Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконала: *студентка групи КН-20-2* _____ *Зоряна БОРЕВІЧ*
Група виконавця | Підпис | Ім'я, ПРІЗВИЩЕ

Керівник: *зав. каф. КН, д.т.н., проф.* _____ *Олександр БАРМАК*
Науковий ступінь, посада | Підпис | Ім'я, ПРІЗВИЩЕ

Нормоконтроль: *к.т.н., доц. каф. КН* _____ *Руслан БАГРІЙ*
Науковий ступінь, посада | Підпис | Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
зав. кафедри КН, д.т.н., професор _____ *Олександр БАРМАК*
Підпис | Ім'я, ПРІЗВИЩЕ

11 червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

(підпис)
д.т.н., професор Олександр БАРМАК
«*16*» *02* *2024* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим»*

2. Завдання видано студентці *Зоряні БОРЕВІЧ*
(Ім'я, прізвище)

3. Керівник роботи *завідувач кафедри КН, д.т.н., проф. Олександр БАРМАК*
(посада, ім'я, прізвище)

4. Затверджено наказом університету від «*15*» *02* *2024* р. № *8*

5. Дата видачі завдання студенту: «*16*» *02* *2024* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – покращення ефективності доставлення допомоги у вебсистемах для малозабезпечених людей. Необхідно розробити метод доставлення з алгоритмом мурахи для вебсистеми допомоги малозабезпеченим. При формуванні маршруту доставлення потрібно враховувати статус готовності замовлення до відправки. Слід забезпечити виконання функцій створення та редагування нової заявки на допомогу, автоматизоване створення нового маршруту доставлення з використанням методу доставлення з передбаченим вибором початкового пункту, перегляд історії замовлень із статусом їх виконання, перегляд історії маршрутів.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником, складання календарного графіка виконання роботи*	*січень 2024*	*виконано*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*лютий 2024*	*виконано*
3	*Робота над розділом 1 Аналіз методів планування маршрутів для використання у вебсистемах*	*березень 2024*	*виконано*
4	*Робота над розділом 2 Метод доставлення за мурашиним алгоритмом*	*квітень 2024*	*виконано*
5	*Робота над розділом 3 Програмна реалізація вебсистеми допомоги малозабезпеченим*	*травень 2024*	*виконано*
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2024*	*виконано*
7	*Попередній захист кваліфікаційної роботи*	*червень 2024*	*виконано*
8	*Захист кваліфікаційної роботи бакалавра*	*червень 2024*	*виконано*

Виконавець: *студентка групи КН-20-2* _____ *Зоряна БОРЕВІЧ*
Група виконавця — Підпис — Ім'я, ПРІЗВИЩЕ

Керівник: *зав. каф. КН, д.т.н.,проф.* _____ *Олександр БАРМАК*
Науковий ступінь, посада — Підпис — Ім'я, ПРІЗВИЩЕ

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим»*

Виконавець кваліфікаційної роботи бакалавра: *студентка групи КН-20-2 Зоряна БОРЕВІЧ*

Керівник кваліфікаційної роботи бакалавра: *зав. кафедри КН, д.т.н., професор Олександр БАРМАК*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
75	46	5	33	3

Метою кваліфікаційної роботи бакалавра є підвищення ефективності доставлення допомоги у вебсистемах для малозабезпечених людей. Для досягнення мети необхідно розробити та програмно реалізувати метод доставлення з алгоритмом мурахи та вебсистему, що буде його використовувати. Для розробки інформаційної системи було використано мови програмування C# та JS, фреймворк ASP.NET Core WEB API, бібліотеки React та Ionic, а також систему керування базами даних MongoDB.

Розроблена вебсистема призначена для малозабезпечених людей та адміністраторів фонду допомоги таким людям. Реалізований метод доставлення з алгоритмом мурахи для формування маршруту дозволяє підвищити ефективність доставки допомоги, формуючи оптимальний маршрут доставки.

Напрямками практичного використання розробленої вебсистеми є застосування запропонованого методу для імплементації у системах допомоги малозабезпеченим людям.

Ключові слова: метод доставлення, мурашиний алгоритм, маршрут, вебсистема, допомога малозабезпеченим.

Виконавець: *студентка групи КН-20-2* _____ *Зоряна БОРЕВІЧ*

Група виконавця | Підпис | Ім'я, ПРІЗВИЩЕ

Зміст

Перелік скорочень....4
Вступ....5
Розділ 1 Аналіз методів планування маршрутів для використання у вебсистемах....7
1.1 Огляд методів планування маршрутів....7
1.1.1 Алгоритм Дейкстри....9
1.1.2 Генетичні алгоритми....10
1.1.3 Мурашиний алгоритм....11
1.2 Огляд вебреалізацій систем допомоги....13
1.3 Мета, задачі та вимоги до реалізації інформаційної системи....19
Розділ 2 Метод доставлення за мурашиним алгоритмом....21
2.1 Мурашиний алгоритм у задачах планування маршрутів....21
2.2 Метод доставлення за мурашиним алгоритмом....23
2.3 Особливості реалізації мурашиних алгоритмів у бібліотеках....26
2.4 Проектування вебсистеми допомоги малозабезпеченим....29
2.4.1 Серверна частина....29
2.4.2 Клієнтська частина....32
2.5 Проектування бази даних....37
2.6 Висновки до розділу 2....41
Розділ 3 Програмна реалізація вебсистеми допомоги малозабезпеченим....42
3.1 Загальна архітектура вебсистеми допомоги малозабезпеченим....42
3.2 Структура та особливості реалізації методу доставлення за мурашиним алгоритмом....44
3.3 Структура та особливості реалізації вебсистеми допомоги малозабезпеченим....49
3.3.1 Структура та особливості реалізації серверної частини....49
3.3.2 Структура та особливості реалізації клієнтської частини....53
3.3.3 Структура бази даних....56
3.4 Тестування інформаційної системи....57
3.5 Вимоги до публікації вебсистеми....59
3.6 Функціональність вебсистеми....60

3.7. Оцінка покращення ефективності доставлення допомоги за запропонованим методом.....67
3.8 Висновки до розділу 3.....70
Загальні висновки 72
Перелік посилань 74
Додатки

Перелік скорочень

Скорочення, термін, позначення	Пояснення
БД	База даних
ІС	Інформаційна система
КРБ	Кваліфікаційна робота бакалавра
ACO	Ant colony optimization
API	Application program interface
CSS	Cascading style sheets
DTO	Data transfer object
ERD	Entity relationship diagram
GPS	Global positioning system
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
HTTPS	Hypertext transfer protocol secure
JWT	Json WEB token
REST	Representational state transfer
SPA	Single page application
SQL	Structured query language
TSP	Travelling salesman problem
URL	Uniform resource locator

Вступ

Кваліфікаційна робота бакалавра присвячена розробці методу доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим, що дає можливість побудувати найоптимальніший маршрут доставки допомоги.

Актуальність. В умовах сучасних викликів, таких як економічна нестабільність та соціальні кризи, забезпечення надання основних ресурсів малозабезпеченому населенню стало особливо важливим. В Україні існує значна потреба в ефективних і надійних системах розподілу гуманітарної допомоги. Одним з основних питань, з яким стикаються благодійні організації, є оптимізація процесу доставки, що тягне за собою мінімізацію часу і витрат при забезпеченні своєчасної і точної доставки.

Ця проблема стала особливо значущою в умовах повномасштабної війни, яка почалася в Україні в 24 лютого 2022 році. Військові дії призвели до масового переміщення населення, руйнування інфраструктури та значного збільшення кількості вразливих осіб, які гостро потребують допомоги. У таких умовах ефективна система доставки допомоги є не тільки питанням соціальної відповідальності, але й життєво важливим елементом забезпечення виживання постраждалих та збереження населення України.

Алгоритми мурашиного типу мають високу здатність адаптуватися до змінних умов середовища, таких як зміни в дорожній інфраструктурі, погодні умови або інші непередбачувані фактори, що особливо актуально в умовах військових дій. Це забезпечує стабільну та надійну роботу системи доставки навіть в умовах невизначеності.

Саме тому розробка методу доставлення за мурашиними алгоритмом для вебсистем допомоги малозабезпеченим є дуже актуальною у нашій країні.

Об’єкт дослідження – процес доставлення допомоги у вебсистемі для малозабезпечених людей.

Предмет дослідження – методи оптимізації, еволюційні алгоритми, мурашиний алгоритм у системах планування маршрутів

Мета кваліфікаційної роботи бакалавра – підвищення ефективності доставлення допомоги у вебсистемах для малозабезпечених людей.

Завдання кваліфікаційної роботи бакалавра – Провести аналіз методів планування маршрутів для використання у вебсистемах. Оглянути методи планування маршрутів та існуючі вебреалізації систем допомоги. Дослідити використання алгоритму мурахи в задачах планування маршрутів та його особливості для вирішення проблеми допомоги малозабезпеченим. Розробити метод доставлення з алгоритмом мурахи для вебсистем допомоги малозабезпеченим, що буде формувати та віддавати як результат оптимальний маршрут для почергового відвідування кожного пункту призначення. Спроектувати серверну та клієнтську частини для вебсистеми допомоги малозабезпеченим. Спроектувати базу даних для вебсистеми допомоги малозабезпеченим. Розробити загальну архітектуру вебсистеми. Програмно реалізувати метод доставлення з мурашиним алгоритмом. Програмно реалізувати клієнтську та серверну частини вебсистеми допомоги малозабезпеченим. Провести тестування розробленої інформаційної системи та методу доставлення. Провести аналіз функціональності вебсистеми. Оцінити покращення ефективності доставлення допомоги малозабезпеченим з використанням методу доставлення з алгоритмом мурахи.

Розділ 1 Аналіз методів планування маршрутів для використання у вебсистемах

1.1 Огляд методів планування маршрутів

Задача планування маршрутів полягає у визначенні оптимального шляху для досягнення пункту призначення, зважаючи на різні фактори, наприклад: трафік на дорогах, погодні умови, стан доріг на шляху [1].

Планування маршрутів є важливою складовою сучасного транспортного управління та логістики. Історія планування маршрутів налічує багато етапів. Перші спроби систематизації маршрутів відбувалися в давні часи, коли торговці та подорожні використовували картографію та знання місцевості для визначення оптимальних шляхів переміщення[1]. З розвитком технологій, зокрема винайденням комп'ютерів та географічних інформаційних систем, планування маршрутів стало більш точним та ефективним.

У сучасному світі, сповненому різними видами транспорту, планування маршрутів є важливою і вже невід’ємною частиною використання засобів пересування. Заздалегідь спланований оптимальних шлях економить ресурси, такі як: паливо для транспорту, час у дорозі та сили, витрачені водієм, приносить менше шкоди навколишньому середовищу через зменшення кількості викидів CO2 у атмосферу [2]. Також правильно сплановані маршрути збільшують продуктивність людей чи компаній, та зменшують рівень стресу водіїв, знімаючи з них відповідальність за це. Однак, важливо розуміти, що не завжди оптимальний шлях буде найкоротшим або найшвидшим, адже при пошуку такого маршруту, будуть враховуватись всі переваги та недоліки та дистанції, а також всі вимоги, які були вказані користувачем.

Планування маршрутів є широко застосованим рішенням у різних сферах діяльності людини. Найбільш популярним є у логістиці та сервісах доставок, наприклад, пошта, доставка їжі чи квітів, та різні транспортні перевезення [3]. Без правильно спланованих маршрутів, кур’єри б часто привозили вже холодні страви (бо не врахували час приготування), фермери б доставляли вже зіпсовану їжу до магазинів, а розрахунок орієнтовного часу доставлення посилок був би майже

неможливим. Також, зросли б ціни на такі послуги, оскільки людина не здатна визначати оптимальні маршрути так точно, як машина, тому клієнти платили б за похибки у виборі шляху, розрахунку часу та пального.

Саме тому зараз існує багато навігаційних систем, такі як GPS, які планують маршрути до пунктів призначення користувачів, враховуючи їх метод пересування(ходьба, авто, потяг, велосипед тощо) та інші фактори.

Також без планування маршрутів важко уявити існування різних транспортних служб (громадський транспорт, потяги, літаки, кораблі). Адже тут важливо не лише як їде єдиний транспорт, а і те, як він рухається відносно інших засобів. Це важливо для уникнення зіткнення потягів на коліях, одночасного прибуття різних трамваїв на одну зупинку, для надання достатнього часу посадки клієнтам та можливості пересадки.

Остання популярна сфера використання планування маршрутів – туризм [4]. Якщо у попередніх сферах, зазвичай переважали фактори швидкості та відстані шляху, то у туризмі люди часто спеціально обирають найдовші маршрути, аби насолодитися краєвидами та побачити більше нових для себе місць.

Вибір алгоритму маршрутизації або методу планування маршруту залежить від конкретних вимог. Серед поширених вимог можна виділити мінімальний час, мінімальну відстань та максимальну безпеку. Зазвичай можна задовольнити лише одну вимогу одночасно. Наприклад, якщо безпека має високий пріоритет, то шлях може бути обраний з урахуванням цього, навіть якщо це означає деяке збільшення дистанції. Також вибір залежить від технологій, які будуть використовуватися: комбінаторна оптимізація, жадібні алгоритми, лінійне програмування, динамічне програмування.

Однієї з найпоширеніших задач комбінаторної оптимізації є задача комівояжера або TSP (Travelling salesman problem)[5]. Основне завдання даної задачі – пошук найкоротшого шляху, який відвідує кожен пункт призначення лише один раз. Вперше цю проблему було висвітлено у 1930 році [6] і вона досі користується попитом вже майже століття. Завдяки великій кількості проведених

досліджень, люди сучасності володіють багатьма методами, які можуть оптимізувати побудування маршрутів, зберігаючи їхній час та ресурси.

Розглянемо існуючі підходи до планування маршрутів: жадібний алгоритм Дейкстри для пошуку найкоротшого шляху, генетичні та мурашиний алгоритми комбінаторної оптимізації для вирішення задачі комівояжера.

1.1.1 Алгоритм Дейкстри

Алгоритм Дейкстри є одним з найпростіших методів пошуку найкоротшого шляху між точками в мережі з позитивними вагами ребер. Його створив Едсгер Дейкстра в 1956 році[7]. Алгоритм прокладає найкоротший шлях від однієї точки мережі до всіх інших, керуючись правилом "жадібності": на кожному кроці вибирає точку, до якої найкоротший шлях від стартової точки є найкоротшим, а потім оновлює відстані.

Алгоритм Дейкстри складається з наступних основних кроків[8].

1. Ініціалізація відстаней. Спочатку відстані від початкового вузла до всіх інших вузлів встановлюються на максимальні значення. Відстань від початкового вузла до нього самого дорівнює 0.

2. Обробка вузлів. Для кожного вузла алгоритм вибирає вузол, який має найкоротшу відстань до початкового вузла з усіх не оброблених вузлів. Потім цей вузол позначається як оброблений.

3. Оновлення відстаней. Для всіх сусідніх вершин обраної вершини перевіряється, чи можна досягти сусідніх вершин із меншою відстанню через обрану вершину. Якщо це можливо, відстань до сусідніх вершин оновлюється.

Ці кроки повторюються, поки залишаються необроблені вершини.

Алгоритм Дейкстри триває, поки всі вершини графа не будуть розглянуті або поки не буде знайдена цільова вершина, до якої шукається найкоротший шлях. Після виконання алгоритму він визначає найкоротший шлях від початкової вершини до всіх інших вершин графа [8].

Даний метод має свої переваги та недоліки.

Із переваг варто зазначити, що він є досить простим у реалізації, його концепцію легко зрозуміти та втілити в життя. Також він дієвий для невеликих графів, забезпечуючи швидку та ефективну роботу [9].

Значущими недоліками алгоритму є такі:

1) не підтримує графи з від'ємними вагами: застосовується лише до графів, у яких усі ваги ребер позитивні, при наявності від'ємних ваг ребер алгоритм виходить з ладу [10];

2) не дієвий для великих графів: алгоритм буде працювати буде повільно, оскільки на кожному кроці вимагає оновлення відстаней до всіх сусідніх вершин;

3) визначає найкоротші шляхи лише від визначеного стартового вузлаю. Він не знаходить найкоротші шляхи між довільними парами вузлів у графі [11];

4) надмірне споживання ресурсів, а саме пам’яті, оскільки для роботи алгоритму Дейкстри необхідно зберігати інформацію про відстані до всіх вузлів графа.

1.1.2 Генетичні алгоритми

Генетичні алгоритми – це методи оптимізації, які імітують природний відбір і генетичні мутації, щоб знайти найкращі рішення для проблем [12]. Вони часто використовуються для розв’язання задач, таких як маршрути доставки та задача комівояжера. Алгоритм створює групу можливих рішень, так званих популяцій, а потім використовує операції схрещування та мутації для створення нових популяцій, які з часом наближаються до оптимального рішення.

Генетичний алгоритм складається з таких основних компонентів [13]:

– популяція – група кандидатних рішень (індивідів) на початку процесу оптимізації, що представлені у вигляді рядків бітів, чисел або інших структур даних, які представляють рішення проблеми;

– функція пристосованості – вимірює ефективність окремих членів популяції у вирішенні певної задачі;

– схрещування – створює нових нащадків, поєднуючи гени обох батьків;

– мутація – вносить випадкові зміни в геном особини та допомагає досліджувати нові області потенційних рішень;

– відбір – залучає індивідів до наступного покоління виходячи з їхніх адаптаційних можливостей;

– елітарний відбір – передбачає, що лише найкращі особини з кожного покоління без змін переходять у наступне, зберігаючи найефективніші риси.

Генетичні алгоритми визначаються як ітеративний процес оптимізації, який проходить послідовні покоління. Цей процес завершується, коли певна умова зупинки, наприклад задана якість рішення або максимальна кількість поколінь, не буде досягнута [14].

Як і інші методи, генетичні алгоритми мають свої недоліки на переваги.

Вони здатні працювати з великими обсягами даних (кількість маршрутів, умови доставки тощо). Також, генетичні алгоритми є гнучкими, адже можуть легко модифікуватись та налаштовуватись для врахування різних обмежень (наприклад, час, трафік, вага вантажу) для знаходження близького до оптимального рішення [15].

Висока ефективність алгоритму забезпечується довшим часом виконання, в порівнянні з іншими методами оптимізації. Також, складним завданням є правильне налаштування та параметризація алгоритму, оскільки допущені помилки на цьому етапі можуть спричинити зациклення алгоритму на локальних мінімум, що не дасть можливості знайти оптимальне рішення [16].

1.1.3 Мурашиний алгоритм

Алгоритм оптимізації колонії мурах (мурашиний алгоритм оптимізації, ACO) є одним з ефективних поліноміальних методів для знаходження наближених розв’язків задачі комівояжера [17]. Цей алгоритм використовує модель поведінки мурашок – вони відзначають успішні маршрути великою кількістю феромонів [18].

У природі одна мурашка - примітивне створіння, яке не може аналізувати середовище, робити висновки та приймати рішення. Але завдяки своїй соціальності, мурахи є дуже розвиненими як вид, будують великі мурашники, розподіляють роботу та забезпечують себе усім необхідним для життя. Кожна мураха має інформацію тільки про місцеве оточення. Механізми пошуку найкоротшого шляху від мурашника до джерела їжі ґрунтуються на взаємодії мурашок між собою. Кожен раз, коли комаха проходить від мурашника до їжі і назад, вона залишає за собою слід феромонів. Інші мурахи, відчувши такі сліди, будуть прагнути до них, тобто слідувати за ними. Чим більше мурах проходить певним шляхом, тим привабливішим він стає для інших. При цьому, чим коротший маршрут до джерела їжі, тим менше часу потрібно мурашкам на його проходження - отже, тим швидше сліди стають помітними [19].

Для початку роботи з алгоритмом мурашиного колонії необхідно встановити значення параметрів: вага феромонів α, коефіцієнт близькості β, коефіцієнт випаровування феромонів ρ, кількість ітерацій N та початкове значення феромонів на шляхах D. На вхід алгоритм отримує матрицю обернених відстаней. Також потрібно визначити кількість агентів (мурах), яка дорівнює кількості точок у маршруті, тобто на кожну точку припадає один агент [20].

Спочатку кожним агентом обчислюється ймовірність вибору шляху до наступного пункту за такою формулою [21]:

$$P = \frac{\tau(r,u)^{\alpha} * \eta(r,u)^{\beta}}{\sum_k \tau(r,u)^{\alpha} * \eta(r,u)^{\beta}}, \quad (1.1)$$

де $\tau(r,u)$ – інтенсивність ферменту на шляху між вузлами r й u, $\eta(r,u)$ –зворотна відстань для шляху, α – вага ферменту, β– коефіцієнт видимості, k – кількість доступних вузлів із поточного.

Агенти проходять усі точки, формуючи маршрути. Після цього обчислюється довжина кожного маршруту – сума довжин усіх пройдених шляхів.

Обчислюється кількість ферамонів, яку кожен агент залишив на кожному шляху [21]:

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}, \quad (1.2)$$

де Q – константа, $L^k(t)$ – довжина маршруту, прокладеного мурахою k.

На кожній ітерації оновлюється кількість феромонів на шляхах, враховуючи випаровування феромонів. Розрахунки проводяться відповідно до формули [21]:

$$\Delta\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}(t), \quad (1.3)$$

де ρ – коефіцієнт випаровування ферамонів.

Якщо довжини всіх маршрутів рівні, алгоритм завершує роботу. Якщо довжини маршрутів не рівні, агенти формують нові маршрути, і процес повторюється. В кінці алгоритм формує оптимальний маршрут [22].

Варто зазначити переваги мурашиного алгоритму. Він має просто реалізацію та високу ефективність. Він може адаптуватись до вирішення різних задач оптимізації. Є можливість застосувати його для пошуку оптимальних маршрутів у реальному часі [20].

Недоліки, звичайно ж, також є. Алгоритм може застрягати в локальних максимумах, не знаходячи глобального оптимального рішення. Налаштування параметрів алгоритму, таких як швидкість випаровування феромонів, вага феромонів тощо, може бути нетривіальним завданням і вимагати експериментів. Для досягнення задовільного рішення може знадобитися велика кількість ітерацій, що збільшує час виконання алгоритму [21].

Отже, було розглянуто три методи планування маршрутів. Алгоритм Дейкстри простим та зрозуміли, однак не зможе працювати із великою кількість пунктів маршруту та не гарантує знаходження найкоротшого шляху. Генетичні алгоритми часто використовують для знаходження оптимального маршруту, однак він є складним у реалізації та налаштуванні. Алгоритм мурахи втілити у життя досить легко, він може працювати у реальному часі та дає наближене до оптимуму рішення. Зважаючи на такі результати мурашиний алгоритм було обрано як метод планування маршрутів доставлення.

