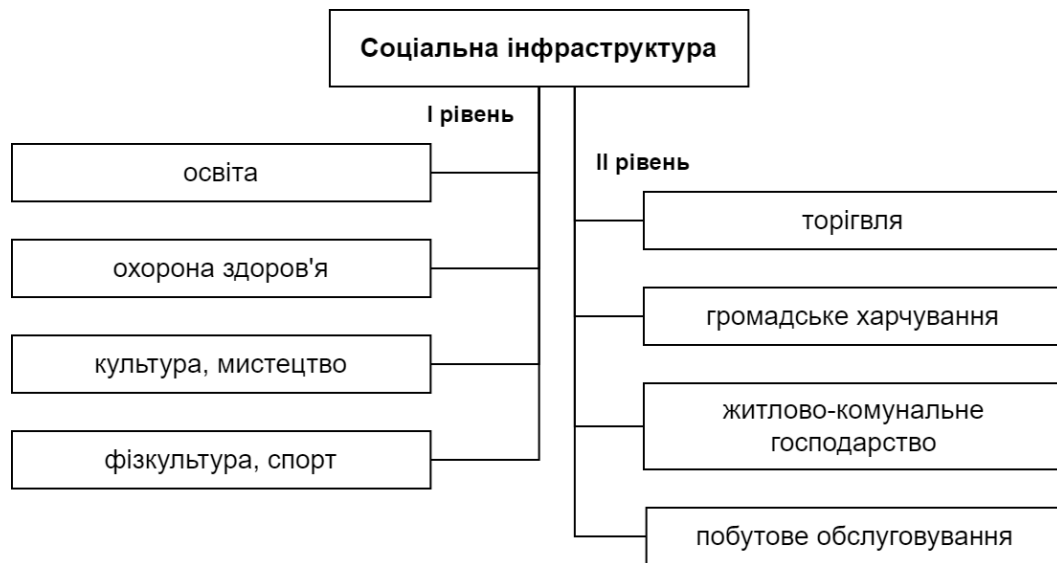


Рисунок 1.1 – Соціальна інфраструктура України

За інформацією Державної служби статистики України, соціальна інфраструктура відіграє ключову роль у розвитку кожної області та населеного пункту. Вона є показником якості життя населення та здатності регіону приваблювати інвестиції [3].

Тут також варто звернути уваги на наступну проблему, яка буде розглянута пізніше, вже в контексті пунктів незламності. Особлива природа інфраструктури, зокрема й на регіональному рівні, зумовлена тим, що економіка вільного ринку в принципі нездатна забезпечувати громадян та економічних агентів певними видами послуг. Цю прогалину за допомогою інших факторів виробництва заповнюють інфраструктурні ланки, якими традиційно опікується публічний сектор [4].

Спираючись на наведені джерела, можна зробити висновок про те, що соціальна інфраструктура є невід'ємною частиною життя суспільства, яка визначає якість життя громадян і впливає на економічний розвиток регіонів.

В умовах постійної динаміки суспільства, зміни демографічної ситуації, технологічних нововведень та економічних викликів, система соціальної інфраструктури потребує постійного адаптування та оптимізації.

Оптимальне розташування та функціонування об'єктів соціальної інфраструктури може забезпечити ефективніше задоволення потреб населення, зниження витрат та підвищення якості наданих послуг [5]. Тому актуальним є питання вдосконалення механізмів управління та оптимізації соціальної інфраструктури, що забезпечить збалансований розвиток територій та підвищення якості життя громадян.

1.1.2 Пункти незламності

Восени 2022 року під час російсько-української війни Росія почала масовані терористичні обстріли української критичної інфраструктури. У відповідь ще в другій половині жовтня в Офісі Президента почали працювати над розгортанням пунктів обігрівання на випадок відключень енергії, однак з посиленням обстрілів цей проєкт збільшився за масштабами та переріс у те, що 22 листопада 2022 року Президент України Володимир Зеленський оголосив про розгортання по всій Україні пунктів незламності [6].

Пункт незламності (ПН) – це місце де люди можуть безкоштовно отримати базові блага, такі як електроенергія, світло та тепло. ПН являє собою приміщення, або намет, яке має відповідне оснащення для задоволення перелічених потреб. ПН – це місце короткострокового одночасного перебування від 40 до 500 осіб, хоча час перебування ніяк не регламентується і не обмежується [7].

Перевага пунктів незламності у їх мобільності, швидкості розгортання та невеликій вартості на один такий пункт – 17 тисяч доларів. Це дозволило швидко і вчасно забезпечити створення 4 тис. пунктів на день їх оголошення, а через рік, на 17 жовтня 2023 року в Україні розгорнуто вже понад 13 тисяч пунктів незламності, у той час коли додатково 2,5 тисячі потребують доукомплектування [8].

Виходячи з наведеної класифікації у попередньому пункті, ПН складно одностайно віднести до I або II групи, адже вони як такі бувають саме двох

відповідних типів. Перший тип – це державні пункти, тобто такі, які створюються на базі Державної служби України з надзвичайних ситуацій (ДСНС), поліції, об'єктів державної власності чи комунальних підприємств, та фінансуються із бюджету [9]. А от другий тип, це так звані ПН відповідального або сумлінного бізнесу, які забезпечують ряд базових потреб, однак ряд їх послуг може бути платний на розсуд власників. До такого типу входять магазини, аптеки, відділення банків, кафе тощо. Таким чином можна виділити першу специфічну проблему ПН, яка буде розглянута і подалі, – різна відповідальність за них.

Проблему саме розташування пунктів незламності можна поділити на три рівні. На першому рівні стверджується, що не всі міста України однорідним чином, чи принаймні в достатній, забезпеченні саме тими пунктами, які є державними. Якщо проаналізувати карту ПН у містах Дніпро та Запоріжжя, до прикладу, то виявиться, що доля саме державних пунктів буде всього лише 17,2% у першому, і аж 85,81% у другому місті. Усе інше займають ПН бізнесу [7].

Проблема другого рівня, говорячи саме про розташування ПН, у нерівномірному розподілі за районами одного окремого взятого міста або області. Цілком природньо, що і оснащення буде розподілено нерівномірним чином. Для доведення неоптимального розташування закладів за районами було проведено аналіз Полтавської області, із перерахуванням кількості ПН та одиниць обладнання на кількість мешканців кожного з районів [10]. Результати такого аналізу зображено на рисунку 1.2.

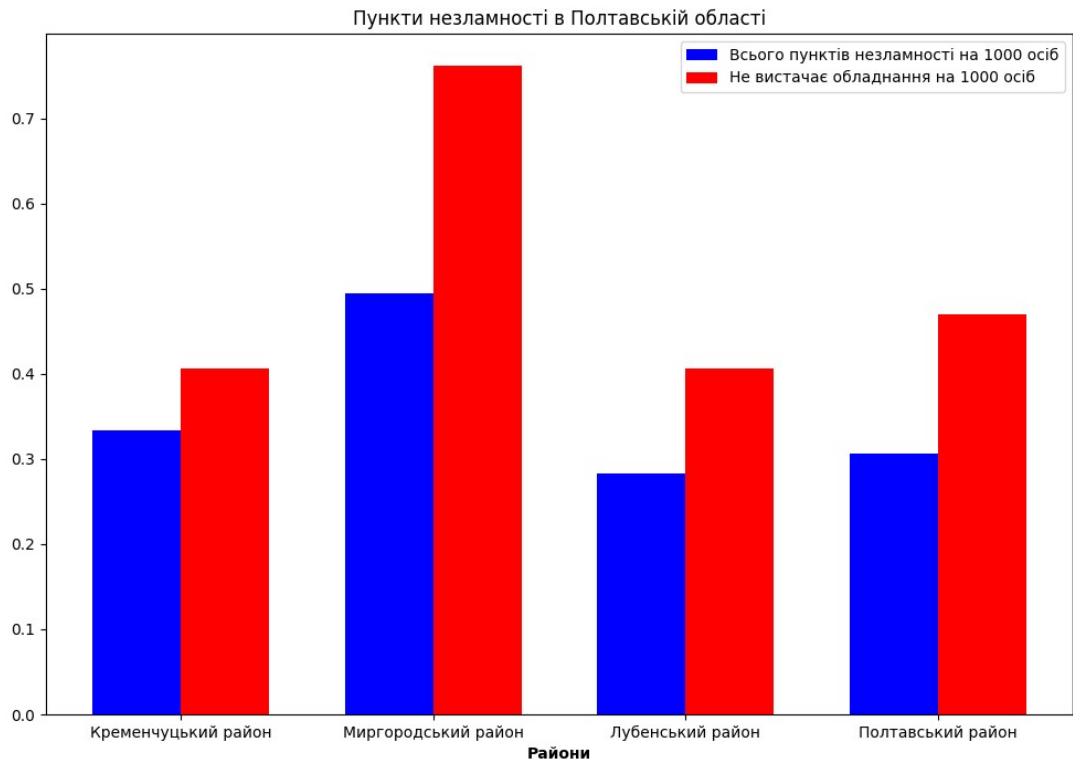


Рисунок 1.2 – Нерівномірний розподіл ПН в Полтавській області

Також неоднорідність розміщення ПН можна побачити на прикладі Амур-Нижньодніпровського та Індустріального районів міста Дніпро, розділених трасою E50, як зображено на рисунку 1.3.

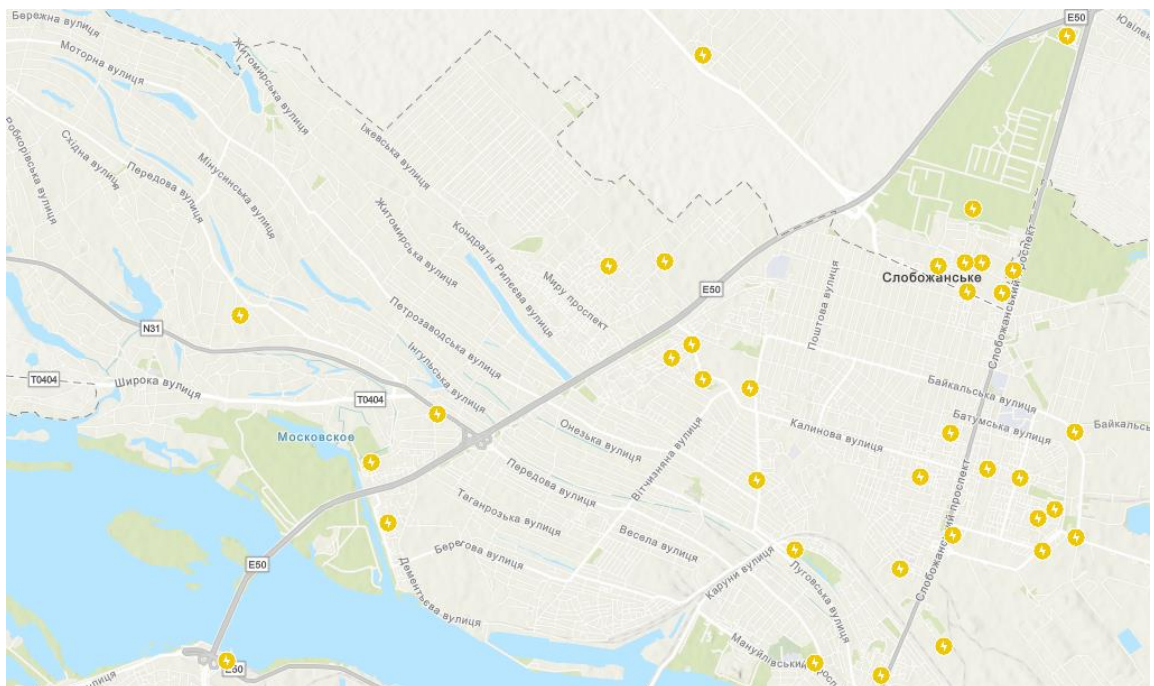


Рисунок 1.3 – Нерівномірна кількість ПН у двох районах Дніпра

До цього фактично було обговорено проблему розташування закладів у різних масштабах, однак окремо слід зазначити і проблему оснащення ПН обладнанням. На відміну від попереднього року, у 2023 році кожний такий пункт повинен обов'язково забезпечувати громадян водою, обігрівом, генератором, зарядкою для телефонів, інтернетом, мобільним зв'язком та аптечкою [7]. Серед додаткового матеріального забезпечення можна побачити булер'ян, старлінк та оптоволоконне підключення до мережі інтернет [11].

Якщо звернутися до графіку саме масованих обстрілів, а не поодиноких, то можна побачити підвищення кількості ракетних обстрілів під опалювальний сезон, що зображено на рисунку 1.4.

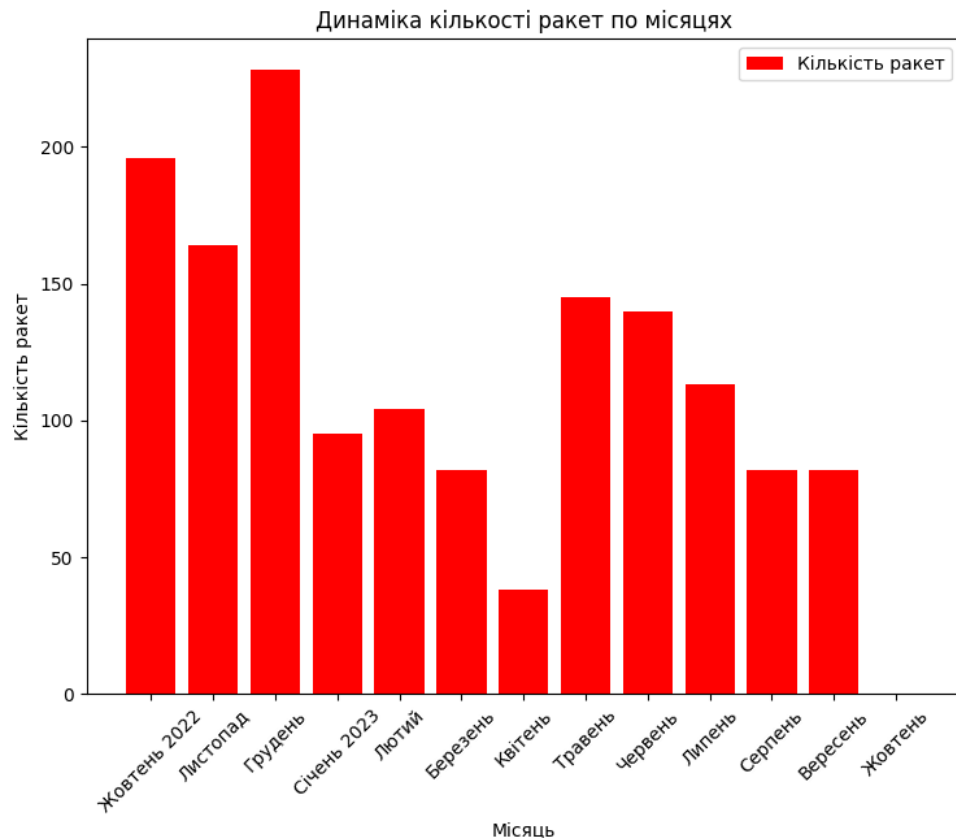


Рисунок 1.4 – Динаміка масованих ракетних обстрілів України [12]

На особисту думку автора, у тривалій війні є сенс та час між, скажімо, сезонами опалювання, коли держава може змінити конфігурацію системи ПН. Як було зазначено, сюди входить як оптимізація ПН у окремо взятому місті загалом так і в одному конкретному районі цього міста.

Екстраполюючи динаміку, яка зображена на рисунку 1.4, на поточний рік та наступний сезон опалювання 2023/2024, можна стверджувати про живу актуальність дослідження, а різке зменшення обстрілів у березні та квітні, а також їх менша інтенсивність продовж наступного півріччя свідчать про реально існуюче вікно для завчасних змін у системі соціальної інфраструктури та системі ПН зокрема.

Підсумовуючи вищесказане, система пунктів незламності вимагає оптимізації з точки зору розташування закладів. Це дозволить вивільнити певні ресурси, та застосувати їх у інших напрямках, сферах, чи навпаки, створити ті ж самі ПН але у інших місцях, де в них є попит і які поки не мають потрібної уваги від держави. Звичайно що в такі скрутні часи люди справедливо вимагають щоб пунктів незламності було якомога більше, та в усіх місцях одночасно. Однак в даних непростих умовах питання стає у тому, щоб принаймні кожний громадянин України мав доступ до хоча б одного такого закладу. Саме таку, чітку та формалізовану постановку має задача покриття множини, яка розглянута у наступному пункті.

1.1.3 Задача покриття множини

У широкому сенсі задача оптимізації системи соціальної інфраструктури відповідає так званій проблемі розташування закладів [13]. Її визначними характеристиками є встановлена ціна для кожного закладу або певна відстань до вибраного клієнту. У свою ж чергу, подібна проблема вирішується як задача покриття множини.

На практиці завдання про покриття виникає при розміщенні пунктів обслуговування, зокрема як у даній роботі пунктів незламності, у системах інформаційного пошуку, при призначенні екіпажів на транспорт, проектуванні інтегральних схем та конвеєрних ліній тощо. Задача про вершинне покриття, що є окремим випадком задачі про покриття множини, може використовуватися при розробці систем контролю та спостереження, у

будівництві завадостійких алгоритмів передачі інформації, при діагностиці обладнання та в інших галузях [14].

Задача покриття множини (ЗПМ) є важливою NP-складною задачею в комбінаторній оптимізації. Маючи колекцію елементів, завдання покриття множини полягає в тому, щоб знайти мінімальну кількість підмножин, які включають (покривають) всі ці елементи [15].

Формально ЗПМ визначається бінарною матрицею A (1.1) із m рядків та n стовпців [16]:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}, \quad (1.1)$$

де $a_{i,j} \in \{0, 1\}$ – значення поля (i, j) , $i \in \{1, 2, \dots, m\}$, $j \in \{1, 2, \dots, n\}$.

Вважається, що рядок i покриває стовпець j , якщо $a_{i,j}$ дорівнює 1, або, навпаки, не покриває, якщо дорівнює 0. Кожний рядок може бути асоційовано із ваговим значенням $c_j \in C$, $C = \{c_1, c_2, \dots, c_n\}$. Тоді нехай $I = \{1, 2, \dots, m\}$ та $J = \{1, 2, \dots, n\}$ – це множини рядків та стовпчиків відповідно. ЗПМ вимагає знайти таку підмножину $S \subseteq J$, щоб кожен стовпчик $j \in J$ був покритий принаймні одним рядком $i \in I$.

Таким чином, задача оптимізації множини визначається наступною формулою (1.2):

$$\min \sum_{i \in I} c_i x_i \quad (1.2)$$

за умов (1.3)-(1.4):

$$\sum_{i \in I} a_{i,j} x_i \geq 1, \quad \forall j \in J \quad (1.3)$$

$$x_i \in \{0, 1\}, \quad \forall i \in I, \quad (1.4)$$

де x_i дорівнює 1, якщо рядок i у рішенні S та 0 у протилежному випадку.

Виходячи з подібної постановки завдання, ціль оптимізації полягає в тому, щоб знайти мінімальну кількість рядків (1.2), урахувуючи їх ваги, з обмеженням, що кожний рядок покриває хоча б один стовпчик (1.3).

Переходячи від формальної постановки ЗПМ до практичного прикладу, необхідно описати головні об'єкти предметної області, які надалі будуть використовуватися у роботі:

- сервісний центр (сервіс, сервісна вершина, сервісний об'єкт) – це заклад що надає певні послуги, які споживають клієнти. Головною характеристикою сервісного центру є радіус обслуговування R , який визначає, чи покриває такий сервіс клієнта. Сервіс у ЗПМ відповідає рядку i . В даній дипломній роботі сервісним центром буде пункт незламності;

- клієнтський центр (клієнт, клієнтська вершина, клієнтська область) – це швидше за все область, або об'єкт, який приймає послуги від сервісу. Головною характеристикою такого об'єкту є відповідь на питання, чи потрапляє він у радіус обслуговування конкретного сервісу. Клієнт у ЗПМ відповідає стовпчику j . В даній дипломній роботі клієнт – це область проживання людей, визначена багатокутником.

Як сервіс, так і клієнт у реальному світі мають певне положення у просторі, за яким можна визначити їх взаємне розташування та від матриці відстаней, урахувуючи радіус обслуговування, перейти до бінарної матриці.

Для розуміння задачі є сенс розглянути наступний приклад. Припустимо, що необхідно надавати послуги пожежної безпеки за найнижчою можливою вартістю у місті, яке складається з шести зон, як зображено на рисунку 1.5.

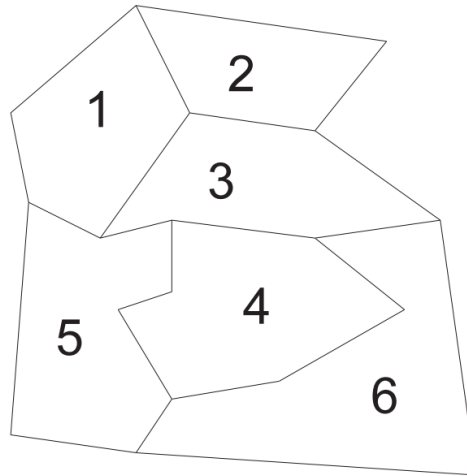


Рисунок 1.5 – Зони розташування сервісів [16]

Для спрощення будемо вважати, що клієнти та сервіси тут розташовуються по тих самих зонах. Тобто зона № 1 – це сервіс, який може покрити усі інші п'ять клієнтів, та у той же час зона № 1 – це і клієнт, який може бути покрити іншим сервісом.

Також задача має певні обмеження:

– замість радіуса обслуговування візьмемо, що сервіс покриває лише прилеглі зони, тобто сервіс зони № 1 покриває усі зони крім № 6, адже вони не мають спільного кордону;

– усі клієнти без виключення, тобто усі зони, мають бути покриті у фінальному рішенні;

– кожна зона являю собою один сервіс та одного клієнта одночасно.

Виходячи із умов задачі, можна сформувану бінарну матрицю (1.5), яка вказує, які зони охоплені гіпотетичною пожежною станцією відповідно до зони, в якій вона знаходиться:

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \quad (1.5)$$

де $a_{i,j} \in \{0, 1\}$ – значення поля (i, j) A , що дорівнює 1 якщо пожежна станція в j зоні покриває зону i , де i та $j \in \{1, 2, \dots, 6\}$.

Тепер для кожної зони визначено вагові коефіцієнти, згідно з якими будемо оптимізувати функцію:

$$C = (3, 5, 6, 4, 2, 1).$$

Виходячи із постановки задачі оптимізації за формулами (1.2)-(1.4) та враховуючи вагові коефіцієнти, задача покриття множини у розглянутому прикладі зводиться до оптимізації наступної цільової функції:

$$\min 3x_1 + 5x_2 + 6x_3 + 4x_4 + 2x_5 + 1x_6$$

за умови:

$$\begin{cases} a_{1,1}x_1 + a_{2,1}x_2 + a_{3,1}x_3 + a_{4,1}x_4 + a_{5,1}x_5 + a_{6,1}x_6 \geq 1 \\ a_{1,2}x_1 + a_{2,2}x_2 + a_{3,2}x_3 + a_{4,2}x_4 + a_{5,2}x_5 + a_{6,2}x_6 \geq 1 \\ a_{1,3}x_1 + a_{2,3}x_2 + a_{3,3}x_3 + a_{4,3}x_4 + a_{5,3}x_5 + a_{6,3}x_6 \geq 1 \\ a_{1,4}x_1 + a_{2,4}x_2 + a_{3,4}x_3 + a_{4,4}x_4 + a_{5,4}x_5 + a_{6,4}x_6 \geq 1 \\ a_{1,5}x_1 + a_{2,5}x_2 + a_{3,5}x_3 + a_{4,5}x_4 + a_{5,5}x_5 + a_{6,5}x_6 \geq 1 \\ a_{1,6}x_1 + a_{2,6}x_2 + a_{3,6}x_3 + a_{4,6}x_4 + a_{5,6}x_5 + a_{6,6}x_6 \geq 1 \end{cases} \quad (1.6)$$

$$x_i \in \{0, 1\}, \quad \forall i \in 1, \dots, 6$$

У рівняння (1.6) можна вставити значення $a_{i,j}$ із бінарної матриці 1.5, що дорівнює 1 якщо пожежна станція в j зоні покриває зону i , після цього спростивши вираз (1.6) наступним чином:

$$\begin{cases} x_1 + x_2 + x_3 + x_5 \geq 1 \\ x_1 + x_2 + x_3 \geq 1 \\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 1 \\ x_3 + x_4 + x_5 + x_6 \geq 1 \\ x_1 + x_3 + x_4 + x_5 + x_6 \geq 1 \\ x_3 + x_4 + x_5 + x_6 \geq 1 \end{cases}$$

Із матриці видно, що оптимальне рішення цієї задачі, це побудувати пожежні станції у зонах № 1 та № 6, отримуючи фінальне значення цільової функції рівним 4.

Однак даний приклад є лише спрощеною практичною версією реальних задач, які не можуть бути вирішені засобами лінійного програмування. Тут постають питання більшого числа вхідних параметрів, і, відповідно, більшої обчислювальної складності та часу роботи, питання адаптації під гнучкі умови задачі, які можуть різнитися в залежності від предметної області, необхідність отримання точного рішення та інші. Можливі, існуючі методи вирішення ЗПМ, під призмою удосконалення системи пунктів незламності, обговорено у наступному пункті.

1.2 Порівняння існуючих методів вирішення задачі покриття множини

Усі фундаментальні методи вирішення проблеми покриття множини можна поділити на три категорії: жадібні методи, точні методи та метаевристики.

Для порівняння таких класів методів (класи, тому що є існує велика кількість різноманітних реалізацій) доцільно сформулювати критерії порівняння, згідно із задачею оптимізації пунктів незламності, специфіку яких було проаналізовано у п.1.1.2. Пропонується скласти порівняльну характеристику за наступними критеріями:

– гнучкість та можливість підлаштовуватися під змінні умови задачі – так як планується модифікувати вихідний обраний метод під бізнес логіку

задачі, а саме наявність обов'язкових незмінних пунктів незламності та можливість зміни радіусу, то подібна умова є визначною;

– здатність до оптимізації великих та складних проблем – актуально у тому випадку, коли буде проводитись оптимізація пунктів незламності для цілих міст. Розмірність задачі, приблизно, буде починатися від декількох десятків сервісних центрів та сотні клієнтських областей, однак програма має бути масштабованою та мати змогу працювати і з більшим об'ємом даних;

– гарантія знаходження точного оптимального рішення – насамперед не є ключовою особливістю у контексті розглянутої проблеми, більш того навіть є сенс отримувати різні рішення при одних і тих самих умовах, перекладаючи таким чином відповідальність за прийняття рішення на оператора, та даючи йому змогу розглянути декілька альтернатив. Тобто тут є сенс не тільки знайти глобальне рішення, але й вивчити локальні;

– швидкість та простота реалізації – швидкість є важливою, адже користувач у реальному часі буде очікувати результат роботи. Так як ЗПМ є NP-важкою проблемою, очікується висока обчислювальна складність будь-якого методу. Простота реалізації також є важливою, оскільки вона дозволяє швидше розробляти та впроваджувати алгоритми, а також забезпечує легкість підтримки та модифікації коду;

– ресурсоємність – оцінюється, скільки комп'ютерних ресурсів (таких як пам'ять, час процесора) потрібно для виконання алгоритму. Для великих ЗПМ це особливо актуально, адже висока ресурсоємність може обмежити можливість застосування алгоритму в умовах з обмеженими ресурсами або вимагати дорогих, у плані апаратного забезпечення, операцій.

Також важливими критеріями є якість наближеного рішення та стійкість до помилок та шумів даних, однак ці критерії було опущено. У таблиці 1.1 наведено порівняльну характеристику методів рішення ЗПМ.

Таблиця 1.1 – Порівняння методів рішення ЗПМ

Критерій	Метаевристики	Жадібні методи	Точні алгоритми
Гнучкість	+	$\pm$	-
Здатність оптимізації складних рішень	+	-	$\pm$
Гарантія точного рішення	-	-	+
Швидкість реалізації	$\pm$	+	-
Ресурсоємність	-	+	-

Згідно з таблицею 1.1 та проаналізованих джерел можна підсумувати та окремо виділити переваги та недоліки кожного з методів:

– жадібні методи – прості евристики, які крок за кроком вибирають локально оптимальні рішення з надією на знаходження глобального оптимуму або приблизного рішення. В контексті ЗПМ сутність жадібного методу полягає у виборі тієї множини, яка на кожному кроці покриває максимальну кількість елементів, що ще не покриті, до тих пір, поки всі елементи не будуть покриті або доки не будуть використані всі доступні множини [17]. Перевагами гнучких методів безумовно є швидкість реалізації та низькі обчислювальні витрати, а недоліки – це відсутність здатності до оптимізації складних рішень та гарантії точного рішення. Тим не менш, існує наприклад модифікований жадібний метод великого кроку [18], який на кожному кроці одночасно вибирає p множин так, щоб їх об'єднання покривало найбільшу кількість ще не покритих елементів. Це покращує точність оптимізації за рахунок швидкості роботи. Також для підвищення ефективності жадібного методу може використовуватися гібридизація з локальним пошуком, де після знаходження жадібного рішення виконується локальна оптимізація для покращення знайденого рішення, використовуючи, наприклад, методи локального пошуку або симульованого відпалу [19];

– точні алгоритми гарантують знаходження оптимального рішення задачі. Вони працюють шляхом систематичного перебору всіх можливих варіантів та звуження пошукового простору. В рамках ЗПМ, точні методи зазвичай базуються на техніках цілочисельного програмування, де задача ставиться як математична модель з обмеженнями, і використовуються алгоритми, такі як симплекс-метод або метод гілок та меж [20]. Перевагою точних методів є здатність знаходити дійсно оптимальні рішення, тоді як недоліком є значно вищі обчислювальні витрати, особливо для великих та складних задач, де простір рішень є надміру великим, що може привести до експоненційного зростання часу виконання. Однак існують різні підходи для покращення точних методів, наприклад, впровадження ефективних стратегій гілок і меж, що включають використання відсічень для зменшення розміру дерева пошуку та застосування евристик для вибору гілок [21];

– метаяевристики – це вищий клас евристичних алгоритмів, призначених для розв'язування складних оптимізаційних задач, де класичні методи можуть бути неефективними або занадто повільними. Вони зазвичай використовують стратегії для пошуку на широкому просторі рішень і часто застосовують правила, що дозволяють виходити за межі локального оптимуму для пошуку глобального оптимуму [22]. Метаяевристичні методи краще пристосовуються до специфічних умов задачі, як наприклад, балансування між швидкістю пошуку та якістю рішення. Прикладами метаяевристичних методів, які можуть бути використані у ЗПМ, є генетичний алгоритм, алгоритм рою частинок, що імітує соціальну поведінку тварин; метод імітації відпалу, який використовує концепцію термодинаміки для "охолодження" рішення до стану оптимуму. Ці методи часто включають параметри, які можна налаштувати для покращення їхньої ефективності у конкретних умовах задачі.

Зокрема в цій роботі буде використано саме один із метаяевристичних методів, які перед цим необхідно порівняти, серед них: Chemical Reaction Optimization (CRO), Genetic Algorithm (GA), Global Simple Evolutionary Multi-Objective Optimizer (GSEMO), Jumping Particle Swarm Optimization

(JPSO), Simulated Annealing (SA), Stochastic Evolutionary Integer Programming (SEIP) [23].

Порівняння методів було проведено на даних із бібліотеки OR-library [24] за критеріями: середнє значення кількості ітерацій, що були потрібні для знаходження рішення, середня тривалість виконання (в секундах), середня тривалість однієї ітерації, середнє та найкраще значення співвідношення наближення рішення (чим ближче до 1, тим точніше рішення). Результати експериментів наведено у таблиці 1.2 [23].

Таблиця 1.2 – Порівняння метаевристичних методів рішення ЗПМ [23]

Критерій	CRO	GA	GSEMO	JPSO	SA	SEIP
Кількість ітерації	2126,0 ± 53,0	2558,0 ± 478,0	3608,0 ± 1073,0	2001,0 ± 0,0	2649,0 ± 839,0	4768,0 ± 1250,0
Середня тривалість (с)	1,048 ± 0,16	4,342 ± 1,109	20,952 ± 8,107	701,123 ± 130,38	14,047 ± 5,01	0,666 ± 0,242
Середня тривалість ітерації	0,00049 ± 7e-05	0,00172 ± 0,0030	0,00586 ± 0,0007	0,35039 ± 0,0864	0,00537 ± 0,0224	0,00014 ± 3e-05
Середній коефіцієнт апроксимації	1,644 ± 0,07	1,078 ± 0,091	1,111 ± 0,091	1,433 ± 0,082	1,433 ± 0,063	1,167 ± 0,131
Найкращий коефіцієнт апроксимації	1,556	1,0	1,0	1,333	1,333	1,0

Виходячи з проведеного аналізу (таблиці 1.1 та 1.2), для вирішення задачі оптимізації та вивчення було обрано генетичний алгоритм. Він відрізняється в першу чергу високою гнучкістю, адже дозволяє кодувати параметри як гени хромосом, та здатністю оптимізувати складні,

багатовимірні задачі. У порівняння з іншими метавевристиками, класичний генетичний алгоритм має найкращий показник точності рішення.