1.2 Огляд вебреалізацій систем допомоги

Важкі обставини сьогодення змусили українців створити багато вебсистем допомоги, щоб мати можливість підтримати один одного. Є багато варіантів реалізацій застосунків із такою ідеєю.

«Платформа допомоги врятованим» – це онлайн-майданчик, що створений за сприяння Офісу віцепрем'єрки з питань європейської та євроатлантичної інтеграції, урядової уповноваженої з питань гендерної політики та Фонду ООН у галузі народонаселення в Україні. Знайти цей сервіс можна за посиланням https://www.help-platform.in.ua/ [23]. Цей вебсайт є центром для людей, які постраждали від війни, де пропонуються актуальні програми, ресурси і контактні дані основних служб підтримки. Сюди включені як державні послуги, так і партнерські ініціативи з громадськими та міжнародними організаціями. Наприклад, на рисунку 1.1 показано розділ «Гаманітарна допомога», де зібрані сервіси, куди людині потрібно звернутись якщо їй необхідні певні матеріальні речі. На рисунку 1.2 перераховані сайти, де можна знайти психологічну підтримку, що в час війни дуже важливо.

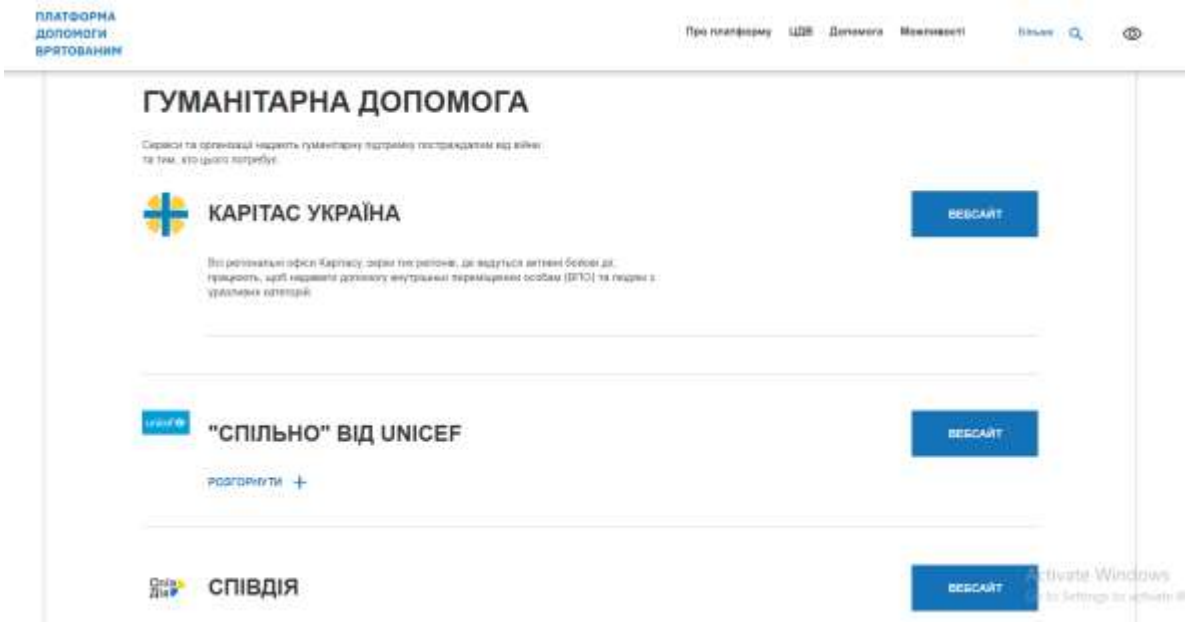


Рисунок 1.1 – Зібраний список організацій, що надають гуманітарну допомогу за версією платформа допомоги врятованим [23]

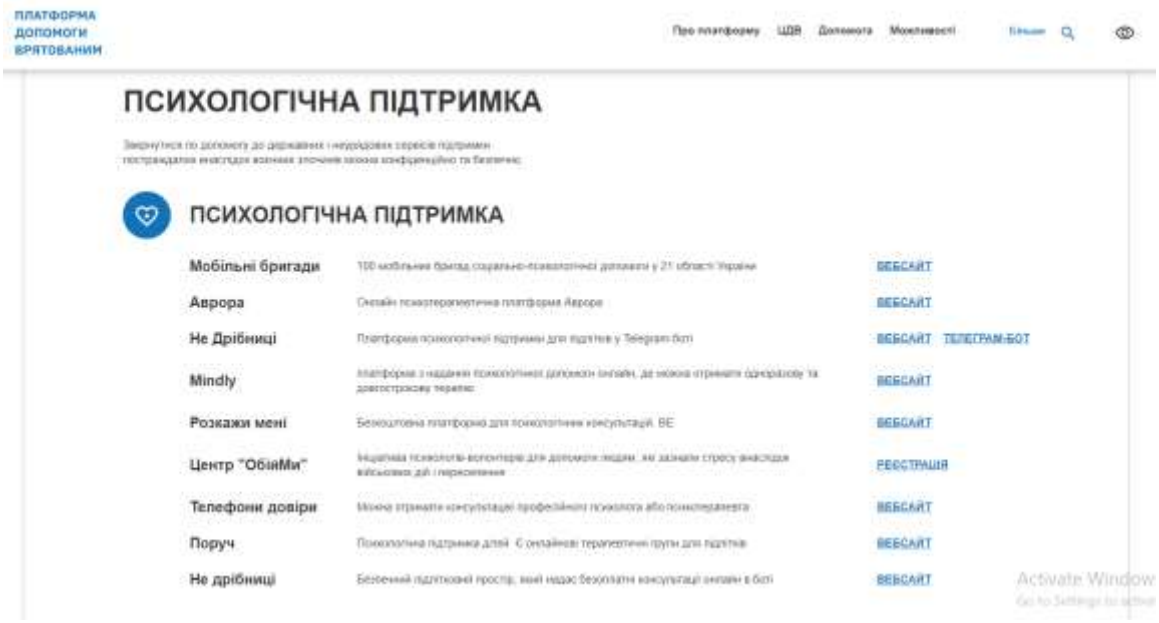


Рисунок 1.2 – Зібраний список організацій, що надають психологічну допомогу за версією платформа допомоги врятованим [23]

Тобто, вебзастосунок «Платформа допомоги врятованим» виконує роль навігатора, що направляє людей до організації, яка найбільше відповідає їх потребам.

Також пропонується багато сайтів благодійних організацій та фондів. Одна з найбільших мереж благодійних організацій світу Caritas існує і в Україні під назвою «Карітас України» (рисунок 1.3). Відвідати їх сайт можна за посиланням https://caritas.ua/about/ [24].

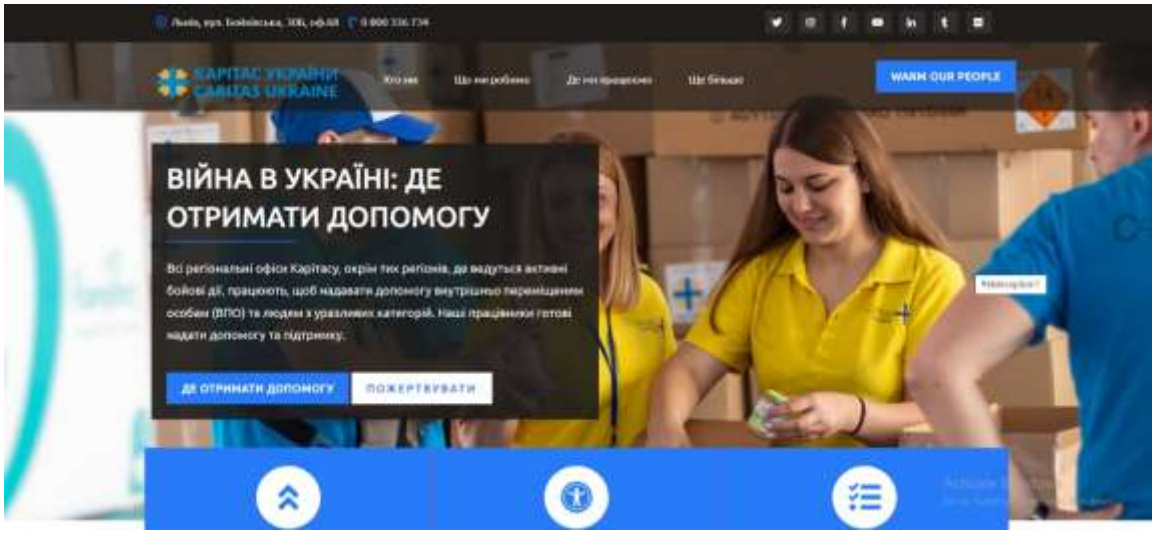


Рисунок 1.3 – Головна сторінка сайту «Карітас України» [24]

Сайт виконує здебільшого інформативну функцію, тобто надає інформацію про вид діяльності організації, про місцезнаходження їх пунктів допомоги (рисунок 1.4) тощо.

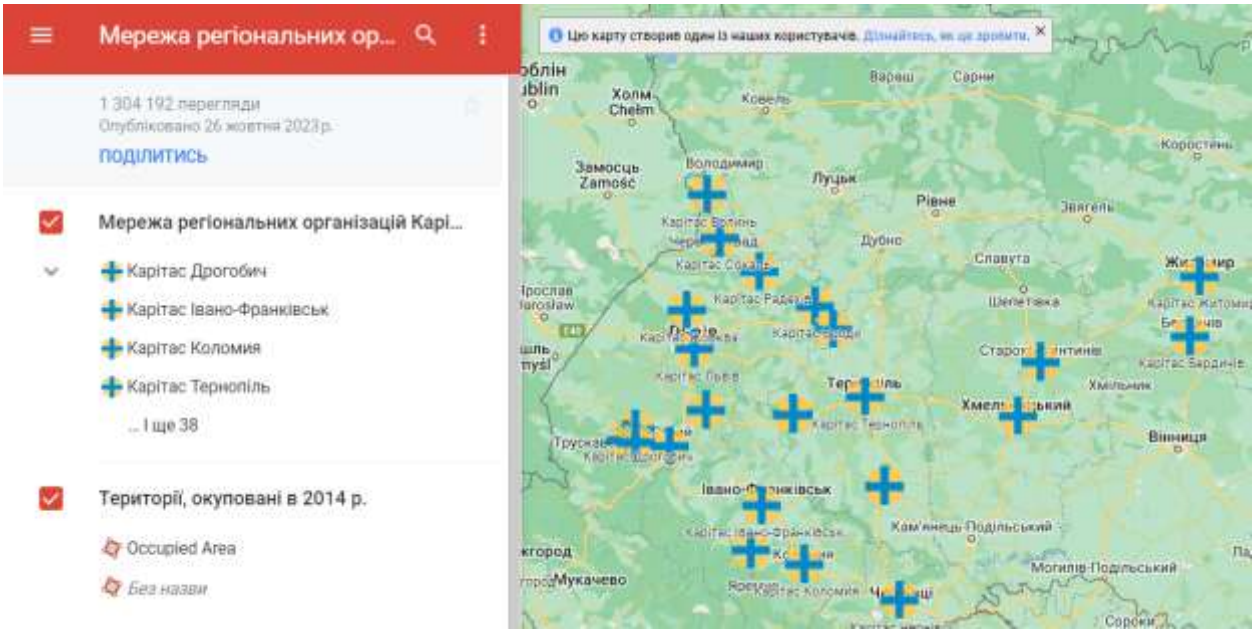


Рисунок 1.4 – Мапа місцезнаходження центрів допомоги «Карітас України» [24]

Для відвідувачів є можливість пожертвувати кошти, продивитися вакансії, стати членом організації. Також, чудова ідея, що кожен бажаючий може переглянути фото-звіт роботи фонду (рисунок 1.5).

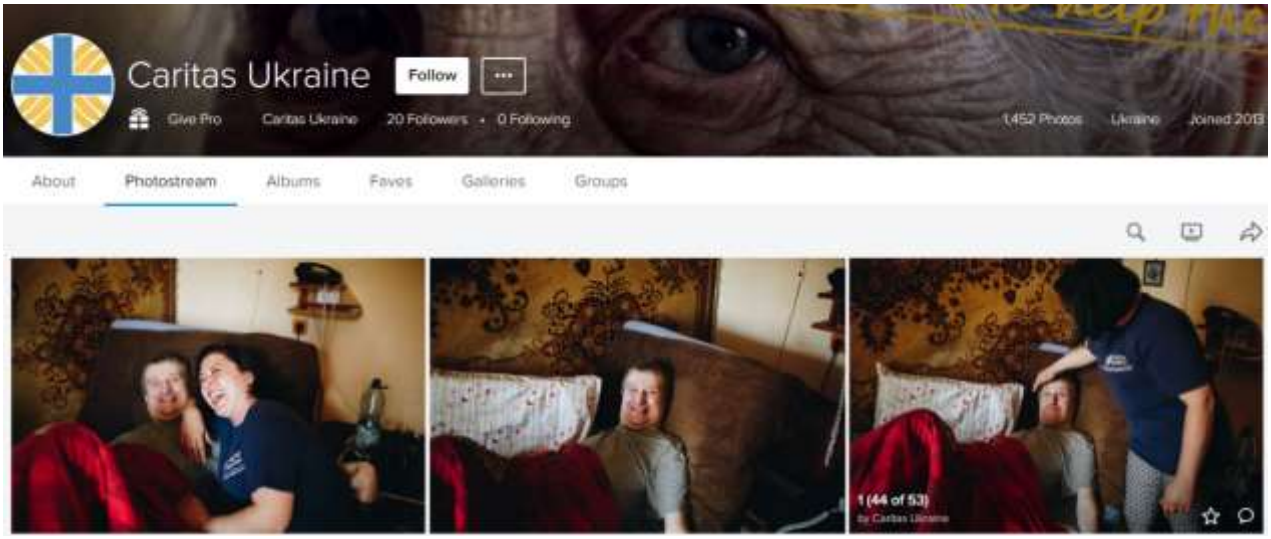


Рисунок 1.5 – Фото роботи членів команди «Карітас україни» [22]

Також було розглянуто вебсайт «ЄДопомога» за посиланням https://social.edopomoga.gov.ua/uk/ [25]. Він дає можливість людям залишати запити про свої потреби, а також дає можливість допомогти (не лише грошима).

На рисунку 1.6 показано сторінку, де зареєстрований користувач може створити заявку на допомогу. Він може обирати категорії, певні позиції та кількість потреб. Наприклад, у категорії «Їжа» обрати продукт «Молоко» та необхідну кількість.

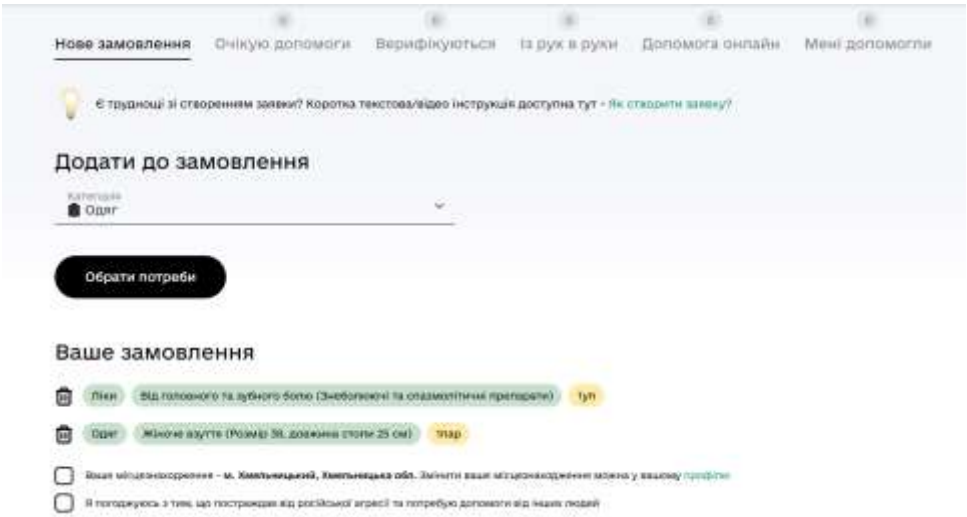


Рисунок 1.6 – Сторінка створення заявки [25]

Після верифікації заявки, вона розміщується на сайті, щоб кожен бажаючий міг її виконати зручним для нього способом (рисунок 1.7).

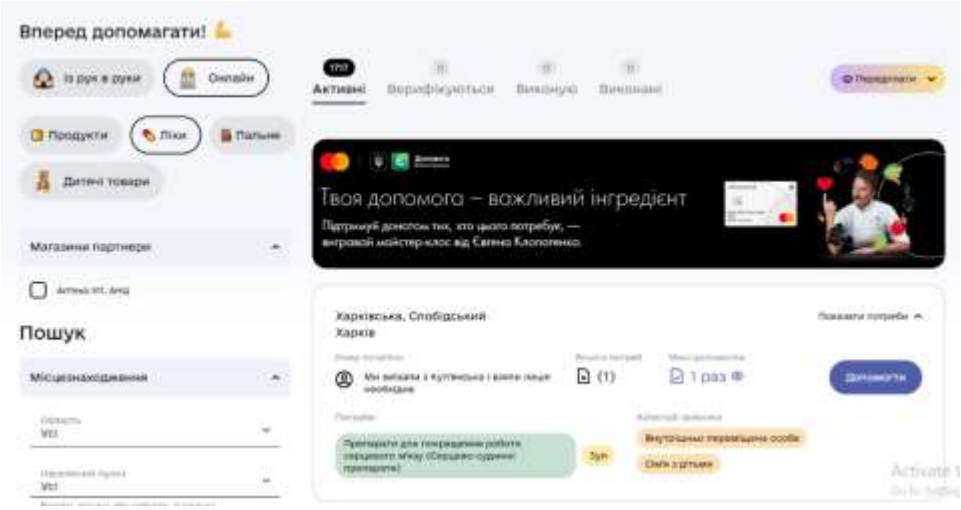


Рисунок 1.7 – Сторінка для бажаючих допомогти [25]

Людина може частково чи повністю оплатити потреби запиту, як показано на рисунку 1.8. Тоді ці кошти будуть передані магазинам-партнерам, де людина зможе забрати необхідін товари.

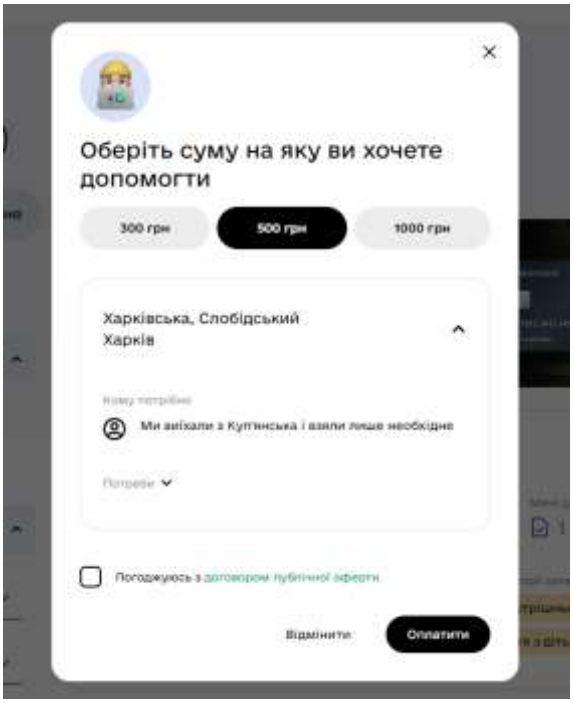


Рисунок 1.8 – Діалогове вікно оплати заявки [25]

Також є можливість допомогти матеріально, ця опція на сайті називається «Із рук в руки». Тобто ви обираєте окремі потреби із заявки (рисунок 1.9), купуєте та передаєте товари людині зручним для вас способом.

Рисунок 1.9 – Вікно вибору товарів для задовільнення потреби в них [25]

«ЄДопомога» – це зручна платформа для взаємодопомоги, що не потребує залучення людей для доставлення допомоги людям.

Отже, було оглянуто популярні вебсистеми допомоги людям. На сайтах «Карітас України» та «Платформа допомоги врятованим» не вистачає функціоналу для можливості залишати свої запити прямо на сайті, люди можуть зробити це лише в їхніх центрах допомоги. «ЄДопомога» чудово втілила ідея запитів, однак недоліком є те, що люди з обмеженими можливостями не зможуть прийти до магазинів-партнерів, щоб отримати необхідні речі. Тому потрібна вебсистема, яка буде передбачати безкоштовне доставлення допомоги за адресами та планувати для цього маршрути з алгоритмом мурахи.

1.3 Мета, задачі та вимоги до реалізації інформаційної системи

Метою кваліфікаційної роботи бакалавра є підвищення ефективності доставлення допомоги у вебсистемах для малозабезпечених людей. Для досягнення поставленої мети необхідно виконати наступні завдання.

1. Огляд методів планування маршрутів та існуючих вебреалізацій систем допомоги.

2. Дослідити використання алгоритму мурахи в задачах планування маршрутів та його особливості для вирішення проблеми допомоги малозабезпеченим.

3. Розробити метод доставлення (планування маршруту) за мурашиним алгоритмом, що буде формувати та віддавати як результат оптимальний маршрут для почергового відвідування кожного пункту призначення.

4. Створити вебсистему допомоги малозабезпеченим для користувачів із різними ролями (адміністратор та споживач), що буде виконувати такі функції:

- авторизація користувача;
- редагування персональних даних користувача;
- перегляд історії своїх заявок користувачем;
- оформлення заявки про допомогу;

- можливість анонімної фінансової допомоги фонду;
- (для адміністратора) фільтрація адрес доставки за їхніми станом готовності до відправки;
- (для адміністратора) можливість відбору заявок для створення нового маршруту;
- (для адміністратора) перегляд історії побудованих маршрутів.

5. Протестувати метод доставлення допомоги за мурашиним алгоритмом та вебсистему, що використовує його, на коректність роботи усіх складових.

6. Оцінити покращення ефективності доставлення допомоги за запропонованим методом.

Розділ 2 Метод доставлення за мурашиним алгоритмом

2.1 Мурашиний алгоритм у задачах планування маршрутів

На рівні підсвідомості люди завжди планували свої маршрути, аби заощадити свій час та сили. Застосовувалися паперові мапи, потім з’явились навігаційні системи GPS, з допомогою яких цю задачу можна було вирішити більш точно. Сучасність – це ера розвитку штучного інтелекту, який вже пронизує багато сфер діяльності людини, і планування маршрутів не виключення [2].

Є різні алгоритми планування маршрутів, зокрема у розділі 1.1 були розглянуті: алгоритм Дейкстри, генетичні алгоритми та мурашиний алгоритм. Було визначено що мурашиний алгоритм буде найкращим рішення для вирішення задачі планування маршрутів.

Мурашиний алгоритм (також відомий як ACO – Ant Colony Optimization) створений на основі природної поведінки мурах, які шукають найкоротший шлях від колонії до джерела їжі. Мурахи залишають феромонний слід на своєму шляху, який служить “путівником” для інших мурах. Чим більше мурах проходить по певному шляху, тим більше феромону вони залишають, і тим привабливішим стає цей шлях для наступних мурах [26].

Під час виконання КРБ було детально розглянуто переваги використання алгоритму мурахи для планування маршрутів.

1. Ефективність у роботі зі складними умовами. Алгоритм дозволяє знаходити оптимальні або найближчі до оптимальних рішень у складних умовах, коли інші методи можуть бути менш ефективними. Тобто, при плануванні маршрутів потрібно враховувати безліч факторів (засіб пересування, трафік на дорогах, погодні умови, стан доріг, кількість зупинок тощо) – це і є складні умови для методів.

2. Паралельна робота. ACO добре підходить для паралельних обчислень, що дозволяє прискорити процес пошуку оптимального маршруту. Наприклад, у великих містах з великою кількістю точок доставки, паралельне виконання

алгоритму може допомогти збільшити швидкість знаходження оптимального маршруту для кур'єрської служби.

3. Адаптивність до змін. Алгоритм швидко адаптується до змінних умов та обставин. Наприклад, у випадку зміни маршруту через дорожні ремонти або інші перешкоди, АСО може швидко змінити маршрут доставки, щоб забезпечити ефективну доставку вантажу.

4. Простота реалізації та налаштування. Це дозволяє швидко застосовувати його для рішення реальних задач. Наприклад, у компанії, що займається доставкою, можна використовувати мурашиний алгоритм для планування оптимальних маршрутів для кур'єрів без значних витрат на розробку складних алгоритмів.

5. Додатковий фактор безпеки. Використання феромонів у АСО може забезпечити додатковий фактор безпеки. Наприклад, у задачі планування маршруту для транспортних засобів, феромони можуть показувати шляхи, які були безпечно пройдені, допомагаючи уникнути ризиків або перешкод на шляху.

6. Велика кількість готових рішень. АСО – це вже не новий алгоритм, однак надійний, саме тому на просторах інтернету можна знайти вже готові рішення, які можна використати у своїй програмі, не витрачаючи додаткового часу на написання алгоритму з нуля.

Основними недоліками використання алгоритму мурахи для планування маршрутів є [26]: довгий час виконання через велику кількість ітерацій, якщо на маршруті буде багато зупинок та потреба у налаштуванні параметрів(швидкість випаровування феромонів, інтенсивність випаровування, кількість мурах-агентів тощо).

Зважаючи на те, що недоліків значно менше ніж переваг мурашиного алгоритму для задачі планування маршрутів, АСО було обрано для розробки методу доставлення допомоги малозабезпеченим.

2.2 Метод доставлення за мурашиним алгоритмом

Малозабезпечені сім’ї – це ті, які мають дохід на одну особу нижчий ніж прожитковий мінімум в країні [27]. В Україні таких сімей досить багато, і на це є дійсно вагомі причини, зокрема війна, яка забрала багато годувальників родин. Тому держава допомагає таким сім’ям [27], але цього, зазвичай, замало.

Для допомоги малозабезпеченим існує не багато фондів, а особливо з передбаченою доставкою додому. Оскільки людям з обмеженими можливостями та матусям і маленькими дітьми складніше поїхати в магазин та придбати те, що потрібно якщо прислали грошову допомогу чи забрати з відділення пошти посилку (вона може бути важкою, великою, чи в іншому населеному пункті), то доставка допомоги додому потребуючого є одним з найважливіших та найскладніших завдань.

Якщо створювати локальні фонди, що будуть працювати лише в одному місті чи максимум області, то запланувати правильний шлях можна навіть на папері з ручкою. Однак, якщо фонд діє по всій Україні, тоді це вимагає набагато складнішого алгоритму, оскільки потрібно врахувати і відстань між пунктами призначення (домівки сімей), і стан доріг, і рівень небезпечності (залежить від кількості вибухів, швидкості прильоту ракети), аби не наражати водіїв на небезпеку також, і не забувати про спеціальні умови перевезення продуктів харчування чи медичних препаратів. Людина може наближено скласти найкращий шлях, однак він буде не точним та це займе багато часу, який можна було б використати на щось важливіше. Тому тут ідеально допоможе алгоритм мурахи для планування маршруту доставлення допомоги малозабезпеченим сім’ям, адже він швидко виконає поставлену задачу та розвантажить працівників чи волонтерів фонду від цього обов’язку.

На рисунку 2.1 детально та поетапно описано особливості використання алгоритму мурахи для доставлення допомоги малозабезпеченим сім’ям.

Вхідними даними для алгоритму є дані маршруту: множина усіх пунктів призначення, початковий пункт (пункт відправки допомоги), готовність

замовлень для відправки та максимальна кількість зупинок у пунктах призначення на один маршрут. Інший тип вхідних даних – параметри налаштування алгоритму мурахи, що включають: вагу сліду ферамонів, коефіцієнт близькості, коефіцієнт випаровування ферамонів, початкове значення ферамонів на шляхах та кількість ітерацій.

Першим етапом алгоритму мурахи є підготовка подорожі. Спочатку формується множина пунктів призначення A для конкретного маршруту. Для цього із множини усіх можливих замовлень малозабезпечених спочатку відбираються ті, які вже готові до відправки, тобто повністю укомплектовані всіма необхідними речами, та враховуючи дистанцію між домівками сімей відбирається кількість K або менше. Після цього визначається кількість мурах-агентів, що рівна кількості елементів множини A, відбується їх розподіл по зупинках: по одному агенту на будинок сім'ї. Формується матриця відстаней та обернених шляхів, та встановлюється початкова кількість ферамонів на шляхах.

Наступний етап – моделювання обходу шляхів агентами. Спочатку обчислюється ймовірність вибору кожним агентом шляху до наступного пункту маршруту. Після цього агенти проходять усі пункти маршруту, підраховуючи довжину пройдених шляхів. Крім того, визначається кількість феромонів, яку кожен агент залишив на пройденому маршруті. В кінці даного етапу відбувається поновлення феромонів на шляхах. Щоб зберегти оптимальність маршрутів для майбутніх агентів враховується також інтенсивність випаровування ферамонів.

Далі визначається збіжність маршрутів. Якщо довжини усіх маршрутів, які мурахи рахували, не дорівнюють одному шляху та кількість ітерацій менша за N, то повторюється другий етап, тобто розпочинається нова ітерація. Якщо ж довжини усіх побудованих маршрутів збігаються або кількість ітерацій досягла числа N, то алгоритм переходить до наступного кроку: формування оптимального шляху.

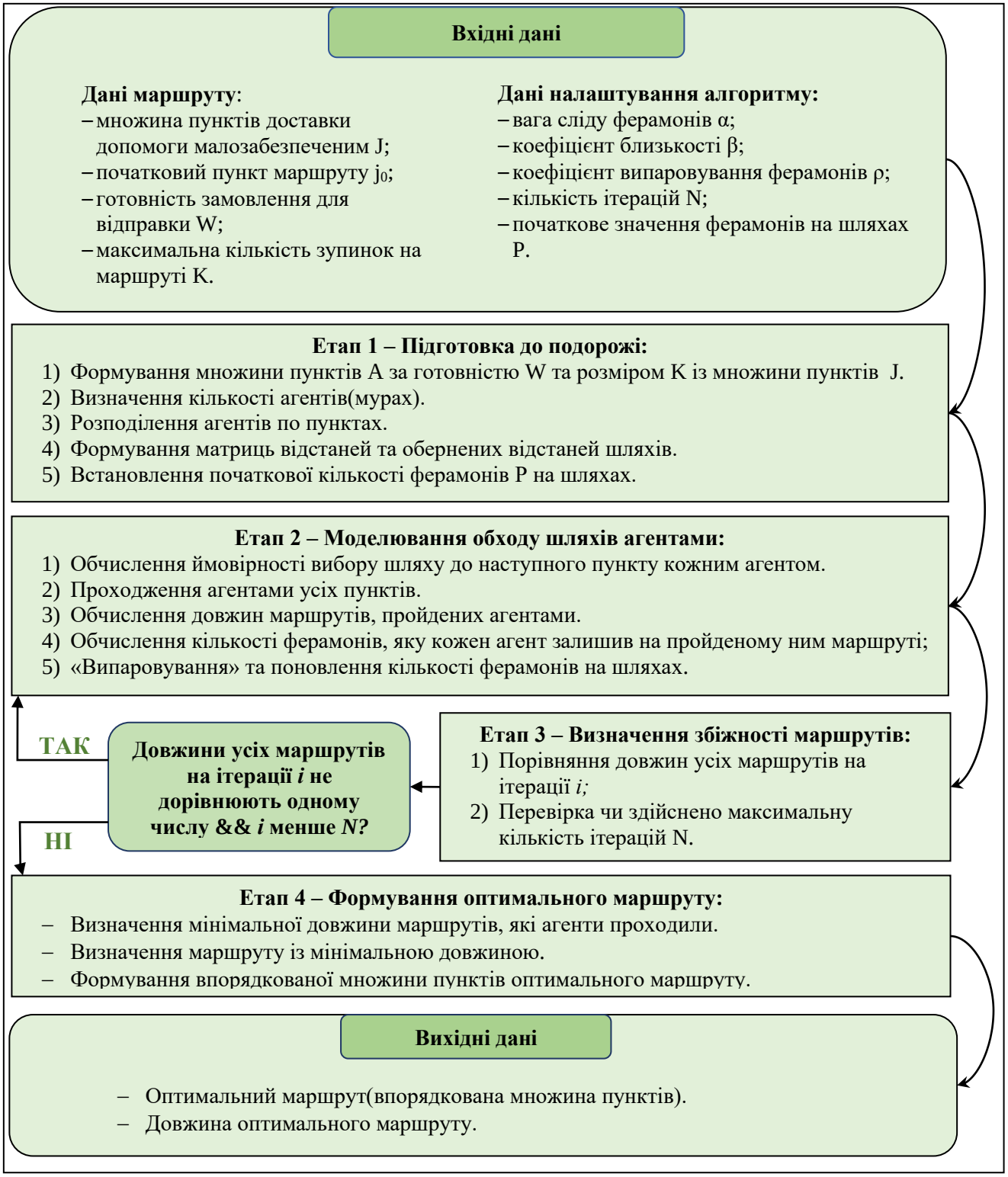


Рисунок 2.1 – Схема методу доставлення за мурашиним алгоритмом

Спочатку відбувається пошук мінімальної довжини маршруту, який мураха проклала на будь-якій ітерації. Після цього визначається сам маршрут із мінімальною довжиною, і в результаті формується впорядкована множина

домівок малозабезпечених сімей, що очікують на допомогу. Саме ця множина і буде оптимальним маршрутом, що водій повинен здійснити.

Відповідно, вихідними даними алгоритму мурахи для доставлення допомоги малозабезпеченим буде побудований оптимальний маршрут та його довжина.

2.3 Особливості реалізації мурашиних алгоритмів у бібліотеках

На просторах мережі Інтернет існує багато статей, що навчають правильно будувати алгоритм мурахи та різні його підвиди. Однак, існує і досить велика кількість готових рішень, які потрібно лише додати у код програми аби вона запрацювала, як того очікує розробник. Це дає можливість заощадити час на створення алгоритму та його відлагодження.

Одним з таких рішень є бібліотека AntColony [28], що написано мовою програмування C#. На рисунку 2.2 зображено діаграму класів цього проекту. Першою особливістю є те, що бібліотека містить у собі два класи, що реалізовують інтерфейс IAnt: клас мурахи та елітної мурахи. Це ознака покращеного алгоритму, адже елітні мурашки можуть залишати за собою більше ферамонів, частіше обирають шлях з вищою концентрацією ферамонів, що дозволяє їм швидше знайти найкоротший маршрут.

Клас графу, що реалізовує інтерфейс IGragh зберігає та обробляє дані про матрицю відстаней між пунктами призначення. За допомогою класу Config можна налаштувати параметри алгоритму мурахи для кращого пошуку оптимального маршруту. Великою перевагою бібліотеки є те що, задати налаштування методу та матрицю відстаней можна файлами, які будуть правильно знайдені та серіалізовані відповідними класами.

Основними класами, з якими має працювати застосунок є AntColonyAlgorithm, що реалізовує інтерфейс IAlgorithm, та Result. Клас AntColonyAlgorithm виконує всю роботу алгоритму, який можна запустити методами Solve() та TrySolve() та отримати результат відповідного типу. Клас

Result містить дані не лише про найкращий складений маршрут та його довжину, а також скільки було витрачено ітерацій та часу для цього пошуку.

Рисунок 2.2 – Діаграма класів бібліотеки AntColony [28]

Отже, для методу доставлення допомоги малозабезпеченим сім’ям за алгоритмом мурахи, алгоритм буде виконуватись саме бібліотекою AntColony, яку було обрано на цю роль, зважаючи на всі її особливості та переваги. Було побудовано схему, що показує як метод доставлення використовує бібліотеку (рисунок 2.3).

Спочатку надходять вхідні дані про пункти призначення, готовність замовлень для них та кількість пунктів на маршрут. Далі ці дані обробляються, щоб правильно їх до алгоритму мурахи. Формується множина пунктів призначення для конкретного маршруту. За допомогою Google Maps Distance Matrix API із вибірки формується матриця відстаней, на основі якої потім

створюється об’єкт типу Graph. Заздалегідь визначені та записані в конфігураційному файлі налаштувань алгоритму мурахи передаються до об’єкту типу Config. Також необхідно ініціалізувати екземпляр класу Random, що допомагає урізноманітнити вибір наступної зупинки мурашкою. В кінці створюється об’єкт головного класу бібліотеки AntColonyAlgorithm, до конструктора якого передаються усі творені раніше екземпляри типів Graph, Config та Random.

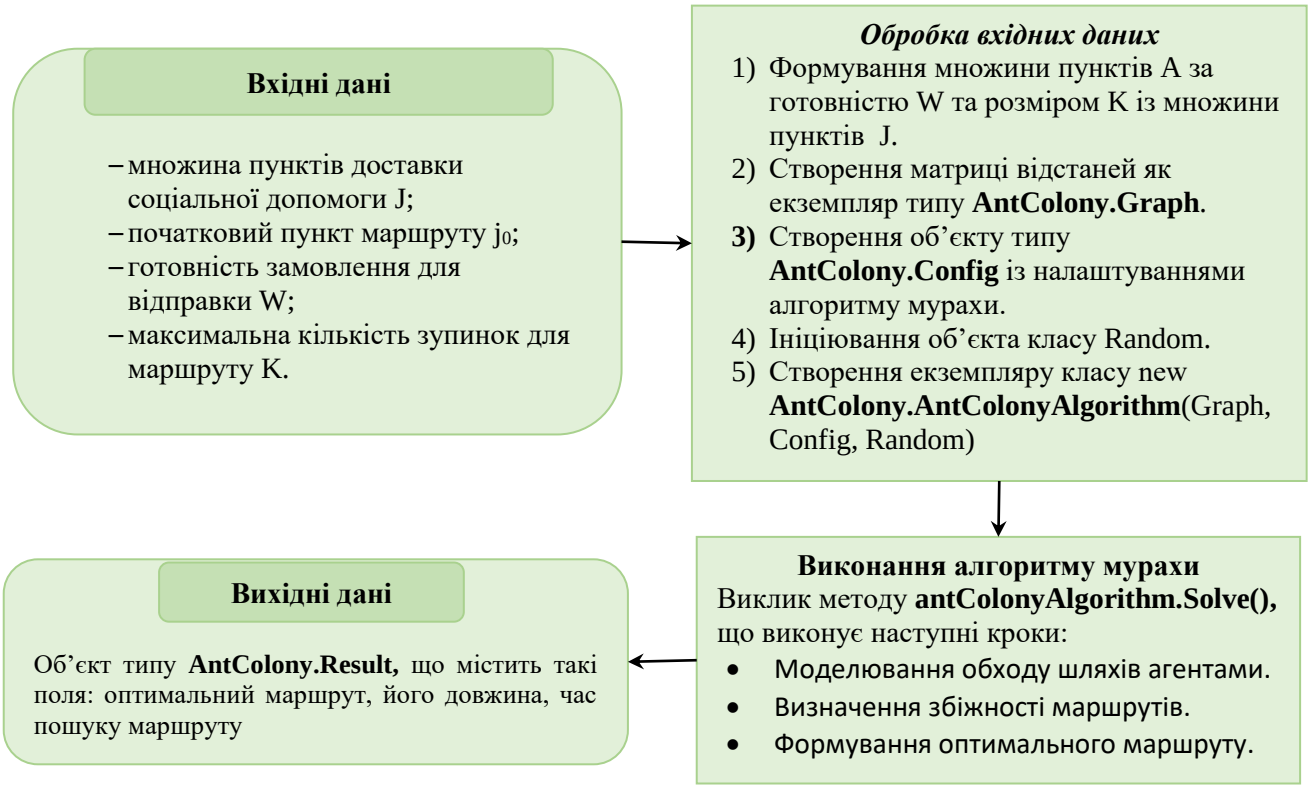


Рисунок 2.3 – Схема методу доставлення допомоги із використанням бібліотеки AntConoly

Найскладнішу роботу, а саме виконання алгоритму мурахи, бібліотека здійснює самостійно. Для цього потрібно лише викликати метод об’єкту Solve(). Тоді AntColony моделює обходи шляхів мурахами, визначає коли маршрути збігаються, формує оптимальний маршрут, та повертає результат. Вихідними даними і є результат виконання методу Solve() типу Result, що містить маршрут, його довжину та час виконання алгоритму для його пошуку.

Отже, порівнюючи рисунок 2.1 та 2.3, легко помітити наскільки сильно спрощено завдання створення методу доставлення допомоги малозабезпеченим, завдяки бібліотеці AntColony.

2.4 Проектування вебсистеми допомоги малозабезпеченим

Для безпеки та надійності вебсистеми допомоги малозабезпеченим було вирішено створити роздільний проект, що складається з двох частин: серверної частини та клієнтської частини. З'єднання за допомогою HTTP запитів дає можливість обом частинам працювати як разом, так і з іншими компонентами.

2.4.1 Серверна частина

Для розробки серверної частини було обрано мову програмування C#, платформу .NET та фреймворк ASP.NET Core Web API, що є потужним інструментом для створення вебсервісів на основі HTTP протоколу та за REST принципами [29]. Фреймворк кросплатформний, що дає можливість розгортати проект на різних операційних системах, а також має багато вбудованих інструментів (наприклад Dependency Injection), що скорочує час написання програми та її масштабування.

Компоненти сервера мають наступні відповідальності. Модуль «Entities» містить усі необхідні типи для реалізації програми. За допомогою модуля «Repositories» сервер отримує та передає інформацію до бази даних системи. У модулі описано як правильно зберігати, витягувати, сортувати, оновлювати дані в бази для кожного окремого запиту із клієнтської частини.

Далі дані обробляються у модулі «Services», що містить усю бізнес-логіку застосунку. Саме у цьому шарі використовується бібліотека AntColony для знаходження оптимального шляху, а також описано формування справжньої матриці відстаней з реальними довжинами шляхів між зупинками.

На рисунку 2.4 зображено усі компоненти серверної частини вебсистеми, що має багаторівневу архітектуру.

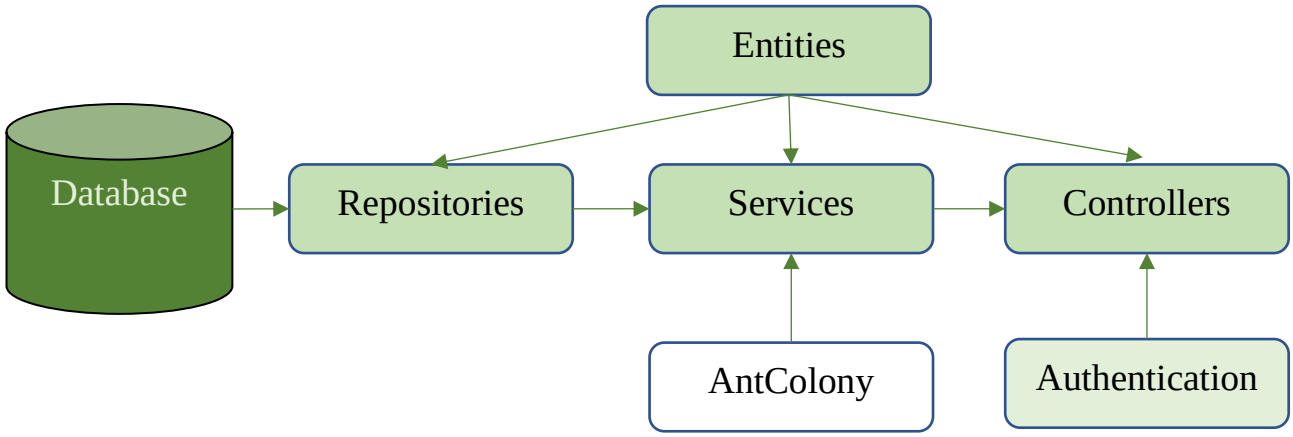


Рисунок 2.4 – Схема архітектури серверної частини вебсистеми

Компонент «Controllers» містить контролери, які визначають url-шляхи запитів з їхніми параметрами та типами результатів. Через контролери отримуються запити, що клієнтська частина надсилає, тому він використовує ще один окремий модуль «Authentication». Він відповідає за реєстрацію та логін користувача, а також передавання даних про користувача за допомогою JWT токена.

Json Web Token (JWT) – це сучасний метод для автентифікації (розпізнавання) користувача [30], також ним можна передати не персональну інформацію про користувача у вигляді вимог. Потрібно бути дуже обережним з тим, що записується до токену, оскільки токени не є зашифрованими і витягнути вимоги, прописані в ньому, надзвичайно легко. Однак JWT – це прекрасний інструмент для того, щоб запобігти логіну користувача щоразу як він відкриває вкладку вебсайту у браузері. Для такої задачі використовується два токени: access та refresh. На рисунку 2.5 детально зображено діаграму JWT автентифікації.

Спершу клієнтська частина надсилає запиту на логіну користувача, якщо пароль та електронна пошта коректні, тоді сервер генерує пару access і refresh токенів, в які записується унікальний ключ користувача (id), та віддає токени і їх термін дії у відповіді на запит. Термін дії access токену є досить коротким 10-30

хв, а ось refresh токен можу бути дійсним цілий тиждень чи навіть більше. Це робиться для безпеки від хакерів, адже якщо вкрадуть access токен, то скоро він працювати не буде, а для того, щоб отримати новий потрібно або увійти знову за допомогою паролю, або мати refresh token для поновлення.