Також варто зазначити що, згідно з «No Free Lunch» теоремою, не існує універсального алгоритму оптимізації, який працював би краще за всі інші алгоритми на всіх можливих задачах [25]. Іншими словами, алгоритм, який є дуже ефективним для одного класу проблем, може не бути таким ефективним для іншого. Тому, вибираючи алгоритм для конкретної оптимізаційної задачі, важливо враховувати специфіку проблеми. У будь-якому випадку, в даній роботі доцільно працювати саме з генетичним алгоритмом та його модифікаціями.

1.4 Постановка задачі

Задача оптимального розташування об'єктів інфраструктури вирішується як задача покриття множини, яка в свою чергу визначається формулою (1.2), за умов (1.3)-(1.4).

Згідно із проаналізованими методами вирішення ЗПМ, найбільш доцільним для дослідження та застосування на прикладі пунктів незламності обрано генетичний алгоритм. Таким чином, у подальшому у роботі необхідно визначити принцип кодування задачі у хромосомі, визначити фітнес-функцію для обрахування рівня пристосованості та налаштувати параметри на прикладі конкретної задачі.

Для надання можливості користувачу проводити оптимізацію реальних об'єктів необхідно інтегрувати у розроблену програму геоінформаційну систему, а також розробити відповідний користувацький та програмний інтерфейс.

1.5 Висновки за розділом 1

В даному розділі розглянуто поняття соціальної інфраструктури України, а також передумови появи у 2022 році пунктів незламності як важливої ланки інфраструктури. Додатково проаналізовано проблеми пунктів незламності та виявлено можливості покращення системи розтушування таких об'єктів у цілому за допомогою вирішення задачі покриття множини. Також порівняно методи вирішення такої задачі та обрано генетичний алгоритм для подальшого застосування та розвинення.

Завдання дипломної роботи визначені наступні:

- аналіз предметної області;
- дослідження методів і моделей вирішення задачі покриття множини;
- проєктування програмного забезпечення для оптимізації системи соціальної інфраструктури;
- розробка програмного забезпечення для оптимізації системи соціальної інфраструктури.

2 МАТЕРІАЛИ І МЕТОДИ

В даному розділі проведена обробка вихідних даних, модифікація генетичного алгоритму з оператором відновлення та виконано налаштування еволюційних параметрів на основі абстрактних даних.

2.1 Обробка вихідних даних

Для створення рішення проблеми покриття множини в роботі використано абстрактні дані з OR-Library [24]. Ця бібліотека наборів даних пропонує широку колекцію вихідних даних для багатьох проблем дисципліни дослідження операцій (operations research), та задачі покриття множини зокрема [24]. Бібліотека, як і багато алгоритмів вирішення ЗПМ, була розроблена Джоном Бізлі, який зробив неоцінений внесок у розвинення методів комбінаторної оптимізації [26].

Набори даних представлені не у звичному вигляді бінарної матриці, а у описовому форматі, як цю матрицю отримати. Такий формат складений за наступними принципами:

- у першому рядку записана кількість строк m та кількість стовпчиків n матриці A ;
- наступний блок зберігає ваги для кожного стовпчика, однак у цій роботі ваги опущено, задля оптимізації лише кількості об'єктів, а не їх загальної ціни;
- кожен наступний блок має у першому рядку число записів, а у наступних рядках – номер колонки, що покриває i рядок.

Надалі для тестування розробленого алгоритму буде використано набір даних `scr41`, що складається з 200 рядків (сервісних центрів) та 1000 стовпчиків (клієнтів).

На рисунку 2.1 зображено приклад фрагмента тестового формату вихідних даних.

```
50 500
1 1 1 1 1 1 1 1 1 1 1 2
2 2 2 2 2 2 2 3 3 3 3 3
3 3 3 3 3 4 4 4 4 4 4 4
5 5 5 5 5 5 5 6 6 6 6 6
...
81
4 8 10 11 12 14 15 16 18 23 26 28
45 52 53 54 56 57 60 67 74 80 81 86
106 108 112 118 121 125 126 133 137
176 177 183 190 195 199 201 204 206
250 252 253 261 262 263 267 272 277
320 324 325 327 330 334 335 345 347
376 377 380 386 396 410 415 425 430
469 473 477 478 479 483 496
73
1 2 4 5 13 19 21 28 36 44 45 49 50
59 61 62 67 68 71 73 74 77 82 83 92
```

Рисунок 2.1 – Фрагмент набору даних scr65

На рисунках 2.2 та 2.3 зображено блок-схему алгоритму обробки вихідного набору даних та створення бінарної матриці.

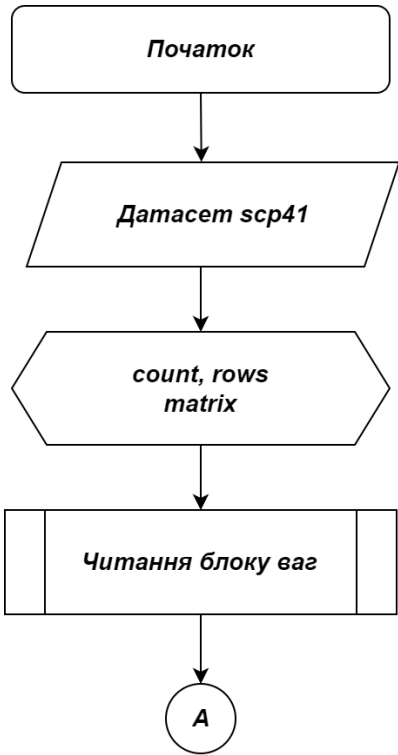


Рисунок 2.2 – Алгоритм виведення вихідних даних

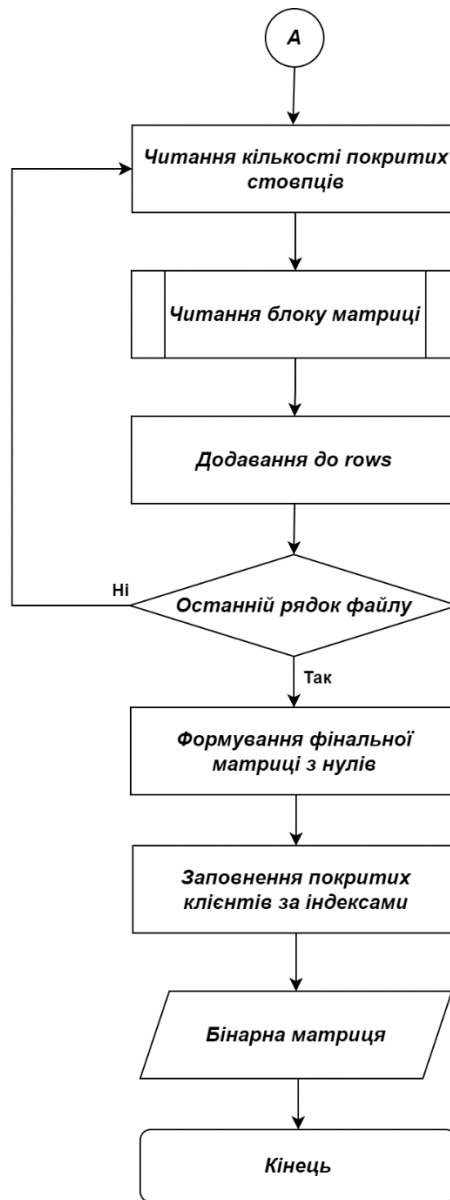


Рисунок 2.3 – Продовження алгоритму виведення вихідних даних

В результаті роботи алгоритму на виході отримано бінарну матрицю, яка буде використана генетичним алгоритмом (ГА) при вирішенні ЗПМ.

2.2 Реалізація ГА та модифікація оператора відновлення

На першому кроці створення ГА під вирішення конкретної задачі необхідно визначитись з представленням даних. Для ЗПМ логічним рішенням буде використати бінарне кодування 0-1, для позначення включено чи ні у фінальне рішення той чи інший рядок. Варто відзначити, що подібна

хромосома навряд може вважатися гомологічною, адже застосування еволюційних операторів може призвести до появи недопустимих рішень [27]. Бінарне представлення індивіда для ЗПМ зображено на рисунку 2.4.

Номер рядка	1	2	3	4	5	...	n - 1	n
Хромосома	1	0	1	1	0	...	0	1

Рисунок 2.4 – Бінарне представлення хромосоми

Фітнес функція обчислюється за наступною формулою (2.1):

$$f_i = \sum_{j=1}^n x_{ij}, \quad (2.1)$$

де x_{ij} значення j -го біту i -го індивіда.

Використовуючи ГА до вирішення ЗПМ, у роботі [28] вводиться два оператори: оператор імітації війни та оператор відновлення. Перший необхідний для підвищення рівня диверсифікації популяції у визначений момент часу, та є оригінальною розробкою авторів. А от другий – оператор відновлення – це лише реалізація одного з двох найбільш поширених методів забезпечення допустимості рішення. Насправді, у ЗПМ існує серйозна імовірність отримати таких індивідів, які не задовольняють певним умовам. По-перше, індивід може не покривати усі клієнтські центри, наприклад, у випадку коли певного клієнта можу обслужити лише один конкретний сервіс, який у рішення не включено. По-друге, індивід може не містити необхідні сервісні центри, які є обов’язковими через специфіку задачі (наприклад, ПН відповідального бізнесу є обов’язковими, тому що ми не маємо права їх виключати із рішення чи змінювати їх конфігурацію).

Оператор відновлення – це евристичний оператор для перетворення недопустимого рішення в допустиме. Ще одним способом контролю повноти рішення є штрафна функція, що зменшує значення пристосованості

недопустимих рішень. Однак її недоліком є складність налаштування та відсутність повної гарантії у повній допустимості хромосоми [29].

Головною проблемою оператору відновлення є велика обчислювальна вартість та високий рівень впливу на показники ефективності роботи алгоритму, попри безпосередньо еволюційні оператори. Через це пропонується впровадити евристику для підвищення продуктивності кожної ітерації пошуку.

Для модифікації оператора відновлення можна впровадити поняття «конфліктних сервісних центрів» для покращення розподілу покриття клієнтів. Ця модифікація базується на ідентифікації та розв'язуванні потенційних конфліктів між сервісними центрами, які намагаються покрити одних і тих самих клієнтів. Її сенс у наступному:

- після визначення непокритих клієнтів, для кожного такого клієнта знаходяться всі сервісні центри, що його покривають. Якщо декілька сервісних центрів покривають одного клієнта, вони вважаються «конфліктними»;

- для кожного конфліктного сервісного центру розраховується показник ефективності, який визначається як кількість унікальних клієнтів, яких цей центр може покрити;

- для кожного непокритого клієнта вибирається той сервісний центр, що має найвищий показник ефективності. Це дозволяє забезпечити більш оптимальне розподілення ресурсів між сервісними центрами;

- після вибору оптимальних сервісних центрів для кожного непокритого клієнта, оновлюється стан кожного індивіда у популяції, зазначаючи відповідні сервісні центри як активні;

- якщо після вибору сервісних центрів деякі з них стають надмірними (тобто, їх внесок у покриття клієнтів є низьким або нульовим), їх можна відключити, зменшуючи загальні витрати.

Ця модифікація дозволяє більш ефективно розподіляти клієнтів між сервісними центрами, забезпечуючи краще покриття і зменшуючи загальні витрати.

Для оцінки ефективності впровадженого оператора відновлення використано стандартні метрики максимальної та середньої пристосованості популяції. Для оцінки рівня диверсифікації було попарно знайдено відстань Хеммінга між хромосомами популяції, а потім знайдено їх середнє значення. Відстань Хеммінга обраховується за формулою (2.2):

$$H(x, y) = \sum_{j=1}^n [x_i \neq y_i], \quad (2.2)$$

де n – кількість генів хромосоми;

x_i та y_i – i -ті біти хромосом на x та y відповідно;

$[x_i \neq y_i]$ – індикаторна функція, яка приймає значення 1, якщо x_i не дорівнює y_i , і 0 в іншому випадку.

У ході дослідження було проведено по 10 експериментів на звичайний та модифікований оператор відновлення із максимальною кількістю ітерацій 100 та розміром вибірки 50 особин. Результати експерименту зображено на рисунку 2.5 та рисунку 2.6.

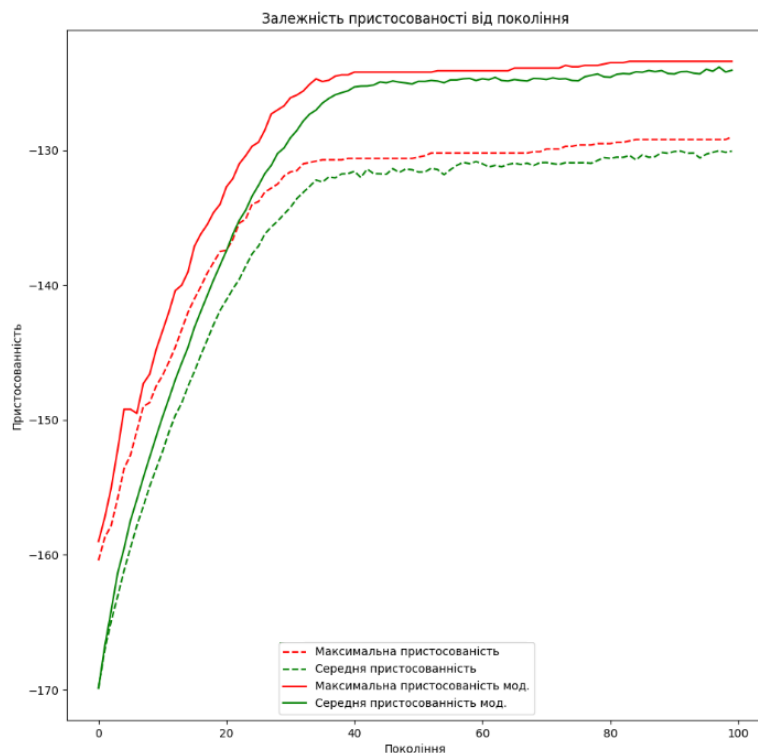


Рисунок 2.5 – Залежність пристосованості від покоління

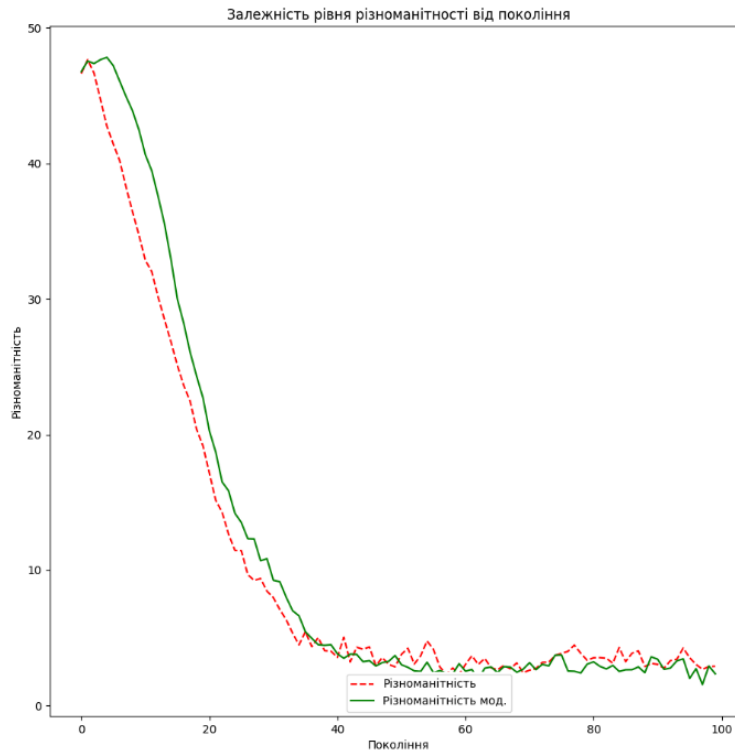


Рисунок 2.6 – Залежність рівня різноманітності під покоління

Як видно з результатів експерименту, оператор відновлення що використовує «конфліктні сервісні центри» показує значно кращі результати на 100 ітерацій пошуку як з точки зору максимальної пристосованості вибірки, так і середньої. Щодо різноманітності то її рівень зберігається на попередньому рівні, а швидкість однієї ітерації більша лише на 2,5%, тобто обчислювальна вартість майже не змінена. Насправді, найбільш вимоглива до часу частина оператора залишилася не змінна – це пошук тих клієнтів які не були покриті.

2.3 Метод пошуку оптимального розташування об'єктів соціальної інфраструктури

Виходячи із постановки задачі, її специфічних особливостей та обраних засобів її вирішення, які було описано вище, в дипломній кваліфікаційній роботі пропонується наступний метод пошуку оптимального розташування об'єктів соціальної інфраструктури, зображений на рисунку 2.7.

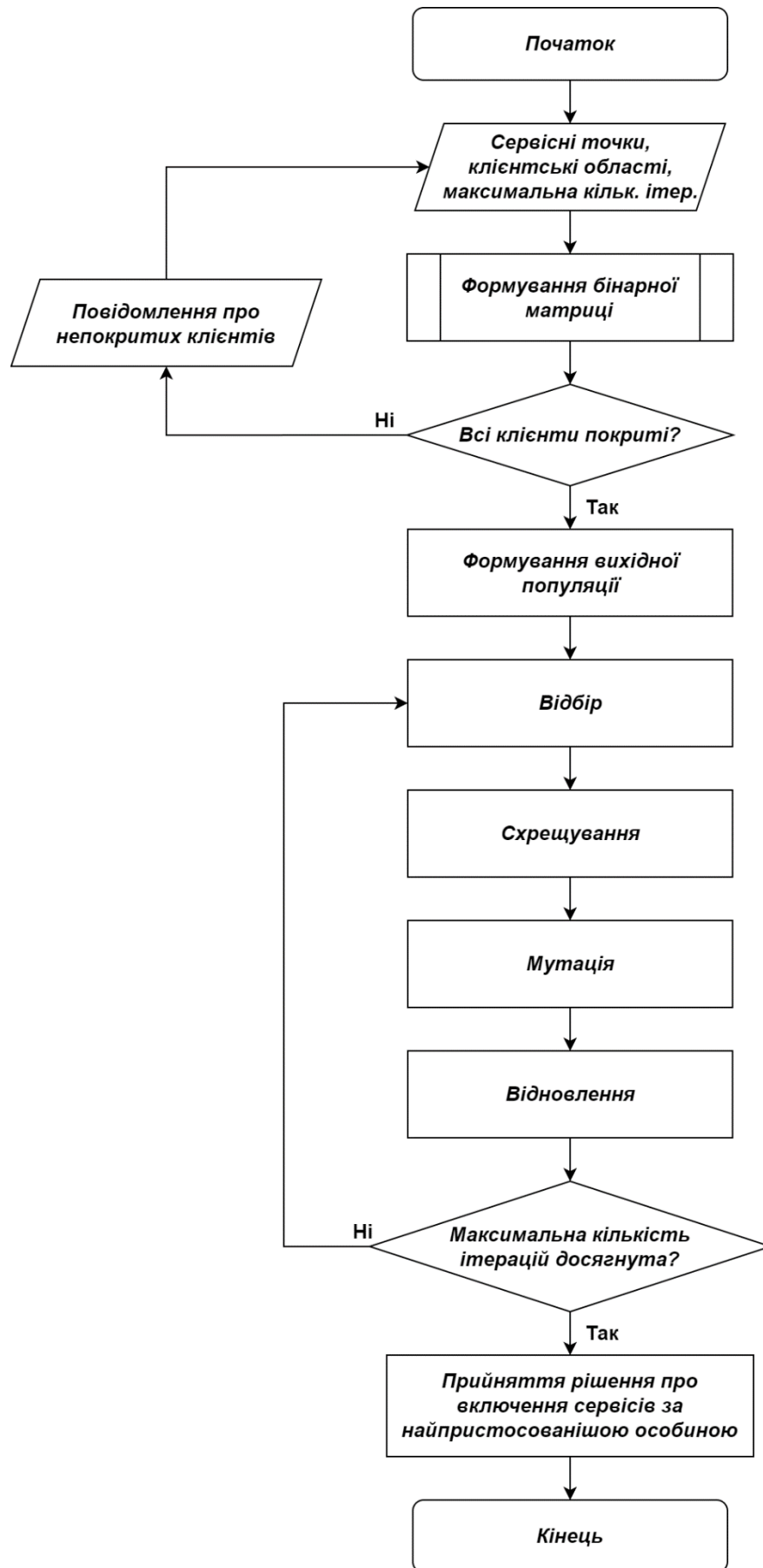


Рисунок 2.7 – Запропонований метод пошуку оптимального розташування об’єктів соціальної інфраструктури

Запропонований метод складається з наступних етапів:

- формування вихідних даних – передбачає позначення користувачем сервісних центрів та клієнтських областей на мапі, більш детально цей процес описано у пункті 4.1;

- формування бінарної матриці – згідно з вихідними даними, а саме координатами, що відповідають за просторове розміщення об'єктів, а також радіус обслуговування, визначається бінарна матриця. Її елементи позначають, які клієнти знаходяться у зоні досяжності певного сервісу. Клієнтам відповідають колонки, а сервісам рядки. Формальне визначення бінарної матриці подано формулою (1.1);

- перевірка, чи всі клієнти є покритими – процедура необхідна для задоволення умови задачі, де кожний клієнт має доступ хоча б до одного сервісного центру;

- формування вихідної популяції – визначення початкових значень генів індивідів згідно з обраним бінарним принципом їх кодування, наведеним у пункті 2.2;

- послідовне застосування еволюційних операторів – підвищення пристосованості індивідів вибірки та проведення еволюційного пошуку, налаштування параметрів якого детально описано у наступному підрозділі;

- застосування оператора відновлення – для забезпечення гарантії наявності у рішенні необхідних сервісних центрів та виключення наявності недопустимих рішень застосовується оператор відновлення, модифікацію якого, базуючись на концепції «конфліктних сервісних центрів», наведено у попередньому підрозділі;

- припинення оптимізації та формування фінального рішення – при досягненні ліміту на кількість ітерації обирається найбільш пристосований індивід та згідно зі значеннями його генів приймається рішення щодо того, чи залишати той чи інший сервісний центр, чи прибрати.

2.4 Налаштування параметрів ГА

В ході роботи застосовано турнірний відбір. Це пояснюється тим, що пороговий відбір займає багато часу, адже містить сортування на кожному кроці, а також вирізняється доволі низьким ступенем підтримки різноманітності, навіть при відносно невеликому значенні порога, а пропорційний відбір демонструє погану вибірковість у тому випадку, коли в ході еволюції всі елементи популяції стають приблизно рівними по значенню цільової функції. До того ж, турнірний відбір є найбільш часто використовуваним у вирішенні задач еволюційного пошуку.

Щодо інших еволюційних операторів то було обрано рівномірне схрещування, з огляду на результати роботи [28], а мутація – інвертування біту, з імовірністю для кожного гену $1/n$, де n – розмір хромосоми.

Важливою характеристикою еволюційного процесу, що вимагає налаштування, є розмір популяції. Для вибору оптимального значення було проведено експерименти для 50 особин, 100, 150, 200 та 300. Результат експериментів зображено на рисунку 2.8.

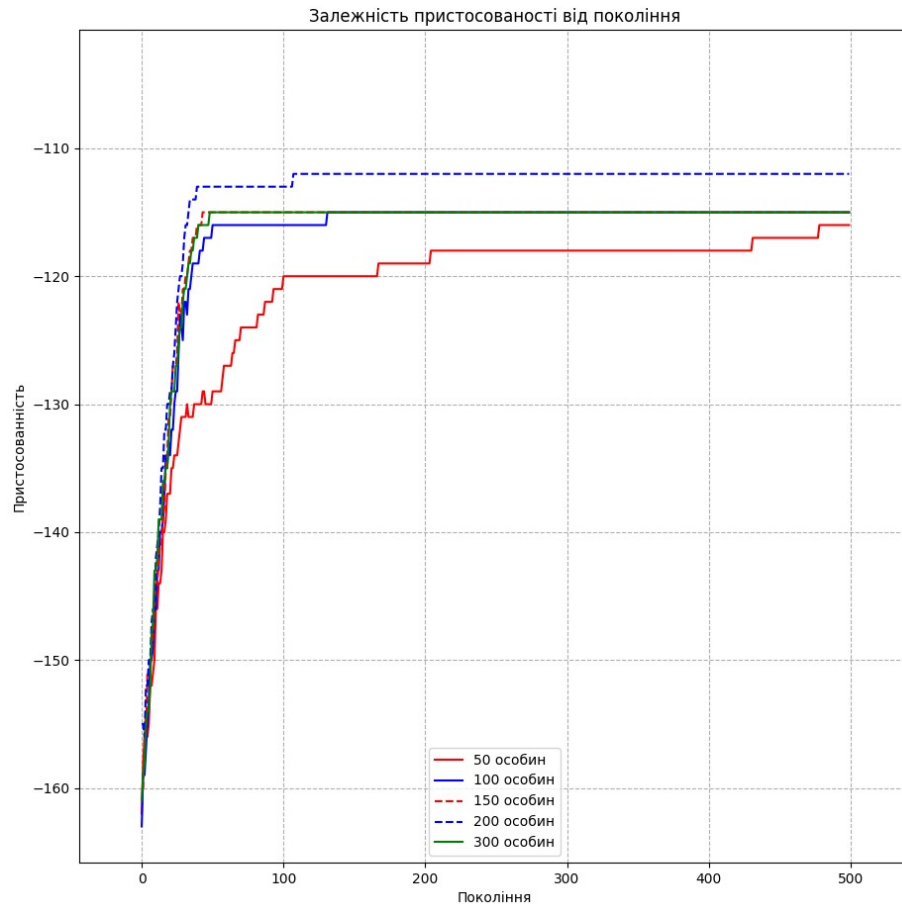


Рисунок 2.8 – Залежність пристосованості найкращого індивіду від покоління на 500 ітерацій

Як видно з рисунку 2.9, оптимальне рішення отримано лише в одному випадку, проте головним показником в ЗПМ все ж таке є не стільки необхідність отримати точне рішення, скільки швидкість знаходження приблизно точного. Через це є сенс проаналізувати також метрику часу, зміну якої зображено на рисунку 2.9.

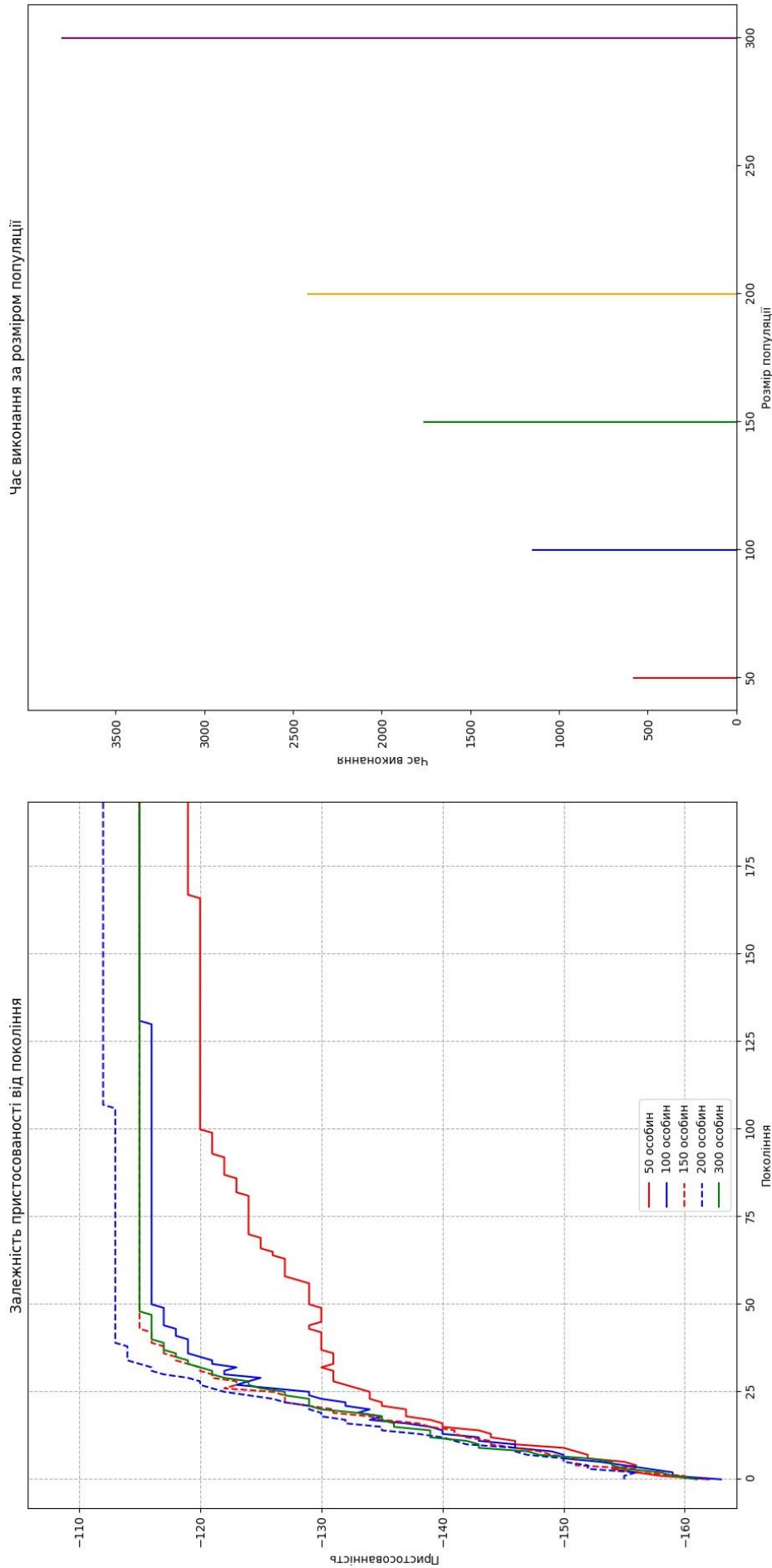


Рисунок 2.9 – Залежність пристосованості та часу від покоління зі зміною кількості особин

Час роботи ГА лінійно залежить від розміру популяції. Найбільш прийнятним значення розміру популяції є 100 особин, адже у порівнянні із 50 він дає якісний стрибок у швидкості знаходження прийнятного рішення: фактично, за 50 ітерацій вибірка зі 100 особинами демонструє такі ж результати як і з 150 та 300, та набагато краще ніж популяція із 50 особинами. Якщо прийняти до уваги швидкість, то можна стверджувати, що популяції більше ніж 100 особин не є прийнятними принаймні у випадку майбутньої розробки програми під потреби користувача, у разі необхідності швидкого отримання рішення.

Так само проведено 10 експериментів із різним показником імовірності схрещування результати яких зображено на рисунку 2.10.

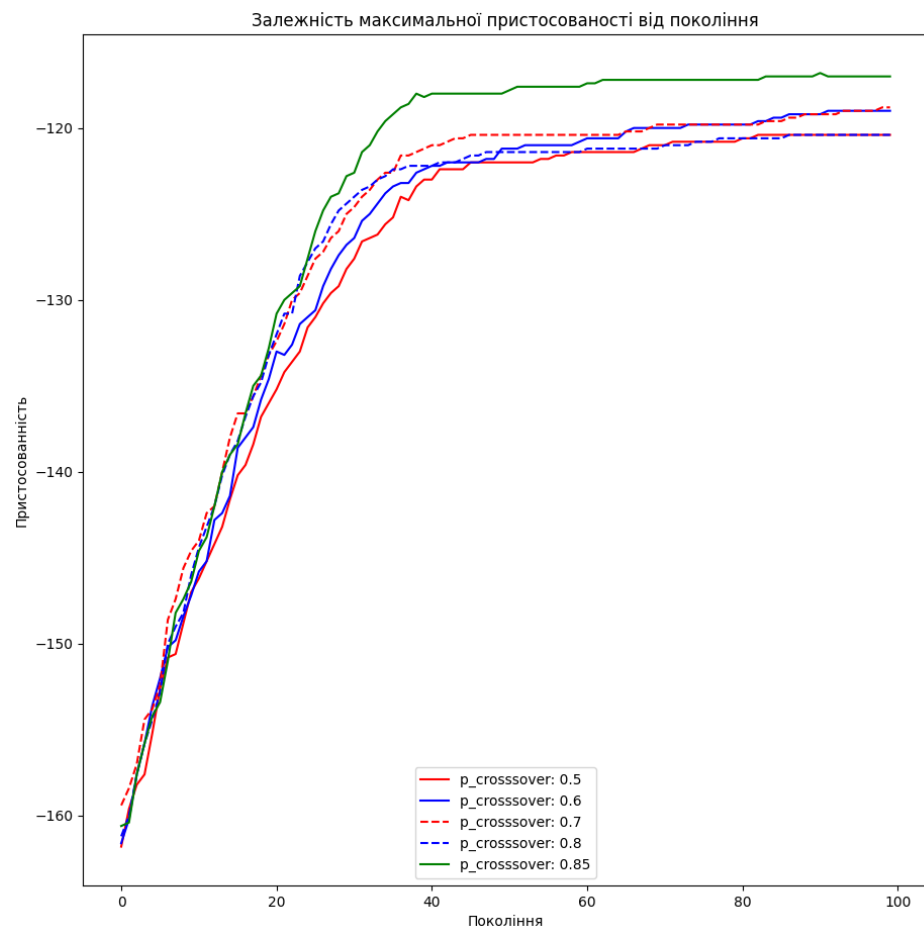


Рисунок 2.10 – Залежність максимальної пристосованості від покоління із різними показниками ймовірності кросинговеру

Як видно з рисунку 2.10, найбільш наближені до оптимуму рішення ГА отримує у випадку значення імовірності схрещування рівної 0,85, всі ж інші значення є приблизно однаково менш ефективними. Щодо показника середньої пристосованості популяції, то графік його зміни майже не відрізняється від того, що зображений на рисунку 2.9. Таким чином знайдено найкраще значення імовірності схрещування, що дорівнює 0,85.