Після того як клієнтська частина отримала токени, їй необхідно зберегти їх, але у безпечному місці. Щоразу як користувач надсилає запит до сервера, у хедері Authorization потрібно надсилати access токен – так сервер буде знати, який саме користувач надіслав запит, і які дані йому потрібно віддати. Кожного запиту сервер перевіряє токен на валідність, і якщо з ним щось не так, або закінчився час дії, тоді поверне помилку.

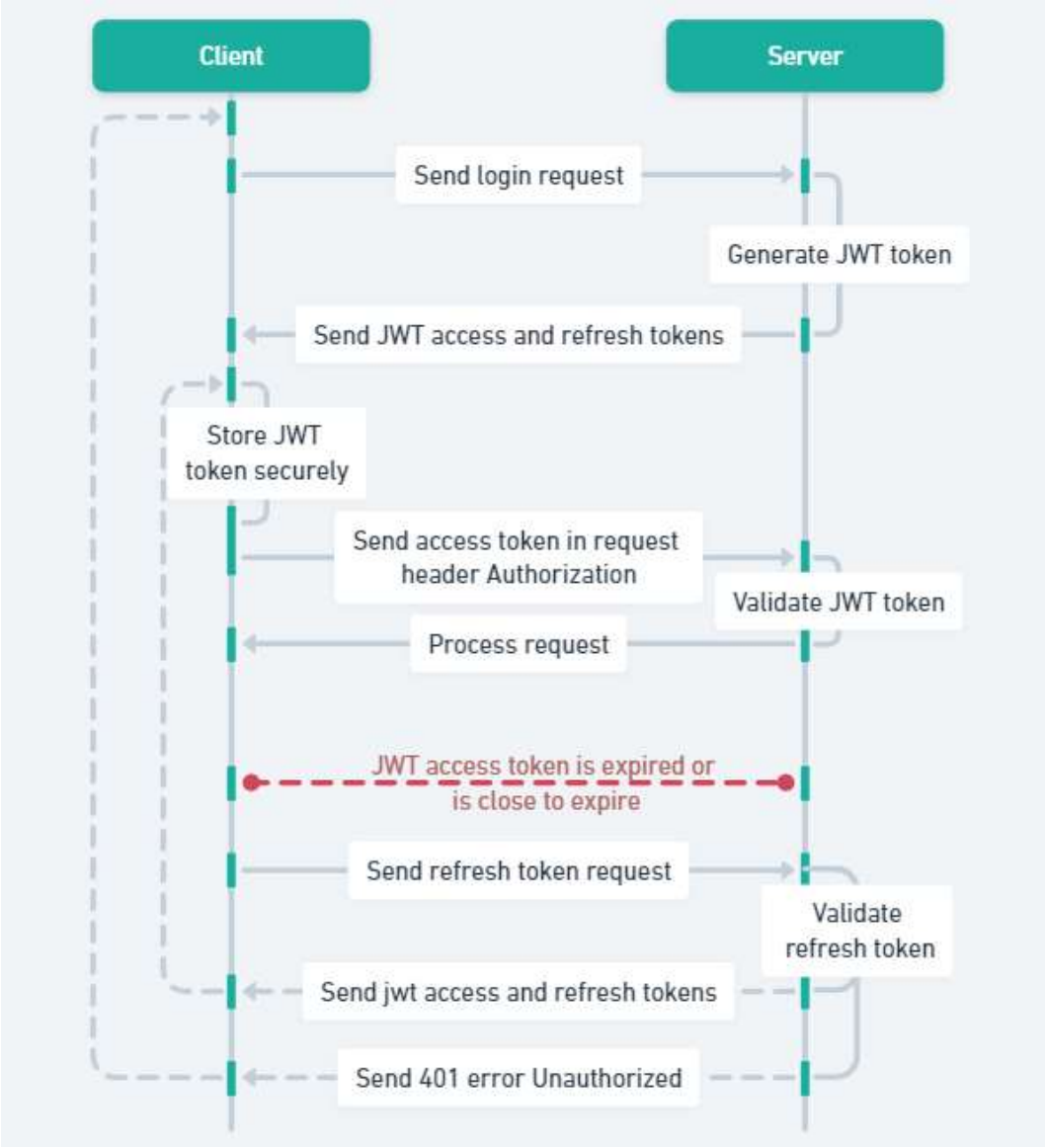


Рисунок 2.5 – Діаграма послідовності JWT автентифікації

Після помилки про access токен, клієнтська частина мусить надіслати запит на поновлення пари токенів, для цього в параметри необхідного запиту потрібно вставити refresh токен. Якщо операція пройшла успішно, то у відповідь прийде нова пара токенів, яку необхідно також зберегти і використовувати з подальшими запитами. Якщо ж refresh токен не пройшов валідацію, тоді потрібно заходити у свій кабінет за допомогою електронної пошти та пароля.

2.4.2 Клієнтська частина

Для створення клієнтської частини було обрано та об’єднано дві технології: React та Ionic.

React JS – це JavaScript бібліотека для розробки інтерфейсів користувача, створена компанією Facebook. Вона дозволяє ефективно розробляти високопродуктивні та масштабовані вебзастосунки. Однією з ключових концепцій у React JS є компоненти – незалежні блоки коду, які відповідають за відтворення певних частин користувацького інтерфейсу [31].

Ionic – це відкрите програмне забезпечення для розробки гібридних мобільних додатків [32]. Воно базується на вебтехнологіях, таких як HTML, CSS і JavaScript, і дозволяє створювати додатки, які працюють на різних платформах, таких як iOS і Android. Ionic надає набір компонентів і інструментів, що полегшують розробку і забезпечують однорідний зовнішній вигляд для додатків на різних платформах.

Одним з найважливіших завдань створення вебсистеми є створення правильного інтерфейсу. У процесі проектування користувацького інтерфейсу першорядне значення має розгляд різних елементів, які сприятимуть чіткому розумінню продуктів, пристосованих для конкретної аудиторії. Інтерфейс був розроблений з чіткою метою відобразити цілі вебсайту, перш за все, щоб спонукати людей подавати свої запити щодо своїх потреб та сприяти створенню організації.

Створюючи відмінний користувальницький інтерфейс, ми прокладаємо шлях для більш плавної взаємодії користувача з сайтом. Основна роль вебінтерфейсу виходить за рамки просто представляючи привабливий вигляд; він також прагне запропонувати приємну подорож користувача. При першій зустрічі з компонентами інтерфейсу, користувач повинен мати можливість легко переміщатися і оптимально використовувати всі необхідні інтерактивні інструменти. Це не тільки залишає позитивний вплив на користувачів, але і допомагає їм у досягненні своїх цілей.

Користувачами вебсайту можуть бути як працівники (адміністратори) певного фонду допомоги малозабезпеченим, так і люди, які потребують допомоги. Відповідно, сайт відображає різні сторінки та дає можливість виконувати різні операції в залежності від ролі користувача: “User” або “Administrator”.

На рисунку 2.6 зображена діаграма варіантів використання для адміністратора. Розглянуто сценарії прецедентів для адміністратора.

Назва прецеденту: логін.

Діюча особа: адміністратор.

Мета: увійти до вебсистеми як адміністратор.

Передумови: адміністратор відкрив в браузері вебсайт.

Послідовність:

1. Адміністратор вводить у відповідні поля адресу електронної пошти та пароль.

2. Система перевіряє символи у полях, якщо вони пройшли перевірку, то відправляє запит для логіну на сервер. Якщо одне із значень полів не коректне, тоді система показує адміністраторові помилку.

3. Сервер перевіряє адресу користувача, якщо такої в базі немає: повертається помилка. Якщо адміністратор є в базі, тоді звіряється введений пароль із хешем паролю а базі, якщо значення різні – видається помилка.

4. Здійснивши перевірки даних облікового запису сервер генерує JWT токени та повертає у відповіді на запит.

5. Клієнтська частина зберігає токени та переводить адміністратора на сторінку із таблицею зроблених замовлень.

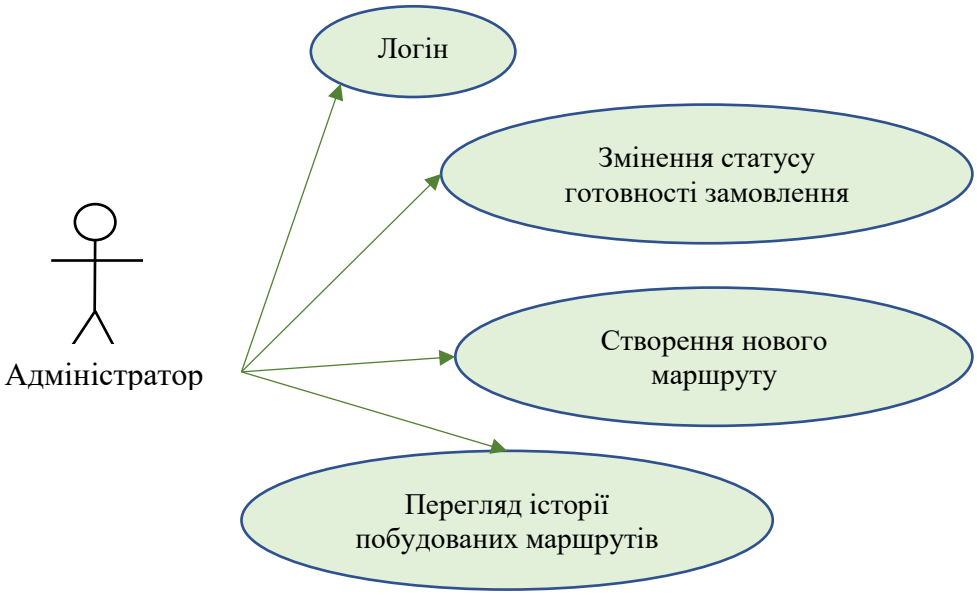


Рисунок 2.6 – Діаграма варіантів використання для адміністратора

Назва прецеденту: змінення статусу готовності замовлення.

Діюча особа: адміністратор.

Мета: позначити/забрати позначку готовності замовлення до відправлення.

Передумови: адміністратор відкрив в браузері вебсайт та увійшов до системи.

Послідовність:

1. Адміністратор відкриває сторінку із таблицею зроблених замовлень.

2. Адміністратор вводить унікальний ключ замовлення в поле пошуку.

3. Система відправляє запит на отримання замовлення із зазначеним id та виводить результат на вебсторінці.

4. Адміністратор ставить або забирає прапорець готовності для стовпця «Done» у відповідному рядку таблиці.

Назва прецеденту: створення нового маршруту.

Діюча особа: адміністратор.

Мета: сформувати новий оптимальний маршрут для доставки допомоги малозабезпеченим.

Передумови: адміністратор відкрив в браузері вебсайт та увійшов до системи.

Послідовність:

1. Адміністратор відкриває сторінку із таблицею побудованих маршрутів.
2. Адміністратор натискає кнопку «+» для створення нового маршруту .
3. Система відриває сторінку із списком готових до відправлення замовлень.
4. Адміністратор ставить прапорці навпроти тих замовлень, які потрібно відправити.
5. Система відправляє запит із вибіркою пунктів на сервер. Сервер за допомогою методу алгоритму знаходить найкращий маршрут, зберігає його в базі та відправляє у відповіді на запит. Система виводить впорядковану вибірку пунктів маршруту.

На рисунку 2.7 детально розглянуто функції вебсайту для користувача із роллю «User». Перш за все, аби виконати будь-яку із зображених функцій необхідно бути авторизованим до систем, тобто або пройти реєстрацію або логін.

Після цього звичайний користувач потрапляє на головну сторінку сайту, де можна почитати трішки інформації про організацію, що володіє фондом, а також перейти із неї на сторінку переказу коштів або створення заявки, натиснувши відповідні кнопки.

Є можливість перегляду та редагування профілю користувача. На цій сторінці відображається історія створених та отриманих замовлень. Користувач може змінити персональні дані, такі як ім’я, прізвище, вік тощо. Також користувач може змінити свій пароль до системи на новий, або взагалі видалити акаунт.

Варто зазначити, що звичайний користувач не зможе попасти на сторінки, що призначені для адміністраторів, оскільки для цього потрібен JWT access токен із роллю «Admin». Адміністратора можна лише створити у базі даних мануально,

тобто люди, що проходять реєстрацію через сайт – всі матимуть виключно роль «User».

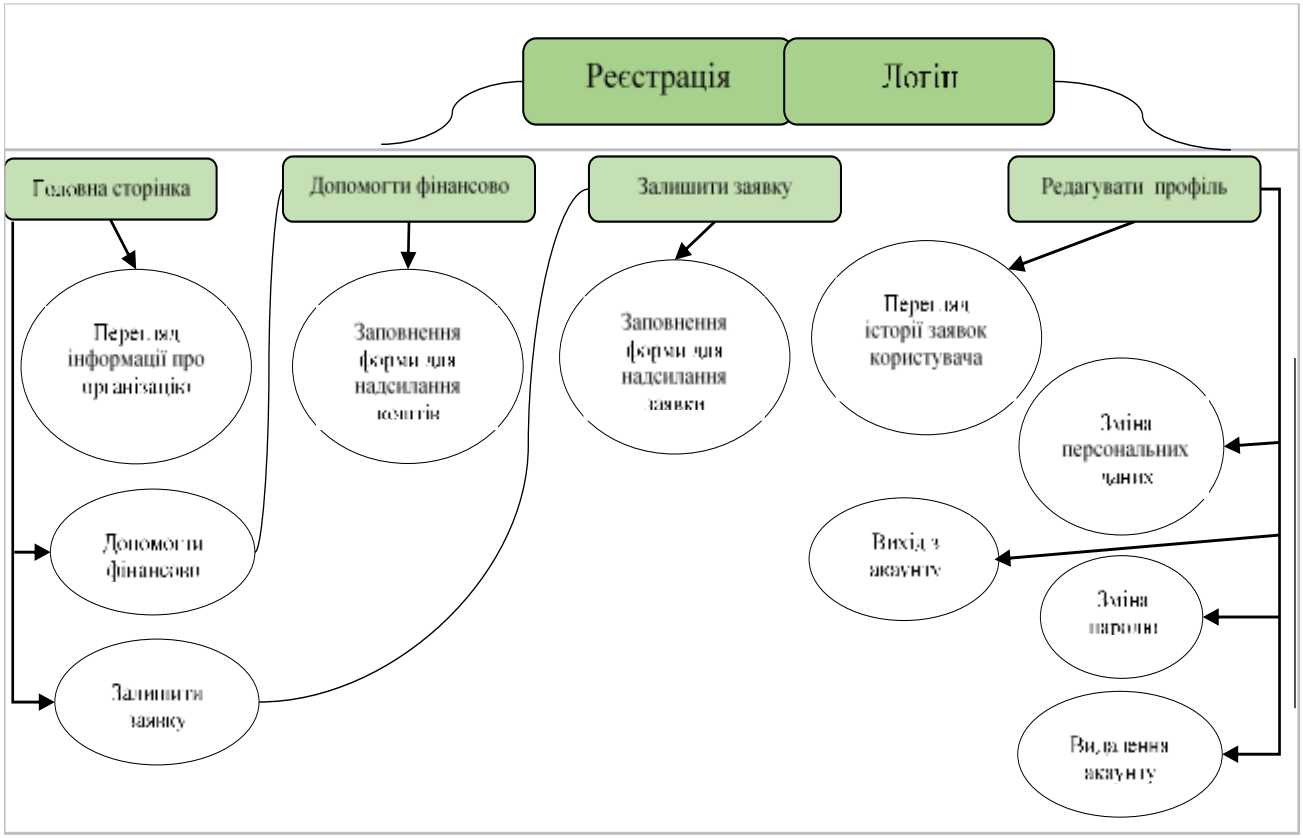


Рисунок 2.7 – Діаграма функцій системи для простого користувача

Для кращого розуміння як проходить процес створення нового замовлення допомоги на рисунку 2.8 зображено діаграму станів для процесу створення нової заявки користувачем.

Отже, після того, як користувач зайшов на сторінку створення нової заявки, система очікує, що він вибере категорію допомоги, яка йому потрібна. Після вибору категорії користувачеві необхідно ввести список та кількість конкретних речей з цієї категорії. Якщо користувач бажає додати ще одну категорію, він повинен знову вибрати її зі списку доступних і ввести необхідний список речей. Якщо це остання категорія в замовленні, система очікує введення адреси для доставки. Цей процес дозволяє користувачеві створити докладний список потреб і забезпечити, що необхідна допомога буде надана в повному обсязі.

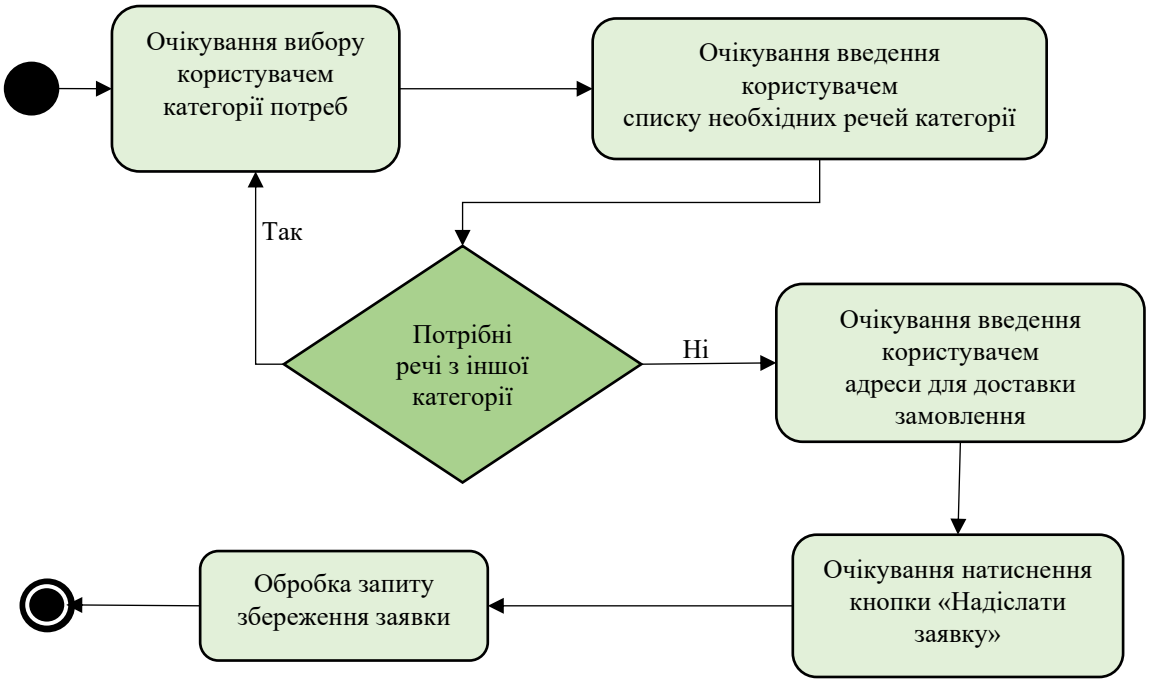


Рисунок 2.8 – Діаграма станів для створення нового замовлення

Після адреси, система очікує натиснення кнопки «надіслати заявку» для надсилання запиту зберігання замовлення на сервер. У випадку успішного збереження заявки система виводить повідомлення про це.

2.5 Проектування бази даних

Для виконання КРБ, а саме для зберігання даних про замовлення, маршрути, користувачів було обрано базу даних MongoDB.

MongoDB – це документ-орієнтована база даних, яка зберігає дані у вигляді документів у форматі BSON (Binary JSON) [33]. MongoDB є популярним вибором для вебзастосунків, аналітики, систем керування контентом та інших сценаріїв, де важлива гнучкість та швидкодія.

Відмінним атрибутом MongoDB є його адаптивна схема, яка дозволяє документам в межах однієї колекції володіти різними полями, виключаючи необхідність фіксованого креслення даних. Вбудовані структури даних MongoDB призначені для полегшення швидких операцій читання та запису

Іншою примітною особливістю є його здатність до масштабування, що дозволяє розподіляти дані по численних серверах, тим самим підвищуючи продуктивність і надійність. MongoDB також надає масив запитів, починаючи від пошуку, агрегації, сортування, серед інших функцій.

Крім того, MongoDB забезпечує гнучку модель розгортання, яка вміщує локальні, а також хмарні сервіси, такі як AWS, Azure і Google Cloud Platform. Для забезпечення доступності найвищого рівня, аварійного відновлення та резервного копіювання даних MongoDB оснащений функцією автоматичної реплікації.

Для зручності роботи з MongoDB було створено MongoDB C# Driver для платформи .NET, що теж використаний для написання вебсистеми доставки допомоги малозабезпеченим.

Перед початком написання програми варто визначити які сутності будуть необхідні для її роботи, які поля вони будуть мати та як вони будуть пов’язані між собою – для цього завдання зазвичай використовують ER-діаграму.

ERD – це абревіатура діаграми відношень сутностей, яку іноді називають діаграмами ER або моделями відношень сутностей. ERD графічно представляють зв'язок між об'єктами, такими як окремі особи, об'єкти або ідеї в базі даних, включаючи їх специфічні атрибути. Метою діаграми ER є зображення логічної структури баз даних шляхом опису сутностей, деталізації їх атрибутів та демонстрації їх взаємозв'язків. Це особливо корисно для інженерів з метою документування існуючої бази даних або планування макета для нової.

Отож було створено діаграму відношення сутностей для вебсистеми доставлення допомоги малозабезпеченим, зображено її на рисунку 2.9. Основними сутностями є:

– користувач, із такими атрибутами, як ім’я, прізвище, адреса електронної пошти, номер телефону, хеш паролю та роль;

– благодійний внесок із атрибутом суми, що може зробити користувач;

– замовлення, що створює користувач, із такими атрибутами як, адреса, час створення, id замовника та булеве значення зібрано/не зібрано;

– адреса, що складається з назви області, району, міста/села, вулиці, номеру будинку та номеру квартири;

– потреби, з яких складаються замовлення, із атрибутами категорія, назва та кількість одиниць;

– маршрут (що складається із замовлень, точніше із їх адрес), який може сформувати лише адміністратор, і має атрибути початкового пункту та довжини.

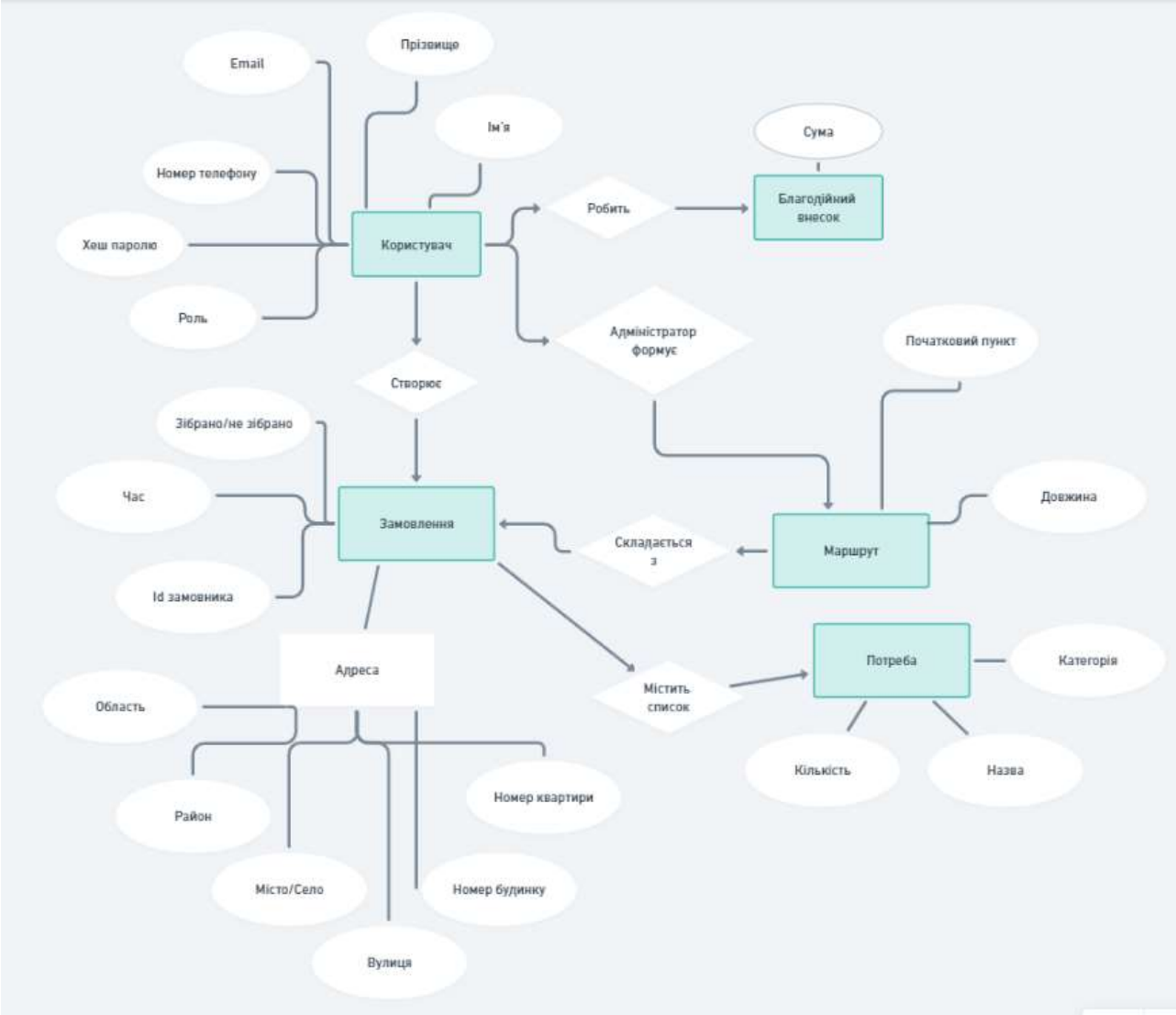


Рисунок 2.9 – ER-діаграма вебсистеми

Оскільки база даних MongoDB документ-орієнтована, тоді усі сутності представлені у вигляді документів, відповідно до їх типу створеного в коді програми. Відповідно було створено такі типи: User, Address, Order, NeedItem, Route.

Усі документи база даних зберігає у колекціях (аналог рядків таблиці у реляційних базах даних). Хоч MongoDB дозволяє зберігати різні типи документів в одній колекції, однак це робиться вкрай рідко та обережно, оскільки при десириалізації такі колекції можуть спричинювати помилки в коді. Отже, для вебсайту було створено такі колекції: users, routes, orders та refreshTokens (для автентифікації користувача з JWT).

Нижче наведено детальну структуру документів із типами даних у MongoDB таких сутностей: Address (таблиця 2.1), NeedItem (таблиця 2.2), Order (таблиця 2.3).

Таблиця 2.1 – Атрибути таблиці “Address”

№ п/п	Назва	Тип даних	Опис
1.	Region	String	Назва області
2.	District	String	Назва району
3.	City	String	Назва міста
4.	Street	String	Назва вулиці
5.	HouseNumber	String	Номер будинку
6.	FlatNumber	String	Номер квартири (якщо є)

Таблиця 2.2 – Атрибути таблиці “NeedItem”

№ п/п	Назва	Тип даних	Опис
1.	Category	String	Назва категорії потреби
2.	Name	String	Назва певної потреби (речі)
3.	Number	String	Кількість

Таблиця 2.3 – Атрибути таблиці “Order”

№ п/п	Назва	Тип даних	Опис
1.	_id	String	Унікальний ключ замовлення
2.	UserId	String	Унікальний ключ замовника
3.	DateTime	Date	Час створення замовлення
4.	Compiled	Boolean	Статус зібрано/не зібрано
5.	Address	Object	Адреса доставки замовлення
6.	NeedItems	Array	Список потребованих речей

На таблицях помітна певна різниця у атрибутах, зокрема унікальний ключ є лише в одному з трьох розглянутих – Order. Address та NeedItem є лише складовою частиною для інших сутностей, вони служать як тип даних для певних полів (як Address) або ж є елементами списків (як NeedItem).

2.6 Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра було розроблено метод доставлення за мурашиним алгоритмом та проект вебсистеми допомоги малозабезпеченим.

Мурашиний алгоритм є ефективним у роботі зі складними умовами, адаптивний до змін, може працювати на паралельних обчисленнях, має додатковий фактор безпеки та є досить простим для реалізації та налаштувань. Він має велику кількість готових рішень та підвидів, які покращують роботу алгоритму. Тому його було обрано для методу покращення доставлення допомоги малозабезпеченим.

Необхідно створити вебсистему допомоги малозабезпеченим, яка буде реалізовувати доставлення допомоги за мурашиним алгоритмом, а точніше його готовим рішенням AntColony. Система має складатись із двох частин: багаторівневої серверної (створена з фреймворком ASP.NET Core Web API) та привабливої користувачеві клієнтської (створена з фреймворком React та бібліотекою Ionic). Вебсайт має бути доступним користувачам із різними ролями: адмінстратори – основна функція сформувати новий оптимальний маршрут для доставлення допомоги; користувачі – основна функція створити своє замовлення на допомогу. Усі дані повинні зберігатись у документ-орієнтованій базі MongoDB. Після створення вебсистеми, її необхідно ретельно випробувати різними методами тестування задля впевненості у коректності її роботи.

Також потрібно показати покращення ефективності доставлення допомоги з використаним методом.

Розділ 3 Програмна реалізація вебсистеми допомоги малозабезпеченим

Загалом, програмна реалізація такої вебсистеми є складним та багатогранним процесом, який вимагає уваги до деталей та глибокого розуміння потреб цільової аудиторії. Прагнучи створити вебсистему, яка буде надійним та ефективним інструментом для надання допомоги малозабезпеченим та полегшення їхнього життя, потрібно враховувати усі деталі, що можуть впливати на бажання користуватись таким інструментом.

Привабливий та зрозумілий інтерфейс сайту є необхідним для споживачів, однак основним його завданням є ефективне використання методу доставлення допомоги з алгоритмом мурахи. Це означає що сайт має швидко реагувати на дії користувача та відображати результати його запитів.

Швидкість роботи вебсистеми залежить від її архітектури, від особливостей взаємодії компонентів між собою, а також реалізації самих компонентів. Метод доставлення допомоги малозабезпеченим є серцем вебсистеми, від його структури на коректності роботи залежить доцільність створення усього сайту.

Отож, основним завданням розділу є огляд та створення правильної архітектури та структури вебсистеми допомоги малозабезпеченим, структури методу доставлення, а також детальне тестування правильності роботи сайту.

3.1 Загальна архітектура вебсистеми допомоги малозабезпеченим

Як було зазначено у підрозділі 2.4, вебсистема допомоги малозабезпечених буде розподіленою системою, тобто буде складатись із клієнтської частини, серверної частини та бази даних, що будуть комунікувати між собою через інтернет (рисунок 3.1). Перевага розподіленої системи в тому, що вона дає можливість поєднувати в проекті різні фреймворки, мови програмування, що в монолітній системі було б важко або і нереально реалізувати.

Отож, компонент “Client” – це клієнтська частина, що повністю несе відповідальність за комунікацію із користувачами вебсистеми. Тобто відображає дані на інтерфейсі, стежить за діями користувача (натискання кнопок, перехід між сторінками, введення даних тощо) та надсилає необхідні запити до іншого компоненту.

Компонент «Server», серверна частина відповідає за логіку вебсистеми: авторизація користувачів, комунікація із базою даних, оброблення запитів, а також найголовніше виконання методу доставлення допомоги малозабезпеченим.

Компонент «Database» – база даних, тобто місце де зберігаються уся інформація, що потрібна для реалізації вебсистеми.

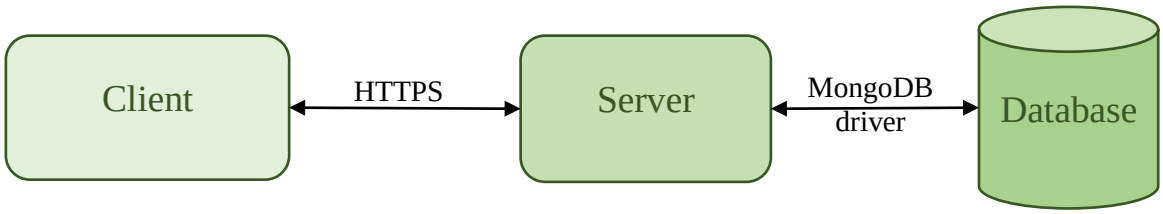


Рисунок 3.1 – Архітертура вебсистеми

“Client” та «Server» спілкуються між собою за допомогою HTTPS (Hypertext Transfer Protocol Secure) протоколу. Це поширена методика комунікацій компонентів розподіленої системи, оскільки вона забезпечує безпечну передачу даних шляхом шифрування даних, що дозволяє захистити інформацію від несанкціонованого доступ та модифікації інформації під час передачі. Наявність HTTPS підвищує довіру користувачів до вашого вебресурсу, оскільки вони бачать, що їх дані захищені (відображається зелений замочок біля url-поля в браузері).

Використовуючи HTTPS, також реалізовується pull-модель отримання даних. Це означає, що клієнт посилає один із HTTP запитів (POST, PUT, GET тощо) до сервера. Отримавши запит, сервер обробляє його, як прописано, та повертає відповідь до клієнта. Тобто для того, щоб щось тримати, клієнт спочатку повинен це попросити.

«Server» та «Database» комунікують за допомогою MongoDB driver, що сильно полегшує створення запитів до бази даних, оскільки драйвер надає інтерфейс для такої взаємодії.

3.2 Структура та особливості реалізації методу доставлення за мурашиним алгоритмом

Реалізація основного функціоналу методу доставлення за мурашиним алгоритмом було завданням, що вирішував сервер вебсистеми. На рисунку 3.2 зображено діаграму класів, що використовуються для розробки методу доставлення.

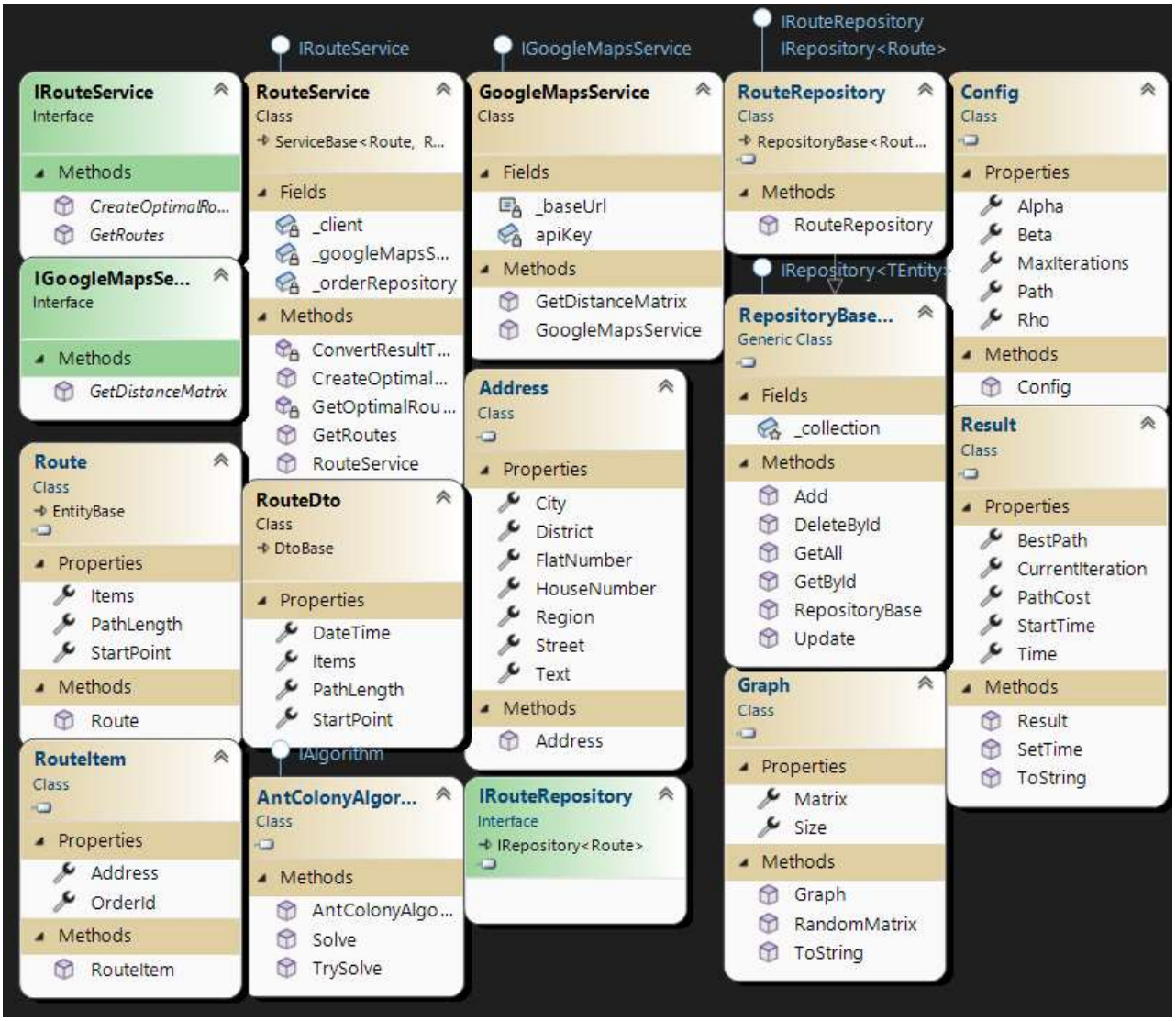


Рисунок 3.2 – Діаграма класів методу доставлення

На діаграмі показано класи, які поступово обробляють запит клієнта на створення нового маршруту доставлення. Розглянемо їх детальніше.

1. RouteService – клас, що виконує основну роль у методі доставлення. Він реалізовує інтерфейс IRouteService, що містить лише два методи CreateOptimalRoute() та GetRoutes(), які відповідаю за створення та отримання маршрутів відповідно.

2. GoogleMapsService – клас, що здійснює не меш важливо роль для методу доставлення. Він за вказаними адресами замовлень та пунктом відправки формує матрицю відстаней, використовуючи Google Maps Distance Matrix API. Він реалізовує інтерфейс IGoogleMapsService та містить єдиний метод GetDistanceMatrix().

3. RouteRepository наслідує клас RepositoryBase та імплементує інтерфейс IRouteRepository. Його завданням є отримання та зберігання даних про маршрути доставлення допомоги малозабезпеченим.

4. AntColonyAlgorithm – клас бібліотеки AntColony, що відповідає за виконання алгоритму мурахи з пошуку оптимального маршруту. Методи Solve() та TrySolve() запускають процес та у відповідь дають об’єкт класу Result, що містить поля BestPath (маршрут) та PathCost (довжина маршруту).

5. Клас Config містить поля параметрів алгоритму мурахи.

6. Клас Graph містить поля Matrix – матриця відстаней та Size – кількість зупинок на маршруті.

7. Клас Route – клас, що містить дані результату виконання методу доставлення допомоги. Поле PathLength (загальна довжина маршруту), поле StartPoint (пункт відправки або початкова точка маршруту) типу Address. Поле Items – це впорядкована множина зупинок маршрут, або список об’єктів класу RouteItem. Об’єкт типу Route зберігається у базі даних, однак клієнту передається RouteDto.

Для того, щоб зрозуміти порядок виконання дій у методі доставлення допомоги малозабезпеченим було створено діаграму послідовності, що зображена на рисунку 3.3.

Коли клієнт надсилає запит для створення нового маршруту, він передає в параметрах адресу пункти відправки допомоги та список унікальних ключів замовлень, що будуть доставлені цим маршрутом.

Запит потрапляє до контролера AdminController, що в своє чергу передає запит до класу RouteService за допомогою методу CreateOptimalRoute().

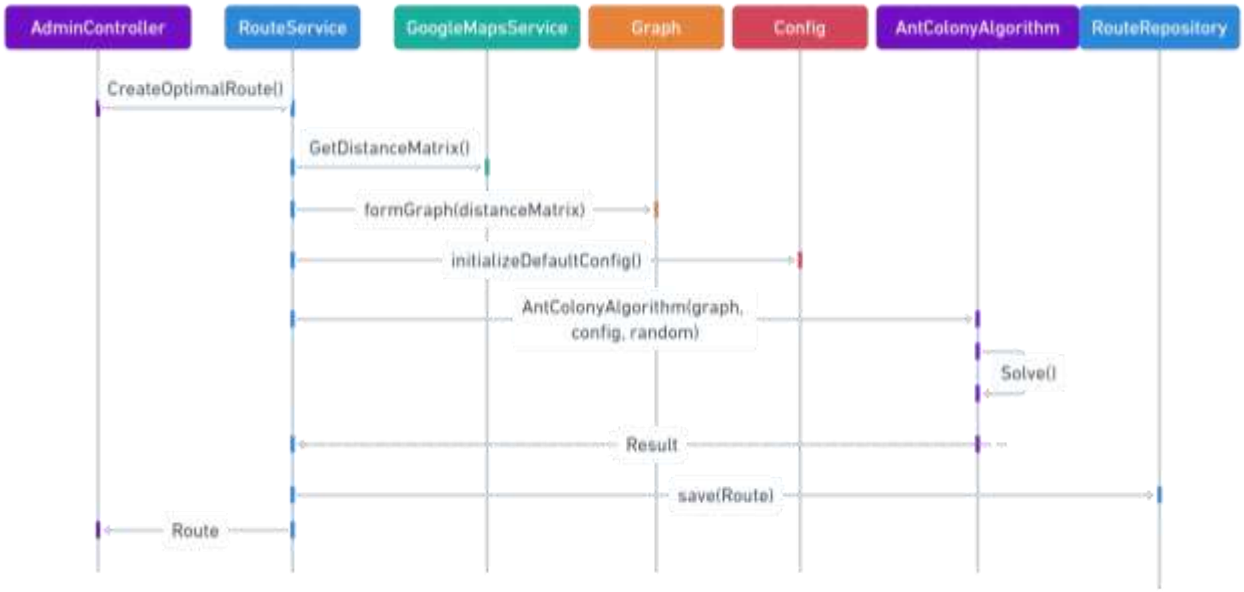


Рисунок 3.3 – Діаграма послідовності створення маршруту

RouteService виконує багато роботи. Спершу він звертається до GoogleMapsService.GetDistanceMatrix(), щоб отримати реальну матрицю відстаней між зупинками (включаючи пункт відправки). На основі матриці відстаней він формує об’єкт класу Graph», також ініціалізує об’єкт класу Config з параметрами мурашиного алгоритму по замовчуванню. Далі створюється об’єкт класу AntColonyAlgorithm, передаючи необхідні параметри (graph, config, random), та викликається метод Solve(). Коли алгоритм мурахи виконався, метод повертає результат типу Result.

Після цього RouteService конвертує результат у тип Route. Сервіс передає об’єкт Route до RouteRepository, що зберігає його до колекції «routes» у базі даних. Тоді сервіс повертає результат контролеру, що віддає його у відповідь на запит користувача.

На клієнтській стороні процес створення нового маршруту виглядає дещо по-іншому. Для початку авторизований користувач із роллю «Admin» має перейти на вкладку «Замовлення» та обрати ті, які мають бути відправленими цим маршрутом, після цього натиснути кнопку «Створити маршрут» (рисунок 3.4).

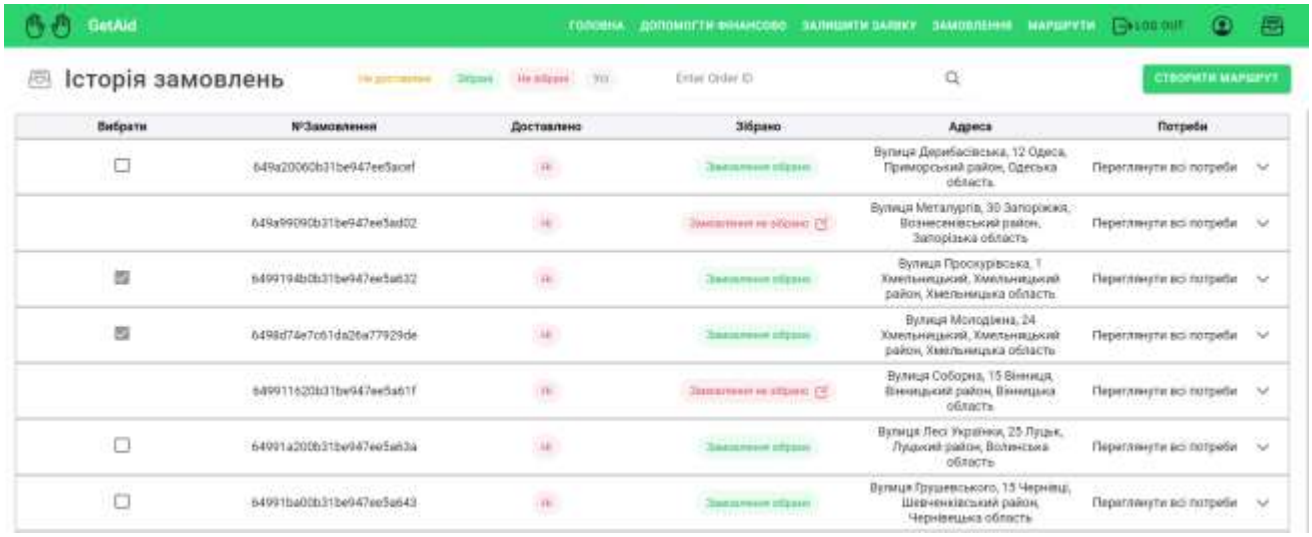


Рисунок 3.4 – Перший етап створення маршруту клієнтом

Після натискання кнопки «Створити маршрут» вебсистема виводить вікно для підтвердження операції (рисунок 3.5). Користувач може змінити пункт відправки, що вписаний по замовчування, та перейти до наступного етапу або ж скасувати операцію.