2.5 Висновки за розділом 2

У розділі 2 було проведено кодування задачі покриття множини у вигляді генів для застосування ГА для її вирішення. Перед дослідженням можливостей алгоритму було наведено схему витягання вихідних даних із стандартного набору даних для цієї задачі – OR Library. Для задоволення умови необхідності включення у рішення певних сервісних центрів та забезпечення допустимості рішення було застосовано оператор відновлення.

Наукова новизна роботи полягає в створеному методі пошуку оптимального розташування об'єктів соціальної інфраструктури, який базується на використанні генетичного алгоритму, визначає необхідні етапи для формування фінального рішення, застосовує модифікований оператор відновлювання. Для підвищення ефективності пошуку оптимального рішення оператор відновлювання було модифіковано, ввівши поняття «конфліктних сервісних центрів», що дозволило отримати кращі рішення за фіксований час пошуку.

Перед застосуванням ГА на реальних даних було проведено налаштування параметрів моделі на абстрактних. Фінальна конфігурація моделі наступна:

- турнірний відбір із трьома учасниками;
- рівномірне схрещування із імовірністю обміну генами 0.3;
- мутація – інвертування із імовірністю для кожного біту в залежності від розміру гену;

- розмір популяції 100 особин;
- імовірність схрещування 0.85;
- імовірність мутації 0.05;
- максимальна кількість поколінь – 50, однак цей показник має виставлятися користувачем.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В даному розділі проаналізовано вимоги до програмного продукту, за ними створено критерії оцінювання допоміжних програмних засобів, обґрунтовано вибір мови програмування, аргументовано вибір геопросторової інформаційної системи для інтеграції та відповідний їй фреймворк, обрано стратегію розробки проєкту та наведено перелік усіх інструментів використаних для створення повноцінного програмного продукту.

3.1 Вибір інструментарію розробки програмного забезпечення

Програмні засоби, за допомогою яких буде здійснюватися розробка та підтримка фінального продукту, були обрані в залежності від кінцевих цілей та вимог до програми, які описані у технічному завданні.

Одразу можна сказати, що складна декларована структура проєкту підштовхує до ретельного вибору саме загальної архітектури – принципу, за яким проєкт буде організовано та розроблено. Також неминучим є вибір геоінформаційної системи та, відповідного до неї, Application Programming Interface (API), задля задоволення центральної потреби користувачів у роботі з мапою реального світу. Вибір же інших інструментів залежить від вимог до програми, тому є сенс їх проаналізувати та визначити найважливіші, водночас викорінивши ті, що задовольняють потреби як звичайних користувачів, так і адміністраторів.

3.1.1 Аналіз технічного завдання та формування вимог до використовуваних програмних засобів

Стисло, вимоги до програмного модуля, що реалізує генетичний пошук оптимального рішення задачі покриття множини, як з реальними так і з абстрактними даними, можна сформулювати наступним чином:

- оброблювати та зберігати дані – зчитувати набори даних з OR-Library та GeoJSON файлів, перетворювати непідготовлені дані у готові структури, як-то матриці відстаней та матриці ваг, записувати та зчитувати результати досліджень у текстові файли;

- аналізувати дані – визначати відстані між об'єктами та інші параметри, проводити аналіз різноманітності популяцій та інших показників, знаходити покриті та непокриті сервісні центри та клієнтські точки;

- візуалізувати та забезпечувати інтерактивність – будувати, зберігати та взаємодіяти з графіками відстеження різних параметрів, забезпечувати можливість змінення масштабу, повороту та інтерактивного аналізу;

- проводити пошук оптимального рішення – задавати, клонувати та змінювати хромосоми та популяції, реалізовувати генетичні оператори (відбору, схрещування, мутації та інші), проводити еволюційний пошук;

- забезпечувати налаштування конфігурації – налаштовувати параметри еволюційного пошуку та проводити паралельні моделі у декількох потоках, вимірювати час роботи методів.

Що стосується центральної функціональності, то програма має задовольняти можливість виконання зазначених нижче функцій:

- проводити авторизацію користувачів – дозволяє користувачеві отримати доступ до системи, використовуючи особистий логін та пароль. Ця дія передує іншим функціям і є обов'язковою для отримання повного доступу до системи, зокрема для редагування карти та пошуку оптимального розташування об'єктів;

- реєструвати нових користувачів – початковий етап для нового користувача, що хоче скористатися системою. Тут користувач має ввести свою існуючу, реальну пошту та пароль з коректним форматом. Після цього перейти за своєю поштовою скринькою та підтвердити реєстрацію, перейшовши за посиланням. Це стандартна реєстраційна процедура з верифікацією особи за електронною поштою;

– взаємодіяти з мапою – користувач може взаємодіяти з картою, зокрема здійснювати масштабування, переглядати координати певних точок на карті для навігації за місцевістю, шукати за адресою та фільтрувати об'єкти за певними критеріями, наприклад, за типом об'єкту чи радіусом обслуговування;

– отримувати інформацію про об'єкти соціальної інфраструктури – користувачеві надається можливість дізнатися деталі про різні об'єкти, розміщені на мапі, у вигляді таблиці атрибутів;

– керувати об'єктами на карті – додавання та видалення точок та клієнтських областей (у вигляді багатокутника), а також зміна положення вже існуючих та редагування інформації про них. Для цього необхідно створити окреме вікно, для взаємодії з об'єктами на мапі;

– проводити оптимізацію – користувач може запускати процес оптимізації соціальних об'єктів на мапі, використовуючи внутрішній алгоритм системи;

– контролювати деякі параметри пошуку рішення, наприклад, максимальний час пошуку чи імовірність мутації, які будуть названі більш звично для користувача, або окремо пояснені, наприклад, здатність моделі відтворювати принципово нові рішення;

– вивантажувати результати – після оптимізації адміністратор може вивантажити отримані результати для подальшого аналізу та використання у різних форматах. У подальшому ці файли може бути знов завантажено, сформувавши з них шар ознак та відобразивши на мапі;

– проходити опитування для отримання відгуку від користувачів системи щодо ефективності проведеної оптимізації.

На рисунку 3.1 представлено схему ключових операцій системи, які планується впровадити.

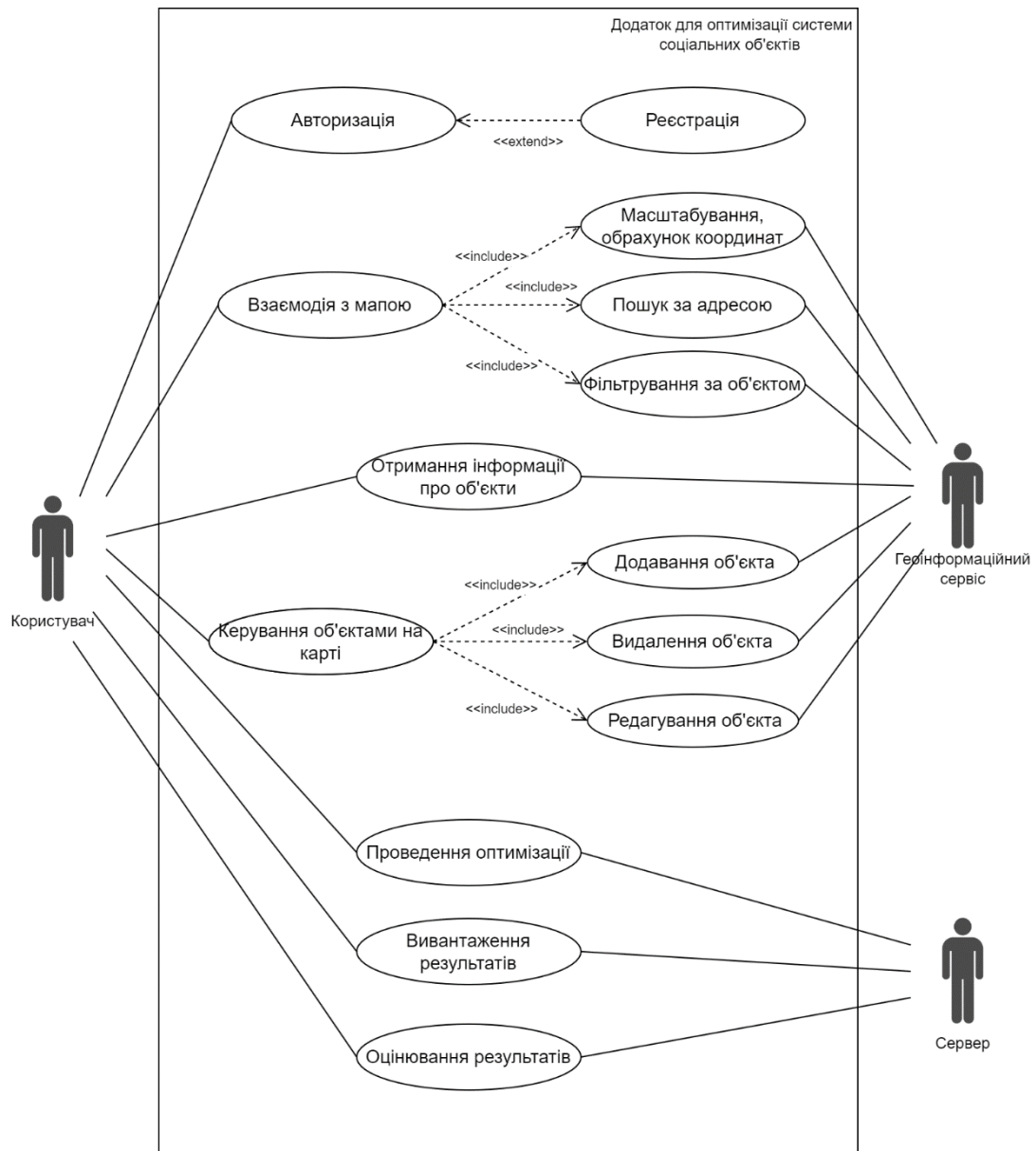


Рисунок 3.1 – Діаграма варіантів використання програми

Виходячи з проаналізованих функціональних та нефункціональних вимог до програми та вмісту технічного завдання, програмні засоби (ПЗ), які будуть допоміжними при розробці, повинні задовольняти наступним вимогам:

– типізація. Бажано, щоб при розробці повноцінного застосунку була використана саме статично типізована мова програмування, адже це покращить безпеку коду, зменшивши кількість помилок ще до компіляції, покращить читабельність та зробить проєкт більш масштабованим у майбутньому. Для створення моделі все ж краще використати мову з динамічною типізацією, адже це просто пришвидшить розробку;

– структури даних. Вони можуть визначати, як інформація зберігається в пам'яті та як можна звертатися до цієї інформації, допомагають зберігати, організовувати та маніпулювати даними;

– веб-API. Серцевина застосунку, взаємодія за допомогою якої повинна бути ключовим фактором. Веб-API визначає набір правил та протоколів для будівництва та взаємодії з вебслужбами. Вони дозволяють різним програмам спілкуватися між собою за допомогою стандартних протоколів, таких як HTTP;

– вебпакети. Бібліотеки та фреймворки, які надають інструменти та функції для розробки вебзастосунків. Вони можуть стосуватися як клієнтської, так і серверної частин вебзастосунку;

– рендеринг. Процес перетворення коду та ресурсів у видимий вебконтент, який може відображатися користувачам у браузері. Бажано реалізувати як клієнтський, так і серверний рендеринг для більшої гнучкості та кращого користувацького досвіду;

– геоінформаційна система. Такий інструмент необхідний для збору, управління, аналізу та візуалізації геопросторових даних. Відповідний їй API дозволить програмі взаємодіяти з мапою реального світу, обмінюючись даними та функціональністю із відповідним сервісом;

– багатопоточність. Визначна вимога у рамках керування процесами у ГА, зокрема для зменшення часу пошуку рішення. Підтримка багатопоточності надає можливість програмі виконувати кілька завдань паралельно в межах одного процесу.

– візуалізація даних. Для повного аналізу та наочного сприйняття результатів роботи генетичних методів, проведення досліджень та пошуку найоптимальнішої конфігурації параметрів необхідно подавати дані у зрозумілій людині формі;

– робота з файловою системою. Читання, запис, редагування та управління файлами особливо важливе у контексті розробки, адже буде проведена робота з файлами різного розширення та формату, з різним

кодуванням та принципами формування вмісту. Тут визначним є саме простота та багатство варіантів взаємодії з файловою системою як через браузер, так і через операційні системи;

- час виконання. Для пришвидшення дослідження та безпосередньо виконання пошуку рішення за допомогою ГА необхідно звернути увагу саме на час виконання коду;

- швидкість розробки. Актуально, адже програму необхідно розробити у чіткі, стислі терміни, враховуючи зручність мови, доступні бібліотеки, інструмент, спільноту та фундаментальні закладені можливості мови до швидкого написання мінімально готового продукту;

- спільнота. Активна та велика спільнота розробників гарантує, що будуть доступні ресурси, документація, бібліотеки та підтримка в вирішенні потенційних проблем, які є неминучими;

- хостинг. Доступність хостингових рішень визначає, наскільки легко можна буде розгорнути та масштабувати розроблене програмне забезпечення, надавши доступ користувачам в інтернеті;

- пакетний менеджмент. Проєкт буде зростати і розширюватися, тому важливо мати можливість легко додавати, оновлювати та видаляти пакети без порушення функціональності, а також слідкувати за версіями та конфліктами упродовж розробки;

- контроль версій. Навіть попри розробки проєкту наодинці важливо слідкувати за версіями та мати бекап на випадок втрати коду.

3.1.2 Вибір мови програмування

Із перелічених у попередньому пункті вимог до програмних засобів викоренимо критерії, за якими можна провести вибір визначного інструменту розробки – мови програмування. Порівняння мов JavaScript (JS) та Python за заданими критеріями наведено у таблиці 3.1.

Таблиця 3.1 – Аналіз мов програмування

Критерій	JavaScript	Python
1	2	3
Типізація	Динамічна типізація, але з введенням TypeScript (TS), що є суперсетою JS, розробникам надано можливість використовувати статичну типізацію. Порівнюючи з іншими типізованими мовами, TS є доволі гнучким та багатим за синтаксисом [30].	Динамічна типізація, однак починаючи з версії 3.5, в Python було введено анотації типів. Це дозволяє розробникам явно вказувати очікувані типи для змінних та функцій, однак все ж таки це не є статичною типізацією, і слугує як «порада» [31].
Структури даних	Об'єкти (пара ключ-значення), масиви (впорядковані колекції елементів), Set (колекція, яка зберігає унікальні значення без визначеного порядку), Map (колекція пар ключ-значення) [32].	Списки (динамічні масиви), словники (асоціативні масиви, які зберігають пари ключ-значення), множини (невпорядковані колекції унікальних елементів), кортежі (впорядковані, незмінні послідовності) [33].
Веб-API	Вбудована підтримка вебтехнологій, ідеально підходить для створення клієнтської частини застосунків. Існує безліч бібліотек, таких як Axios, Fetch та jQuery.	Використовується для створення серверних застосунків та мікросервісів, що пропонують веб-API. Для клієнтської частини існує бібліотека Requests.

Продовження таблиці 3.1

1	2	3
Вебпакети	Величезний репозиторій пакетів NPM. Для клієнта: React, Angular, Vue. Для сервера: Node.js і його фреймворки, такі як Express. Широкий вибір бібліотек для роботи з даними, анімацією і т. д.	Має потужний репозиторій пакетів PyPI. Django та Flask – це два основних фреймворки для веброзробки на Python. Вбудовані бібліотеки для роботи з базами даних, аутентифікацією, безпекою.
Рендеринг	Традиційно, JS був використовуваний для клієнтського рендерингу та створення односторінкових застосунків (SPA). Однак є можливість і серверного рендерингу (SSR) в React, Nuxt.js для Vue, або Next.js знову ж таки для React [34].	Зазвичай використовується для серверного рендерингу, однак деякі рішення, такі як Pyodide, дозволяють запускати Python безпосередньо в браузері, але це менш поширено.
ГІС	Найбільш популярні бібліотеки Leaflet і Mapbox GL JS. Вони дозволяють створювати інтерактивні карти на вебсторінках. Щодо API, множина вебсервісів надає ГІС-API, яке можна легко інтегрувати, наприклад Google Maps та ArcGIS [35].	Geopandas дозволяє легко обробляти та аналізувати просторові дані в структурі DataFrame, в той час як PyQGIS використовується для розширення та автоматизації QGIS – вільної ГІС-платформи.

Продовження таблиці 3.1

1	2	3
Багатопоточність	Фактично однопоточна мова, однак виконання декількох процесів одночасно забезпечується асинхронною моделлю. Для підтримки паралельності обчислень застосовуються WebWorkers.	Багатопоточний, однак, через Global Interpreter Lock (GIL) в CPython традиційні потоки часто не можуть ефективно виконуватися паралельно. Для паралельних завдань, часто використовується модуль multiprocessing [36].
Візуалізація даних	Основний фокус – це вебвізуалізація. Бібліотека D3.js, дає можливість динамічної, інтерактивної, а головне браузерної візуалізації. Chart.js дає можливість створювати різноманітні графіки [37].	Сильний акцент на наукову та аналітичну візуалізацію. Matplotlib є стандартною бібліотекою для створення візуалізацій, а Seaborn працює поверх нього та додає додаткові можливості для створення графіків [38].
Робота з файлами	У контексті браузера, JavaScript має обмежений доступ до файлової системи з міркувань безпеки. Проте, існує API, як-от FileReader. Node.js вже має повноцінний модуль fs та його похідні, з великими можливостями.	Python має вбудовані засоби для роботи з файлами, а також має модулі, як os та shutil, для управління файлами та директоріями. Бібліотеки, як-от Pandas, дозволяють ефективно працювати з даними у форматах CSV, Excel, SQL.

Продовження таблиці 3.1

1	2	3
Швидкість виконання	Інтерпретована мова, що працює в браузері. Завдяки двигунам, як V8, сучасний JS виконується дуже швидко, завдяки внутрішній оптимізації коду. Також покращити ефективність закликані асинхронні операції [39].	Також інтерпретована мова та повільніша порівняно з компільованими мовами. Проте, для наукових обчислень і обробки даних, Python має оптимізовані бібліотеки, такі як NumPy та Cython, які можуть значно прискорити виконання [40].
Швидкість розробки	Величезна кількість простих та докладно пояснених бібліотек, зручний менеджер пакетів, багато збирачів проєкту та доволі легкий функціональний синтаксис. Однак попри це, усі допоміжні інструменти є «нерідними» одним до одного, що зараз виправлено у Bun.	Зрозуміла об'єктно-орієнтована парадигма, водночас із сильними функціональними можливостями, читабельний код, багатий стандартний модуль та широкий вибір сторонніх пакетів із простим їх менеджером PIP.

Розписавши особливості притаманні обраним мовам програмування, за вибраними визначними критеріями, є сенс скоротити ці описи для більш наочного порівняння та вибору області їх застосування, що і було зроблено та наведено у таблиці 3.2.

Таблиця 3.2 – Порівняння мов програмування

Критерій	JavaScript	Python
Типізація	динамічна, TypeScript	динамічна, анотація типів
Структури даних	±	+
Веб-API	клієнтська сторона	серверна сторона
Веб-пакети	+	+
Рендеринг	клієнтський, серверний	серверний
ГІС	для браузеру	для аналізу даних
Багатопоточність	-	+
Візуалізація даних	-	+
Робота з файлами	+	+
Швидкість виконання	±	±
Швидкість розробки	+	+

Аналізуючи інформацію, наведену у таблиці 3.1 та 3.2 можна зробити наступний висновок. Хоча обидві мови є дуже популярними та багатими, за допомогою них дійсно можна створювати цілісний застосунок, використовуючи одну з них, все ж таки прийнято рішення, спираючись на переваги, загальну практику та думки суспільства, застосувати ці мови у різних сферах, а саме для створення інтернет-застосунку використати JavaScript, а для роботи з даними та генетичними алгоритмами використати Python. Тобто виділити на кожен область застосування більш доречний інструмент.

Також за це рішення говорять результати щорічного опитування розробників Python Developers Survey за 2022 рік, проведеного JetBrains [41]. Тут видно, що Python традиційно найчастіше використовується для аналізу даних – 51%, а більш того, друга мова, яку використовують із Python в одних проєктах, це саме JavaScript – 37%. Тому подібне поєднання та взаємна інтеграція мов програмування є зваженим та актуальним рішенням [41].

Загалом, підсумовуючи вищезазначене, повноцінний вебсайт буде розроблено за допомогою JavaScript, а точніше TypeScript, за це говорять наступні чинники:

- статична типізація TypeScript – невід’ємна умова розробки відносно великого проєкту, тим паче при застосуванні сторонніх бібліотек. Можливості TypeScript настільки великі, що він собою представляє Turing Complete; можливості ж анотації типів у Python недостатні;

- наявність веббазованих ГІС-застосунків та інтерактивних карт, тоді як Python надає глибоку підтримку для просторового аналізу та обробки геоданих без браузерної інтеграції;

- більша кількість бібліотек та пакетів в контексті вебзастосунків;

- можливість як клієнтського, так і серверного рендерингу;

- відмінна робота з підтримкою веб-API, особливо важливо для клієнтської частини;

- живе перезавантаження.

Для аналізу даних, проведення дослідження та розробки моделі з використанням ГА, використано Python через його наступні переваги:

- більш різноманітний набір вбудованих структур даних порівняно з JavaScript, що робить його дуже зручним для роботи з комплексними наборами даних;

- різноманітніші підходи до багатопоточності;

- більший арсенал інструментів та бібліотек для наукової та детальної аналітичної візуалізації;

- більш інтуїтивний та потужний інструментарій для роботи з файлами в різноманітних сценаріях, включаючи обробку великих наборів даних;

- швидкий та оптимізований для наукових обчислень завдяки спеціалізованим бібліотекам;

- застосування переважно у розробці серверної частини;

- читабельний.

3.1.3 Вибір стратегії розробки

Зазвичай, вебпроект розробляється у вигляді полірепозиторію, де серверна та клієнтська частини являють собою два фактично незалежні проекти, грубо кажучи це два різні git-репозиторії. Однак останнім часом, навіть для відносно невеликих проектів, набирає популярність протилежна стратегія розробки – монорепозиторій, де код клієнта, сервера та будь-яких інших пакетів зберігається в одному місці і являє собою один великий git-репозиторій. Попри це, монорепозиторій це дещо більше, ніж просто купа коду в одному місці, це також принцип зв'язування коду, його розробка та підтримка. У дипломній роботі пропонується розроблювати проект у вигляді монорепозиторію, а перед цим впевнитись у доцільності такого рішення.

У таблиці 3.3 наведено порівняння двох стратегій розробки проекту, полі- та монорепозиторію.

Таблиця 3.3 – Аналіз стратегій розробки

Критерій	Монорепозиторій	Полірепозиторій
Централізований доступ до коду	+	-
Єдиний набір інструментів	+	-
Здатність вносити атомарні коміти	+	-
Оптимізоване управління залежностями	+	-
Легкість у розумінні структури	+	-
Єдина документація	+	-
Необхідність завантаження всього коду	-	+
Масштабування (великі проекти)	±	±

Із таблиці 3.3 можна виділити певні недоліки монорепозиторію у порівнянні з принципом розділення обов'язків по декількох проєктах:

- необхідність завантаження всього коду. Розробник, який працює над певною частиною проєкту, може відчувати незручності, оскільки йому потрібно клонувати весь репозиторій, незалежно від його поточних завдань;

- масштабування. При збільшенні обсягу коду та кількості розробників можуть виникнути проблеми зі швидкістю збірки та тестування. Це може призвести до збільшення часу на CI/CD процеси і ускладнити масштабування проєкту.

Тим не менш, серед головних переваг монорепозиторію неможливо не зазначити:

- one source of truth. У монорепозиторії весь код знаходиться в одному місці, що гарантує усім проєктам можливість посилатися на одні й ті ж самі допоміжні пакети, які є пропрієтарними. Ця впорядкованість зменшує шанс конфліктів версій, спрощує управління залежностями і робить необов'язковою публікацію цих пакетів, оскільки достатньо просто створити проєкт у тому ж самому репозиторії;

- єдині допоміжні засоби. Всі проєкти в монорепозиторії використовують один набір інструментів, такі як літери, прітієри, конфіги TypeScript тощо. Це не тільки полегшує управління та підтримку, але також забезпечує однорідність коду і стандартизацію розробки;

- атомарні коміти. Монорепозиторій дозволяє вносити зміни до проєкту, одразу змінюючи увесь можливий код задля створення оновлення, впровадження покращень чи виправлені багу. Тобто за один коміт розробник може провзаємодіяти з усією базою коду, у той час коли при полірепозиторії внесення змін може супроводжуватися багатьма комітами у декількох проєктах. Подібні атомарні коміти дають можливість для кращого масштабуванню проєкту в подальшому;

- оптимізація пакетного менеджменту. Завдяки спільному використанню залежностей між проєктами в монорепозиторії, можна

зменшити загальну кількість залежностей. Наприклад, використання `npm` у монорепозиторії може збільшити швидкість завантаження пакетів та сприяти ефективнішому тестуванню проєкту;

- логічна структура. Завдяки монорепозиторію розробнику набагато легше розуміти зв'язки між різними частинами коду, від бази даних (БД) до подання цих даних клієнту у браузері;

- єдина база документації. У монорепозиторії всі файли зібрані в одному місці. Це не тільки спрощує доступ до інформації для розробників, але і дозволяє створити документацію як окремий проєкт, що забезпечує однорідність, повноту і структурованість документації.

Виходячи із потреби створити проєкт для клієнтської частини, сервер та пакет для проведення оптимізації інфраструктурних об'єктів (тобто фактично 3 проєкти), а також із проведеного аналізу, було прийнято рішення розроблювати програму у вигляді монорепозиторію.

3.1.4 Вибір геоінформаційної системи

Програма повинна мати інтеграцію з геоінформаційною системою для того, щоб користувач міг взаємодіяти з мапою та розташовувати об'єкти для проведення за ними оптимізації. Через те що розробка буде відбуватися мовою TypeScript, за допомогою фреймворку React.js, то обрана ГІС повинна мати відповідний API.

При виборі ГІС було розглянуто та порівняно три інструменти:

- Google Maps, як найпопулярніша, найзрозуміліша та, з великою імовірністю, вже знайома простому користувачу;

- ArcGIS, як найбільш розширений за своїми можливостями та інфраструктурою інструмент;

- React Leaflet, як дуже легка та швидко впроваджувальна у вебзастосунок, написаний за допомогою React.js.

У таблиці 3.4 наведено порівняння цих трьох обраних геоінформаційних систем.

Таблиця 3.4 – Аналіз ГІС

Критерій	Google Maps	ArcGIS	React Leaflet
API та інтеграція з React.js	Гарне	Відмінне	Гарне
Документація	Гарна	Відмінна	Достатня
Гнучкість і масштабованість	Висока	Найвища	Висока
Функціональні можливості	Високі	Найвищі	Стандартні
Ліцензування та вартість	Обмежене	Доступне	Вільне
Спільнота та підтримка	Велика	Найбільша	Достатня
Персоналізація	Добра	Найкраща	Добра
Доступ до геоданих	Хороший	Найширший	Стандартний

Вміст таблиці 3.4 можна прокоментувати наступним чином:

– Google Maps та React Leaflet мають відмінну інтеграцію з React.js та кастомізацію за допомогою HTML/CSS, однак ArcGIS має настільки потужний JavaScript API, що пропонує фактично свій власний фреймворк з унікальною концепцією рендерингу, UI систему та принципи написання коду;

– документація Google Maps може бути обмежена для нетривіальних проблем, React Leaflet є мало деталізованою, у той час коли документація ArcGIS високоякісна, детальна та налічує численні практичні приклади;

– React Leaflet не націлена на роботу з великими обсягами даних та застосовується у невеликих проєктах, а от Google Maps та ArcGIS здатні обробляти великі обсяги даних з високою продуктивністю та доволі добре оптимізовані під такі цілі;

– ArcGIS має найбільш розширений спектр інструментів для аналізу, візуалізації та редагування геоданих, у той час коли React Leaflet має лише базові функціональні можливості для вебкартографії;

– Google Maps має безкоштовний план лише на три місяці, після чого потребує платежів, ArcGIS пропонує такий безкоштовний план, який повністю задовольняє вимогам даної дипломної роботи. React Leaflet взагалі надає відкритий код;

– усі три ГІС мають активну спільноту, однак ArcGIS відрізняється тим, що має окремий форум для обговорення проблем. Під час розробки дипломної роботи велика кількість проблем та рішень була знайдена саме не за документацією, а за форумом;

– ArcGIS має незрівнянні можливості персоналізації з розширеними інструментами для налаштувань;

– Google Maps надає стандартний доступ до геоданих, у той час коли ArcGIS має доступ до високоякісних геоданих, з можливістю імпорту та експорту у великій кількості форматів, що є визначним для дотриманням вимогам дипломної роботи.

Якщо просто скласти усі переваги, то виходить що ArcGIS впевнено випереджає своїх конкурентів за можливостями та відповідає усім поставленим вимогам до програмного забезпечення та геоінформаційного модуля окремо. Таким чином, для взаємодії з картою було обрано ArcGIS.

3.2 Структурна схема програми та її опис

Програму спроектовано за архітектурою Model-View-Controller (MVC). Сервер працює під Node.js, загальну структуру якого визначає фреймворк Express.js: наявність роутерів, відповідна обробка помилок, наявність middleware. Також однією з особливостей є винесення бізнес логіки у окремий шар сервісів. Загалом, застосунок характеризується можливостями аутентифікації, обробки запитів, взаємодії з базою даних, відправки електронних листів, виконання зовнішніх API запитів, запуск модуля Python який реалізує логіку генетичної оптимізації.

На рисунку 3.2 зображено структурну схему розробленої програми.

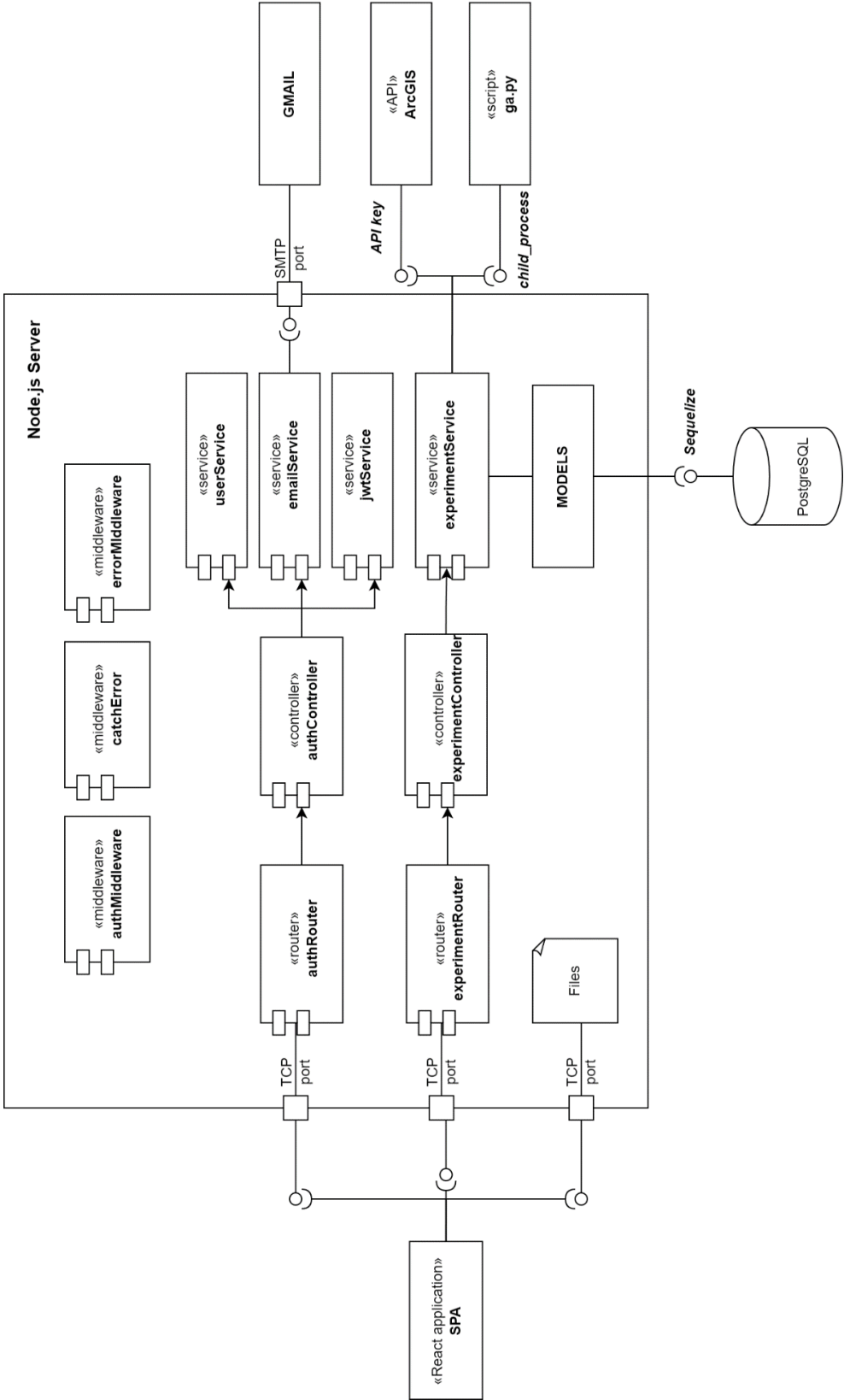


Рисунок 3.2 – Структурна схема програми

У таблиці 3.5 наведено опис визначних компонентів системи, структурна-схема якої зображена на рисунку 3.2.

Таблиця 3.5 – Опис компонентів програми

Назва компоненту	Опис
1	2
SPA	React-застосунок, який функціонує як клієнтська частина, забезпечує інтерактивну взаємодію із користувачем і отримує дані з сервера. Більш докладно структуру клієнту описано у розділі 3.
Routers	Оброблює маршрутизацію запитів, будуючи RESTful API та визначаючи адреси, методи та функції обробки. Містить послідовний ланцюжок викликів проміжного програмного забезпечення (middleware) та контролерів.
Files	Статичні файли серверу, без урахування файлів клієнта. Тут можуть розташовуватися як звичайні файли клієнта, у випадку відсутності потреби у SPA, так і файли даних, фото, іконки тощо. Файли подаються за допомогою вбудованої у Express функції static().
authController	Контролер, що відповідає за логіку аутентифікації користувачів. Визначає 4 функції: реєстрацію; логін; функцію перевірки валідності токена доступу (access token) та його оновлення, щоб користувачу не потрібно було постійно виконувати аутентифікацію (функція refresh); функцію активації акаунту за поштою, для верифікації користувача. Цей контролер взаємодіє з низкою відповідних сервісів.
experimentController	Головною особливістю контролера є визначення потоку проведення експерименту, послідовно: витягти дані із запиту, обробити дані, провести експеримент з відповідними параметрами, зберегти результат та повернути його клієнту. Також цей модуль містить операції читання та видалення окремого експерименту.

Продовження таблиці 3.5

1	2
userService	Визначає функції читання окремого користувача за ідентифікатором, поштою, створення користувача та нормалізацію його даних.
emailService	Створює об'єкт для взаємодії з поштовим клієнтом через пакет Nodemailer. Зберігає функцію перевірки з'єднання, функцію відправки повідомлення на відповідну адресу з визначеним контентом та функцію відправки саме посилання для активації акаунту.
jwtService	Доступ до деяких ресурсів сервера обмежений правами користувачів, наприклад, експерименти можуть проводити лише зареєстровані користувачі. Для верифікації кожен запит підписується унікальним jwt токеном на основі унікальних даних користувача. Модель зберігає функцію створення токена та перевірки його валідності.
experimentService	Визначає функції першочергової обробки даних, їх підготовки до експерименту та перевірки на коректність; функцію проведення експерименту; базові функції читання, збереження та видалення даних про експерименти.
authMiddleware	Визначає функцію перевірки, чи має користувач доступ до певного ресурсу на сервері. Прописується як middleware, працює із Bearer-токеном, у разі помилки викидає кастомну помилку, яка збирається за допомогою catchError та оброблюється у кінці ланцюжку обробки запитів errorMiddleware.

Продовження таблиці 3.5

1	2
catchError	Модуль визначає однойменну функція, яка повинна обгортати кожен функцію у обробник помилок, у разі виникнення якої передавати управління вбудованому в Express.js механізму обробки.
errorMiddleware	Фінальний обробник для кожного запиту. У разі виникнення помилки чи викидання її у вигляді виключення забезпечує їх оборку. Працює з усіма помилками та зокрема з помилками валідації, роботи з БД та кастомними помилками ApiError.
Models	Моделі визначають загальну структуру таблиць БД та методи взаємодії з ними, спрощуючи роботу загалом. Моделі синхронізуються на першому етапі запуску серверу, та не керується за допомогою міграцій. Обраною технологією для зв'язування БД з програмою є Sequelize.
PostgreSQL	Реляційна база даних застосунку, під керуванням PostgreSQL.
Gmail	Поштовий клієнт, який налаштовано для автоматичного відправлення повідомлень із серверу. Для цього пройдено двофакторну верифікацію gmail, створено клієнт та налаштовано пароль для застосунку.
ArcGIS API	Обширний програмний інтерфейс для роботи із мапою та географічними об'єктами. На сервері необхідний для визначення бінарної матриці за шарами розтушування серверних об'єктів та клієнтських областей.
main.py	Модуль, що визначає логіку роботи генетичного алгоритму та проведення оптимізації. Запускаються як окремий процес за допомогою пакету child_process.

3.3 Проектування структури бази даних

Дані програми зберігаються у базі даних під керівництвом системи керування БД PostgreSQL, а також у хмарному сховищі, яке надає ArcGIS. Таблиці створюються автоматично при синхронізації моделей, описаних за допомогою Sequelize, при виконанні відповідного програмного сценарію. Додатково відбувається заповнення початковими даними.

Концептуальна схема БД програми зображена на рисунку 3.3.

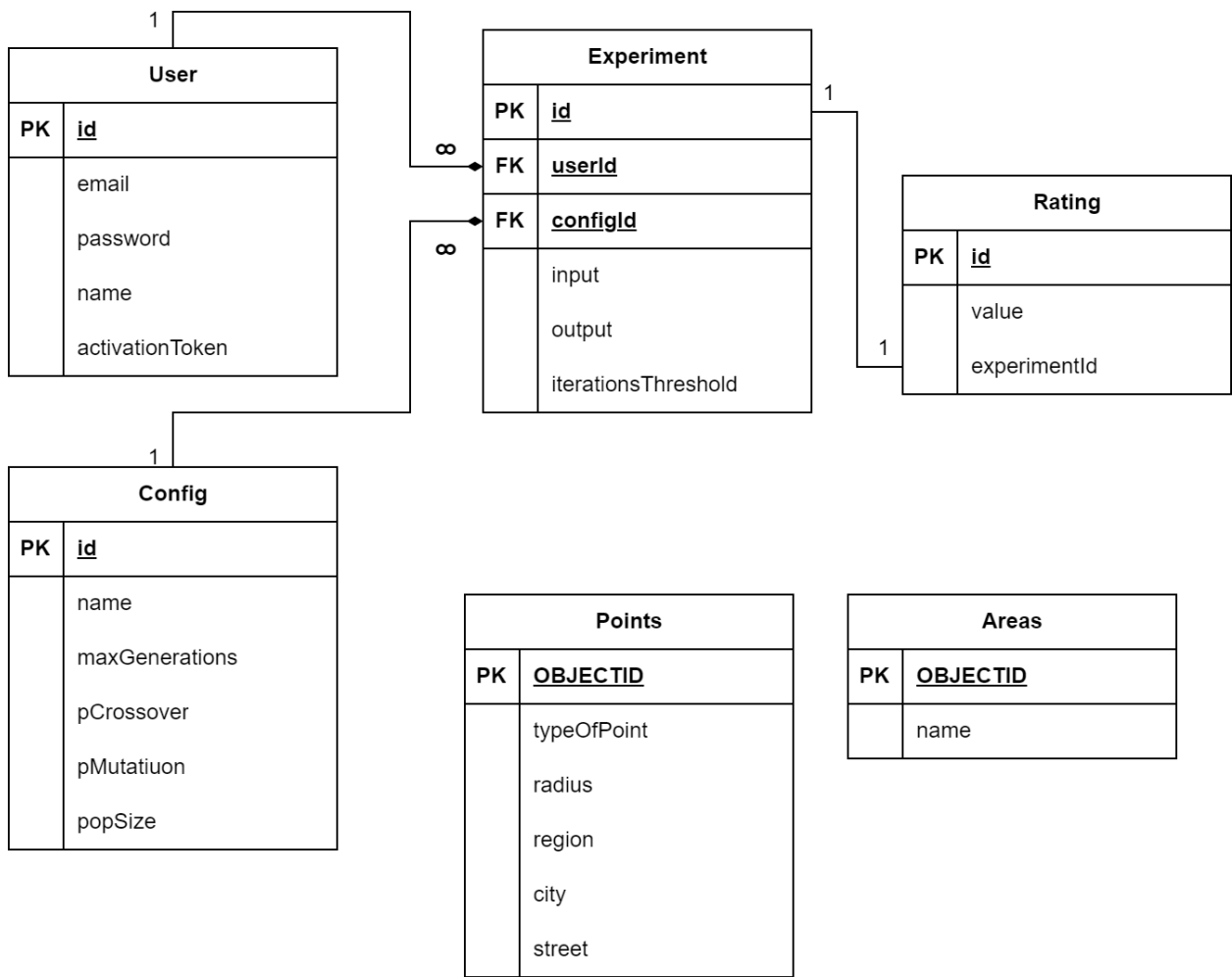


Рисунок 3.3 – Логічна схема БД програми

Опис таблиць та даних, які вони зберігають та які були розроблені в ході створення програми, наведено у таблиці 3.6.

Таблиця 3.6 – Опис основних сутностей БД

Назва сутності	Опис
User	Зберігає дані про користувача. Пошта має бути реальною електронною поштою, представляти унікальне значення, так само як і пароль. Пароль в свою чергу гешується без використання солі, однак із заданою кількістю раундів за допомогою пакета bcrypt. Ім'я користувача це те, як його буде відображено у системі, однак це значення може бути неунікальним. Токен активації акаунту створюється тоді ж, коли вперше додається запис користувача до БД. При активації акаунту це поле стає null.
Experiment	Зберігає дані про експеримент, щоб користувач мав доступ до них будь-коли. Запис створюється лише при успішному завершенні експерименту. Містить показник максимальної кількості ітерації для проведення пошуку, що виставляє користувач.
Rating	Містить показник від 0 до 5 на скільки користувач оцінює ефективність проведеної оптимізації. Потрібен для довідкової інформації та ніяк не впливає на хід пошуку.
Config	Містить дані про конфігурацію алгоритму. Програміст може за унікальним значенням тієї чи іншої конфігурації задавати, з якими еволюційними параметрами буде запускатися оптимізація.
Points	Фактично це атрибути шару ознак ArcGIS, які зберігаються у хмарі, доступ до якої надається через унікальний ключ. Зміна цих даних відбувається або через панель адміністратора, або користувачем через розроблену програму. Дана таблиця зберігає дані про сервісні центри.
Areas	Зберігає атрибути для клієнтських областей, що фактично лише один рядок – що позначає назву області, навіть не адреса, тому що область може, наприклад, захопити декілька вулиць одночасно.

3.4. Висновки за розділом 3

У розділі 3 було описано функціональність програми та, згідно зі сформованими критеріями, порівняно мови програмування – було прийнято рішення використати TypeScript для створення вебзастосунку та Python для розробки моделі з використанням генетичного алгоритму.

Сама розробка буде проходити у монорепозіторії, а робота з картою буде забезпечена інтеграцією з ArcGIS.

Сервер буде розроблено на Node.js з використанням фреймворку Express.js, а клієнтська частина з використанням React.js. У процесі розробки буде застосовано технології: Sequelize для взаємодії з БД під керівництвом PostgreSQL, Nodemailer для роботи із електронною поштою, Yup для валідації запитів, axios та React-Query для виконання запитів від клієнту до серверу, Redux Toolkit як стейт-менеджер, Bootstrap для написання стилів, Framer Motion для створення анімацій та Formik для створення форм. Робота буде проходити у середовищах Spyder та Visual Studio Code.

Також було розроблено структурну схему програми та логічну схему бази даних.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В даному розділі наведено діаграми головних бізнес-процесів та визначні алгоритми функціонування програмного забезпечення, описано головні модулі реалізованої програми.

4.1 Алгоритми функціонування програмного забезпечення

Одним з ключових реалізованих бізнес-процесів є авторизація користувача у системі.

Процес починається з того, що користувач завантажує головну сторінку. Якщо у локальному сховищі присутній access token, система перевіряє його валідність. У разі, якщо він недійсний, користувач переходить до сторінки авторизації, де він вводить параметри реєстрації. Далі система перевіряє валідність цих параметрів. Якщо вони недійсні, користувач отримує повідомлення про помилку, у протилежному випадку користувач продовжує реєстрацію. Потім відбувається перевірка на наявність унікальності введених даних. Якщо дані не унікальні, користувач знову повертається до етапу введення параметрів реєстрації.

У випадку, коли дані унікальні, користувачу відправляється лист із посиланням на активацію аккаунту на електронну пошту. Після переходу за посиланням клієнт зчитує активаційний токен та, відправляючи його на сервер, акаунт активується. Далі користувач може увійти в систему, завершуючи процес реєстрації.

Бізнес-процес реєстрації користувача зображено на рисунку 4.1.

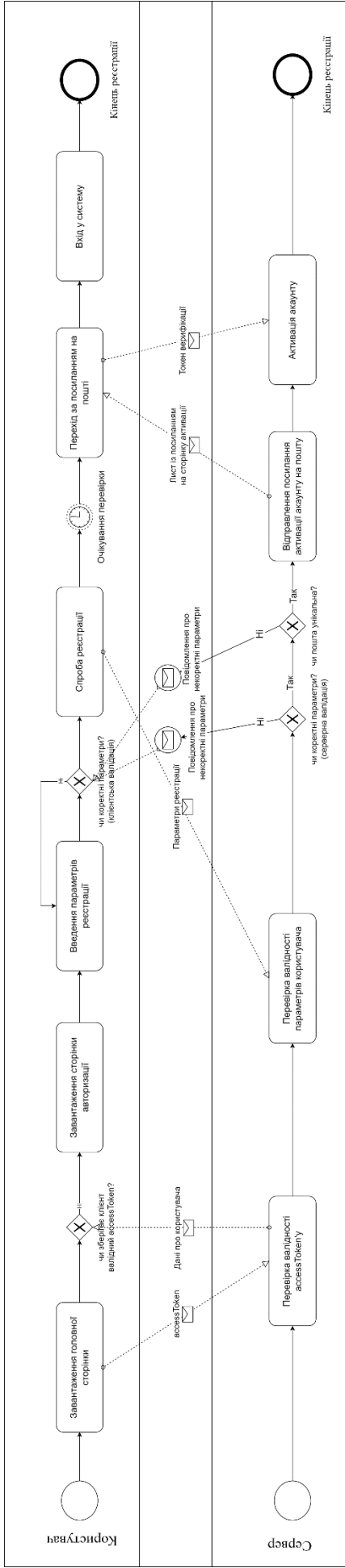


Рисунок 4.1 – Бізнес процес реєстрації нового користувача у системі

На рисунку 4.2 зображено алгоритм визначення початкової конфігурації системи соціальної інфраструктури для проведення оптимізації, що передбачає розташування на інтерактивній мапі як сервісних центрів, у вигляді точок на мапі, так і клієнтських областей, у вигляді багатокутників.

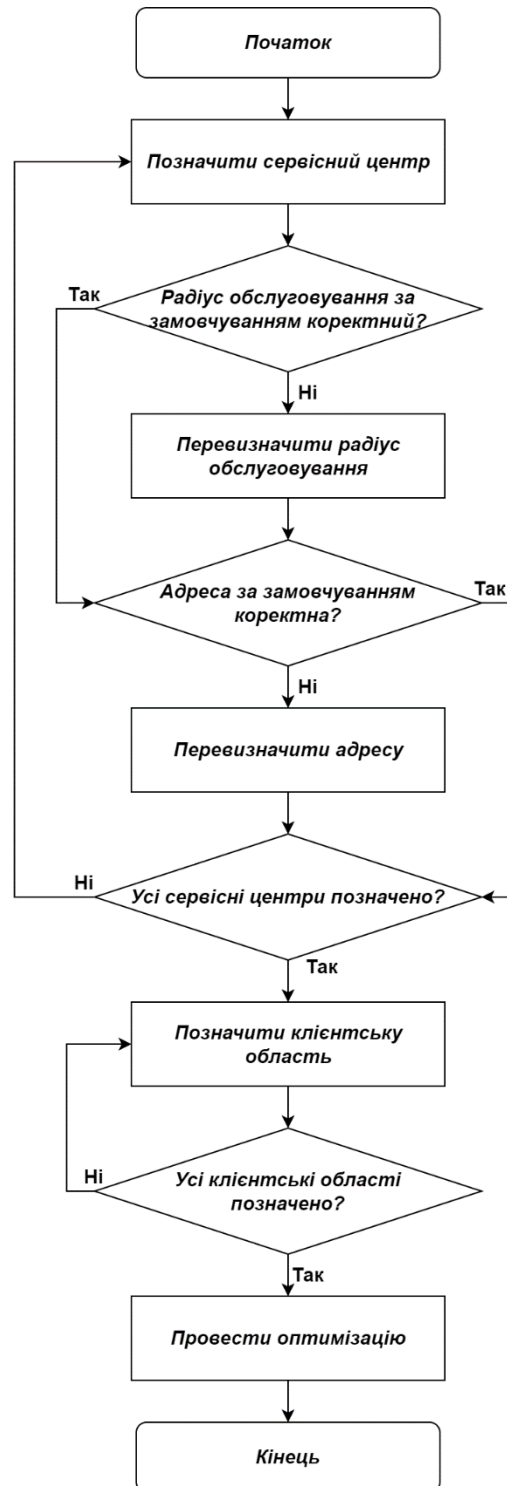


Рисунок 4.2 – Алгоритм позначення об’єктів соціальної інфраструктури на мапі для оптимізації

4.2 Опис модулів реалізованої програми

4.2.1 Опис клієнтського застосунку

Якщо у пункті 3.2 описано структурну схему програми з акцентом на серверну частину, де клієнт фактично було представлено одним окремим компонентом – SPA, то на рисунку 4.3 зображено структуру саме клієнтського застосунку.

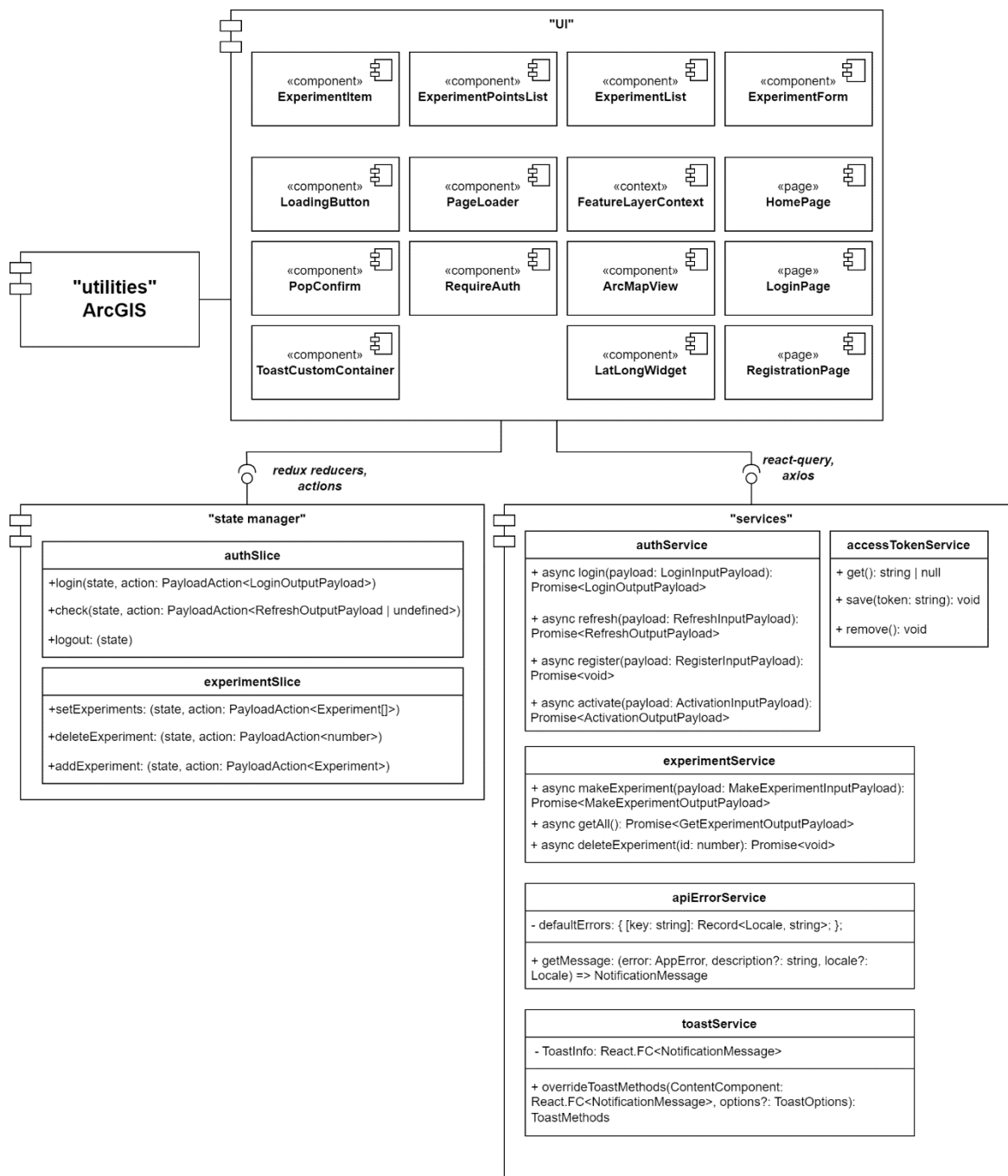


Рисунок 4.3 – Структурна схема клієнтської частини програми

Сама реалізація обумовлена фреймворком React.js та принципами, які закладають супутні бібліотеки та допоміжні пакети. Умовно програма поділяється на 4 складові частини: користувацький інтерфейс, менеджер стану застосунку, сервіси по взаємодіє із сервером та функції інтеграції ГІС. Менш важливими компонентами, але вартими зазначеннями, є набір типів, допоміжні функції, інструменти встановлення з'єднання із сервером та інтерцептори, стилі та медіа елементи, такі як шрифти та картинки.

4.2.1.1 Функції ArcGIS

Інтеграція інтерактивної мапи відбувається при завантаженні головної сторінки сайту, при першому рендерингу головного компоненту. Спочатку встановлюється з'єднання із хмарним сховищем за секретним ключем, після цього відбувається послідовна побудова двох шарів ознак – для клієнтських областей та сервісних центрів. Потім створюється безпосередньо об'єкт карти, який включає попередньо створені шари ознак, та об'єкт його інтерактивного представлення на сайті. Після цього відбувається додавання допоміжної функціональності, а саме:

- створюється спадаючий список для фільтрації клієнтів за типом (звичайні пункти незламності або пункти незламності бізнесу);
- додається слухач події натискання на сервіс для малювання на мапі радіусу обслуговування;
- послідовно додаються вікна Editor для редагування шару ознак, Search для пошуку місця за його адресою та CoordinateConversion для визначення поточного положення курсору на мапі;
- додається обробник події зміни положення курсору на сторінці для відображення поточних координат довготи та широти;
- додається обробник події додавання нового сервісного центру на мапу для запити поточної адреси за його положенням, та додавання цієї адреси у атрибуту ознаки.

Перелік функцій та їх призначення, які розроблені для інтеграції на сайт можливостей ArcGIS наведено у таблиці 4.1.

Таблиця 4.1 – Опис функцій ArcGIS

Функції	Опис
1	2
<code>createAreasLayer(): FeatureLayer</code>	Створює об'єкт шару ознак для клієнтських областей за посиланням, встановлює його назву.
<code>createPointsLayer(): FeatureLayer</code>	Створює об'єкт шару ознак для сервісних точок за посиланням. Встановлює різні позначення для різних типів пунктів незламності. Визначає структуру таблиці атрибутів, яка відображається при натисканні на точку.
<code>getAreasData(pointsLayer: FeatureLayer)</code>	Робить запит на сервер ArcGIS для витягання атрибутів для кожної області та проводить нормалізацію.
<code>drawRadiusEvent(view: MapView, layerToFilter: FeatureLayer, areasLayerView: __esri.FeatureLayer View): IHandle</code>	Додає обробник на подію кліку лівою кнопкою миші на мапі. У разі, якщо натискання відбулося на точку, відбувається запит до серверу на отримання атрибутів точки, які потім нормалізуються. Далі будується радіус обслуговування та усі клієнти, які не входять у нього, зникають на мапі. Для повернення у вихідний стан необхідно ще раз натиснути у довільній точці карти.

Продовження таблиці 4.1

1	2
<code>getAddressByLocation(location: Point __esri.ViewClickEvent)</code>	Асинхронна функція, яка визивається при створенні нового сервісного центру та витягає адресу за координатами точки. У разі якщо адресу не визначено повертає null, у протилежному випадку нормалізує дані, беручи лише регіон, місто та вулицю.
<code>getPointsData(pointsLayer: FeatureLayer)</code>	Виконує запит на сервер для витягання даних про конкретну точку на мапі.
<code>getRadiusById(pointsLayer: FeatureLayer, objectId: number)</code>	Виконує запит на сервер для витягання радіусу серверної ознаки. У протилежному випадку повертає null.

4.2.1.2 Реалізація користувацького інтерфейсу

Користувацький інтерфейс розроблено у формі SPA, тобто таким чином, що при переході на інші сторінки контент не перезавантажується, що у свою чергу суттєво покращує користувацький досвід. Важливо також зазначити що усі будівельні блоки – це функціональні React-компоненти.

З певним ступенем абстракції усі компоненти можна поділити на ті, що представляють сторінки, ті, що забезпечують взаємодію користувача із мапою, ті, що забезпечуються проведення експерименти та перегляд їх же результатів, а також просто допоміжні компоненти, наприклад для відображення контекстного меню чи стану завантаження.

Також необхідно підкреслити, що для покращення продуктивності роботи було використано компоненти вищого порядку, що зменшують кількість необов'язкових рендерів.

Опис реалізованих компонентів наведено у таблиці 4.2.

Таблиця 4.2 – Опис функціональних компонентів та сторінок

Компоненти	Опис
1	2
LoginPage	Реалізує форму авторизації у системі під керівництвом formik. Форма складається із поля для пошти та поля для паролю. Передбачена клієнтська валідація. При натисканні на кнопку «Увійти» форма блокується до завершення запиту на авторизацію. У випадку помилки користувач автоматично переходить на головну сторінку, у протилежному випадку він бачить текст помилки.
RegistrationPage	Повторює сторінку логіну, однак додатково має поле для введення ім'я користувача. У випадку успішної реєстрації користувач побачить повідомлення про необхідність активації акаунту за посиланням на пошті.
HomePage	Головна сторінка визначає структуру інтерфейсу застосунку: праворуч панель проведення експерименту та перегляд результатів, ліворуч мапа та інструменти по взаємодії із нею.
ExperimentItem	Визначає структуру одного запису про експеримент у списку. Компонент обгорнуто у React.memo() з ціллю оптимізації. Містить інформацію про унікальний код експерименту, час його проведення, а також кнопки що забезпечують вивантаження результатів у форматі JSON, перегляд детальної інформації та видалення експерименту.
ExperimentPointsList	Список що можна згорнути/розгорнути для перегляду інформації про рішення по кожному сервісному центру.

Продовження таблиці 4.2

1	2
ExperimentList	Список експериментів. Його особливість у наявності плавної анімації на додавання та видалення елементів у списку, що забезпечено розширеними можливостями бібліотеки framer-motion по контролю життєвим циклом компонентів.
ExperimentForm	Форма, з якої відбувається проведення експерименту. Містить поле для введення максимальної кількості ітерації та інформаційну панель скільки сервісів позначено на карті та яких типів. Перед початком експеримент серверу відправляються дані, що нормалізуються саме на клієнті.
FeatureLayerContext	React контекст, який зберігає шари ознак для клієнтів та сервісів на верхньому рівні представлення, щоб мати до них доступ як на мапі, так і у формі.
ArcMapView	Компонент, що фактично реалізує усю описану логіку з попереднього підпункту одразу після того, як компонент монтовано у DOM дерево.
LatLongWidget	Компонент, що прив'язується до курсору та виводить на екран широту та довготу відносно його положення на карті.
LoadingButton	Компонент кнопки, головною особливістю якого є позначення стану завантаження.
PageLoader	Оверлей, що відображається на час перевірки валідності токenu доступу.
PopConfirm	Функціональний компонент, який реалізує можливість перепитати у користувача, чи хоче він підтвердити дію. Приймає функцію, яка буде виконана у разі підтвердження, функцію для виконання у разі скасування дії, а також функцію, як визначений

Продовження таблиці 4.2

1	2
	параметр children, що сама є функціональним компонентом для визначення розмітки такого вікна.
RequireAuth	Зручний компонент у разі необхідності обмежити деякі сторінки у доступі, наприклад, HomePage доступна лише для зареєстрованих користувачів. Фактично це найбільш оптимальний сценарій створення приватних роутів при використанні react-router.
ToastCustomContainer	Визначає компонент, який приймає параметри, що описують поведінку контейнера повідомлень на сторінці: його положення, час затримки, максимальна кількість, наявність панелі прогресу тощо.

4.2.1.3 Управління станом застосунку

Для уникнення проблеми props drilling, за якої розробник повинен по декілька разів передавати аргументи все глибше і глибше у дереві компонентів, використано менеджер стану – Redux Toolkit. Це фактично обгортка, що покликана спростити роботу зі звичайною реалізацією Redux. Вона дозволяє доволі зручно визначити кожен slice для забезпечення інкапсуляції функціональності зі зміни стану. Для цього потрібно експортувати та використати action-функції, а також reducer-функції.

Також, для спрощення доступу до можливостей менеджера стану створено хуки useAppSelector та useAppDispatch, що дозволяють у будь-якому компоненті або зчитати поточне значення, або виконати якусь action-дію.

У таблиці 4.3 наведено опис action-функцій для авторизації та для управління експериментами.

Таблиця 4.3 – Опис action-функцій менеджера стану

Функції	Опис
login: (state, action: PayloadAction<LoginOutputPayload>)	Приймає на вхід дані про користувача (без паролю) та встановлює їх у об'єкт стану. Приймає згенерований на сервері та встановлює у локальному сховищі access token.
check: (state, action: PayloadAction<RefreshOutputPayload undefined>)	Якщо на вхід подано дані, то виконує дії як при логіні. У будь-якому випадку виставляє значення isChecked у true.
logout: (state)	Видаляє токен доступу із локального сховища та встановлює поточного користувача у null.
setExperiments: (state, action: PayloadAction<Experiment[]>)	Встановлює переданий список експериментів у стан менеджера.
deleteExperiment: (state, action: PayloadAction<number>)	Видаляє експеримент зі списку експериментів за id за допомогою функції filter().
addExperiment: (state, action: PayloadAction<Experiment>)	Додає новий переданий експеримент у список експериментів у стан застосунку у кінець.

4.2.1.4 Сервіси клієнту

Фактично під сервісами тут розуміється певна логіка, що винесена окремо від компонентів, яка реалізує певну важливу допоміжну функціональність. Такі сервіси можуть працювати із різними етапами життя програми та застосовувати не пов'язані між собою інструменти. В ході реалізації програми клієнтської частини було прийнято рішення не використовувати класи, адже це ускладнить сприйняття та логіку доволі легкого коду, тому замість цього тут просто використано ECMAScript модулі.

У таблиці 4.4 наведено опис реалізованих сервісів на клієнтській частині.

Таблиця 4.4 – Розроблені сервіси

Компоненти	Опис
authService	Реалізує запити до серверу за допомогою axios для авторизації, реєстрації, активації та уточнення інформації.
experimentService	Реалізує запити на проведення експерименту, отримання їх списку та видалення одного з них.
accessTokenService	Працює із локальним сховищем, використовуючи ключ «accessToken», отримує, видаляє та зберігає його.
apiErrorService	Реалізує функцію що приймаю помилку типу AppError та в залежності під неї формує повідомлення яке отримує клієнт. Передбачено як повідомлення українською, так і англійською мовами. Можна передавати будь-які повідомлення, або використати вже прописані, наприклад про проблеми із мережею.
toastService	Визначає об'єкт що розширює можливості стандартного об'єкту з бібліотеки react-toastify.