Натиснення кнопки «Створити» на другому етапі створення маршруту провокує відправлення запиту на сервер. Якщо сталася помилка про створенні маршруту доставлення, то вебсистема виводить повідомлення про це. Якщо ж операція успішна, то вебсистема виводить результат – оптимальний маршрут доставлення допомоги малозабезпеченим (рисунок 3.6). На модальному вікні є блакитна кнопка «Зберегти», що дає можливість адміністратору зберегти новий маршрут як файл формату .json та потім поділитися ним із кимось (наприклад, водієм, що буде здійснювати доставлення). Натиснувши зелену кнопку «Закрити» адміністратор повернеться на сторінку «Замовлення» із оновленим списком замовлень.

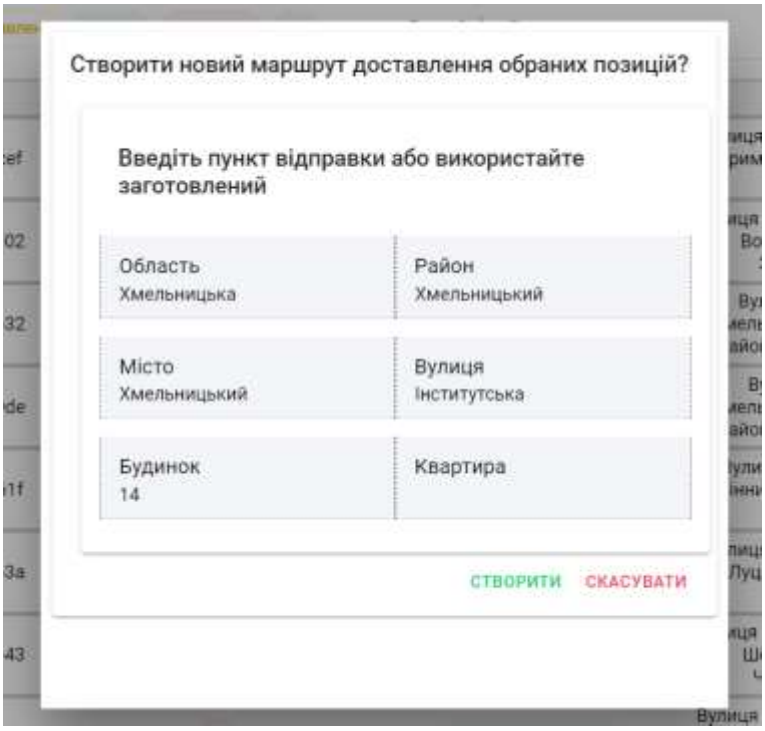


Рисунок 3.5 – Другий етап створення маршруту клієнтом

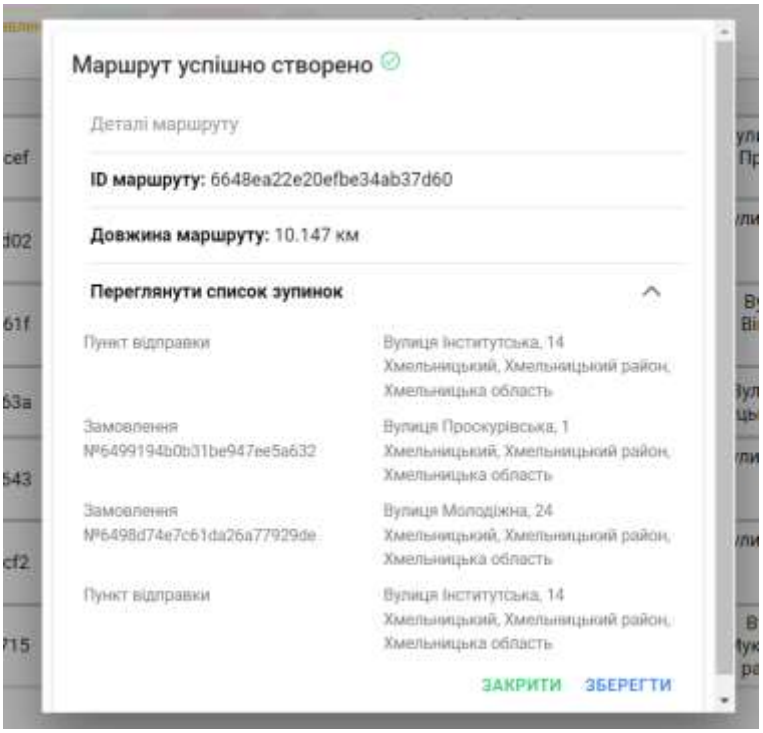


Рисунок 3.6 – Результат створення маршруту клієнтом

Отже структура методу доставлення допомоги з мурашиним алгоритмом є досить складною та багатоетапною для виконання, однак для користувача вебсайту цей процес є дуже легким у виконанні.

3.3 Структура та особливості реалізації вебсистеми допомоги малозабезпеченим

Вебсистема допомоги малозабезпеченим є розподіленою системою, тому було розглянуто структуру та особливості реалізації клієнтської та серверної частин окремо. Також розглянемо структуру бази даних.

3.3.1 Структура та особливості реалізації серверної частини

Сервер був створений на платформі .NET, використовуючи фреймворк ASP.NET Core WEB API. Такі системи мають свою характерну структуру файлів, зокрема виділяються такі основні типи файлів:

- controllers, які обробляють HTTP-запити та відповідають на них;
- models – класи моделей, які визначають структуру даних;
- DTOs – містить класи для передачі даних між клієнтом і сервером;
- services, які містять бізнес-логіку програми;
- repositories – класи для доступу до даних та їх обробки.

Беручи за основу структуру проектів WEB API, сервер вебсистеми допомоги малозабезпеченим було побудовано наступним чином (рисунок 3.7).

Було створено рішення GetAidBackend, що містило в собі декілька різних проектів. Перший проект Domain містив усі класи моделей (models), зокрема:

- базовий клас EntityBase;
- Address (дані про адреси);
- NeedItem (дані про потреби у замовленнях);
- Order (дані про замовлення допомоги);
- Route (дані про побудовані маршрути доставлення);
- User (дані про користувача);
- UserPrivateData (особисті дані користувача);
- UserRole (роль користувача: admin або consumer).

Проект Storage містить усі класи репозиторіїв: RepositoryBase, OrderRepository, RouteRepository, UserRepository та їхній інтерфейси, що розподілені по різних папках. А також клас Configurator, що відповідає за впровадження залежностей.

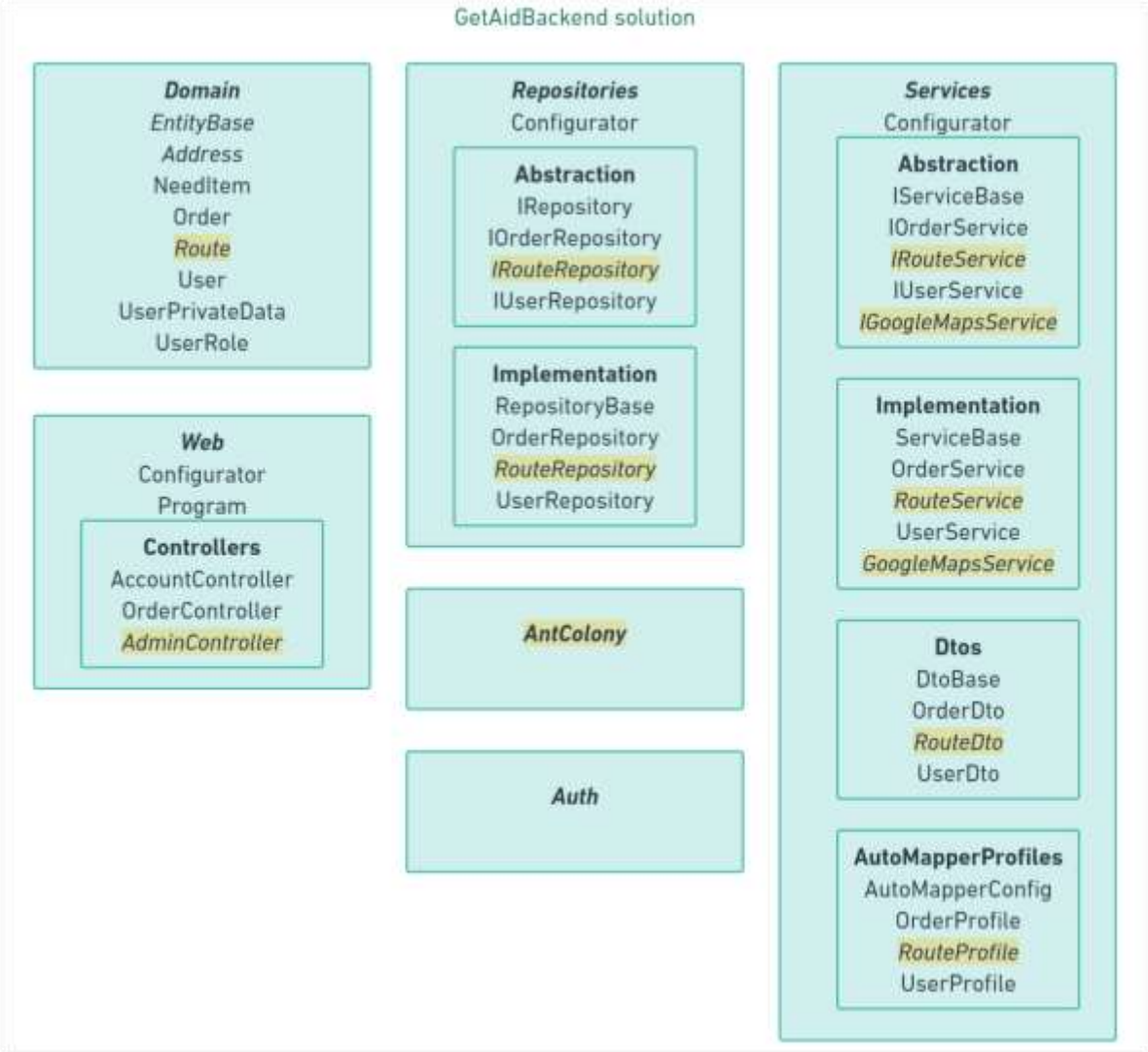


Рисунок 3.7 – Структура серверу вебсистеми

Також є проект Services, що містить сервіси ServiceBase, OrderService? UserService, RouteService, GoogleMapsService та їхні інтерфейси. Також містить папку Dtos, що містить класи DtoBase, OrderDto, RouteDto, UserDto та необхідні файли для їх конвертації за допомогою бібліотеки AutoMapper.

Проект Web містить контролери AccountController, OrderController та AdminController. AccountController містить методи обробки даних пов’язаних із

користувачем (авторизація, змінення паролю, редагування даних профілю тощо). OrderController надає метод для створення замовлень, отримання замовлень певного користувача. AdminController містить методи, що може виконувати лише користувач з роллю адміністратора, а саме створювати маршрути, отримувати історію маршрутів та всіх замовлень.

Також в рішенні присутні проекти AntColony (бібліотека алгоритму мурахи) та Auth (бібліотека авторизації користувача). Проекти Domain, Storage, Services є також бібліотеками, а Web вебзастосунком.

На діаграмі було позначено класи жовтим виділенням, які мають відношення до реалізації методу доставлення за мурашиним алгоритмом. Його не було винесено в окремий проект, оскільки для невеликої вебсистеми доцільності в цьому немає, лише збільшиться кількість класів.

Особливу увагу потрібно надати проекту Auth. Він відповідає за одну з найважливіших частин сайту – авторизація з використанням JWT. Авторизація користувача на сайтах важлива для забезпечення безпеки та конфіденційності даних, контролю доступу до ресурсів та функцій, а також для персоналізації користувацького досвіду. Вона допомагає захистити інформацію від несанкціонованого доступу, дозволяє системі ідентифікувати користувачів та надавати їм відповідні права, а також підвищує загальну довіру та надійність сайту. Авторизація також допомагає запобігти шахрайству і забезпечити відповідність правовим і регуляторним вимогам.

Діаграму класі проекту Auth зображено на рисунку 3.8. Клас RefreshTokenRepository реалізовує інтерфейс та відповідає за доступ до даних про refresh-токени користувачів. Сервіс PasswordHasher містить функції для хешування та перевірки на правильність паролів користувачів. Сервіс JwtTokenService реалізовує методи для створення access та refresh токенів та їх перевірки. Основний сервіс JwtAccountService містить методи для логіну користувачів, їх реєстрації, змінення паролю та поновлення токенів.

Також у проекті створені типи моделей для токенів (JwtTokenInfo, RefreshTokenInfo). Різні типи запитів користувачів: LoginRequest, RegisterRequest,

ChangePasswordRequest, RefreshTokenRequest. А також модель відповіді LoginResponse.

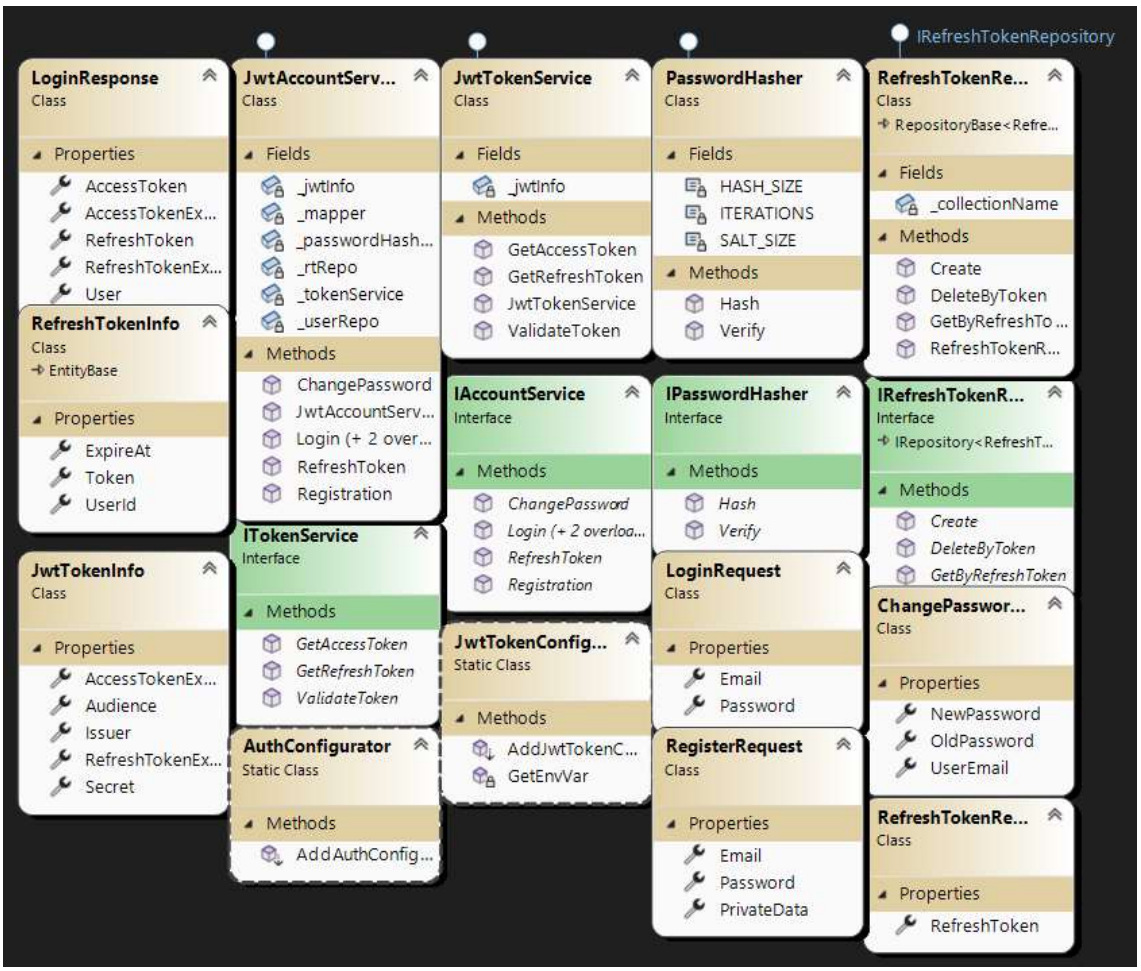


Рисунок 3.8 – Діаграма класів проекту Auth

Для кращого розуміння роботи модуля Auth було створено діаграму послідовностей виконання запиту на логін користувач (рисунок 3.9).

AccountController приймає запит на логін і викликає метод Login() у класі AccountService. Сервіс звертається до методу GetByEmail() класу UserRepository, щоб знайти користувача за його електронною поштою. Якщо користувача не знайдено, виникає виняток.

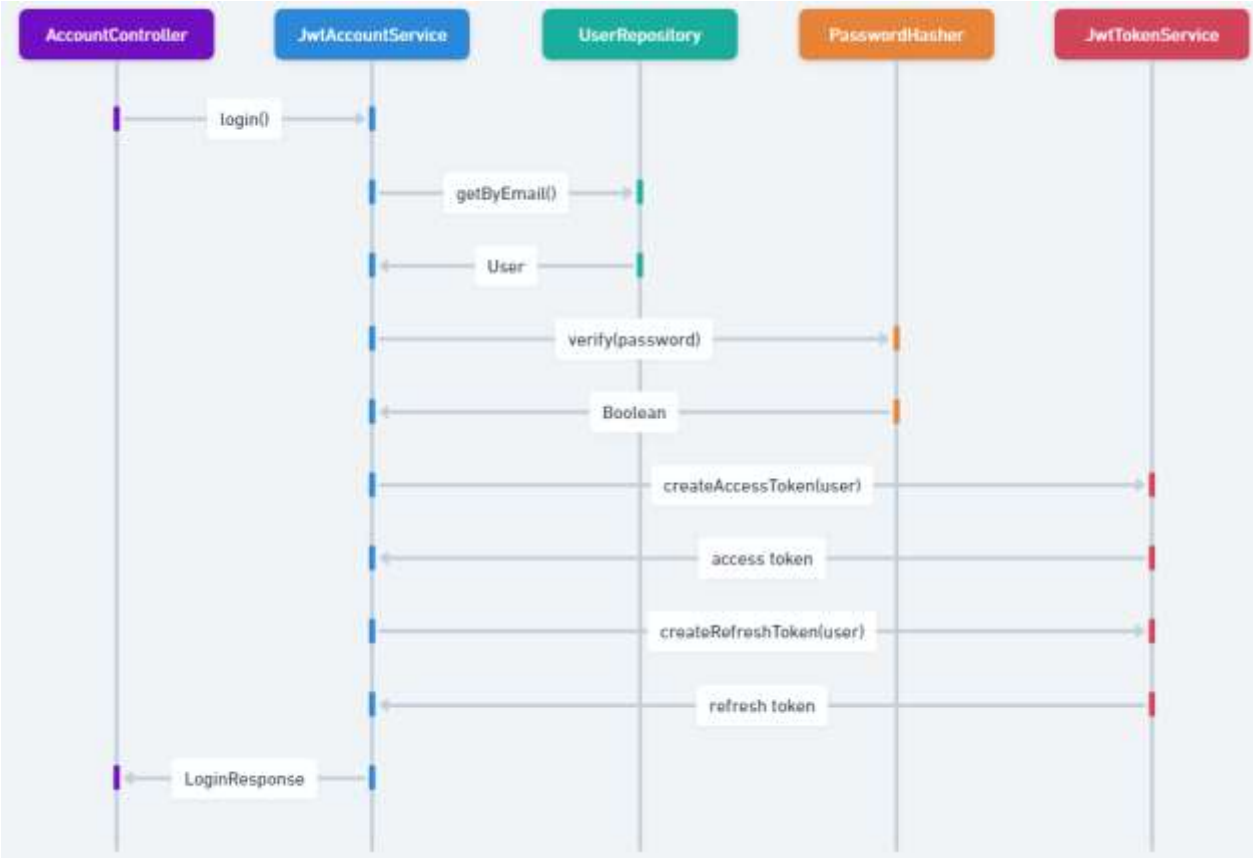


Рисунок 3.9 – Діаграма послідовностей виконання запиту на логін

Якщо користувач знайдений, перевіряється правильність пароля за допомогою методу Verify() класу PasswordHasher. Якщо пароль неправильний, знову виникає виняток. Якщо ж пароль правильний, створюються access і refresh токени методами CreateAccessToken() і CreateRefreshToken() класу JwtTokenService. Результати формуються в об'єкті LoginResponse і повертаються до контролера.

3.3.2 Структура та особливості реалізації клієнтської частини

Клієнтська частина була створена за допомогою JavaScript бібліотеки React та фреймворком Ionic. Такий стек технологій дає можливість зручно та швидко створювати вебсайти, оскільки має містить багато заготовлених компоненті, які легко інтегруються у код. Хоч такі проекти є гнучкими до змін, одна їх структура заздалегідь логічно визначена. Файли з основним функціоналом системи лежать у папці src, що містить в собі головний компонент застосунку наступні папки:

– assets: папка для зберігання статичних ресурсів (іконки, зображення тощо);
– components: папка для зберігання багаторазових React-компонентів;
– pages: папка для сторінок додатку;
– theme: Папка для тем і стилів додатку;

При створенні застосунку було збережено початкову структуру проекту, а також додано до src папки domain, що зберігає основні типи для сайту, та api, що містить методи для комунікації з сервером системи.

Для наочного ознайомлення з структурою клієнтської частини було побудовано діаграму класів (рисунок 3.10) за допомогою плагіну середовища VS Code.

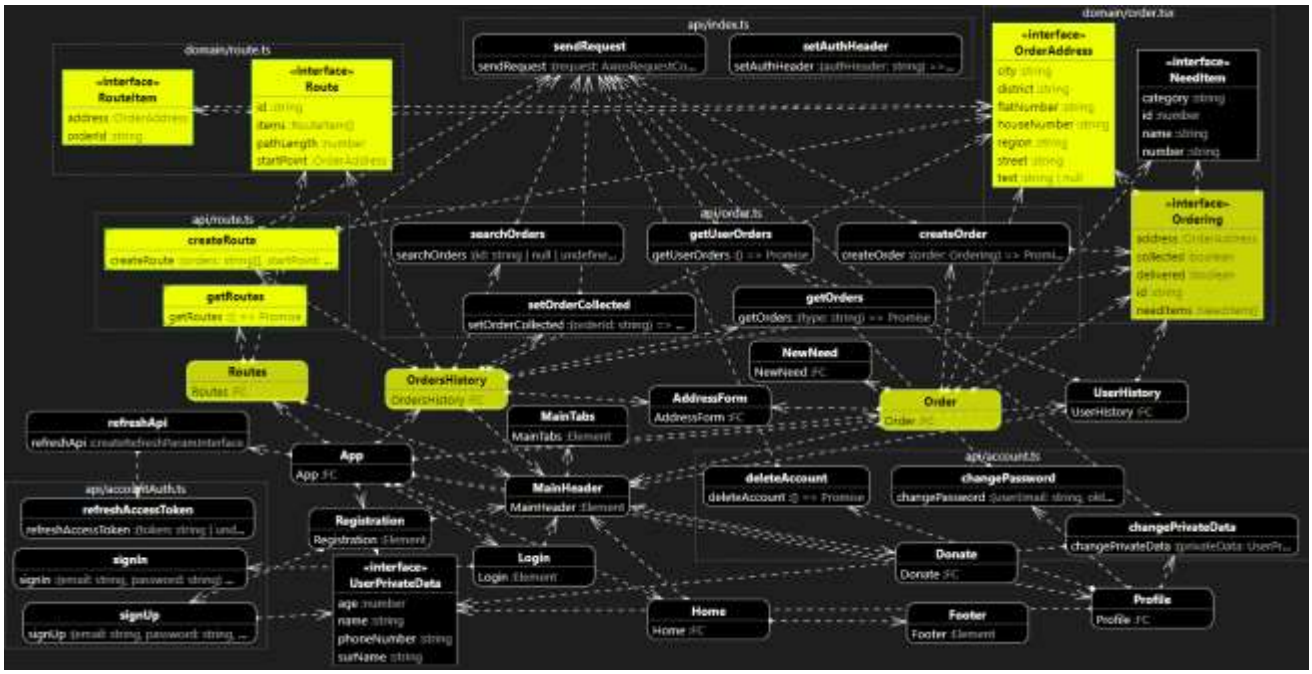


Рисунок 3.10 – Діаграма класів клієнтської частини

Жовті елементи, що позначені на діаграмі, є тими компонентами, які допомагають реалізувати метод доставлення допомоги малозабезпеченим на сайті. Інтерфейси Route, RouteItem, OrderingAddress – це типи, які використовуються для отримання та відображення даних із серверу. Методи файлу api/route.ts getRoutes() та createRoute() є відповідальними за отримання наявних маршрутів та створення нового відповідно. Вже створені маршрути

користувачу виводить сторінка Routes, а побудувати новий можна на сторінці OrdersHistory.

Застосунки, створені з бібліотекою React, використовують SPA(Single Page Application) шаблон. Це означає, що фактично увесь вебсайт відображається на одній HTML-сторінці, і під час навігації між різними частинами застосунку відбувається динамічне оновлення вмісту сторінки без повного перезавантаження.

Під час створення SPA застосунку для допомоги малозабезпеченим використовувались спеціальні ReactIonic компоненти для маршрутизації між сторінками. IonicReatRouter виконує маршрутизацію, IonRouterOutlet відображає вміст відповідно до поточного маршруту, Route визначає маршрут та компонент, що буде відображений. Отож було побудовано діаграму, на якій зображено як реалізована маршрутизація для вебсистеми допомоги малозабезпеченим (рисунок 3.11), тобто який компонент буде відображатись за певним url-шляхом та яка це буде сторінка для користувача.

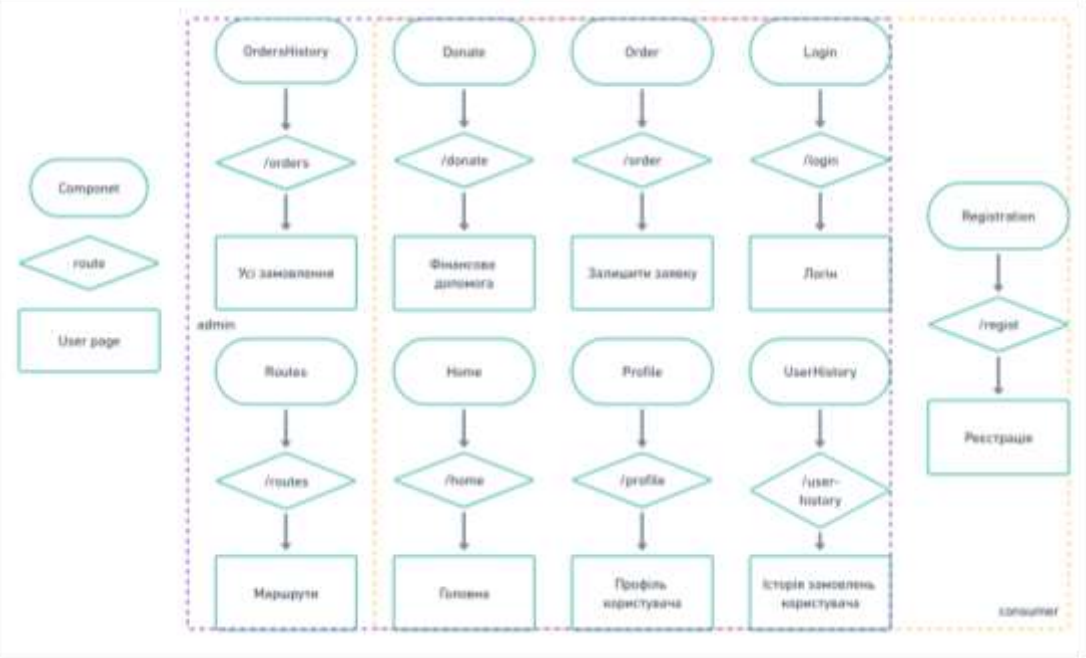


Рисунок 3.11 – Маршрутизація вебсайту допомоги малозабезпеченим

Розглядаючи діаграму, можна зрозуміти, що користувач побачить сторінку створення заявки на допомогу при переході за маршрутом /order, яка відображає компонент Order. При переході за маршрутом /profile, користувач відкриє

сторінку свого профілю, на якій буде відображений компонент Profile. Головна сторінка, за адресою /home, міститиме компонент Home. Перехід на адресу /login відкриє сторінку для входу в систему, на якій буде відображений компонент Login. Сторінка з історією замовлень користувача буде доступна за маршрутом /user-history та міститиме компонент UserHistory. Щоб надати фінансову допомогу, користувач може перейти за адресою /donate, де буде відображений елемент Donate. При переході за маршрутом /register, користувач буде перенаправлений на сторінку реєстрації, на якій буде відображений елемент Registration. Компонент Routes знаходиться за адресою /routes на сторінці з маршрутами. А сторінка з усіма замовленнями, яка містить компонент OrdersHistory, розташована за маршрутом /orders.

Також на діаграмі помітні позначені зони для користувачів із різними ролями. Тобто лише користувач з роллю consumer може зареєструватися на сайті, та лише адміністратори можуть бачити сторінки маршрутів та усіх замовлень. Всі інші сторінки на сайті (головна, фінансова допомога, залишення заявки, перегляд історії замовлень, профіль, логін) відкриті для обох ролей.

3.3.3 Структура бази даних

Для зберігання даних вебсистеми допомоги малозабезпеченим було обрано документ-орієнтовану базу даних MongoDB. Така база даних не є реляційною, що дає можливість зручно формувати дані у документи та створювати невелику кількість колекцій.

Структуру бази даних зображено на рисунку 3.12.

Рисунок 3.12 – Структура бази даних вебсистеми

Отож, БД є досить простою. Вона складається із чотирьох колекцій.

1. Orders: зберігає дані про заявки на допомогу;
2. RefreshTokens: зберігає токени для поновлення доступу авторизованим користувачам;
3. Routes: містить дані про побудовані маршрути доставлення допомоги;
4. Users: містить дані про користувачів вебсистеми.

3.4 Тестування інформаційної системи

Тестування інформаційної системи є надважливим кроком, який ніколи не можна пропускати при розробці будь-яких проектів. Його значимість переоцінити неможливо і на це є ряд причин.

Тестування ІС допомагає виявити та виправити помилки на початкових етапах, що зменшує витрати на їх виправлення. Забезпечує відповідність системи вимогам і специфікаціям. Виявляє та усуває проблеми з продуктивністю. Виявляє вразливості для захисту конфіденційної інформації. Перевіряє роботу системи на різних платформах і конфігураціях. Забезпечує позитивний досвід користувача. Зменшує витрати та час, потрібний для виправлення помилок на пізніших етапах. Гарантує дотримання законодавчих і регуляторних вимог.

Отож вебсистему допомоги малозабезпечених було протестованими різними способами, а саме юніт-тестами та тест-кейсами.

Для тестування правильності роботи методу Solve() класу AntColonyAlgorithm було створено три юніт-тести на отримання правильної довжини маршруту в залежності від вхідних даних: SolveTest_10City(), SolveTest_5City(), SolveTest_3City().

Усі три тести було успішно пройдено (рисунок 3.13). Тобто, у відповідь на вхідні дані вони дали таку довжину маршруту, яка дорівнює очікуваній. Також у результаті виконання можна побачити впорядкований маршрут для відвідування «міст» (зупинок) по черзі.

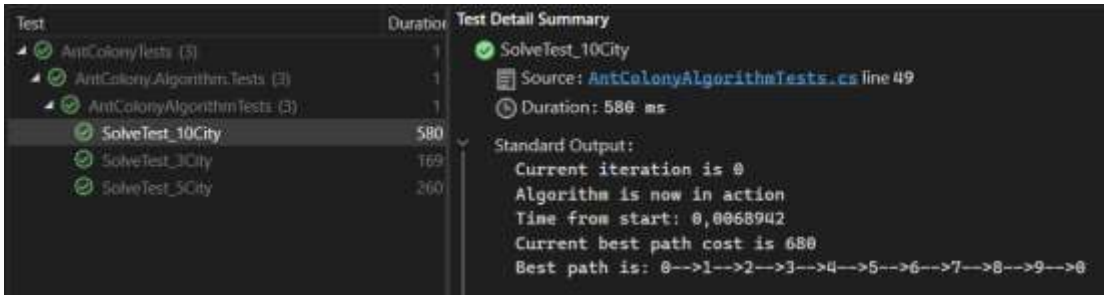


Рисунок 3.13 – Результат виконання юніт-тестів

Після цього вебсистему було протестовано тест-кейсом. Тест сформований у таблиці 3.1, він перевіряє коректність надання доступу до сторінок сайту відповідно до ролі авторизованого користувача.

Таблиця 3.1 – Тест-кейс BZ0001

Тест-кейс ID: BZ0001	Пріоритет**:** 1	Створено:10.05.2024, Боревіч З.Є.
Назва: Перевірка на коректність надання доступу до сторінок сайту відносно ролі авторизованого користувача Вхідні дані: Запущений сервер, запущена клієнтська частина		
Кроки		Очікуваний результат
1. Запустити браузер. 2. Перейти до головної сторінки вебісайту за посиланням «http:// http://localhost:8100/home». 3. На панелі швидкої навігації натиснути кнопку «LogIn». 4. У поле «Електронна адреса» ввести набір букв «bilenka@gmail.com». 5. У поле «Пароль» ввести набір символів: «string» 6. Натиснути кнопку «Увійти»		Буде відображено сторінку авторизації сайту. Авторизація пройде успішно. Користувача буде перенаправлено на сторінку створення нової заявки. Відображаються такі елементи навігаційного меню: головна, допомогти фінансова, залишити заявку, профіль, історія замовлень.
Результат виконання тест-кейсу: пройдено успішно		

Результат виконання тест-кейсу зображено на рисунку 3.14. Користувача за вказаними електронною адресою та паролем було успішно авторизовано. Оскільки такий користувач існує в базі даних із роллю «Consumer», то для нього відображаються лише ті елементи навігаційного меню, які є доступними за цією роллю. Отже, тест-кейс BZ0001 було успішно пройдено.

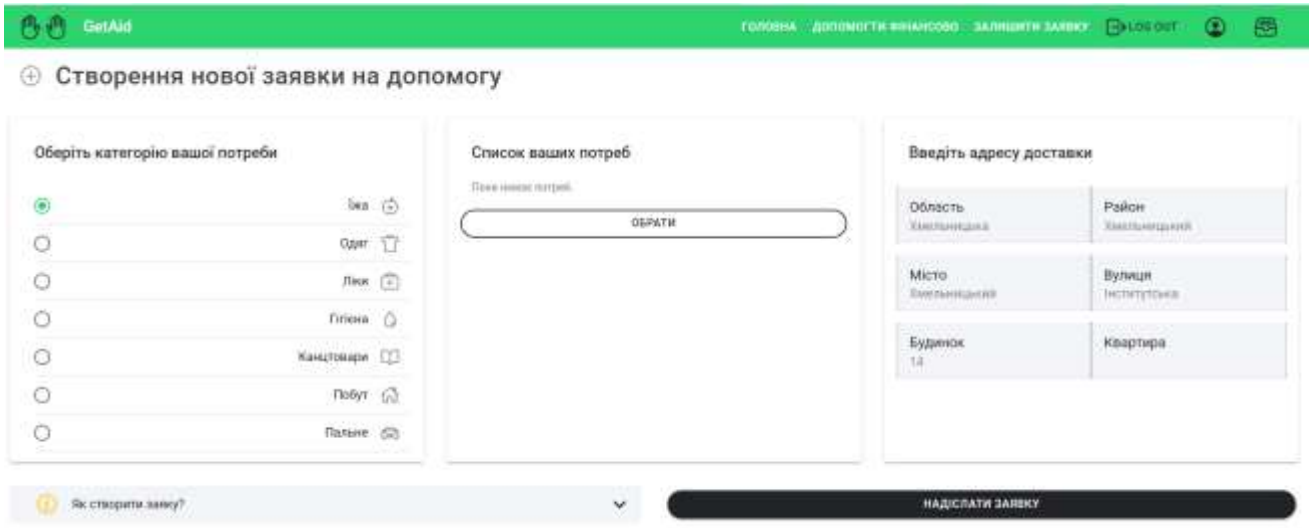


Рисунок 3.14 – Результат роботи тест-кейсу BZ0001

Отже, опираючись на результати виконання різних тестів ІС, можна стверджувати, що система працює відповідно до поставлених їй вимог та правильно виконує поставлені їй завдання.

3.5 Вимоги до публікації вебсистеми

Розгортання вебсистеми іноді є доволі складним завданням, особливо у випадку розподіленої системи. Для того, щоб трохи спростити цю роботу, наведено детальні вимоги до кожної частини вебсистеми.

Вимога до бази даних одна: розробник має мати стрічку підключення до існуючої бази. Її легко створити самостійно. Потрібно мати аккаунт на MongoDB Atlas – на цьому сервісі можна створити безкоштовний кластер, через який можна отримати доступ до баз даних користувачів MongoDB.

При розгортанні сервера та клієнта вимог є дещо більше. Потрібно обрати хостинг, що буде забезпечувати безперебійну роботу та швидкий доступ до вебсайту. Крім того, він повинен підтримувати потрібну версію вебсервера. На сервері повинно бути достатньо місця для зберігання файлів вебсайту та бази даних. Хостинг також повинен підтримувати необхідні технології та мови програмування, такі як C# , JavaScript та ASP.NET.

Клієнтську та серверну частини теоретично можна закидати на різні хостингові сервіси (GoogleCloud, Azure, AWS тощо), однак таке з’єднання може довше виконувати запити, збільшується вартість підтримування системи. Також є вища ймовірність перехоплення даних під час виконання запитів. Тому, варто обирати один хостинг, що зможе розгорнути як сервер ASP.NET Core WEB API, так і клієнт React-Ionic App.

Перед публікацією проектів необхідно заповнити усі конфігураційні файли на хостингу, зокрема змінні середовища, адже без зазначення потрібних даних сервер не зможе запуститись, а клієнтська частина не зможе знайти сервер.

Якщо ж необхідно розгорнути вебсистему для тесту чи налагодження коду, то це можна зробити локально. Відкривши середовище програмування або термінал, наприклад, Visual Studio, потрібно запустити проект GetAidBackend.Web – ці дії запустить сервер локально. Потім необхідно запустити клієнтську частину. Використовуючи термінал чи якийсь редактор коду потрібно виконати команду «ionic serve». Головне прописати у коді клієнтської частини правильний URL до локального сервера.

3.6 Функціональність вебсистеми

Для того, щоб допомогти новим користувачам швидко зорієнтуватися та розпочати роботу з системою, було проведено аналіз функціональності вебсистеми допомоги малозабезпеченим та висвітлено у цьому розділ результат – детальну інструкцію використання вебсистеми.

Як тільки користувач переходить по посиланню на сайт або розгортає його локально (для тесту), то він одразу попадає на головну сторінку вебсайту (рисунок 3.15).

На цій сторінці розміщується інформація про те, чому важливо допомагати малозабезпеченим, як це можна зробити фінансово та як отримати допомогу.

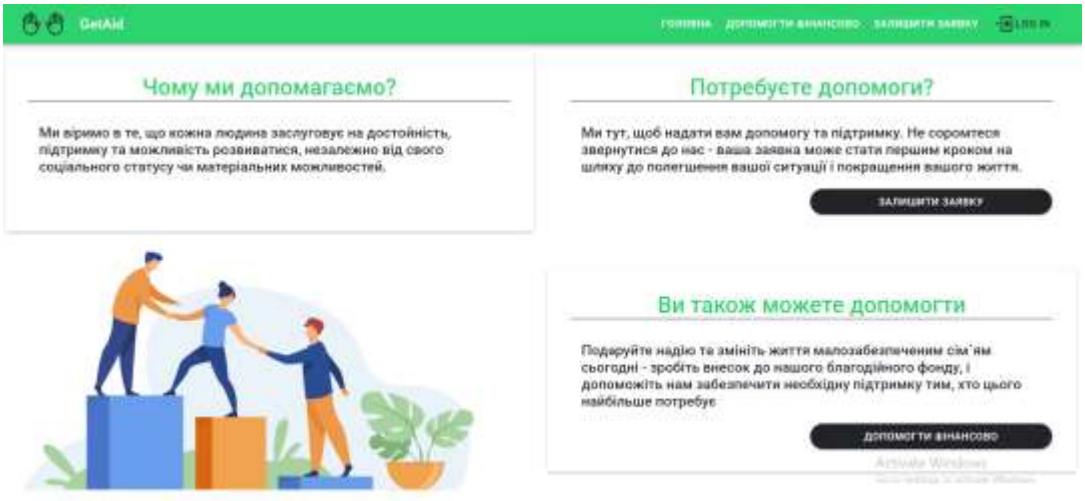


Рисунок 3.15 – Головна сторінка вебсистеми допомоги малозабезпеченим

Із головної сторінки можна потрапити на сторінку фінансової допомоги (рисунок 3.16). Тут можна фінансово підтримати фонд, що займається купівлею, розподілом та доставленням допомоги малозабезпеченим. Ця сторінка не вимагає авторизації, тому задонатити може будь-хто без лишніх кроків.

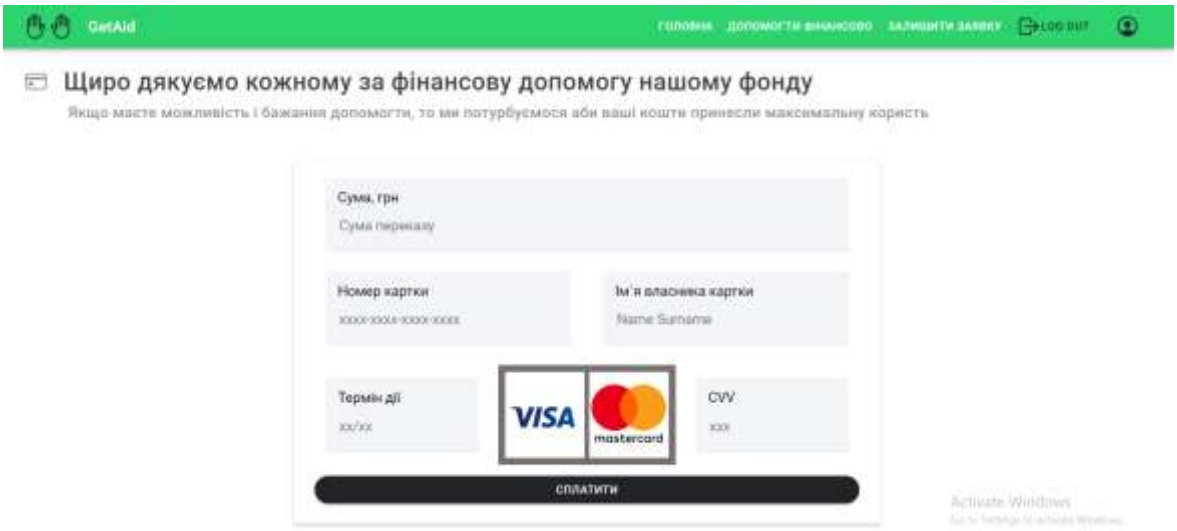


Рисунок 3.16 – Сторінка фінансової допомоги малозабезпеченим

Користувачам, що бажають отримати допомогу, необхідно бути авторизованими для створення нової заявки. Тому у навігаційному меню потрібно натиснути кнопку «Log In», після чого їх буде перенаправлено на сторінку логіну (рисунок 3.17).

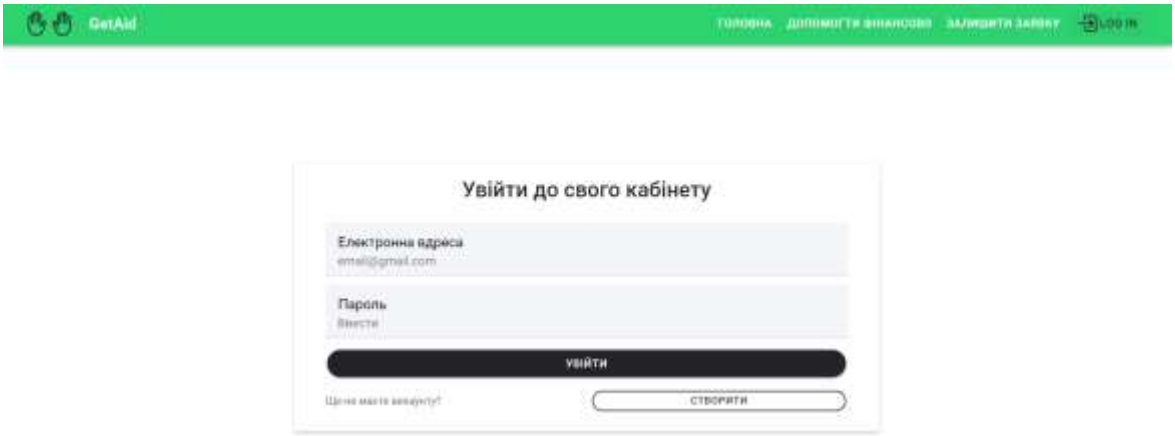


Рисунок 3.17 – Сторінка для входу користувача до системи

Якщо людина раніше не користувалась цим сервісом, то їй необхідно зареєструватись у вебсистемі. Після натискання на кнопку «Створити» на сторінці логіну, користувача буде перенаправлено на сторінку реєстрації (рисунок 3.18). Варто зазначити, що тут можуть зареєструватися лише користувачі, що набудуть роль «Consumer». Для адміністраторів процес реєстрації проходить по-іншому: їх просто додають у базу та видають облікові, як вони потім можуть змінити.