4.2.2 Опис модулів для проведення еволюційного пошуку

Код, який забезпечує пошук мінімальної кількості пунктів незламності, написано мовою Python та завантажується під Node.js за допомогою бібліотеки child_process. Він складається з модулів, що відповідають за роботу генетичних операторів (це селекція, схрещування, мутація), побудову графіків, безпосередньо самого центрального модуля, що містить генетичний алгоритм. Однак докладно необхідно зупинитись на модулі tools, який вбирає усі оператори, які є специфічними для даної задачі, а також класи Individual та

Population, з об'єктами яких і проходить еволюційний процес. На рисунку 4.4 зображено схему пакету для виконання генетичного пошуку.

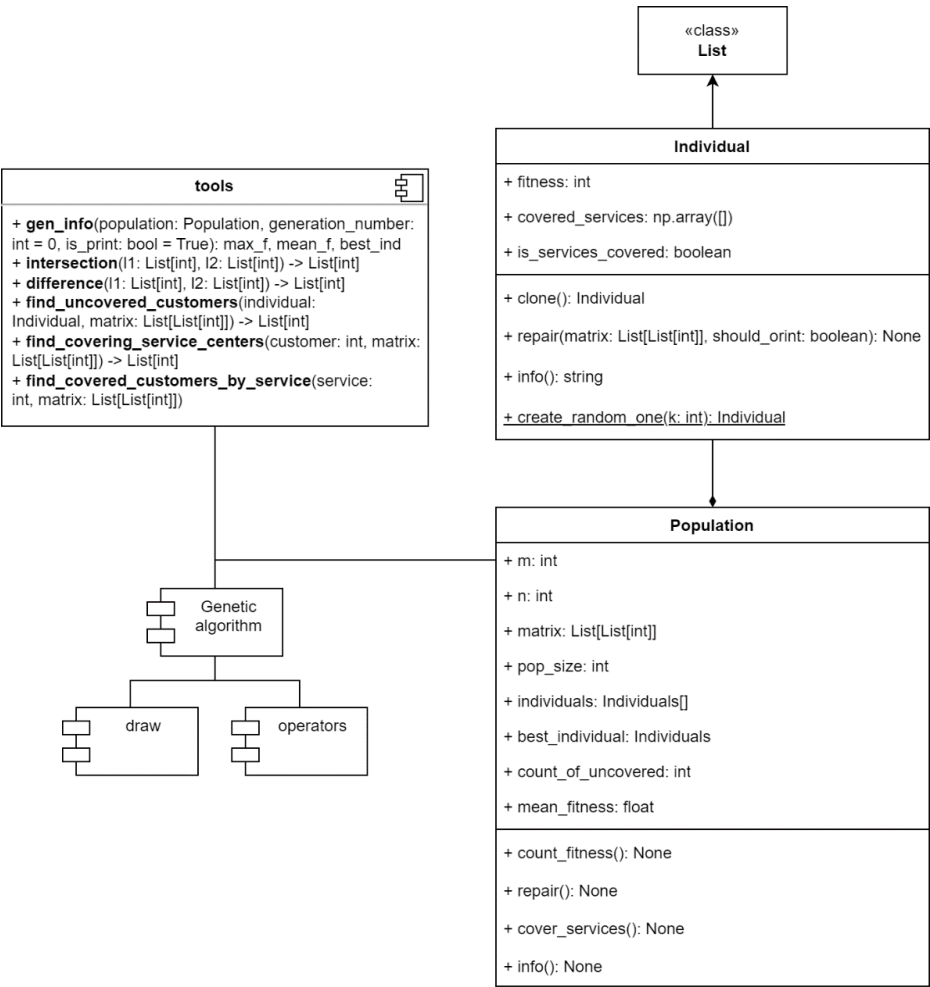


Рисунок 4.4 – Схема модулів для проведення еволюційного пошуку

У таблиці 4.5 наведено опис вмісту модуля tools.py.

Таблиця 4.5 – Опис функцій модуля tools.py

Функції	Опис
1	2
gen_info (population: Population, generation_number: int = 0, is_print: bool = True): max_f, mean_f, best_ind	Приймає об'єкт популяції, номер поточного покоління та логічне значення чи потрібно виводити інформацію у консоль. Повертає значення максимальної

Продовження таблиці 4.5

1	2
	пристосованості, середньої та найбільш пристосованої особини.
intersection (l1: List[int], l2: List[int]) -> List[int]	Реалізує перетин. Приймає два списки із індексами та знаходить ті значення, що є у обох списках.
difference (l1: List[int], l2: List[int]) -> List[int]	Реалізує виключення. Приймає два списки із індексами та знаходить ті значення, що є в першому та відсутні у другому списку.
find_uncovered_customers (individual: Individual, matrix: List[List[int]]) -> List[int]	Приймає об'єкт індивіда та бінарну матрицю. Знаходить індекси тих колонок (клієнтів), які не є покритими сервісами, що представлені у генах хромосоми. Використовує попередньо описані функції, залишає унікальні значення.
find_covering_service_centers (customer: int, matrix: List[List[int]]) -> List[int]	Приймає індекс колонки (клієнту) та, за бінарною матрицею, знаходить ті сервіси, що покривають цього клієнту.
find_covered_customers_by_service (service: int, matrix: List[List[int]])	Приймає індекс рядка (сервісу) та, за бінарною матрицею, знаходить тих клієнтів, що є покритими від цього сервісу.
hamming_distance(chromosome1: Individual, chromosome2: Individual)	Приймає дві хромосоми, знаходить та повертає значення відстані Хеммінга між ними.
average_hamming_distance(population: Population)	Приймає об'єкт популяції. Попарно обраховує відстань Хеммінга між послідовними хромосомами а потім знаходить середнє значення відстані (різноманітність) та повертає його.

4.2.2.1 Клас Individual

Об'єкт такого класу існує тільки як частина популяції – об'єкт класу Population, та призначений для розділення та спрощення логіки по перетворенню та збереження даних про хромосому. Опис полів та методів класу Individual наведено у таблиці 4.6.

Таблиця 4.6 – Опис класу Individual

Поля та методи класу	Опис
+ fitness: int	Зберігає значення пристосованості особини.
+ covered_services: np.array	Зберігає масив індексів сервісів, які включені до рішення та принаймні покривають клієнта.
+ is_services_covered: boolean	Вказує, чи усі клієнти покриті даною особиною.
+ clone(): Individual	Функція глибокого копіювання об'єкта індивіда.
+ repair(matrix: List[List[int]], should_orint: boolean): None	Оператор відновлення, який забезпечує те, що індивід покриває усі необхідні клієнтські центри. Приймає бінарну матрицю та показник, чи потрібно виводити результат роботи у консоль. Використовує tools.py.
+ info(): string	Виводить у консоль інформацію про особину у відформатованому вигляді, зрозумілому користувачу.
+ create_random_one(k: int): Individual	Випадковим чином заповнює гени хромосоми значеннями. Розподіл наступний: на кожні 10 генів 8 мають бути одиницями, а 2 нулями. Приймає розмірність – кількість генів хромосоми.

4.2.2.2 Клас Population

Об'єкт цього класу визначний, адже саме він проходить усі етапи еволюційного перетворення. Загалом, його методи визначають внутрішні зміни хромосоми, а от статичні функції із інших модулів відповідають за перетворення в ході еволюційного процесу. Властивості та методи класу Population наведено у таблиці 4.7.

Таблиця 4.7 – Опис класу Population

Поля та методи класу	Опис
1	2
+ m: int	Зберігає кількість рядків бінарної матриці.
+ n: int	Поле, що зберігає кількість колонок.
+ matrix: List[List[int]]	Бінарна матриця, яка вказує, які клієнтські області покриває конкретний сервіс.
+ pop_size: int	Кількість особин у популяції.
+ individuals: Individuals[]	Особини популяції. Їх генерація відбувається у конструкторі класу.
+ best_individual: Individuals	Властивість, що повертає найбільш пристосованого індивіда.
+ count_of_uncovered: int	Властивість, що повертає кількість непокритих клієнтів.
+ mean_fitness: float	Властивість, що повертає середнє значення пристосованості у популяції.
+ count_fitness(): None	Функція, що обраховує пристосованість для кожної особини популяції.
+ repair(): None	Функція, що застосовує оператор відновлення для кожного індивіда у популяції, передаючи бінарну матрицю покриття.

Продовження таблиці 4.7

1	2
+ cover_services(): None	Проходить усю популяцію та для кожної особини знаходить клієнтів, які ця особина покриває. Якщо не всі необхідні клієнти покриті, то встановлюю відповідне значення параметра is_services_covered.
+ info(): None	Виводить інформацію про популяцію у консоль у вигляді таблиці за допомогою пакету PrettyTable.

4.3 Висновки за розділом 3

В даному розділі було описано дві визначні операції, що забезпечують функціональність розробленого застосунку: процес реєстрації описано у вигляді діаграми бізнес-процесу, алгоритм проведення генетичної оптимізації наведено у вигляді блок-схеми. Також описано визначні програмні модулі для клієнтського застосунку та для програмного пакета, що відповідає за генетичний алгоритм.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

В даному розділі описано призначення програми, перелічено необхідні умови для коректної роботи програми, побудовано та описано сценарії роботи із застосунком. Також було проведено тестування функціональних характеристик програми та наведено результат роботи розробленого генетичного алгоритму на реальних даних одного з міст України.

5.1 Призначення програми

Загалом, розроблена програма призначена для:

- перегляду інформації про об'єкти соціальної інфраструктури;
- редагування інформації про об'єкти інфраструктури, а саме пункти незламності;
- проведення оптимізації системи закладів.

Варто відзначити, що призначення програми, як, до речі, і її візуальний інтерфейс, поділяється на дві частини: перше це робота із мапою, друге – проведення експериментів на основі результатів взаємодії з мапою. Ще одне розділення – це можливість чи неможливість редагувати об'єкти на карті, адже одні користувачі можуть просто передивлятися інформацію про пункти незламності, для власної довідки, а інші експлуатувати повністю наявні можливості, проводячи оптимізацію. Більш того, програма включає комплекс розроблених зручних інструментів по роботі із генетичними алгоритмами, наприклад, створені еволюційні оператори чи засоби побудови графіків. Усі вони можуть використовуватись якщо не кінцевим споживачем продукту, то іншими розробниками, які зацікавлені у розвиненні запропонованої теми.

Коротко визначаючи групи користувачів, які повинні експлуатувати застосунок, необхідно виділити дві наступні групи:

– державні установи, адже все ж таки головне застосування – це оптимізація саме системи пунктів незламності, згідно з їх реальним розташуванням та тим обладнанням, яким вони оснащені. Більш того, мова програми українська, а початкові дані, що завантажені у систему, орієнтовані саме на географію України;

– приватні особи чи компанії, адже навіть попри суттєву орієнтацію на відповідну ціль та певний ринок, соціальна інфраструктура у програмі розглядається більш широко. Головне – це наявність радіусу обслуговування, а урахування обладнання не є обов’язковим. Більш того, географія застосунку необмежена, клієнти та сервіси можуть позначатися у будь-якій точці мапи, тому і роботи можу бути проведена, де завгодно. Так, навіть попри деякі незручності, оптимізація інфраструктури це не лише оптимізація пунктів незламності, але й будь-яких закладів бізнесу.

5.2 Умови виконання програми

Якщо вказувати умови виконання програми на клієнтській частині, то тут достатньо будь-якої операційної системи, наявності безперебійного доступу до мережі інтернет та довільного, сучасного браузера (Firefox, Chrome, Safari, Edge, Internet Explorer починаючи із 10 версії, мобільні браузери Chrome on Android, Safari in iOS, Opera Mini). У такому випадку можна казати про кросплатформність застосунку, що безумовно є великою перевагою.

Умови виконання програми зводяться до наступних пунктів: додержання цілісної файлової структури, яка продиктована вибраною архітектурою розробки (мова про монорепозиторій), дотримання пакетної цілісності та слідкування за їх версіями, задоволення мінімальних апаратних вимог для розгортання серверу.

5.2.1 Файлова структура

Упродовж розробки програми головними директоріями та файлами, що визначають зручність процесу написання коду та гарантують фінальний результат, є:

- `node_modules` – директорія, що зберігає усі скомпільовані та готові до використання залежності проєкту, які були встановлені через пакетний менеджер `npm`. Такі папки знаходяться як у корені репозиторію та зберігають спільні модулі, так і для кожного проєкту окремо;

- `apps` – папка, що у собі зберігає усі проєкти, зазначена як `workspace`. Саме тут знаходиться вихідний код як клієнта, так і сервера. У подальшому тут можуть з'являтися і інші проєкти. Це прерогатива підходу з використанням монорепозиторію;

- `packages` – ще один `workspace`, який також зберігає проєкти, однак які по суті є допоміжними, а не повноцінними застосунками. Наприклад, тут зберігаються модулі, спільні для проєктів, із типами та схемами валідації, та модулі проведення генетичного пошуку;

- `dist` – папка містить збірку проєкту або його частини, які були транспільовані та оптимізовані. Вона може містити статичні файли, такі як HTML, CSS, JavaScript, зображення тощо;

- `.env` – файл, який зберігає глобальні змінні середовища, які будуть встановлені на початку запуску проєкту. Бажано мати декілька таких файлів для конфігурації розробки та безпосередньо для запуску готового проєкту в експлуатацію. Перевага такого файлу в тому, що він дозволяє встановити конфігураційні параметри, які не повинні бути включені до репозиторію з кодом через їх чутливість або специфічність для середовища;

- `tsconfig.base.json` – такий файл зберігається у корені проєкту, його ціль – відслідковувати зміни у перелічених у ньому проєктах та, у разі запуску команди компіляції `tsc`, компілювати усі ці проєкти один за одним;

– `tsconfig.options.json` – звичайний кореневий файл налаштування TypeScript, він має бути успадкований всіма іншими підпорядкованими, внутрішніми проєктами. Це дозволить на всіх проєктах використовувати однакові параметри налаштування;

– `package.json` – цей файл є ключовим для проєктів Node.js. Він містить метадані про проєкт, такі як назва, версія, автори, та найважливіше за все – він перелічує залежності проєкту, тобто інші пакети, які проєкт використовує;

– `.gitignore` – файл, що використовується системою контролю версій git. Він вказує на файли та папки, які git повинен ігнорувати і не відслідковувати. Це можуть бути приватні файли, файли конфігурації, папки з залежностями;

– `npnm-lock.yaml` – файл, створений менеджером пакетів npnm для забезпечення цілісності залежностей. Він заморожує версії всіх пакетів, які використовуються в проєкті, щоб гарантувати, що всі розробники працюють з однаковим набором залежностей;

– `npnm-workspace.yaml` – файл, використовуваний у проєктах з множинними пакетами (монорепозиторіями), який управляється npnm. Він дозволяє налаштувати роботу з декількома пакетами в одному репозиторії, вказуючи, які папки входять до складу робочого простору;

– `.eslintignore` – файл, який використовується утилітою ESLint для визначення файлів або папок, які не повинні перевірятися на відповідність правилам правильності коду. Це може бути корисно для ігнорування файлів, які не потребують перевірки коду;

– `.eslintrc.js` – це конфігураційний файл для ESLint, інструменту для статичного аналізу коду, який допомагає знаходити і виправляти проблеми в коді;

– `main.py` – початковий файл який містить вихідний код для запуску генетичного алгоритму для оптимізації розтушування закладів. Організований так само як окремий допоміжний пакет.

На рисунку 5.1 зображено дерево файлів та директорій, які повинні бути включені у проєкт.

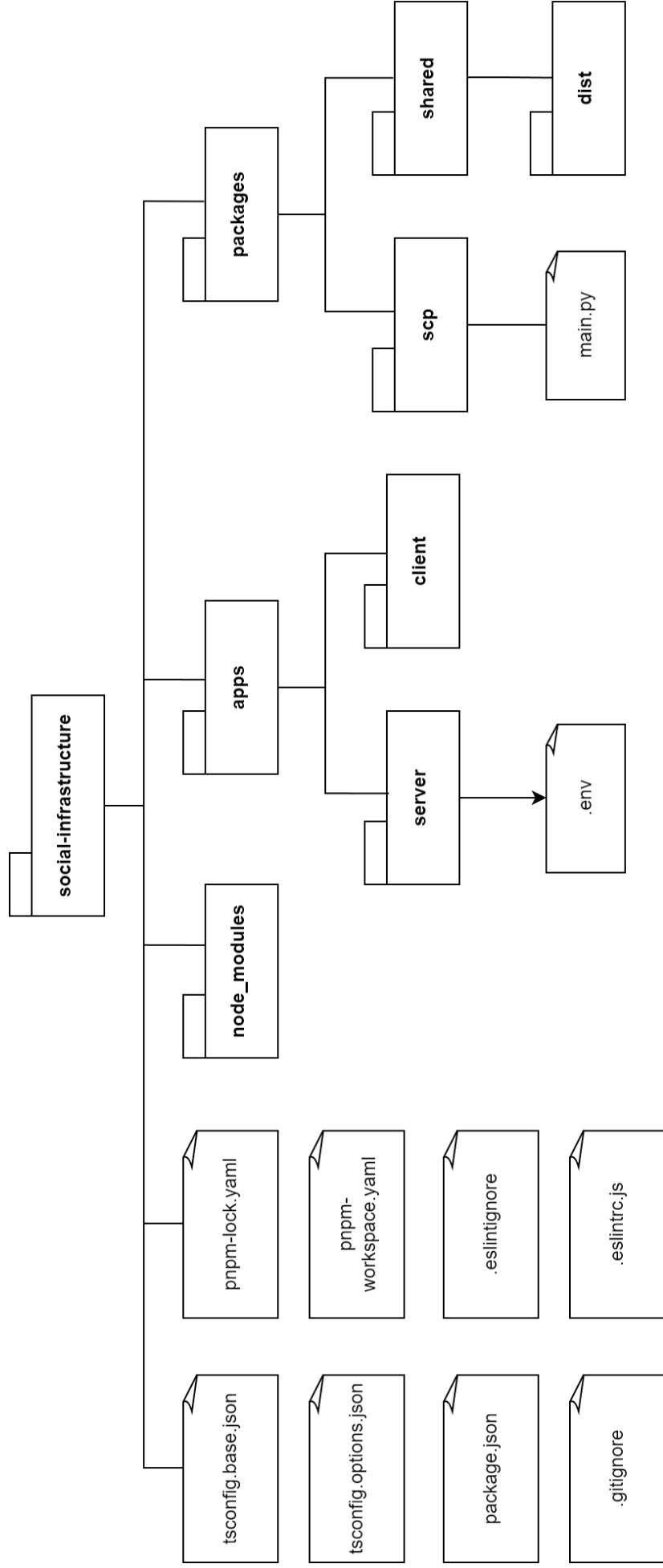


Рисунок 5.1 – Файловая структура проекту

5.2.2 Наявність зовнішніх залежностей

Якщо для створення модулю ГА було використано більшою мірою вбудовані пакети та таким чином розроблено власні модулі по роботі із ГА, то для створення повноцінного вебзастосунку було використано велику кількість вже готових рішень.

У таблиці 5.1 наведено опис використаних зовнішніх залежностей при розробці клієнтського застосунку.

Таблиця 5.1 – Пакетні модулі клієнтського застосунку

Назва	Версія	Опис відповідальності
1	2	3
@arcgis/core	4.27.6	Геопросторовий аналіз, візуалізація даних на карті та керування геоданими.
@reduxjs/toolkit	1.9.7	Спрощення конфігурації та використання Redux у застосунках. Включає функції такі як налаштування store, створення reducer, асинхронну логіку.
axios	1.6.0	Виконання HTTP-запитів як у браузерях, так і з Node.js. Підтримує проміси, перехоплення запитів та відповідей, трансформацію даних і скасування запитів.
bootstrap	5.3.2	Розширений фреймворк CSS і JavaScript для швидкого створення візуально привабливих, адаптивних вебінтерфейсів. Містить готові до використання стилі та компоненти.
bootstrap-icons	1.11.1	Офіційний набір іконок від Bootstrap, який включає в себе сотні оптимізованих іконок для вебзастосунків. Легко інтегрується з Bootstrap або може використовуватися окремо.

Продовження таблиці 5.1

1	2	3
classnames	2.3.2	Допоміжна утиліта для умовного об'єднання імен класів CSS в JavaScript. Використовується для динамічного формування класів.
debounce	1.2.1	Функція для зменшення частоти виклику подій, зокрема при швидкому послідовному виклику. Часто використовується для оптимізації продуктивності, наприклад, при обробці подій скролу або клавіатури.
formik	2.4.5	Бібліотека, спеціалізована на створенні та управлінні формами в React. Включає в себе засоби для управління станом форм, валідації та обробки подій.
framer-motion	10.16.4	Бібліотека для реалізації анімації та жестів у React застосунках. Дозволяє легко створювати складні анімаційні ефекти, включаючи переходи між сторінками.
react-query	3.39.3	Бібліотека для ефективного управління асинхронними запитами, кешуванням даних та оновленням стану у React-застосунках. Підвищує продуктивність та спрощує управління даними.
react-redux	8.1.3	Офіційна бібліотека Redux для інтеграції з React. Дозволяє легко з'єднати компоненти React з Redux store для управління глобальним станом програми.
react-router-dom	6.18.0	Бібліотека для маршрутизації у односторінкових React-застосунках (SPA). Підтримує динамічну маршрутизацію, ліниве завантаження компонентів та захист роутів.

Продовження таблиці 5.1

1	2	3
react-tiny-popover	8.0.4	Легка та гнучка бібліотека для створення спливаючих вікон (popovers) у React-застосунках. Підтримує кастомізацію позиціонування та стилізації.
react-toastify	9.1.3	Бібліотека для створення сповіщень (toasts) в React. Пропонує багато опцій для кастомізації, включаючи позиціонування, анімацію та автоматичне закриття.
uuid	9.0.1	Бібліотека для генерації унікальних ідентифікаторів (UUID). Використовується для створення унікальних ключів, ідентифікаторів сесій, токенів тощо.
yup	1.3.2	Легка бібліотека для схемної валідації об'єктів, особливо корисна для валідації форм. Підтримує кастомізацію правил валідації та синхронну та асинхронну валідацію.

Деякі пакети, що використовуються на клієнті, також використовуються і на сервері. Виключно серверні зовнішні залежності із описом версій та функціональності наведено у таблиці 5.2.

Таблиця 5.2 – Пакетні модулі серверу

Назва	Версія	Опис відповідальності
1	2	3
bcrypt	5.1.0	Бібліотека для гешування та забезпечення безпеки паролів. Використовується для захисту паролів перед зберіганням у базі даних, використовуючи раунди гешування або сіль.

Продовження таблиці 5.2

1	2	3
cors	2.8.5	Проміжний обробник (middleware) для включення CORS у вебзастосунках. Дозволяє контролювати, які домени можуть взаємодіяти із сервером.
jsonwebtoken	9.0.1	Бібліотека для створення, перевірки та декодування JSON Web Tokens. Часто використовується для аутентифікації та обміну інформацією між сервером та клієнтами.
morgan	1.10.0	Проміжний обробник логування HTTP-запитів для Node.js. Дозволяє докладно відслідковувати запити до сервера, полегшуючи моніторинг.
nodemailer	6.9.4	Модуль для Node.js, який спрощує відправку електронних листів. Підтримує різноманітні протоколи та особливості, такі як HTML-шаблони, вкладені файли, SSL/STARTTLS тощо.
pg	8.11.3	Клієнт PostgreSQL для Node.js. Використовується для взаємодії з базами даних PostgreSQL, надаючи функції для виконання запитів, підключення до бази даних та роботи з результатами.
sequelize	6.32.1	ORM бібліотека для Node.js. Підтримує різні діалекти SQL, такі як MySQL, PostgreSQL. Спрощує роботу з базами даних через об'єктно-орієнтований інтерфейс.
sequelize-typescript	2.1.5	Розширення для Sequelize, яке дозволяє використовувати TypeScript-декоратори для визначення моделей. Поліпшує інтеграцію.

5.2.3 Мінімальні апаратні вимоги

Важливо врахувати використання перелічених вище різноманітних пакетів для роботи з вебтехнологіями, базами даних та серверним середовищем. Мінімальні апаратні вимоги до програмного забезпечення:

- процесор Intel Core i3 або еквівалент AMD (мінімум 2 ядра, 2.5 ГГц). Такого оснащення достатньо адже проєкт використовує різноманітні пакети, такі як React.js, Node.js та Sequelize, які вимагають достатньої обробки для рендерингу вебсторінок та обробки серверних запитів;

- вільне місце на диску мінімум 10 Гб – необхідно для зберігання коду проєкту, бази даних та лог-файлів. Залежності Node.js теж вимагають доволі багато вільного простору;

- оперативна пам'ять мінімум 4 Гб – з урахуванням вимог Node.js, бази даних та фронтенд-технологій, таких як анімації та інтеграція мапи із частим рендерингом, потрібно достатньо пам'яті для ефективного виконання застосунків та серверних процесів;

- операційна система Windows 10/11 або Linux – програмне забезпечення повинно бути сумісним з найпоширенішими ОС, оскільки використані технології (Node.js, React, PostgreSQL) є кросплатформними;

- мережеве з'єднання: стабільне інтернет-з'єднання – для доступу до зовнішніх API, взаємодії з вебсервером та базами даних, а також для використання вебзастосунків, інтернет-з'єднання є обов'язковим;

- підтримка SSL сервером для забезпечення безпечного з'єднання. Використання SSL/TLS є важливим для захисту даних, які передаються між клієнтом та сервером, особливо при обробці конфіденційної інформації. Це вимагає від сервера наявності відповідних сертифікатів SSL та конфігурації для підтримки шифрованих з'єднань.

Ці вимоги засновані на типових характеристиках для проєктів, що використовують вказані технології. Вони можуть бути кориговані в залежності від конкретних вимог та масштабу фінального проєкту.

5.3 Виконання програми

Спершу необхідно запустити сервер, в даному випадку використано саме локальний сервер. Для цього використовується команда `pnpm`, до неї додається фільтр, що позначає, який саме проєкт запускається, та безпосередньо виконується команда `run start`. Якщо все пройшло успішно то оператор бачить на екрані послідовні повідомлення про успішне з'єднання з БД, успішне розгортання серверу на зазначеному 3000 порті та під'єднання менеджера по роботі із поштовим клієнтом. У разі помилки запуск серверу обривається та оператор бачить опис виключної ситуації. Описані дії зображено на рисунку 5.2.

```
Sergey@SLW10 MINGW64 ~/projects/social-infrastructure-api (master)
$ pnpm api run dev

> sadok-monoc@1.0.0 api C:\Users\Sergey\projects\social-infrastructure-api
> pnpm --filter server "run" "dev"

> server@1.0.0 dev C:\Users\Sergey\projects\social-infrastructure-api\apps\server
> nodemon ./src/index.ts

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node ./src/index.ts`
Connection to db has been established successfully.
Server is established successfully on PORT=3000
Server is ready to work with email
```

Рисунок 5.2 – Запуск серверу

Після цього клієнт може проводити звертання до ресурсів серверу. На початку роботи користувачу потрібно авторизуватися у системі, щоб отримати доступ до API за токеном доступу.

Якщо користувач не має акаунту, йому необхідно попередньо зареєструватися на відповідній сторінці, або перейшовши за посиланням зі сторінки логіну. Далі потрібно вказати будь яке ім'я користувача, унікальну,

реальну електронну пошту та пароль, що складається не менше ніж з 5 символів. У разі некоректності параметрів користувач одразу, як тільки прибере фокус, побачить помилку під конкретним текстовим полем із поясненням, чому формат некоректний. Після того як користувач натисне реєстрація він теж може побачити помилки – або неунікальну пошту, що позначає що такий акаунт вже існує, або інші помилки валідації. Нарешті якщо всі параметри вірні користувач побачить повідомлення для підтвердження дій та активації свого акаунту.

Перейшовши до пошти та передивившись листи необхідно обрати лист з темою «Активация акаунт» та перейти за посиланням. Невдовзі піде підтвердження за тим активаційним токеном, який закладений у рядок посилання. В залежності від швидкості інтернету, користувача майже миттєво буде переведено на головну сторінку, завершивши активацію.

Процес логіну у систему повторює реєстрацію з точки зору користувацького досвіду, при цьому користувач побачить повідомлення про некоректний пароль, у разі якщо пошта вірна, а пароль до неї – ні. На рисунку 5.3 зображено сценарій авторизації у системі.

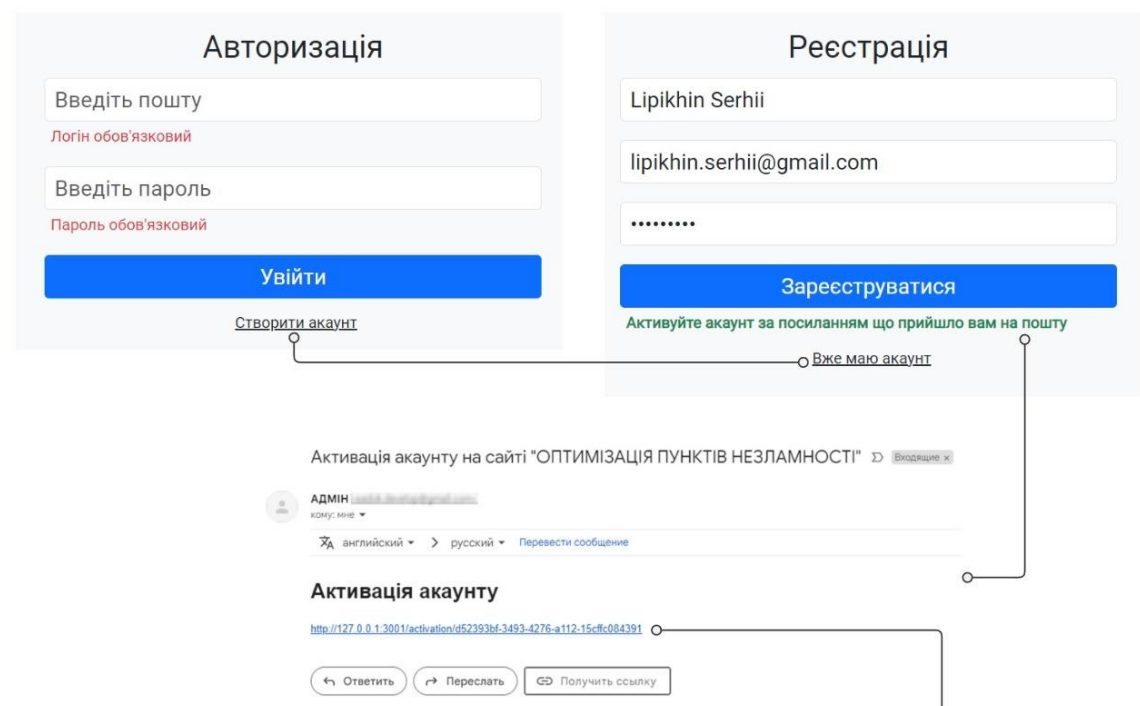


Рисунок 5.3 – Вікна авторизації

Як тільки користувач потрапляє на головну сторінку, він ліворуч бачить інтерфейс для проведення та контролю експериментів, а праворуч – мапу для управління об'єктами. Щодо взаємодії з картою реалізований наступний функціонал:

- пошук адреси або місця. У верхньому лівому куті є текстове поле, куди користувач може ввести адресу або назву місця, яке необхідно знайти на карті;

- створення об'єктів. У правій частині екрану є панель «Editor» з двома вкладками: «Create features» та «Edit features». Під вкладкою «Edit features» користувач може вибрати інструмент «Select» для вибору та редагування існуючих об'єктів на карті. В розділі «Create features» можна створювати нові об'єкти, наприклад, «Область проживання», будуючи багатокутник, або «Пункт незламності», додаючи точку на карту;

- налаштування для створення об'єктів. Під час створення об'єктів є форма, де потрібно вказати параметри для нового об'єкта інфраструктури: радіус обслуговування, тип, регіон, місто та вулиця. Після введення даних необхідно натиснути кнопку «Create» для створення об'єкта;

- взаємодія з мапою. Для масштабування призначені кнопки «+» та «-» у лівому нижньому куті, при зміні положення курсора у нижньому лівому куті можна побачити поточні координати у широті та довготі, а при натисненні на конкретний об'єкт у тому ж місці з'явиться таблиця з атрибутами;

- при натисканні на об'єкт, без вибраного параметру «Select» буде побудовано коло, що позначає радіус обслуговування, а усі клієнтські області буде відфільтровано, залишаючи лише ті що перетинають площину;

- праворуч знаходиться також меню, у якому можна відфільтрувати або лише звичайні пункти незламності, або пункти незламності бізнесу. У разі необхідності відмінити фільтрації достатньо натиснути у довільну область.

На рисунку 5.4. зображено вигляд картографічного блоку, функціональність та використання якого було описано вище.

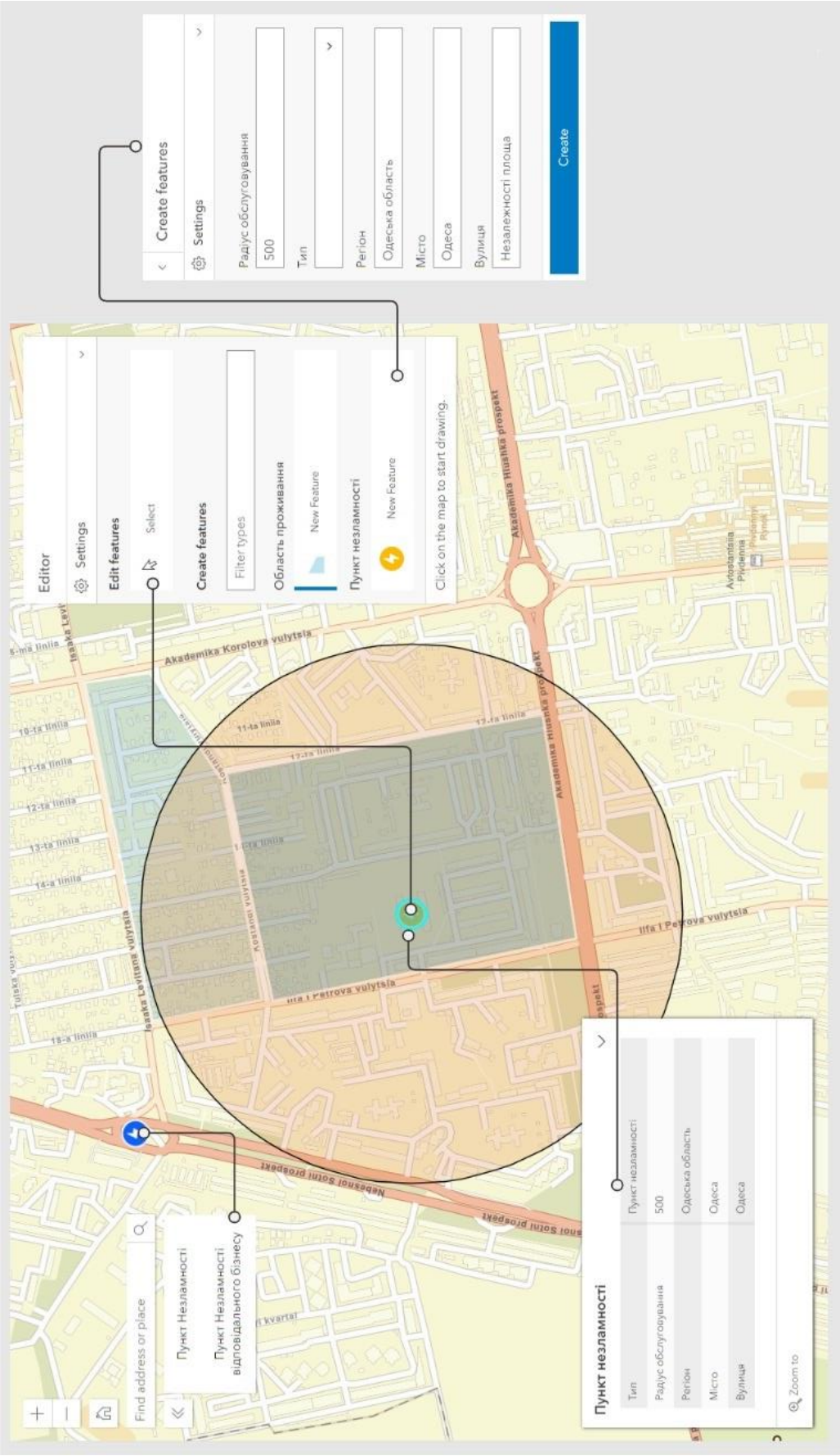


Рисунок 5.4 – Взаємодія користувача із картою

Як тільки користувач встановив необхідну конфігурацію об'єктів на карті для оптимізації, для проведення самого експерименту призначено панель праворуч.

Тут можна встановити кількість ітерацій, які будуть максимальною планкою при зупинені еволюційного пошуку. Також вказана кількість обов'язкових і необов'язкових пунктів.

При натисканні на кнопку «Провести оптимізацію» буде запущено алгоритм, користувач побачить завантаження, а сама кнопка буде заблокована для виключення можливості повторного натиснення. У разі, якщо при проведенні оптимізації на сервері виникнуть якісь проблеми, користувач побачить відповідне повідомлення у спливаючому вікні. У разі успіху нових експеримент, а точніше його результати, буде додано до таблиці.

Користувач має можливість вивантажити результати експерименту у вигляді файлу формату JSON. Також можна переглянути додаткову інформацію про кожний пункт: тип пункту, його адреса, яка складається з регіону, міста та вулиці, рішення, яке прийнято по цьому об'єкту та позначено відповідним кольором. Більш того, якщо експеримент виявився некоректним, неінформативним, зайвим, то його можна видалили натиснувши відповідну кнопку з хрестом.

Також варто зазначити, що перелічена функціональність доступна лише авторизованим користувачам, адже доступ до серверних ресурсів обмежений. У разі виходу користувача із системи, натиснувши відповідну кнопку, він не зможе а ні потрапити на сторінку, а ні здійснити запит до сервера, адже не буде мати токена доступу.

На рисунку 5.5 зображено обидві панелі, що визначають головну функціональність програми та з якими взаємодіє користувач для досягнення цілі застосування програми.

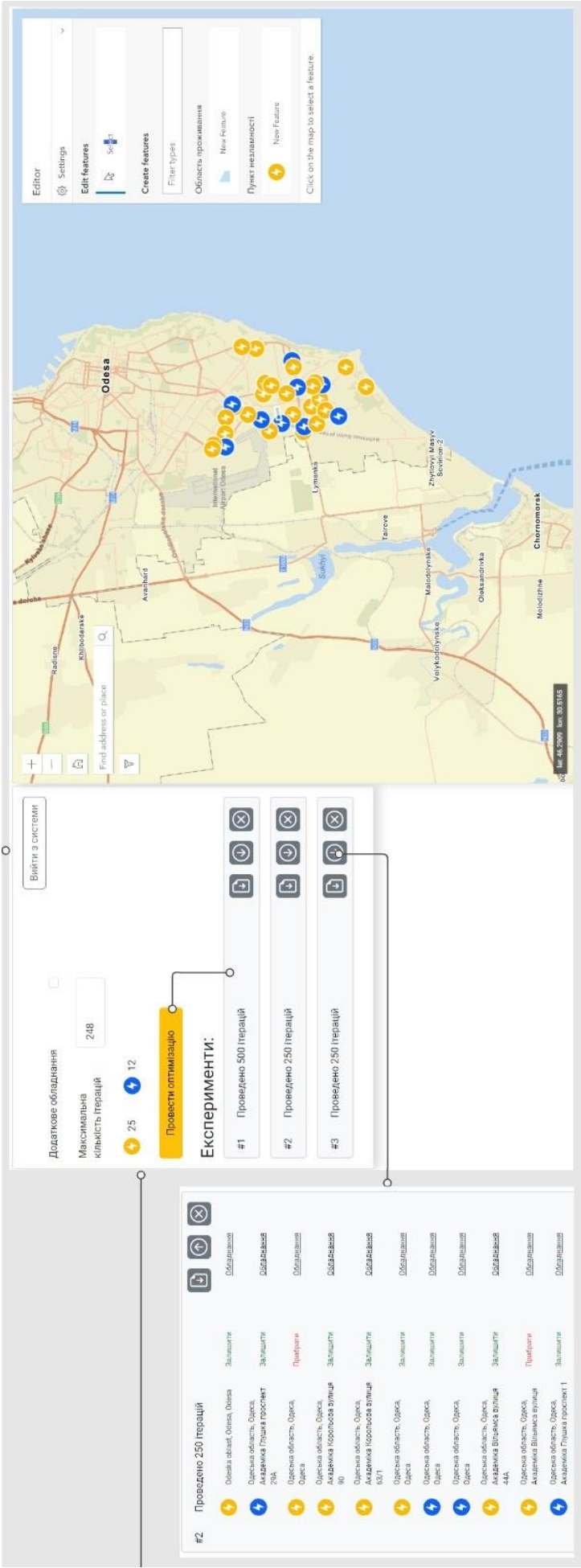


Рисунок 5.5— Проведення експерименту

5.4 Тестування програми

5.4.1 Тестування функціональності програми

Тестування програмного забезпечення проведено щодо відповідності програми її технічним вимогам та забезпечення її надійності та безпеки. Основні види тестування, які були використані в рамках цього дипломного проєкту, включають ручне тестування та модульне тестування.

Для проведення тестування коду в дипломній роботі були використані наступні інструменти:

- jest – фреймворк для модульного тестування в JavaScript та TypeScript, який забезпечує зручний API для написання тестових сценаріїв;
- supertest – цей модуль використовується для тестування HTTP-інтерфейсів. Він дозволяє легко створювати запити до Express-застосунку та перевіряти відповіді;
- jest-mock – використовуються для створення імітаційних об'єктів (моків), які можуть замінювати реальні залежності в тестах.

Функціональність, що відповідає за проведення експерименту та взаємодію із мапою, було протестовано в ручному режимі, а от функціональність з авторизації користувача у системі, яка не вимагає інтеграції з іншими фреймворками та реалізована повністю без готових рішень, покрито модульними тестами. Для імітації відповідних сценаріїв логіну було написано окремі тести для перевірки наступного функціоналу:

- валідація введених даних – коректна обробка валідних та невалідних даних, тестування реакції на відсутність електронної пошти або пароля;
- перевірка існування користувача – тестування поведінки системи, коли користувача з вказаною електронною поштою не існує в базі даних;
- перевірка статусу активації акаунта – переконатися, що функція логіну враховує, чи був активований акаунт користувача;
- перевірка відповідності пароля – тестування того, як функція обробляє випадки, коли введений пароль не збігається з паролем в базі даних;

– перевірка генерації токена – переконатися, що при успішному логіні користувач отримує правильний токен доступу.

Список перевірки тестування, згідно функціональних вимог до програми наведено у таблиці 5.3.

Таблиця 5.3 – Список перевірки функціонального тестування

Функція	Результат
Авторизація зареєстрованого користувача з коректними параметрами	Пройдено
Авторизація користувача з некоректними параметрами	Пройдено
Реєстрація нового користувача з коректними параметрами	Пройдено
Активація акаунту за посиланням на пошті	Пройдено
Оновлення токена доступу до серверних ресурсів	Пройдено
Створення нового об'єкта інфраструктури	Пройдено
Створення нової області проживання	Пройдено
Редагування існуючого об'єкта на карті	Пройдено
Видалення об'єкта на карті	Пройдено
Фільтрація пунктів незламності за типом	Пройдено
Фільтрація усіх областей проживання за радіусом обслуговування певного закладу	Пройдено
Пошук географічного місця за адресою	Пройдено
Нормалізація даних для проведення експерименту	Пройдено
Перевірка покриття усіх клієнтів хоча б одним сервісом	Пройдено
Проведення експерименту за створеними об'єктами	Пройдено
Видалення запису про експеримент із історії	Пройдено
Вивантаження результатів експерименту	Пройдено
Відображення детальної інформації про експеримент	Пройдено
Оновлення списку експериментів	Пройдено

5.4.2 Тестування моделі на реальному прикладі

Розроблений генетичний алгоритм інтегровано у вигляді серверного процесу, що дозволяє проводити оптимізацію реально існуючих на мапі об'єктів. Для перевірки життєздатності методу використано дані про розміщення пунктів незламності для Київського району міста Одеса.

Тестування розробленого генетичного алгоритму для пошуку оптимального розміщення об'єктів проведено використовуючи підібрані у підрозділі 2.3 еволюційні параметри.

Перед проведенням оптимізації було нанесено на мапу реально існуючі пункти незламності, адреси яких узято із офіційного сайту. Щодо областей проживання, то вони нанесені на мапу у вигляді багатокутників згідно з вулицями міста.

У таблиці 5.4 наведено адреси пунктів незламності, радіус їх обслуговування та рішення щодо їх залишення чи скасування.

Таблиця 5.4 – Результат проведення оптимізації пунктів незламності

№	Адреса	Радіус	Результат
1	2	3	4
1	Академіка Глушка проспект 2	500	Залишити
2	Академіка Глушка проспект 29А	500	Залишити
3	Академіка Корольова вулиця 90	500	Залишити
4	Академіка Корольова вулиця 63/1	450	Прибрати
5	Вулиця Академіка Вільямса, 79	500	Прибрати
6	A3C Sun Oil	500	Залишити
7	Вулиця Академіка Вільямса, 46	500	Залишити
8	Академіка Вільямса вулиця 44А	500	Залишити
9	Академіка Вільямса вулиця	500	Прибрати
10	Академіка Глушка проспект 1	400	Залишити

Продовження таблиці 5.4

1	2	3	4
11	Люстдорфська дорога 152	500	Залишити
12	Вулиця Ільфа і Петрова, 15А	500	Залишити
13	Проспект небесної сотні 30Б	500	Залишити
14	Небесної Сотні проспект 8	500	Залишити
15	Школа №82	500	Прибрати
16	Проспект Небесної Сотні, 4Б	500	Залишити
17	Люстдорфська дорога	500	Залишити
18	Ліцей «Гармонія»	500	Залишити
19	Івана Шишкіна вулиця 60А	500	Прибрати
20	Львівська вулиця 15	600	Залишити
21	Антоніна Макаренка вулиця 2А	500	Залишити
22	Вулиця Інглезі ТЦ «ТАВРІЯ-В»	500	Залишити
23	Космонавта Комарова вулиця 8	400	Прибрати
24	Космонавта Комарова вулиця 6	500	Залишити
25	Дитячий садок 190	500	Залишити
26	Космонавтів вулиця 70	500	Залишити
27	Інглезі вулиця 13Б	500	Залишити
28	Інглезі вулиця 13Б	500	Залишити
29	Іцхака Рабіна вулиця 45	500	Залишити
30	Іцхака Рабіна вулиця 37А	500	Прибрати
31	Героїв Крут вулиця	500	Прибрати
32	Іцхака Рабіна вулиця 14А	500	Залишити
33	Іцхака Рабіна вулиця 7А	500	Прибрати
34	Івана Та Юрія Лип вулиця 1	500	Залишити
35	Варненська вулиця 4А/1	500	Залишити
36	Варненська вулиця 2/2	500	Прибрати
37	Академіка Філатова вулиця 1	500	Залишити
38	Ромашкова вулиця 55	500	Залишити

На рисунку 5.6 зображено конфігурацію системи пунктів незламності до та після оптимізації.



Рисунок 5.6 – Результати оптимізації пунктів незламності

Алгоритм враховує, що клієнтська область входить у зону сервісу у тому випадку, якщо 10 % її площі покрито площею обслуговування. В результаті оптимізації було зменшено кількість пунктів незламності на 32 %.

Важливо зазначити, що алгоритм залишає усі пункти незламності відповідального бізнесу, а покриття залежить від того самого принципу перетину. У подальшому можна ввести умову про те, що клієнт буде вважатися покритим якщо не менше 2 сервісів одночасно його обслуговують. Також важливою нішею подальшої модифікації може бути врахування кількості населення на одиницю клієнтської області. Порівнюючи результати оптимізації на реальних даних із абстрактними, можна стверджувати, що час на оптимізацію є прийнятним (13,2 с), а тому користувачу не потрібно довго

чекати результатів. Тим не менш, час буде збільшуватись зі збільшенням кількості сервісних об'єктів, наприклад у разі оптимізації не окремого району міста, а усього міста в цілому.

5.5 Висновки за розділом 5

В даному розділі було описано можливі цілі використання програми державними та приватними установами, наведено структуру проєкту, із переліченням визначних файлів, описано важливі залежності, без яких застосунок не буде функціонувати коректним чином. Також було наведено сценарії використання програми та проведено функціональне тестування застосунку. Для доведення життєздатності розробленого методу оптимізації було зменшено кількість пунктів незламності у Київському районі міста Одеса, з умовою, що кожен мешканець зможе дістатися хоча б одного такого пункту у певному фіксованому радіусі.

ВИСНОВОК

З метою створення програми для проведення оптимізації системи об'єктів соціальної інфраструктури було досліджено задачу покриття множини, вирішення якої у контексті розтушування громадських закладів, дає оптимальну конфігурацію таких об'єктів. Виходячи із порівняння методів вирішення задачі покриття множини, через високу гнучкість та ефективність, було обрано генетичний алгоритм.

Наукова новизна роботи полягає в запропонованому методі пошуку оптимального розташування об'єктів соціальної інфраструктури з використанням генетичного алгоритму, що відрізняється від існуючих рішень модифікованим оператором відновлювання. Такий оператор використовує концепцію «конфліктних сервісних центрів», що передуює появі недопустимих рішень, – за той же самий час виконання та кількість ітерацій модифікований оператор дозволяє отримувати більш наближені до оптимуму рішення.

Інтеграція у розроблену програму сучасної геоінформаційної системи дозволяє користувачу створювати та розташовувати об'єкти на мапі, задаючи також їх атрибути. У поєднанні із високою швидкістю пошуку оптимального рішення, застосунок надає користувачу змогу зручно та швидко проводити оптимізацію будь-яких соціальних об'єктів.

Використовуючи стандартні, абстрактні дані було проведено налаштування еволюційних параметрів моделі, а використовуючи реальні дані про розташування пунктів незламності у Київському районі Одеси проведено оптимізацію на практиці. В результаті вдалось зменшити кількість об'єктів на 32 %, залишаючи при цьому усі пункти незламності відповідального бізнесу та задовольняючи умову, що кожен клієнт повинен мати доступ принаймні до одного сервісного центру.

Подальший супровід застосунку вимагає розгляду більш ретельного принципу формування системи вже не соціальних об'єктів, а клієнтських областей, звертаючи увагу на щільність населення у кожній такій області.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кириченко, С.О. Економіка. Підходи до визначення сутності поняття «соціальна інфраструктура» та її функціональне призначення [Текст] / С.О. Кириченко // Демографія, економіка праці, соціальна економіка і політика, 2016. – № 1 (55). – С. 56–60.
2. Швайка, Л.А. Завдання державного регулювання розвитку галузей соціальної інфраструктури [Текст] / Л.А. Швайка // Державне регулювання економіки. – Київ : Знання, 2006. – 435 с.
3. Державна служба статистики України [Електрон. ресурс]. – Режим доступу : <https://www.ukrstat.gov.ua>.
4. Регіональне управління : підручник [Текст] / за заг. ред. Ю. В. Ковбасюка, В. М. Вакуленка, М. К. Орлатого. – К. : НАДУ, 2014. – 740 с.
5. Кочемировська, О. О. Основні напрями оптимізації системи соціального захисту в Україні : аналіт. доп. [Текст] / О. О. Кочемировська, О. М. Пищуліна. – К. : НІСД, 2012. – 88 с.
6. По всій країні розгортаємо новий проєкт підтримки людей – «Пункти незламності» – звернення Президента України [Електрон. ресурс]. – Режим доступу : <https://www.president.gov.ua/news/po-vsij-krayini-rozgortayemo-novij-proekt-pidtrimki-lyudej-p-79361>.
7. Пункт незламності [Електрон. ресурс]. – Режим доступу : <https://nezlamnist.gov.ua>.
8. Стало відомо, скільки Пунктів незламності вже розгорнули в Україні [Електрон. ресурс]. – Режим доступу : <https://www.slovoidilo.ua/2023/10/17/novyna/suspilstvo/stalo-vidomo-skilky-punktiv-nezlamnosti-vzhe-rozhornuly-ukrayini>.
9. На Одещині всі «пункти незламності» готові майже на 97% – ОВА [Електрон. ресурс]. – Режим доступу : <https://suspilne.media/596817-na-odesini-vsi-punkti-nezlamnosti-gotovi-majze-na-97-odeska-ova>.

10. На Полтавщині вже працює 141 пункт незламності, а готові до роботи ще 333 [Електрон. ресурс] : що в них має бути. – Режим доступу : <https://kg.ua/news/na-poltavshchini-vzhe-pracyuie-141-punkt-nezlamnosti-gotovi-do-roboti-shche-333-shcho-v-nih>.

11. «Пункти незламності» [Електрон. ресурс] : зиму переживемо – світло, зв'язок, тепло і вода будуть. – Режим доступу : <https://www.ukrinform.ua/rubric-ato/3621302-punkti-nezlamnosti-zimu-perezivemo-svitlo-zvazok-teplo-i-voda-budut.html>.

12. Як змінювалася динаміка та інтенсивність масованих ракетних атак Росії [Електрон. ресурс]. – Режим доступу : https://media.slovoidilo.ua/media/infographics/19/189083/189083-1_uk_origin.png.

13. Multi-period hub set covering problems with flexible radius: A modified genetic solution [Text] / [A. Ebrahimi-zade, H. Hosseini-Nasab, Y. Zare-mehrjerdi, A. Zahmatkesh] // Applied Mathematical Modelling. – 2016. – Vol. 40, № 4. – P. 2968-2982.

14. Lan, G. An effective and simple heuristic for the set covering problem [Text] / G. Lan, G.W. DePuy, G.E. Whitehouse // European Journal of Operational Research. – 2007. – Vol. 176. № 3. – P. 1387–1403.

15. Grossman, T. Computational experience with approximation algorithms for the set covering problem [Text] / T. Grossman, A. Wool // European Journal of Operational Research – 1997. – Vol. 101, № 1. – P. 81-92.

16. Analyzing the Effects of Binarization Techniques when Solving the Set Covering Problem Through Swarm Optimization [Text] / [J. M. Lanza-Gutierrez, B. Crawfordb, R. Sotob, N. Berriosb, J. A. Gomez-Pulidoa, F. Paredes] // Expert Systems with Applications. – 2017. – Vol. 70, № 15. – P. 1–19.

17. Mount, D. Greedy Approximation [Electronic resource] : Set Cover / D. Mount. – Washington : 2017. – CMSC 451 : Lecture 9 – Access mode : <https://www.cs.umd.edu/class/fall2017/cmsc451-0101/Lects/lect09-set-cover.pdf>.

18. Chandu, D. P. Improved Greedy Algorithm for Set Covering Problem [Text] / D. P. Chandu // SSRG International Journal of Computer Science and Engineering. – 2015. – P. 1–4.
19. Gupta, A. A Local Search-Based Approach for Set Covering [Text] / A. Gupta, E. Lee, J. Li // Symposium on Simplicity in Algorithms. – 2022. – P. 1–11.
20. Caprara, A. Algorithms for the Set Covering Problem [Text] / A. Caprara, P. Toth, M. Fischetti // Annals of Operations Research. – 2000. – Vol. 98. – P. 353–371.
21. Yaghini, M. A Local Branching Approach for the Set Covering Problem [Text] / M. Yaghini, M.R. Sarmadi, M. Momeni // International Journal of Industrial Engineering and Production Research. – 2014. – Vol. 25. № 2. – P. 95–102.
22. Gandomi, A. H. Metaheuristic Algorithms in Modeling and Optimization [Text] / A. H. Gandomi, X.-S. Yang, S. Talatahari, A. H. Alavi // Metaheuristic Applications in Structures and Infrastructures. – 2013. – P. 1–24.
23. Metaheuristics for the Minimum Set Cover Problem [Text] : A Comparison / L. Rosenbauer, A. Stein, H. Stegherr, J. Hahner // 12th International Conference on Evolutionary Computation Theory and Applications. – 2020. – P. 123–130.
24. OR-Library, Set Covering [Electronic resource]. – Access mode: <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html>.
25. Macready, W. G. No Free Lunch Theorems for Optimization. [Text] / W. G. Macready, D. H. Wolpert // IEEE Transactions on Evolutionary Computation. – 1997. – Vol. 1. № 1. – P. 67–82.
26. Beasley, J. R. OR-Library: Distributing Test Problems by Electronic Mail [Text] / J. R. Beasley // Journal of the Operational Research Society. – 1990. – P. 1068–1072.
27. Субботін, С. О. Ітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей [Текст] : Монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник ; під заг. ред. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2009. – 375 с.

28. The Minimization of Public Facilities with Enhanced Genetic Algorithms Using War Elimination [Text] / [Seda P., Mark M., Su Kuan-Wu et. al.] // IEEE Access. – 2019. – Vol. 7 – P. 9395 – 9405.

29. Beasley, J. R. A Genetic Algorithm for the Set Covering Problem [Text] / J. R. Beasley, P. C. Chu // European Journal of Operational Research. – 1996. – Vol. 94, Issue 2. – P. 392–404.

30. Чому важлива типізація в JS [Електрон. ресурс]. – Режим доступу : <https://foxminded.ua/typizaciya-js>.

31. Типізація Python: огляд статичної, динамічної та строгої [Електрон. ресурс]. – Режим доступу : <https://foxminded.ua/typizatsiia-python>.

32. JavaScript data types and data structures [Electronic resource]. – Access mode : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures.

33. Структури даних Python [Електрон. ресурс]. – Режим доступу : <https://docs.python.org/uk/3/tutorial/datastructures.html>.

34. SSR vs CSR Explained [Electronic resource]. – Access mode : <https://www.freecodecamp.org/news/server-side-rendering-javascript>.

35. Comprehensive Comparison Interactive Web Maps [Electronic resource]. – Access mode : <https://norcomdentaire.medium.com/a-comprehensive-comparison-leaflet-openlayers-mapbox-gl-js-google-maps-javascript-api-esri-430c98b657fe>.

36. Python Global Interpreter Lock (GIL) [Electronic resource] : Understanding, Workarounds, and Parallelism. – Access mode : <https://dev.to/bshadmehr/python-global-interpreter-lock-gil-understanding-workarounds-and-parallelism-4dkn>.

37. D3 or Chart.js for Data Visualisation [Electronic resource]. – Access mode : <https://www.createwithdata.com/d3js-or-chartjs>.

38. Comparing Python Libraries for Visualization [Electronic resource]. – Access mode : <https://medium.com/mlearning-ai/comparing-python-libraries-for-visualization-b2eb6c862542>.

39. Why is JavaScript so fast? [Electronic resource]. – Access mode : <https://www.quora.com/Why-is-JavaScript-so-fast>.
40. Understanding Python's Performance [Electronic resource]. – Access mode : <https://bluebirdinternational.com/python-vs-javascript-performance>.
41. Python Developers Survey 2022 Results [Electronic resource]. – Access mode : <https://lp.jetbrains.com/python-developers-survey-2022>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

A.1.1 Назва програми

Найменування – «Дослідження та програмна реалізація методів оптимізації системи об'єктів соціальної інфраструктури». Найменування англійською – «Research and Software Implementation of Methods for Optimization of System of Social Infrastructure Objects».

A.1.2 Призначення та область застосування

Програма має використовуватися у соціальній сфері, будь-то державні проекти чи проекти незалежного бізнесу, підприємстві, галузі обслуговування, будівництві. Вирішальними чинниками, що визначають область застосування програми, є наявність сервісів (установ) обслуговування та клієнтів, які повинні отримати певні услуги, а також необхідність пошуку оптимальної конфігурації таких установ (оптимального розташування на мапі).

Розроблювана програма призначена для автоматизації процесу формування системи об'єктів соціальної інфраструктури. Під цим процесом розуміється пошук оптимального положення низки закладів на реальній мапі області, міста, району тощо для покращення обслуговування та зменшення реальних витрат на забезпечення таких установ. Також програма може використовуватися і загалом для позначення місцезнаходження установ. Головним об'єктом оптимізації є пункти незламності, однак програма здатна працювати з будь-якими організаціями, які мають певний радіус обслуговування.

Фактично програма розроблена з метою задовольнити потреби клієнтів та покращити якість обслуговування в різних галузях соціальної сфери та зменшити витрати сервісів на обслуговування.

А.2 Підстави для розробки

Підставою для проведення розробки програмного забезпечення є наказ № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка» із зазначенням теми роботи «Дослідження та програмна реалізація методів оптимізації системи об'єктів соціальної інфраструктури» та завдання на роботу, видане 4 вересня 2023 року.

Повне найменування теми розробки – «Дослідження та програмна реалізація методів оптимізації системи об'єктів соціальної інфраструктури». Умовне позначення теми розробки (шифр теми) – «ОптимСоцІнфраструктур», та англійською – «OptimizeSocialInfrastructure».

А.3 Призначення розробки

А.3.1 Функціональне призначення

Функціональним призначенням програми є забезпечення користувачеві можливостей:

- переглядати, фільтрувати та шукати на мапі місцезнаходження сервісних точок, що надають певні послуги у визначеній області, та місцезнаходження клієнтських точок, що ці послуги споживають;

- працювати з мапою – перемикатися на необхідну локацію, здійснювати пошук за адресою, скролити та зумити мапу, переглядати поточні координати у широті та довготі;

- створювати, редагувати та видаляти на карті як установи обслуговування, так і клієнтські області, задаючи при цьому значення відповідних їм параметрів;

- проводити визначення оптимального розташування установ, згідно з радіусом їх обслуговування чи обладнання, яке вони мають у розпорядженні; зберігати результати подібних експериментів, вивантажувати інформацію;

– оцінювати результати проведених обчислень, позначаючи наскільки добре, згідно із суб'єктивною оцінкою користувача, була дана відповідь про зміну конфігурації системи соціальної інфраструктури

Останні три пункти відповідають лише користувачам, які зареєстровані та верифіковані у системі.

А.3.2 Експлуатаційне призначення

Програма має експлуатуватися в громадських інтересах, зокрема для оптимізації системи пунктів незламності. Завдяки своїм можливостям, програма також може бути використана комунальними та приватними підприємствами для оптимізації структури своїх установ.

Цілями експлуатації програмного продукту користувачами можуть бути:

– збирання та зберігання інформації щодо місцезнаходження та характеристик об'єктів соціальної інфраструктури. Це дозволяє створити базу даних, яка містить детальну інформацію про об'єкти та послуги, доступні в конкретній області. Ця інформація може бути використана для подальшого аналізу та прийняття стратегічних рішень;

– моніторинг функціонування цих об'єктів та надання відповідної інформації користувачам. Це дозволяє оперативно дізнатися про розташування сервісу та його забезпеченість обладнанням;

– оптимізація розташування об'єктів відповідно до різних параметрів, таких як радіус обслуговування;

– вивчення та застосування генетичного алгоритму для вирішення задачі покриття множин;

– розширення функціоналу програмного пакету що представляє генетичний алгоритм та суміжний інструментарій у напрямку визначення функціональної залежності між параметрами що характеризують сервіс та

радіусом, кодування цих параметрів та вивчення та введення до моделі показників демографії клієнтів.

A.4 Вимоги до програми

A.4.1 Вимоги до функціональних характеристик

A.4.1.1 Вимоги до складу виконуваних функцій

Фактично функціональність програми може бути умовно поділена на три частини:

- модель – фактично визначається у межах вирішення задачі покриття множини за допомогою генетичного алгоритму, та інших методів, реалізації генетичних операторів, дослідження та тестування, обрахунку найкращих параметрів моделі;

- геоінформаційний модуль – забезпечує представлення інформації безпосередньо на мапі, та відповідає за роботу користувача з установами на карті;

- супровідна функціональність – це ті процедури, які забезпечують зв'язок моделі і клієнта та роблять із програмного продукту повноцінний, придатний до експлуатації та вирішення декларованих цілей вебсайт.

Цілком справедливо можна відзначити, що модель повинна бути реалізована на сервері, а геоінформаційний модуль на клієнті.

Програма повинна виконувати зазначені нижче функції, які пов'язані з розробкою моделі та безпосередньо реалізують модель системи:

- зчитувати набори даних із стандартної бібліотеки даних (OR-Library) для вирішення задачі покриття множини;

- зчитувати та зберігати в оперативній пам'яті дані про місцезнаходження клієнтських та сервісних об'єктів, а також пов'язану з ними інформацію, представлену у форматі GeoJSON;

- перетворювати подібні непідготовлені вихідні дані у готову для використання структуру даних (мається на увазі побудова матриці відстаней та матриці wag);

- вираховувати за показниками довготи та широти відстань між об'єктами та визначати, потрапляє чи ні клієнтська точка у радіус обслуговування;

- ініціалізувати хромосому випадковим чином, де певним чином закодовано значення її генів – вказана їх кількість, гомологічний тип та принцип кодування; забезпечити можливість до розширення можливостей кодувати хромосоми;

- ініціалізувати випадковим чином цілу популяцію, враховуючи не лише розмір популяції, але й необхідні дані для створення хромосом;

- реалізувати низку перевірок виняткових випадків, а саме: наявність недоступних або непотрібних сервісних центрів, наявність необхідних сервісних центрів, які обов'язково повинні бути у кінцевому рішенні;

- клонувати хромосому зі значенням її фенотипу (значення функції пристосованості) та геному;

- підраховувати значення функції пристосованості як для однієї особини, так і для всіх особин популяції;

- інвертувати знак значення фітнес-функції для пошуку як мінімального так і максимального значення функції (робити позитивне значення від'ємним та навпаки);

- забезпечити наявність інформації про одне окремо взяте покоління експерименту, а саме: номер покоління, значення максимальної пристосованості у популяції, середнє значення пристосованості та геном найбільш пристосованого індивіда;

- реалізовувати оператор відбору, а саме такі його види як: турнірний із трьома кандидатами, пороговий із можливістю задавання порогу та пропорційний із визначенням особин за допомогою рулетки;

- реалізовувати оператор схрещування, а саме такі його види як: одноточечне, рівномірне із задаванням імовірності мутації одного гену, order crossover, position-based crossover та order-based crossover;
- реалізовувати оператор мутації, а саме такі його види як: рівномірна мутація та нерівномірна мутація із задаванням коефіцієнту нерівномірності;
- реалізовувати оператор відновлення для забезпечення гарантії, що конкретний індивід задовольняє усім умовам задачі;
- реалізовувати оператор імітації війни для збільшення рівня диверсифікації популяції на етапі сталого розвитку середнього значення пристосованості;
- знаходити ті клієнтські точки, які вже є покритими сервісними центрами, та ті, які ще не є покритими, працювати з їх індексами у матриці відстаней;
- знаходити ті сервісні центри, які покривають чи не покривають відповідних клієнтів;
- проводити аналіз рівня різноманітності популяції на кожному поколінні;
- будувати графіки для відстеження залежності середнього та максимального значень пристосованості від ітерацій у процесі еволюційного пошуку;
- зберігати створені графіки;
- забезпечувати можливість взаємодії з графіками в спеціальному вікні: змінювати масштаб, повертати, виводити координати значень відповідно до розміщеного курсору користувача;
- записувати результати дослідження у файли, включаючи оптимальні параметри робочої моделі для вирішення задачі покриття множини. Рекомендується зберігати дані повністю, щоб уникнути проблем при повторних моделюваннях;
- налаштовувати параметри процесу еволюційного пошуку, такі як розмір популяції або імовірність мутації;

- проводити N серій експериментів, кожна з яких складається з M досліджень для встановлення оптимальної конфігурації генетичної моделі;
- записувати та зчитувати результати досліджень у відповідні текстові файли;
- візуалізувати результати досліджень у формі графіків, які відображають залежність одного параметру від іншого. Зазвичай залежний параметр представляє собою значення функції пристосованості;
- забезпечувати можливість обробки у декілька потоків на випадок реалізації паралельних моделей. Кожну окрему, автономну популяцію обгорнути в окремий потік і синхронізувати їх для можливої взаємодії між популяціями;
- заміряти та відображати час виконання операцій, алгоритмів та досліджень;
- забезпечувати виведення записів до лог-файлу, що показують динаміку та результати роботи програми у вигляді таблиці з кодуванням стовпчиків та рядків;
- знаходити оптимальне рішення проблеми покриття множини за допомогою ГА та реалізованих операторів на основі представлених початкових даних, або абстрактних, або просторових;
- забезпечити рекомендації щодо змінення конфігурації обладнання (у разі якщо радіус задано неявно) та змінення системи інфраструктурних об'єктів в цілому.

Геоінформаційний модуль при взаємодії з ним користувача повинен:

- забезпечувати користувача можливістю взаємодіяти з мапою, з маркуванням головних географічних об'єктів, зокрема міст, вулиць, будинків. Акцент має робитися саме на географії України;
- залишати на карті прості позначки з визначенням поточної їх адреси та координат у довготі та широті. Таки чином користувач повинен мати інформацію про будь-яку точки на карті, зокрема відносно свого курсора;

- шукати місцезнаходження, вводячи адресу необхідного місця у пошуковому рядку;
- змінювати масштаб;
- зумити до визначеного місця на мапі;
- наносити на мапу позначки сервісних центрів у вигляді відповідного маркера. Передбачити різні типи об'єктів, зокрема для пунктів незламності – пункти незламності, пункти незламності відповідального бізнесу та інші (наприклад АЗС), кожному типу відповідає така позначка. Створити легенду;
- зберігати додаткову інформацію про сервісні центри. Обов'язковим та мінімальним параметром є радіус обслуговування та відповідний тип установи. Також сервісний центр може містити і іншу супровідну, але не необхідну інформацію;
- наносити на мапу позначення клієнтських областей у вигляді багатокутника – у такому разі система визначає клієнтську точку як центр такого багатокутника;
- зберігати для сервісів та клієнтів реальну адресу, яка повинна автоматично підтягуватись із відповідного сервісу, як тільки користувач закінчив наносити географічну позначку;
- фільтрувати поточні сервісні центри за типом, тобто залишати на мапі лише ті, що вибрані користувачем;
- відображати на мапі області обслуговування навколо вибраного сервісного центру;
- фільтрувати клієнтські області, залишаючи на мапі лише ті, що потрапляють у радіус обслуговування сервісу;
- переглядати інформацію у вигляді таблиці, натискаючи на відповідний об'єкт;
- додавати, редагувати та видаляти сервісні та клієнтські об'єкти на мапі;
- змінювати положення вже доданих об'єктів.

Також програма повинна забезпечувати виконання наступних важливих функцій:

- виконувати обробку станів – помилок, успіхів, завантаження. Там, де можливо, створити спливаючі вікна з пояснювальною інформацією користувача по кожній дії;
- авторизувати користувачів у системі за електронною поштою та паролем;
- підтверджувати існування пошти, на яку реєструється акаунт користувача через відповідний лист, з лінком на сторінку підтвердження акаунту;
- захищати відповідні точки доступу до API за допомогою підписки токеном, використовувати access token, а також refresh token щоб користувачу не треба було турбуватися про постійну авторизацію у зв'язку з простроченням токена з плином часу;
- проводити визначення найоптимальнішої конфігурації системи соціальних установ, отримуючи відповідь у вигляді рекомендації про зменшення кількості таких установ або зміни їх оснащення;
- зберігати результати експериментів, із зазначенням стану об'єктів до та після, та зазначенням конфігурації еволюційних параметрів;
- проводити опитування користувачів згідно з тим, чи зміг оптимізаційний алгоритм знайти рішення, та чи наскільки воно виявилось ефективним чи життєздатним;
- вивантажувати дані про розміщення сервісів чи клієнтів як шарів ознак у форматах GeoJSON, csv, shapefile.
- реалізувати систему моніторингу на стороні клієнту, щоб відстежувати та аналізувати діяльність системи, виявляти помилки та вдосконалювати її ефективність.
- забезпечувати можливість відновлення даних та резервне копіювання інформації, щоб запобігти втраті даних в разі непередбачуваних ситуацій або аварій;

– забезпечити хешування паролів користувачів.

А.4.1.2 Вимоги до організації вхідних даних

Вхідними даними програми з точки зору геоінформаційного модуля є шар ознак (feature layer) з фреймворку ArcGIS. Основні компоненти такого шару наведено у таблиці А.1.

Таблиця А.1 – Основні компоненти шару

Назва	Опис компоненту
Геометрія об'єктів	Це частина, що визначає географічне положення об'єктів на карті. Геометрія може бути точками, лініями або полігонами, залежно від типу об'єкта.
Атрибути	Це дані, що відносяться до кожного геоінформаційного об'єкта на шарі. Атрибути представлені у вигляді полів і можуть включати інформацію, таку як назви, числові значення, дати, текстові описи та інші характеристики об'єктів.
Таблиця атрибутів	Таблиця, яка містить дані атрибутів для всіх об'єктів на шарі. Кожен рядок в таблиці відповідає одному геоінформаційному об'єкту, а кожний стовпець відповідає певному атрибуту. Таблиця атрибутів дозволяє здійснювати запити та аналізувати дані.
Система координат	Кожен шар ознак пов'язаний із певною системою координат, що визначає географічне положення об'єктів на площині. Це може бути географічна система координат (наприклад, WGS 84) або проекційна система координат.
Метадані	Інформація про шар, така як назва, опис, джерело даних та інші характеристики, які допомагають ідентифікувати та описувати шар

Специфіка програми потребує визначення структури атрибутивних полів. Основні поля для сервісного центру (фактично пункту незламності) наступні:

- радіус обслуговування;
- регіон;
- місто;
- вулиця;
- графік роботи;
- генератор;
- булер'ян;
- твердопаливний котел;
- старлінк;
- оптоволоконне підключення до мережі інтернет;
- підзарядка гаджетів.

Що стосується клієнтської області, яка визначається на точкою на мапі, а саме багатокутником (полігоном), то тут у атрибутах потрібно визначити лише адресу, адже головне це положення у просторі (центр полігону). При розширенні функціональності програми пропонується також визначати кількість населення.

A.4.1.3 Вимоги до організації проміжних даних

В результаті формування шару ознак користувач отримує файл формату GeoJSON, який далі передається на вхід до моделі, та потребує у подальшому декодування у матрицю відстаней. Формат такого файлу наведено у таблиці A.2.

Таблиця А.2 – Формат файлу типу GeoJSON

Ключ	Опис поля
Тип об'єкта (type)	Вказує тип GEOJSON-об'єкта. Для шару ознак це завжди буде «FeatureCollection» що означає колекцію геопросторових об'єктів.
Масив об'єктів (features)	Масив геометричних об'єктів та атрибутів. Кожен об'єкт у цьому масиві представляє окремий геопросторовий об'єкт.
Тип об'єкта (type)	У цьому контексті, це завжди «Feature», що позначає окремий геопросторовий об'єкт.
Геометрія (geometry)	Ця частина містить геометричні дані, такі як точки, лінії або полігони, які визначають місцезнаходження об'єкта на карті.
Властивості (properties)	Це об'єкт, який містить атрибути або властивості геопросторового об'єкта. Ці атрибути можуть бути будь-якими даними, які описують об'єкт, наприклад, ім'я, розмір, категорія тощо.
Система координат (crs)	Якщо потрібно, GEOJSON може включати інформацію про систему координат, яка використовується для геометрії у шарі.

А.4.1.4 Вимоги до організації вихідних даних

Також користувач може витягнути сформований шар ознак, як для сервісного центру так і для клієнтів у вигляді shapefile (файлу з розширенням .shp). Основні компоненти структури shapefile включають наступне:

- .shp файл: Це бінарний файл, який містить геометричну інформацію про векторні об'єкти. Він може містити точки, лінії або полігони, а також їхні координати;

– .shx файл: Це файл індексу, який містить інформацію про розташування геометричних об'єктів у .shp файлі. Це допомагає прискорити доступ до конкретних об'єктів у .shp файлі;

– .dbf файл: Це файл бази даних, який містить атрибутивну інформацію про геометричні об'єкти. Він може містити структуровані дані, такі як імена, числові значення, текстові описи та інші характеристики, які відносяться до кожного об'єкта;

– .prj файл (не завжди присутній): це текстовий файл, який містить інформацію про систему координат, в якій зберігаються геометричні об'єкти. Це важливо для правильного відображення та аналізу геоданих;

Shapefile може також містити інші файлові компоненти, такі як .prj, .sbn, .sbx, .shp.xml, які допомагають покращити ефективність та аналіз геоданих в різних ГІС-системах.

Що стосується моніторингу та опрацювання результатів роботи ГА, то передбачено візуалізувати дані у вигляді графіка, який зображено на рисунку А.1.

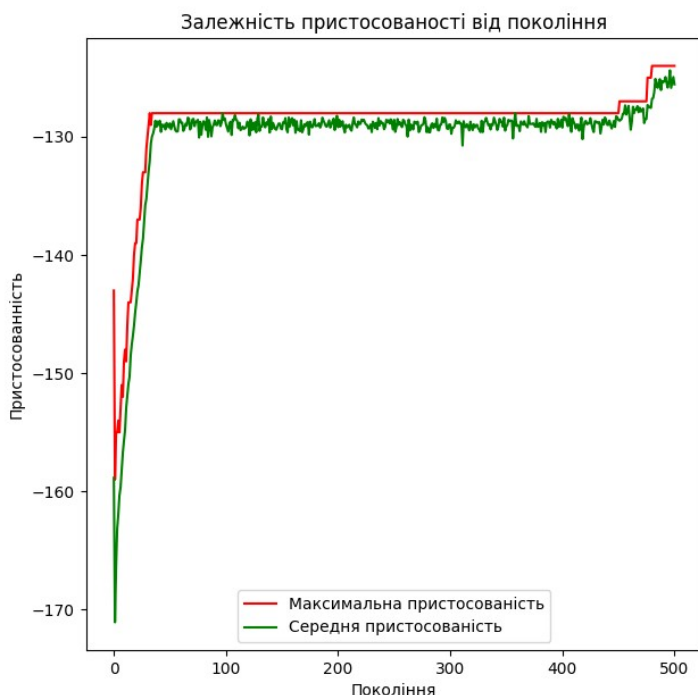


Рисунок А.1 – Результат роботи ГА у вигляді динаміки зміни середнього та максимального показника пристосованості

А.4.2 Вимоги до надійності

А.4.2.1 Вимоги до забезпечення надійного функціонування програми

Для забезпечення надійного функціонування програми важливо враховувати ряд наступних аспектів:

- безперервне живлення – забезпечення безперервного постачання живлення для технічних пристроїв, на яких встановлена програма, є критичним. Це допомагає уникнути негативних наслідків через вимкнення живлення та можливих втрат даних;
- ліцензоване програмне забезпечення обов'язкове для забезпечення законності та забезпечення підтримки та оновлень;
- виконання вимог щодо захисту інформації, передбачених ДСТУ 3396.0-96 з питань захисту інформації для забезпечення безпеки та конфіденційності даних;
- антивірусний захист – проведення регулярних перевірок на предмет вірусів та шкідливого програмного забезпечення є важливим елементом забезпечення надійності програми та запобігання можливим загрозам для безпеки;
- управління станом програми – у разі некоректного закриття програми або випадкового збою система не повинна автоматично відновлювати попередній стан при наступному запуску.

А.4.2.2 Збійні ситуації

Наступні збійні ситуації мають оброблятися програмою:

- відсутнє інтернет з'єднання;
- помилка при з'єднанні з сервісами ArcGIS;
- помилка при створенні, редагуванні чи видаленні нової властивості;
- некоректна адреса для точки на мапі;

- некоректний формат електронної пошти;
- некоректний формат паролю;
- такий користувач вже зареєстрований у системі;
- неонована помилка клієнту, серверу чи стороннього сервісу;
- помилка форматування;
- неіснуючий ресурс;
- перевищення обсягу пам'яті сервера;
- зайнятість серверного потоку;
- помилка у внутрішньому ГА;
- відсутній зв'язок з сервісом;
- відсутній зв'язок з базою даних;
- занесення до БД даних, що не відповідають обмеженням;
- занесення неунікальних даних до БД;
- невказаний обов'язковий параметр;
- некоректні дані для побудови графіку;
- помилка в мережевій комунікації;
- втрата з'єднання з користувачем;
- помилка при збереженні результатів експерименту;
- помилка при проведенні опитування;
- перевищення ліміту за часом для оптимізації;
- несподівані зміни у середовищі виконання;
- неможливо виконати запит на зазначений ресурс;
- некоректно вказана дата та некоректне її перетворення;
- помилка у компоненті користувацького інтерфейсу;
- звернення до відсутнього поля об'єкту;
- некоректне перетворення файлу GeoJSON у шар ознак;
- занадто великий розмір оброблюваного GeoJSON файлу;
- некоректне розширення файлу;
- помилка CORS.

A.4.3 Вимоги до технічних та програмних засобів

Програма розробляється як вебсайт з клієнтською та серверною частинами. Вимоги до технічних засобів на цей момент виглядають наступним чином. Клієнтська частина:

- сучасний веббраузер, сумісний зі стандартами веброзробки;
- роздільна здатність екрану повинна бути не менше 1280x720 пікселів для покращення користувацького досвіду.

Серверна частина:

- центральний процесор з частотою не менше 1 ГГц або швидший;
- оперативна пам'ять: 2 ГБ для оптимальної роботи;
- доступний диск з мінімум 32 ГБ для зберігання даних та файлів серверної частини;
- графічний процесор, не нижче версії 9 DirectX, для обробки графічних даних та візуалізації;
- підтримка технології роботи з віддаленими запитами (HTTP/HTTPS) для взаємодії з клієнтською частиною;
- операційна система, яка підтримує відповідне середовище виконання для серверної частини вебсайту (Node.js та Python).

Ці вимоги дозволять забезпечити оптимальну роботу вебсайту як на стороні клієнта, так і на стороні сервера.

A.5 Вимоги до програмної документації

Вимоги до програмної документації мають включати наступні компоненти:

- технічне завдання – опис теоретичних відомостей, пов'язаних з предметною областю проєкту, мета і завдання проєкту, вимоги до функціональності програмного продукту;

– пояснювальна записка – опис проблемної області, опис проведеного дослідження, включаючи опис типу вирішуваної задачі, опис конструювання та налагодження генетичного алгоритму; опис проєктування системи, включаючи архітектурну схему та взаємодію компонентів, опис програми, включаючи структурну схему програмного забезпечення; опис окремих модулів, з визначенням основних функцій і описом їх реалізації;

– перелік джерел інформації – усі використані джерела інформації повинні бути відзначені із зазначенням авторів, назв, років видання та посилань на них.

Зазначена програмна документація повинна бути докладною, структурованою та повністю відповідати вимогам і потребам проєкту, забезпечуючи можливість зрозумілого ознайомлення з роботою, її виконанням і розвитком в майбутньому.

A.6 Стадії та етапи розробки

Розробка складається з наступних етапів:

– постановка завдання роботи (1 тиждень) – складання головних функціональних вимог, опис вхідних та вихідних даних та їх структури, планування, тощо;

– аналіз предметної області (2 тиждень). В ході цього етапу буде проведено вивчення стану проблеми, видалені головні об'єкти та сутності проблеми, вивчені аналогічні рішення та зроблено постановку завдання;

– дослідження моделей для розв'язання проблеми (3-5 тиждень). Вибір із можливих рішень одного, його покращення та модифікація або створення власної моделі для пошуку оптимального розташування об'єктів соціальної інфраструктури;

– розробка архітектури програми (6 тиждень). На цьому етапі розробки системи визначаються ключові об'єкти та основні компоненти майбутньої системи. Здійснюється глибокий аналіз функціональних обов'язків, які

виконують виокремлені компонент програмного забезпечення. Цей аналіз спрямований на створення працюючого варіанту архітектури системи, включаючи пакети, модулі, класи, бібліотеки тощо. Кожен компонент детально описується з вказівкою його призначення та функціональних можливостей;

- розробка програми (7-9 тиждень). Розробка програмних модулів для досягнення поставлених цілей за допомогою вибраних інструментів, використання найкращих архітектурних рішень та практик, поєднання створених модулів у цілісну систему;

- тестування та експериментальне дослідження програми (10-11 тиждень) – виділення ключових вимог для коректної роботи програми, створення плану тестування, проведення тестування коду та інтеграційного тестування, опис результатів;

- оформлення пояснювальної записки та документів до неї (12 тиждень). Склад та зміст програмної документації, який повинен бути у наявності, описано у розділі А.5.

ДОДАТОК Б
Текст програми

Б.1 Текст файла main.py

```

import random
import time

from models.population import Population

from operators import select
from operators import crossover
from operators import mutation
from operators import tools

# random.seed(1)

def gaSimple(
    matrix=[
        [1, 1, 1, 0, 1, 0],
        [1, 1, 1, 0, 0, 0],
        [0, 1, 1, 1, 1, 1],
        [0, 0, 1, 1, 1, 1],
        [1, 0, 1, 1, 1, 1],
        [0, 0, 1, 1, 1, 1],
    ],
    population_size=100,
    max_generations=100,
    p_crossover=0.85,
    p_mutation=0.05,
    is_soft_repair=False,
    is_print=True
):
    start_time = time.time()
    population = Population(population_size, matrix)

    generation_counter = 0
    fitness_stat = []

    while generation_counter < max_generations:
        generation_counter += 1

        offspring = select.select_tournament(population.individuals,
        population_size)
        offspring = [ind.clone() for ind in offspring]

        random.shuffle(offspring)

        for parent1, parent2 in zip(offspring[::2], offspring[1::2]):
            if random.random() < p_crossover:
                crossover.crossover_uniform(parent1, parent2)

        for mutant in offspring:
            if random.random() <= p_mutation:
                mutation.mutation_bit_flip(mutant, p_mutation)

        population.individuals = offspring

        population.repair(is_soft_repair)
        population.count_fitness()

        # population.info()

```

```

        fitness_stat.append(
            tools.gen_info(population, generation_counter, is_print=is_print)
        )

    end_time = time.time()

    return fitness_stat, end_time - start_time

```

Б.2 Текст файла test_crossover.py

```

import matplotlib.pyplot as plt
from main import gaSimple

from draw import fitness_on_gen_graph
from draw import fitness_on_gen_graph_one
from customio.orlib import read_or_data

N_EXPERIMENTS = 5
MAX_GENERATIONS = 100

matrix = read_or_data('scp41.txt')

def conduct_experiment(p_crossover):
    # Initialize lists to store aggregated fitness and diversity data
    max_fitness = [0] * MAX_GENERATIONS
    mean_fitness = [0] * MAX_GENERATIONS
    diversity = [0] * MAX_GENERATIONS

    for _ in range(N_EXPERIMENTS):
        fitness_stats, time = gaSimple(
            matrix=matrix,
            max_generations=MAX_GENERATIONS,
            p_crossover=p_crossover,
            population_size=100,
        )

        for i, generation in enumerate(fitness_stats):
            max_fitness[i] += generation[0]
            mean_fitness[i] += generation[1]
            diversity[i] += generation[2]

    # Average the results over experiments
    max_fitness = [f / N_EXPERIMENTS for f in max_fitness]
    mean_fitness = [f / N_EXPERIMENTS for f in mean_fitness]
    diversity = [d / N_EXPERIMENTS for d in diversity]

    return max_fitness, mean_fitness, diversity

fs05, fs05mean, _ = conduct_experiment(0.5)
fs06, fs06mean, _ = conduct_experiment(0.6)
fs07, fs07mean, _ = conduct_experiment(0.7)
fs08, fs08mean, _ = conduct_experiment(0.8)
fs085, fs085mean, _ = conduct_experiment(0.85)

plt.figure(figsize=(14, 7))

```

```

plt.subplot(1, 2, 1)
plt.plot(fs05, color='red')
plt.plot(fs06, color='blue')
plt.plot(fs07, color='red', linestyle='--')
plt.plot(fs08, color='blue', linestyle='--')
plt.plot(fs085, color='green')
plt.xlabel('Покоління')
plt.ylabel('Пристосованність')
plt.title('Залежність максимальної пристосованості від покоління')
plt.legend([
    'p_crosssover: 0.5',
    'p_crosssover: 0.6',
    'p_crosssover: 0.7',
    'p_crosssover: 0.8',
    'p_crosssover: 0.85',
], loc='lower center')

plt.subplot(1, 2, 2)
plt.plot(fs05mean, color='red')
plt.plot(fs06mean, color='blue')
plt.plot(fs07mean, color='red', linestyle='--')
plt.plot(fs08mean, color='blue', linestyle='--')
plt.plot(fs085mean, color='green')
plt.xlabel('Покоління')
plt.ylabel('Пристосованність')
plt.title('Залежність середньої пристосованості від покоління')
plt.legend([
    'p_crosssover: 0.5',
    'p_crosssover: 0.6',
    'p_crosssover: 0.7',
    'p_crosssover: 0.8',
    'p_crosssover: 0.85',
], loc='lower center')

plt.tight_layout()
plt.show()

```

Б.3 Текст файлу tools.py

```

import numpy as np
from typing import List

from models.individual import Individual
from models.population import Population

def gen_info(population: Population, generation_number: int = 0, is_print: bool
= True):
    best_ind = population.best_individual
    max_f = best_ind.fitness
    mean_f = population.mean_fitness
    diversity = average_hamming_distance(population)

    if is_print:
        print(f'-----Покоління #{generation_number} -----
--'\
            f'\nМаксимальная пристосованість: {max_f}'\
            f'\nСередня пристосованість: {mean_f}'\
            f'\nНайпристосованіша особина: {best_ind}')

```

```

    return max_f, mean_f, diversity, best_ind

def intersection(l1: List[int], l2: List[int]) -> List[int]:
    return [element for element in l1 if element in l2]

def difference(l1: List[int], l2: List[int]) -> List[int]:
    return [element for element in l1 if element not in l2]

def find_uncovered_customers(
    individual: Individual,
    matrix: List[List[int]]
) -> List[int]:
    # m - services count, n - clients count
    m, n = np.shape(matrix)
    # find all uncovered customers
    picked_services = [matrix[index] for index, service in
        enumerate(individual) if service != 0]
    covered_customers = [[index for index, customer in enumerate(service) if
        customer != 0] for service in picked_services]

    covered_customers_unique = np.unique(np.concatenate(covered_customers)) if
    covered_customers else []
    all_customers_indexies = list(range(0, n))
    uncovered_customers = difference(all_customers_indexies,
        covered_customers_unique)

    return uncovered_customers

def find_covering_service_centers(
    customer: int,
    matrix: List[List[int]]
) -> List[int]:
    return [index for index, covering in enumerate(matrix) if
        covering[customer] != 0]

def find_covered_customers_by_service(
    service: int,
    matrix: List[List[int]]
):
    return [index for index, customer in enumerate(matrix[service]) if customer
        != 0]

def hamming_distance(chromosome1: Individual, chromosome2: Individual) ->
None:
    return sum(c1 != c2 for c1, c2 in zip(chromosome1, chromosome2))

def average_hamming_distance(population: Population):
    distances = []
    for i in range(len(population.individuals)):
        for j in range(i + 1, len(population.individuals)):
            distances.append(hamming_distance(population.individuals[i],
                population.individuals[j]))
    return sum(distances) / len(distances) if distances else 0

```

Б.4 Текст файла individual.py

```

import numpy as np
import random
import operators.tools as tools

class Individual(list):
    def __init__(self, *args):
        super().__init__(*args)
        self.fitness = 0
        self.covered_services = np.array([])
        self.is_services_covered = False

    def info(self):
        red_color_code = "\033[31m"
        green_color_code = "\033[32m"
        reset_color_code = "\033[0m"
        mark = f'{green_color_code}•{reset_color_code}'
        if self.is_services_covered else f'{red_color_code}•{reset_color_code}'
        return f'Fitness = {self.fitness}, Genes = {self}, {mark} {self.covered_services}'

    @staticmethod
    def create_random_one(k=10):
        return Individual(random.choices([0, 1], [0.2, 0.8], k=k))

    def clone(self):
        clone_individual = Individual(self)
        clone_individual.fitness = self.fitness
        clone_individual.covered_services = self.covered_services.copy()
        clone_individual.is_services_covered = self.is_services_covered
        return clone_individual

    def soft_repair(self, matrix):
        uncovered_customers = tools.find_uncovered_customers(self, matrix)

        for uc in uncovered_customers:
            service_to_cover = tools.find_covering_service_centers(uc,
matrix)[0]
            self[service_to_cover] = 1

        self.is_services_covered = True

    def repair(self, matrix, should_print=False):
        uncovered_customers = tools.find_uncovered_customers(self, matrix)
        if should_print:
            print(f'Individual {self} uncovers customers {uncovered_customers}')

        for uc in uncovered_customers:
            service_potentially_coverage = {}

            for service in tools.find_covering_service_centers(uc, matrix):
                potentially_covered_customers =
tools.find_covered_customers_by_service(service, matrix)
                can_be_covered_customers = tools.intersection(
                    uncovered_customers,
                    potentially_covered_customers

```

```

    )

    service_potentially_coverage[service] =
len(can_be_covered_customers)

    service_potentially_coverage = {
        key: val
        for key, val
        in sorted(
            service_potentially_coverage.items(),
            key = lambda ele: ele[1],
            reverse=True
        )
    }

    new_service = list(service_potentially_coverage.keys())[0]

    if should_print:
        print(f'Customer #{uc} covered by service #{new_service} with
,
            f'interaction
rate
=
{service_potentially_coverage[new_service]}. '
            f'All
candidates:
{list(service_potentially_coverage.keys())}')

    self[new_service] = 1

    self.is_services_covered = True
    if should_print:
        print(f'Repaired individual is {self}')

```

Б.5 Текст файла population.py

```

import numpy as np
from prettytable import PrettyTable
from models.individual import Individual

class Population:
    def __init__(self, pop_size, matrix):
        self.m, self.n = np.shape(matrix)
        self.matrix = matrix
        self.pop_size = pop_size
        self.individuals = [Individual.create_random_one(self.m) for i in
range(pop_size)]

        self.count_fitness()
        self.repair()

    @property
    def best_individual(self) -> Individual:
        return max([ind for ind in self.individuals if
ind.is_services_covered], key=lambda x: x.fitness)

    @property
    def count_of_uncovered(self):
        return len([ind for ind in self.individuals if not
ind.is_services_covered])

```

```

    @property
    def mean_fitness(self):
        return np.mean([ind.fitness for ind in self.individuals if
ind.is_services_covered])

    @property
    def diversity(self):
        return np.mean([ind.fitness for ind in self.individuals if
ind.is_services_covered])

    def count_fitness(self):
        for i in self.individuals:
            i.fitness = -1 * np.sum(i)

    def repair(self, is_soft_repair=False):
        for individual in self.individuals:
            if is_soft_repair:
                individual.soft_repair(self.matrix)
            else:
                individual.repair(self.matrix)

    def cover_services(self):
        for i in self.individuals:
            picked_services = [self.matrix[index] for index, service in
enumerate(i) if service != 0]
            covered_customers = [[index for index, customer in
enumerate(service) if customer != 0] for service in picked_services]

            if covered_customers:
                covered_customers_unique =
np.unique(np.concatenate(covered_customers))
            else:
                covered_customers_unique = []

            i.covered_services = covered_customers_unique
            i.is_services_covered = len(covered_customers_unique) == self.n

    def info(self):
        t = PrettyTable(['M', 'N', 'POP_SIZE', 'BEST', 'MEAN'])
        t.add_row([self.m, self.n, self.pop_size,
self.best_individual.fitness, self.mean_fitness])
        print(t)
        print('Matrix:')
        [print(row) for row in self.matrix]
        print('Best individual:')
        print(self.best_individual.info())
        [print(f'#{index + 1} {i.info()}') for index, i in
enumerate(self.individuals)]

```

Б.6 Текст файла orlib.py

```

import os
import json
from typing import List
import numpy as np

```

```

PATHNAME = os.path.dirname(os.path.dirname(os.path.realpath(__file__))) + \
    '\\data\\'

```

```

def read_or_data(filename: str) -> List[List[int]]:

```

```

    with open(PATHNAME + filename, 'r') as file:
        m, n = file.readline().strip().split(' ')
        m, n = map(int, [m, n])

```

```

        count = 0
        row = []
        matrix = []

```

```

        # weight block
        for line in file:
            line = line.strip().split(' ')
            if not len(line) > 1:
                count = int(line[0])
                break;

```

```

        # matrix block
        for line in file:
            line = line.strip().split(' ')
            line = list(map(int, line))

```

```

            if (len(row) + len(line) <= count):
                row += line
            else:
                matrix.append(row)
                row = []
                count = line[0]

```

```

        matrix.append(row)
        result_matrix = np.zeros((m, n))

```

```

        for index, row in enumerate(matrix):
            for point in row:
                result_matrix[index][point - 1] = 1

```

```

        return result_matrix

```

```

def read_json_data(filename: str) -> List[List[int]]:

```

```

    with open(PATHNAME + filename, 'r') as file:
        return json.load(file)

```

Б.7 Текст файла authController.ts

```

import { Request, Response } from 'express';
import { v4 as uuid } from 'uuid';
import bcrypt from 'bcrypt';

```

```

import { emailService } from '../services/emailService';
import { userService } from '../services/userService';
import { jwtService } from '../services/jwtService';

```

```

import {

```

```

    loginSchema,
    registrationSchema,
  } from '../validation/validationSchema';

import { ApiError } from '../exceptions/ApiError';
import { IUser } from '../types/IUser';
import { User } from '../models/User';

async function register(req: Request, res: Response) {
  const { email, password, name, surname } = req.body;

  await registrationSchema.validate({
    email, password, name, surname,
  }, { abortEarly: false });

  const existingUser = await userService.getByEmail(email);

  if (existingUser) {
    throw ApiError.BadRequest('Неможливо зареєструвати користувача', {
      email: 'Користувач з такою поштою вже існує',
    });
  }

  const activationToken = uuid();
  const hash = await bcrypt.hash(password, 10);

  await userService.create({
    email, password: hash, name, activationToken });
  await emailService.sendActivationLink(email, activationToken);

  res.sendStatus(200);
}

async function activate(req: Request, res: Response) {
  const { activationToken } = req.params;
  const user = await userService.getByActivationToken(activationToken);

  if (!user) {
    throw ApiError.NotFound(
      'Активувати акаунт за цим посиланням не вдалося. '
      + 'Посилання не дійсне',
    );
  }

  // USER SERVICE
  user.activationToken = null;
  await user.save();

  sendAuthentication(res, user);
}

async function login(req: Request, res: Response) {
  const { email, password } = req.body;

  await loginSchema.validate({ email, password }, { abortEarly: false });

  const user = await userService.getByEmail(email);

  if (!user) {
    throw ApiError.BadRequest('Користувача з такою поштою не існує');
  }
}

```

```

    if (user.activationToken) {
      throw ApiError.BadRequest(
        'Активуйте акаунт за посиланням, '
        + 'яке було відправлено на пошту',
      );
    }

    const isPasswordValid = await bcrypt.compare(password, user.password);

    if (!isPasswordValid) {
      throw ApiError.BadRequest('Невірний пароль');
    }

    sendAuthentication(res, user);
  }

  async function refresh(req: Request, res: Response) {
    const { accessToken } = req.body;

    const userData = jwtService.validateAccessToken(accessToken);

    if (!userData) {
      throw ApiError.Unauthorized();
    }

    const user = await userService.getByEmail((userData as IUser).email);

    if (!user) {
      throw ApiError.Unauthorized();
    }

    sendAuthentication(res, user);
  }

  async function sendAuthentication(res: Response, user: User) {
    const userData = userService.normalize(user);
    const accessToken = await jwtService.generateAccessToken(userData);

    res.send({
      user: userData,
      accessToken,
    });
  }

  export const authController = {
    register,
    activate,
    login,
    refresh,
  };

```

Б.8 Текст файлу ApiError.ts

```

type Errors = { [key: string]: string };

export class ApiError extends Error {
  public status: number;

  public errors: Errors;

```

```

constructor(status: number, message: string, errors: Errors = {}) {
    super(message);

    this.status = status;
    this.errors = errors;
}

public static BadRequest(message: string, errors: Errors = {}) {
    return new ApiError(400, message, errors);
}

public static Unauthorized() {
    return new ApiError(401, 'Користувача з такими даними не існує');
}

public static NotFound(message: string) {
    return new ApiError(404, message);
}
}

```

Б.9 Текст файлу errorMiddleware.ts

```

import { Request, Response, NextFunction } from 'express';
import Yup from 'yup';
import { ApiError } from '../exceptions/ApiError';

// without (Error, Request, Response, NextFunction) params structure it will
// not be an error handle
export function errorMiddleware(error: Error, _req: Request, res: Response,
    _next: NextFunction) {
    if (error instanceof ApiError) {
        const { status, message, errors } = error;

        res.status(status).send({ message, errors });

        return;
    }

    if (error instanceof Yup.ValidationError) {
        const errors = error.inner.reduce((errorsAcc, currentError) => ({
            ...errorsAcc,
            [currentError.path as string]: currentError.message,
        }), {});

        res.status(400).send({
            message: 'Помилка валідації',
            errors,
        });

        return;
    }

    res.status(500).send({
        message: 'Внутрішня помилка сервера',
    });
}

```

Б.10 Текст файла Experiment.ts

```
import {
  Table,
  Column,
  Model,
  DataType,
  AllowNull,
  HasOne,
} from 'sequelize-typescript';
import {
  type ExperimentInput,
  type ExperimentOutput,
} from '../types/experiment';
import { Rating } from '../Rating';

@Table({
  tableName: 'experiment',
  collate: 'utf8_general_ci',
  timestamps: false,
})
export class Experiment extends Model {
  @AllowNull(false)
  @Column(DataType.JSON)
  declare input: ExperimentInput;

  @AllowNull(false)
  @Column(DataType.JSON)
  declare output: ExperimentOutput;

  @AllowNull(false)
  @Column(DataType.DOUBLE)
  declare iterationsThreshold: number;

  @HasOne(() => Rating, { foreignKey: 'experimentId' })
  declare rating: unknown;
}
```

Б.11 Текст файла experimentService.ts

```
import { Experiment } from '../models/Experiment';
import { ExperimentInput, ExperimentOutput } from '../types/experiment';
import { runGeneticAlgorithm } from '../pythonService';

async function create(
  input: ExperimentInput,
  output: ExperimentOutput,
) {
  return Experiment.create({
    input,
    output,
    iterationsThreshold: input.iterationsThreshold,
  });
}

async function getAll() {
```

```

    return Experiment.findAll({ raw: true });
  }

  async function getById(id: number) {
    return Experiment.findByPk(id, { raw: true });
  }

  async function remove(id: number) {
    return Experiment.destroy({
      where: { id },
    });
  }

  function checkCoverage(matrix: Array<Array<number>>): Promise<number[]> {
    return new Promise((resolve) => {
      const uncoveredClients: number[] = [];

      if (!matrix.length) {
        resolve(uncoveredClients);
      }

      for (let col = 0; col < matrix[0].length; col += 1) {
        let covered = false;

        for (let row = 0; row < matrix.length; row += 1) {
          if (matrix[row][col] === 1) {
            covered = true;
            break;
          }
        }

        if (!covered) {
          uncoveredClients.push(col);
        }
      }

      resolve(uncoveredClients);
    });
  }

  function normalize({ input, ...rest }: Experiment) {
    return rest;
  }

  async function makeExperiment(matrix: number[][][]) {
    try {
      const result = await runGeneticAlgorithm(matrix);

      return result;
    } catch (error) {
      global.console.error('Error:', error);

      return null;
    }
  }

  export const experimentService = {
    getAll,
    getById,
    create,
    remove,
    checkCoverage,
  }

```

```

    normalize,
    makeExperiment,
  };

```

Б.12 Текст файлу pythonService.ts

```

import { spawn } from 'child_process';
import { Matrix } from '../types/experiment';

export async function runGeneticAlgorithm(
  matrix: Array<Array<number>>,
): Promise<Matrix> {
  return new Promise((resolve, reject) => {
    const pythonProcess = spawn(
      'python',
      [
        'scripts/main.py',
        JSON.stringify(matrix),
      ],
    );

    let resultMatrix: Matrix;

    pythonProcess.stdout.on('data', (data) => {
      resultMatrix = JSON.parse(data.toString()) as Matrix;
    });

    pythonProcess.stderr.on('data', (data) => {
      reject(data.toString());
    });

    pythonProcess.on('close', (code) => {
      if (code !== 0) {
        // eslint-disable-next-line prefer-promise-reject-errors
        reject(`Exited with code ${code}`);
      }

      try {
        resolve(resultMatrix);
      } catch (error) {
        reject(error);
      }
    });
  });
}

```

Б.13 Текст файлу ArcMapView.tsx

```

import React, { useEffect, useRef, useId, useState, useContext } from 'react';
import debounce from 'debounce';
import esriConfig from '@arcgis/core/config';
import MapView from '@arcgis/core/views/MapView';
import Map from '@arcgis/core/Map';
import Point from '@arcgis/core/geometry/Point';

```

```

import Editor from '@arcgis/core/widgets/Editor';
import { createPointsLayer } from '@utils/pointsLayer';
import { drawRadiusEvent } from '@utils/drawRadiusEvent';
import { createAreasLayer } from '@utils/areasLayer';
import Home from '@arcgis/core/widgets/Home';

import Search from '@arcgis/core/widgets/Search';

import './ArcMapView.css';
import FeatureFilter from '@arcgis/core/layers/support/FeatureFilter';
import Expand from '@arcgis/core/widgets/Expand';
import Popup from '@arcgis/core/widgets/Popup';
import { getAddressByLocation } from '@utils/getAddressByLocation';
import { LatLongWidget } from '@components/LatLongWidget';
import CoordinateConversion from '@arcgis/core/widgets/CoordinateConversion';
import GeoJSONLayer from '@arcgis/core/layers/GeoJSONLayer';
import { FeatureLayerContext } from '@components/FeatureLayerContext';

type Coords = {
  latitude: number,
  longitude: number,
};

export const ArcMapView: React.FC = () => {
  const mapDiv = useRef(null);
  const mapRef = useRef<Map | null>(null);

  const { pointsLayerRef, areasLayerRef } = useContext(FeatureLayerContext);

  const expandWidget = useRef<HTMLDivElement | null>(null);
  const pointsLayerView = useRef<__esri.FeatureLayerView | null>(null);
  const areasLayerView = useRef<__esri.FeatureLayerView | null>(null);

  const [coords, setCoords] = useState<Coords>({ latitude: 0, longitude: 0 });

  const latLongWidgetId = useId();

  useEffect(() => {
    esriConfig.apiKey = import.meta.env.VITE_API_KEY;

    const pointsLayer = createPointsLayer();
    const areasLayer = createAreasLayer();

    if (pointsLayerRef) {
      pointsLayerRef.current = pointsLayer;
    }

    if (areasLayerRef) {
      areasLayerRef.current = areasLayer;
    }

    mapRef.current = new Map({
      basemap: 'streets-vector',
      layers: [pointsLayer, areasLayer],
    });

    const view = new MapView({
      map: mapRef.current,
      container: mapDiv.current ?? undefined,
      center: new Point({
        latitude: 46.3986,
        longitude: 30.7259,
      })
    });
  }, [mapDiv, mapRef]);

```

```

    })),
    zoom: 11,
    popup: new Popup({
      dockEnabled: true,
      dockOptions: {
        buttonEnabled: false,
        breakpoint: false,
        position: 'bottom-left',
      },
      visibleElements: {
        closeButton: false,
      },
    })),
  });

let expandEvent: IHandle; let pointClickEvent: IHandle;

view.whenLayerView(pointsLayer).then((layerView) => {
  pointsLayerView.current = layerView;

  if (!expandWidget.current) {
    return;
  }

  const seasonsExpand = new Expand({
    view,
    content: expandWidget.current,
    expandIcon: 'filter',
    group: 'top-left',
  });

  expandEvent = seasonsExpand.watch('expanded', () => {
    if (!seasonsExpand.expanded && pointsLayerView.current) {
      pointsLayerView.current.filter = new FeatureFilter();
    }
  });

  view.ui.add(seasonsExpand, 'top-left');

  new Promise<void>((resolve) => setTimeout(() => resolve(), 300)).then(()
=> {
    if (!expandWidget.current) {
      return;
    }

    expandWidget.current.style.display = 'block';
  });

  view.whenLayerView(areasLayer).then((layerView) => {
    areasLayerView.current = layerView;

    pointClickEvent = drawRadiusEvent(
      view,
      pointsLayer,
      areasLayerView.current,
    );
  });

  // getPointsData(pointsLayer).then(x => console.log(x));
  // getAreasData(areasLayer).then(x
console.log(x[0].geometry.centroid)); =>

```

```

const editor = new Editor({ view });
const homeButton = new Home({ view });
const searchWidget = new Search({ view });
const ccWidget = new CoordinateConversion({ view });

view.on('pointer-move', () => {
  const { latitude, longitude } = ccWidget.viewModel.currentLocation;

  setCoords({ latitude, longitude });
});

editor.viewModel.sketchViewModel.on('update', debounce(
  async (event) => {
    const { geometry } = event.graphics[0];

    if ('longitude' in geometry && 'latitude' in geometry) {
      const address = await getAddressByLocation(new Point({
        latitude: geometry.latitude as number,
        longitude: geometry.longitude as number,
      }));

      if (!address) {
        return;
      }

      editor.viewModel.featureFormViewModel.setValue('region',
address.region);
      editor.viewModel.featureFormViewModel.setValue('city',
address.city);
      editor.viewModel.featureFormViewModel.setValue('street',
address.street);
      editor.viewModel.featureFormViewModel.setValue('radius', 500);
    }
  },
  300,
));

const latLongWidget = document.getElementById(latLongWidgetId);

if (latLongWidget) {
  view.ui.add(latLongWidget, 'bottom-left');
}

view.ui.add(editor, 'top-right');
view.ui.add(homeButton, 'top-left');
view.ui.add(searchWidget, 'top-left');

return () => {
  expandEvent.remove();
  pointClickEvent.remove();
  view.destroy();
};
}, []);

return (
  <>
    <div ref={expandWidget} className="seasons-filter esri-widget">
      {
        { type: 'government', name: 'Пункт Незламності' },
        { type: 'business', name: 'Пункт Незламності відповідального бізнесу'
},

```

```

].map(({ type, name }) => (
  <button
    type="button"
    key={type}
    className="season-item"
    onClick={() => {
      if (!pointsLayerView.current) {
        return;
      }

      pointsLayerView.current.filter = new FeatureFilter({
        where: `typeOfPoint = '${type}'`,
      });
    }}
  >
    {name}
  </button>
))}
</div>

<div ref={mapDiv} className="map-view" />

<LatLongWidget
  coords={coords}
  id={latLongWidgetId}
/>
</>
);
};

```

Б.14 Текст файла pointsLayer.ts

```

import FeatureLayer from '@arcgis/core/layers/FeatureLayer';
import { UniqueValueRenderer } from '@arcgis/core/renderers';
import { PictureMarkerSymbol } from '@arcgis/core/symbols';
import PopupTemplate from '@arcgis/core/PopupTemplate';
import Point from '@arcgis/core/geometry/Point';
import { Point as PointOfInvincibility } from '@cutomTypes/experiment/point';

export function createPointsLayer(): FeatureLayer {
  const governmentSymbol = new PictureMarkerSymbol({
    url: `${import.meta.env.VITE_API_URL}/img/pn_yellow.svg`,
    width: '40px',
    height: '40px',
  });
  const businessSymbol = new PictureMarkerSymbol({
    url: `${import.meta.env.VITE_API_URL}/img/pn_icon.svg`,
    width: '40px',
    height: '40px',
  });

  const template = new PopupTemplate({
    title: '{typeOfPoint}',
    content: [
      {
        type: 'fields',
        fieldInfos: [
          {
            fieldName: 'typeOfPoint',

```

```

        label: 'Тип',
      },
      {
        fieldName: 'radius',
        label: 'Радіус обслуговування',
      },
      {
        fieldName: 'region',
        label: 'Регіон',
      },
      {
        fieldName: 'city',
        label: 'Місто',
      },
      {
        fieldName: 'street',
        label: 'Вулиця',
      },
    ],
  },
],
});

const renderer = new UniqueValueRenderer({
  legendOptions: {
    title: 'Route type',
  },
  defaultSymbol: governmentSymbol,
  defaultLabel: 'Other',
  field: 'typeOfPoint',
  uniqueValueInfos: [
    {
      value: 'government',
      symbol: governmentSymbol,
      label: 'Interstate',
    },
    {
      value: 'business',
      symbol: businessSymbol,
      label: 'US Highway',
    },
  ],
});

return new FeatureLayer({
  url: import.meta.env.VITE_POINTS_LAYER_URL,
  renderer,
  title: 'Пункт незламності',
  popupTemplate: template,
});
}

export async function getPointsData(
  pointsLayer: FeatureLayer,
): Promise<PointOfInvincibility[]> {
  const query = pointsLayer.createQuery();

  query.where = '1=1';
  query.outFields = ['*'];

  const queryResults = await pointsLayer.queryFeatures(query);

```

```

return queryResults.features.map(({
  geometry,
  attributes: { OBJECTID, ...rest },
}) => ({
  geometry: {
    latitude: (geometry as Point).latitude,
    longitude: (geometry as Point).longitude,
  },
  attributes: {
    id: OBJECTID,
    ...rest,
  },
}));
}

export async function getRadiusById(
  pointsLayer: FeatureLayer,
  objectId: number,
) {
  const query = pointsLayer.createQuery();

  query.where = `OBJECTID = ${objectId}`;
  query.outFields = ['radius'];

  const queryResult = await pointsLayer.queryFeatures(query);

  // Ensure that a feature was found
  if (queryResult.features.length > 0) {
    const { radius } = queryResult.features[0].attributes;

    return radius;
  }

  return null;
}

```

Б.14 Текст файла RegistrationPage.tsx

```

import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import { useMutation } from 'react-query';
import {
  Formik, Form, Field, ErrorMessage,
} from 'formik';

import { LoadingButton } from '@components/LoadingButton/LoadingButton';
import { RegisterInputPayload } from '@customTypes/api/auth';
import { authService } from '@services/authService';
import { AppError } from '@customTypes/appError';
import { apiErrorService } from '@services/apiErrorService';

import { registerSchema } from './RegistrationPage.validation';

export const RegistrationPage: React.FC = () => {
  const [error, setError] = useState('');

  const registerMutation = useMutation<void, AppError, RegisterInputPayload>({
    mutationFn: async (payload) => {
      await authService.register(payload);
    },
  });

```

```

    },
    onError: (err) => {
      const { title } = apiErrorService.getMessage(err);

      setError(title);
    },
    onMutate: () => setError(''),
  });

return (
  <div className="vh-100 bg-light d-flex align-items-center justify-content-center flex-column">
    <h1 className="mb-3">Реєстрація</h1>

    <div className="col-10 col-md-6 col-xxl-4">
      <Formik
        initialValues={{ name: '', email: '', password: '' }}
        validationSchema={registerSchema}
        onSubmit={values => registerMutation.mutate(values)}
      >
        <Form className="fs-3" onChange={() => setError('')}>
          <div className="mb-3 mb-md-4">
            <Field
              name="name"
              type="text"
              className="form-control fs-3"
              placeholder="Введіть ім'я користувача"
            />
            <div className="form-text fs-5 ms-2 text-danger">
              <ErrorMessage name="name" />
            </div>
          </div>

          <div className="mb-3 mb-md-4">
            <Field
              name="email"
              type="email"
              className="form-control fs-3"
              placeholder="Введіть пошту"
            />
            <div className="form-text fs-5 ms-2 text-danger">
              <ErrorMessage name="email" />
            </div>
          </div>

          <div className="mb-3 mb-md-4">
            <Field
              name="password"
              type="password"
              className="form-control fs-3"
              placeholder="Введіть пароль"
            />
            <div className="form-text fs-5 ms-2 text-danger">
              <ErrorMessage name="password" />
            </div>
          </div>

          <LoadingButton
            type="submit"
            isLoading={registerMutation.isLoading}
            className="fs-3"

```

```

    >
      Зареєструватися
    </LoadingButton>

    {error && (
      <div className="form-text fs-5 ms-2 text-danger">
        {error}
      </div>
    )}

    {registerMutation.isSuccess && (
      <div className="form-text fs-5 ms-2 text-success fw-bolder">
        Активуйте акаунт за посиланням що прийшло вам на пошту
      </div>
    )}

    <Link
      to="/login"
      className="fs-5 ms-2 mt-3 text-center"
      style={{ textDecoration: 'underline' }}
    >
      Вже маю акаунт
    </Link>
  </Form>
</Formik>
</div>
</div>
);
};

```

ДОДАТОК В
Слайди презентації

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ОПТИМІЗАЦІЇ
СИСТЕМИ ОБ'ЄКТІВ СОЦІАЛЬНОЇ ІНФРАСТРУКТУРИ

Виконав
ст. гр. КНТ-132М

Керівник
к.т.н., доцент

Сергій ЛІПІХІН

Валерій ЛЬОВКІН

Рисунок В.1 – Титульний слайд

Вступ

Об'єкт дослідження – процес оптимізації розташування об'єктів соціальної інфраструктури.

Предмет дослідження – програмне забезпечення для пошуку оптимального розташування об'єктів соціальної інфраструктури.

Мета роботи – створення програмного забезпечення геопросторової оптимізації об'єктів соціальної інфраструктури для знаходження найкращого варіанту їх розташування, сприяючи раціональному використанню ресурсів, зменшенню витрат та покращенню якості обслуговування.

Завдання:

- аналіз предметної області;
- дослідження методів і моделей вирішення задачі покриття множини;
- проектування програмного забезпечення для оптимізації системи об'єктів соціальної інфраструктури;
- розробка програмного забезпечення.

2

Рисунок В.2 – Вступ

Актуальність

Пункт незламності – це місце де люди можуть безкоштовно отримати базові блага, такі як електроенергія, світло та тепло.

На листопад 2023 року в Україні розгорнуто вже понад **13 тисяч** пунктів незламності

Пункти незламності в Потіївській області

Район	Кількість пунктів незламності на 1000 осіб (сині)	Кількість пунктів незламності на 1000 осіб (червоні)
Кременецький район	0.35	0.40
Миргородський район	0.50	0.75
Потіївський район	0.28	0.40
Потіївський район	0.30	0.45

Динаміка кількості пунктів незламності по місяцях

Місяць	Кількість пунктів незламності
Жовтень 2022	190
Листопад 2022	160
Грудень 2022	220
Січень 2023	90
Лютий 2023	100
Березень 2023	80
Квітень 2023	40
Травень 2023	140
Червень 2023	130
Листопад 2023	110
Січень 2024	80
Лютий 2024	80

3

Рисунок В.3 – Актуальність

Постановка задачі

Бінарна матриця покриття:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix}$$

де $a_{i,j} \in \{0, 1\}$ – значення поля (i, j)

Задача оптимізації множини визначається формулою:

$$\min \sum_{i \in I} x_i$$

за умов:

$$\sum_{i \in I} a_{i,j} x_i \geq 1, \quad \forall j \in J$$

$$x_i \in \{0, 1\}, \quad \forall i \in I,$$

Радіус обслуговування Сервісний центр Клієнтська область



4

Рисунок В.4 – Постановка задачі

Порівняння методів вирішення ЗПМ

Критерій	Метаевристики	Жадібні методи	Точні алгоритми
Гнучкість	+	±	-
Здатність оптимізації складних рішень	+	-	±
Гарантія точного рішення	-	-	+
Швидкість реалізації	±	+	-
Ресурсоемність	-	+	-

Критерій	CRO	GA	GSEMO	JPSO	SA	SEIP
Кількість ітерацій	2126,0 ± 53,0	2558,0 ± 478,0	3608,0 ± 1073,0	2001,0 ± 0,0	2649,0 ± 839,0	4768,0 ± 1250,0
Середня тривалість (с)	1,048 ± 0,16	4,342 ± 1,109	20,952 ± 8,107	701,123 ± 130,38	14,047 ± 5,01	0,666 ± 0,242
Середня тривалість ітерацій	0,00049 ± 7e-05	0,00172 ± 0,0030	0,00586 ± 0,0007	0,35039 ± 0,0864	0,00537 ± 0,0224	0,00014 ± 3e-05
Середній коефіцієнт апроксимації	1,644 ± 0,07	1,078 ± 0,091	1,111 ± 0,091	1,433 ± 0,082	1,433 ± 0,063	1,167 ± 0,131
Найкращий коефіцієнт апроксимації	1,556	1,0	1,0	1,333	1,333	1,0

5

Рисунок В.5 – Порівняння методів вирішення ЗПМ

ГА та модифікація оператора відновлення

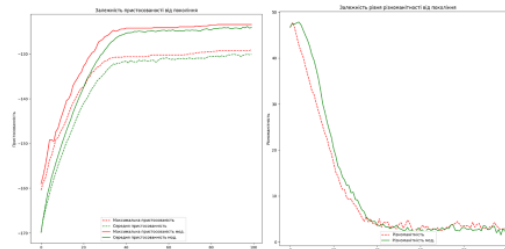
Номер біта 1 2 3 4 5 ... n-1 n
Хромосома 1 0 1 1 0 ... 0 1

Фітнес функція обчислюється за формулою:

$$f_i = \sum_{j=1}^n x_{ij}$$

де x_{ij} значення j -го біта i -го індивіда.

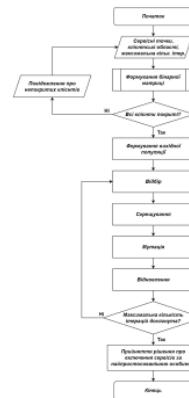
Модифікація передбачає для кожного непокритого клієнта вибір такого сервісного центру, що має найвищий показник ефективності



6

Рисунок В.6 – ГА та модифікація оператора відновлення

Запропонований метод пошуку оптимального розташування об'єктів соціальної інфраструктури



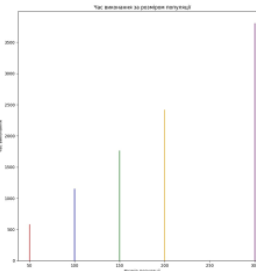
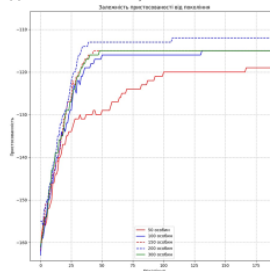
7

Рисунок В.7 – Метод пошуку оптимального розташування об'єктів

Налаштування параметрів ГА

Фінальна конфігурація моделі наступна:

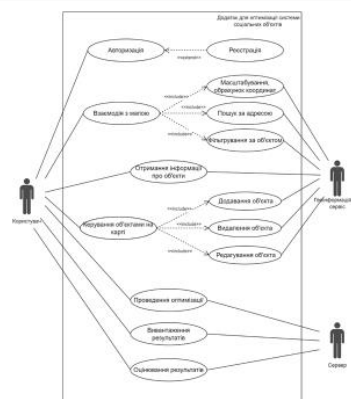
- турнірний відбір із трьома учасниками;
- рівномірне схрещування із імовірністю обміну генами 0.3;
- мутація – інвертування із імовірністю для кожного біту в залежності від розміру гену;
- розмір популяції 100 особин;
- імовірність схрещування 0.85;
- імовірність мутації 0.05;
- максимальна кількість поколінь – 50.



8

Рисунок В.8 – Налаштування параметрів ГА

Діаграма варіантів використання програми



9

Рисунок В.9 – Діаграма варіантів використання програми

Вибір інструментів розробки

- TypeScript для створення вебдодатку
- Python для розробки моделі
- монорепозиторій як стратегія розробки
- ArcGIS для робота з картою

Критерій	JavaScript	Python
Типізація	динамічна, TypeScript	динамічна, анотація типів
Структури даних	±	+
Веб-API	клієнтська сторона	серверна сторона
Веб-пакети	+	+
Рендеринг	клієнтський, серверний	серверний
ГІС	для браузеру	для аналізу даних
Багатопоточність	-	+
Візуалізація даних	-	+
Робота з файлами	+	+
Швидкість виконання	±	±
Швидкість розробки	+	+

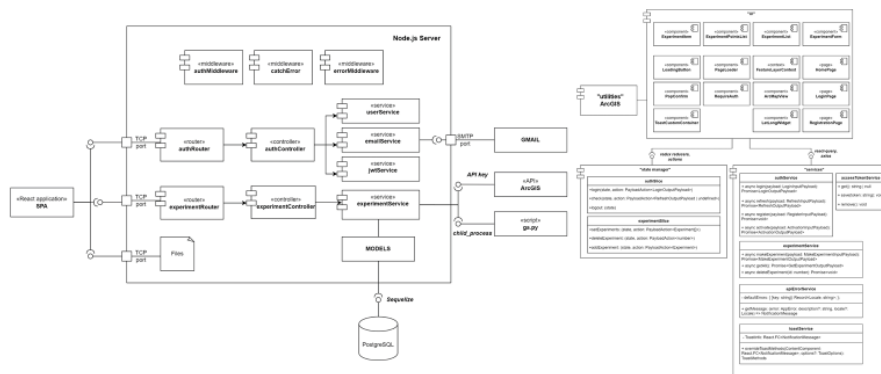
Критерій	Монорепозиторій	Полірепозиторій
Централізований доступ до коду	+	-
Єдиний набір інструментів	+	-
Здатність вносити атомарні коміти	+	-
Оптимізоване управління залежностями	+	-
Легкість у розумінні структури	+	-
Єдина документація	+	-
Необхідність завантаження всього коду	-	+
Масштабування (великі проекти)	±	±

Критерій	Google Maps	ArcGIS	React Leaflet
API та інтеграція з React.js	Гарне	Відмінне	Гарне
Документація	Гарна	Відмінна	Достатня
Гнучкість і масштабованість	Висока	Найвища	Висока
Функціональні можливості	Високі	Найвищі	Стандартні
Ліцензування та вартість	Обмежене	Доступне	Вільне
Спільнота та підтримка	Велика	Найбільша	Достатня
Персоналізація	Добра	Найкраща	Добра
Доступ до геоданих	Хороший	Найширший	Стандартний

10

Рисунок В.10 – Вибір інструментарію розробки

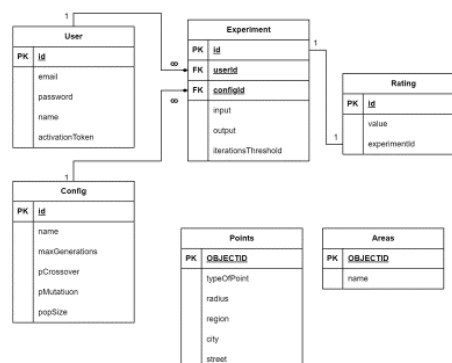
Архітектура застосунку



11

Рисунок В.11 – Архітектура застосунку

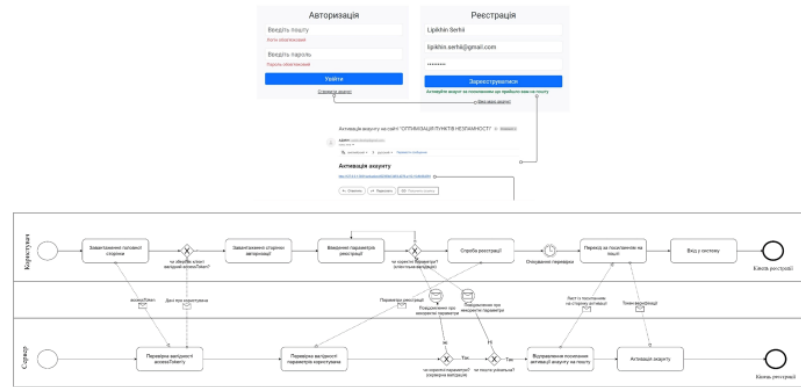
База даних



12

Рисунок В.12 – База даних

Сценарій авторизації

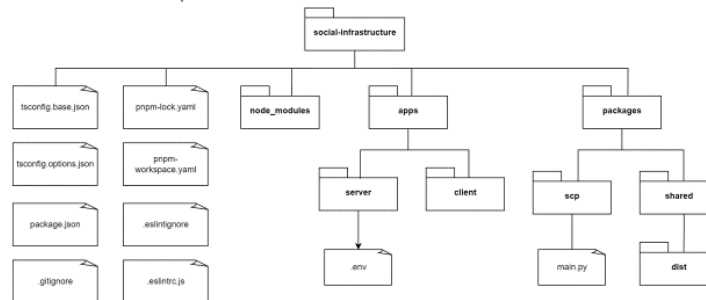


13

Рисунок В.13 – Сценарій авторизації

Умови виконання програми

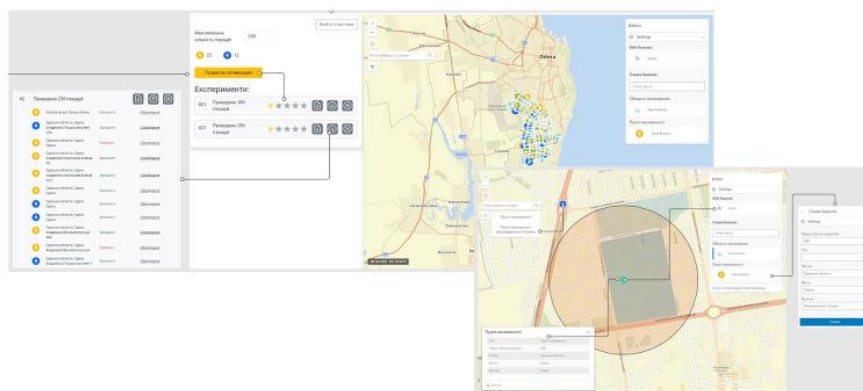
- додержання цілісної файлової структури
- дотримання пакетної цілісності
- задоволення мінімальних апаратних вимог



14

Рисунок В.14 – Умови виконання програми

Робота з програмою



15

Рисунок В.15 – Робота з програмою

Експериментальне дослідження

В результаті оптимізації було зменшено кількість пунктів незламності на 32 % за 13,2 с.

№	Адреса	Радіус	Результат
1	Академіка Глушка проспект 2	500	Залишити
2	Академіка Глушка проспект 29А	500	Залишити
3	Академіка Корольова вулиця 90	500	Залишити
4	Академіка Корольова вулиця 63/1	450	Прибрати
5	Вулиця Академіка Вільямса, 79	500	Прибрати
...	...	...	...
35	Варненська вулиця 4А/1	500	Залишити
36	Варненська вулиця 2/2	500	Прибрати
37	Академіка Філатова вулиця 1	500	Залишити
38	Ромашкова вулиця 55	500	Залишити



16

Рисунок В.16 – Експериментальне дослідження

Висновки

В результаті роботи було досліджено задачу покриття множини та порівняно методи її вирішення. Для проведення оптимізації системи об'єктів соціальної інфраструктури було обрано генетичний алгоритм.

Наукова новизна роботи полягає в запропонованому методі пошуку оптимального розташування об'єктів соціальної інфраструктури з використанням генетичного алгоритму, що відрізняється від існуючих рішень модифікованим оператором відновлення. Такий оператор використовує концепцію «конфліктних сервісних центрів», що передбачає появу недопустимих рішень, – за той же самий час виконання та кількість ітерацій модифікований оператор дозволяє отримувати більш наближені до оптимального рішення.

Інтеграція у розроблену програму сучасної **геоінформаційної системи** дозволяє користувачу створювати та розташовувати об'єкти на мапі, задаючи також їх атрибути. У поєднанні із високою швидкістю пошуку оптимального рішення, додаток надає користувачу змогу зручно та швидко проводити оптимізацію будь-яких соціальних об'єктів.

Використовуючи реальні дані про розташування пунктів незламності у Київському районі Одеси **проведено оптимізацію на практиці**. В результаті вдалось зменшити кількість об'єктів на 32 %, задовольняючи умову, що кожен клієнт повинен мати доступ принаймні до одного сервісного центру.

17

Рисунок В.17 – Висновки