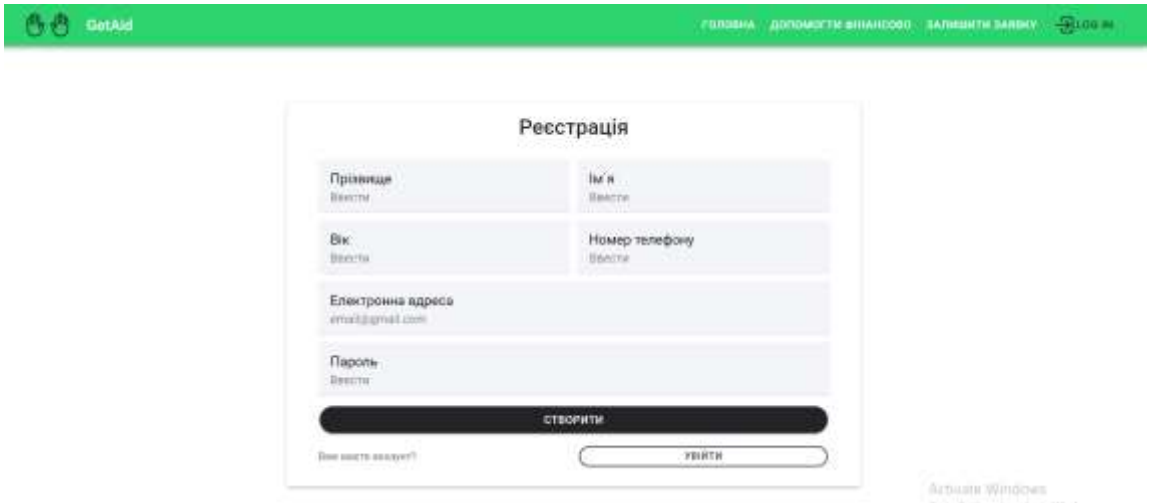


Рисунок 3.18 – Сторінка реєстрації для споживачів

Після успішної авторизації користувача, вебсистема переводить його на сторінку створення нового замовлення про допомогу (рисунок 3.19). Тут споживач може створити замовлення для себе, а адміністратор для людини, що потребує підтримки та не має доступу до інтернету.

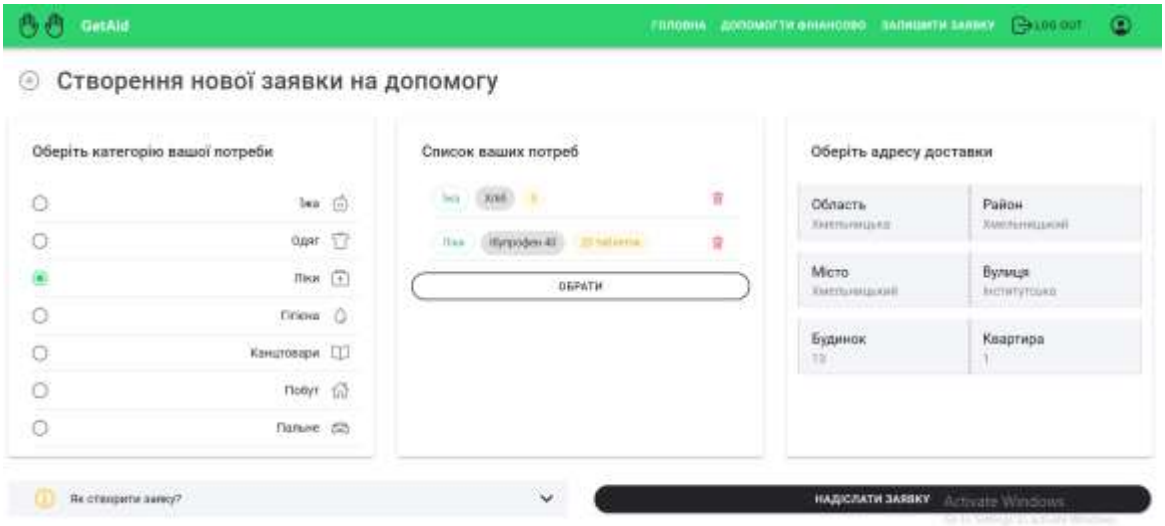


Рисунок 3.19 – Сторінка створення нового замовлення

Обравши певну категорію потреби користувач може сформувати список потреб даної категорії (рисунок 3.20). Також може обрати іншу категорію та додати потреби до неї. Потім необхідно ввести адресу для доставлення допомоги та натиснути кнопку «Надіслати запит». Якщо операція була успішна, то про це з’явиться повідомлення.

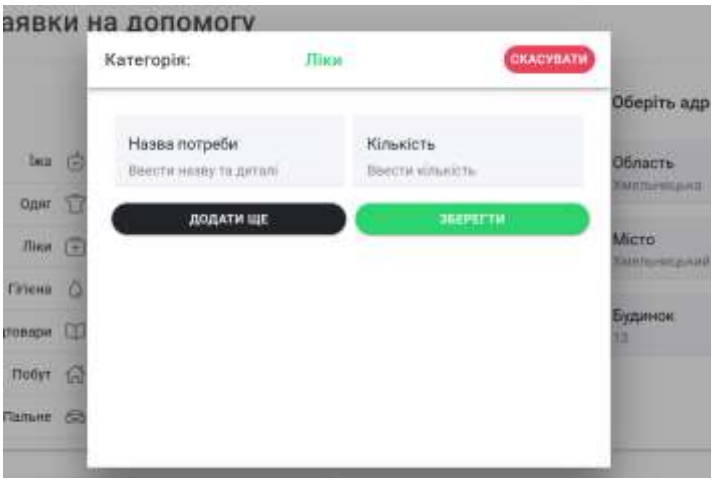


Рисунок 3.20 – Формування списку потреб певної категорії

У разі виникнення питань до процесу створення заявки, користувач може розгорнути блок внизу сторінки «Як створити заявку», де вказана покрокова інструкція.

Натиснувши іконку профілю у навігаційному меню, користувач потрапить на сторінку редагування даних профілю (рисунок 3.21). Тут він може змінити особисті дані, змінити пароль, вийти із акаунту чи навіть видалити свій обліковий запис.

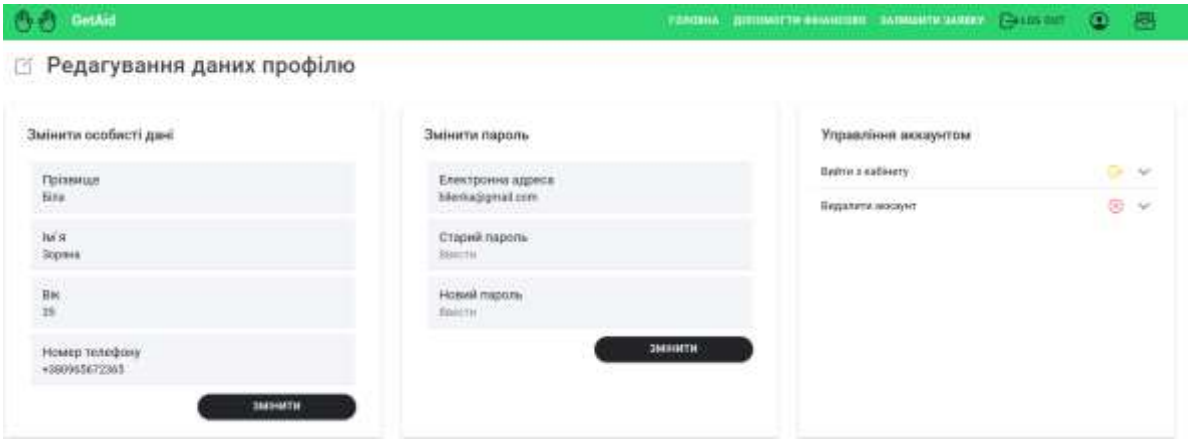


Рисунок 3.21 – Сторінка редагування даних профілю

Користувач може також переглянути історію своїх замовлень (рисунок 3.22), натиснувши крайню праву іконку (стек файлів). Тут можна слідкувати за прогресом виконання замовлення, чи зібране воно вже, чи доставлене, а аткож ще раз переглянути деталі заявки.

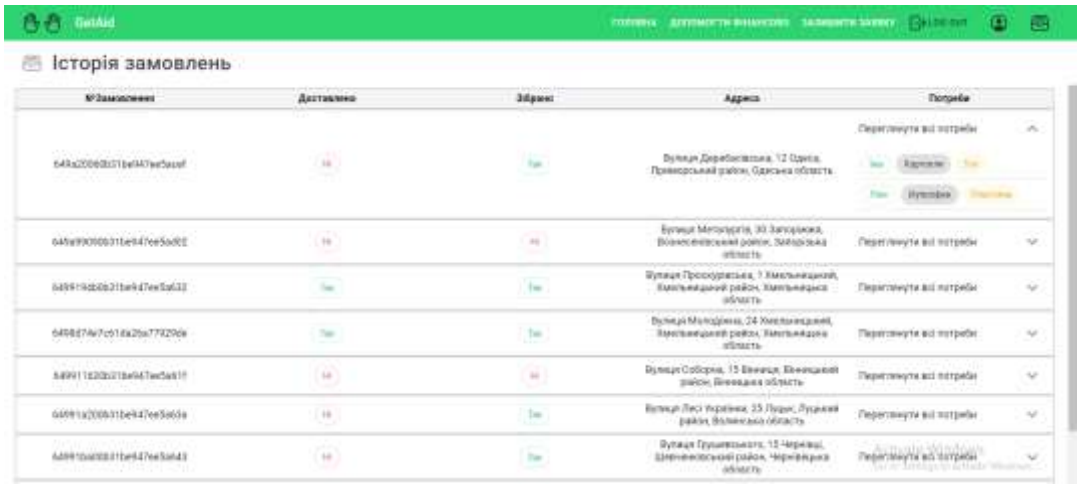


Рисунок 3.22 – Сторінка історії замовлень споживача

Усі наведені сторінки є доступними як споживачам допомоги так і адміністратора фондів. Тепер розглянемо, що особливо може робити адміністратор.

Для початку, він має бути авторизованим у систему, роль користувача «Admin» відриває дві додаткових опції у навігаційному меню: маршрути та замовлення. Натиснувши на кнопку «Замовлення», адміністратор потрапляє на сторінку усіх створених замовлень (рисунок 2.23).

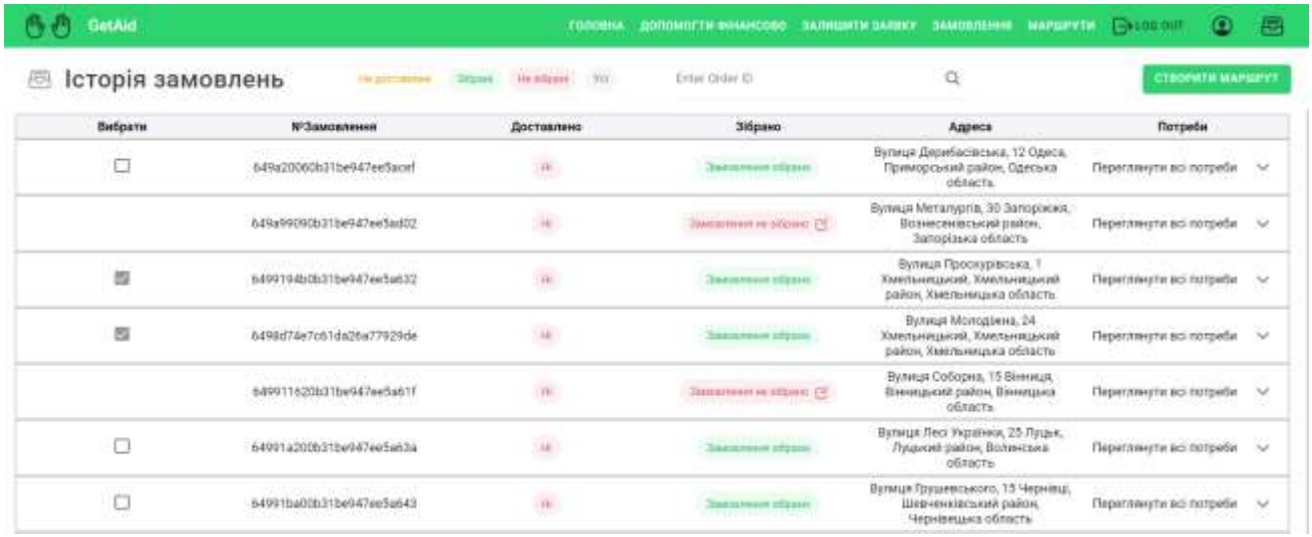


Рисунок 3.23 – Сторінка історії усіх замовлень

Є можливість фільтрації замовлень за статусом їх виконання: зібрані (але не доставлені), не зібрані, не доставлені чи усі. Щоб відфільтрувати записи потрібно просто натиснути на кнопку потрібного фільтру. Також є поле пошуку, куди можна ввести ID замовлення і вебсистема здійснить пошук таких замовлень використовуючи метод порівняння startsWith(). Щоб змінити статус замовлення з «не зібрано» на «зібрано» необхідно натиснути на червону кнопку у колонці «Зібрано» у відповідному рядку замовлення та підтвердити свою дію у модальному вікні.

Саме на цій сторінці адміністратори мають змогу створити новий маршрут для доставлення допомоги. Для цього у крайній лівій колонці необхідно відзначити прапорцями ті замовлення, що повинні бути відправленими у новому маршруті. Позначатись можуть лише ті замовлення, що вже зібрані, тобто готові

до відправки. Після того як було позначено усі замовлення, потрібно натиснути кнопку «Створити маршрут», яка відкриє вікно створення маршруту (рисунок 3.24). Тут необхідно ввести адресу пункти відправки допомоги, або залишити дані по замовчуванню та натиснути кнопку «Створити». Також можна скасувати операцію.

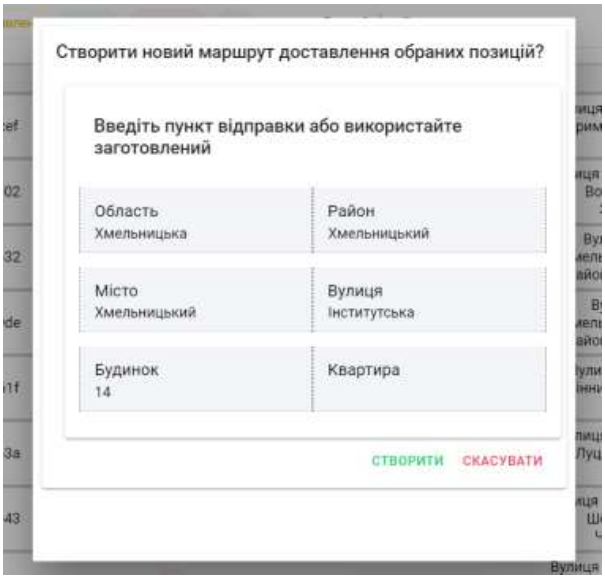


Рисунок 3.24 – Вікно створення маршруту

Після успішного створення маршруту доставлення відображається вікно результату цієї дії(рисунок 3.25).

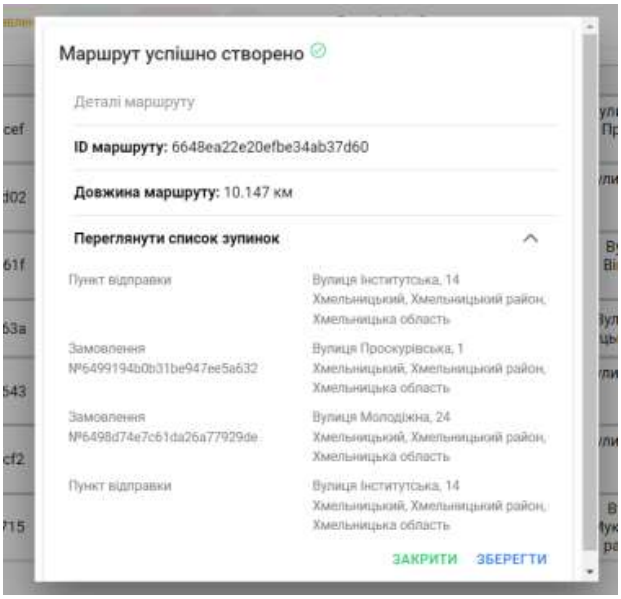


Рисунок 3.25 – Вікно результату створення маршруту

У вікні деталей маршруту показані ID, довжина та впорядкована множина зупинок. Ці результати можна зберегти як .json файл, якщо натиснути на кнопку «Зберегти».

Для того, щоб переглянути історію всіх побудованих маршрутів методом доставленням допомоги з алгоритмом мурахи, адміністратор має перейти на сторінку історії маршрутів (рисунок 3.26). Тут відображаються деталі для кожного побудованого вебсистемою маршруту.

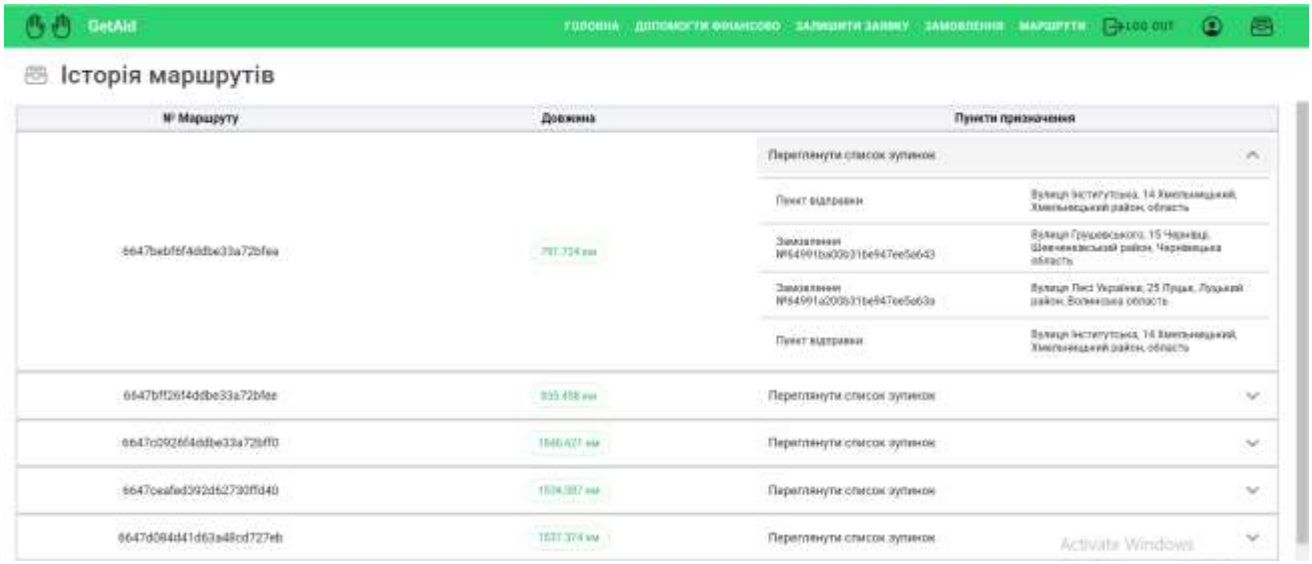


Рисунок 3.26 – Сторінка історії маршрутів

Отже, слідкуючи вказівкам, що зазначені у цьому підрозділі, споживачі та адміністратори зможуть з легкістю виконати поставлені їм завдання.

3.7. Оцінка покращення ефективності доставлення допомоги за запропонованим методом

Цей підрозділ КРБ присвячений оцінці покращення ефективності доставлення допомоги малозабезпеченим з використання створеного методу доставлення за мурашиним алгоритм.

Основним завданням методу доставлення є побудова оптимального маршруту для проходження усіх запропонованих точок доставки, починаючи з

пункту відправки. Як результат метод повертає впорядковану множину зупинок із зазначеними унікальними ключами замовлень на цих адресах та загальну довжину маршруту. На основі цього фонд допомоги малозабезпеченим може розраховувати орієнтовну тривалість та вартість цієї подорожі, правильно розподілити фінанси. Також, дані про побудовані маршрути легко зберігаються, аби можна було відслідковувати діяльність фонду.

До впровадження методу доставлення у вебсистему допомоги малозабезпеченим адміністратори фонду не мали можливості побудувати точний маршрут доставлення. Вони могли лише обрати замовлення, які бажають доставити цим маршрутом та віддати список водієві. Звичайно ж, водій намагатиметься підібрати якнайкращий маршрут для доставлення допомоги, одна людський фактор підвищує ризик помилковості у таких рішеннях.

Порівняємо довжини двох маршрутів доставки (за вхідними даними з таблиці 3.2): побудований методом доставлення з мурашиним алгоритмом (рисунок 3.27) та побудований імітованим водієм транспорту доставлення з використанням застосунку GoogleMaps (рисунок 3.28). Заради спрощення експерименту вхідними даними є невелика кількість зупинок із міста Хмельницький.

Таблиця 3.2 – Вхідні дані для побудови маршруту доставлення

Назва пункту	**№Замовлення**	**Адреса**
Пункт відправки	-	Вулиця Інститутська, 14 Хмельницький
Пункт 1	6499194b0b31be947ee5a632	Вулиця Проскурівська, 18 Хмельницький
Пункт 2	6498d74e7c61da26a77929de	Вулиця Молодіжна, 24 Хмельницький
Пункт 3	664e29e8dd1630bf95a678c8	Вулиця Кармелюка, 5 Хмельницький

Тепер розглянемо результат побудованого маршруту за створеним методом доставлення у вебсистемі допомоги малозабезпеченим (рисунок 3.27). Була отримано загальна довжина маршруту 18,742 км. Порядок зупинок

маршрутів такий: пункт відправки, пункт 2, пункт 1, пункт 3, пункт відправки. Заради чистоти експерименту, пункти призначення у такому ж порядку було уведено на сервісі GoogleMaps для створення маршруту, отриманий результат довжини маршруту: 18,8 км.

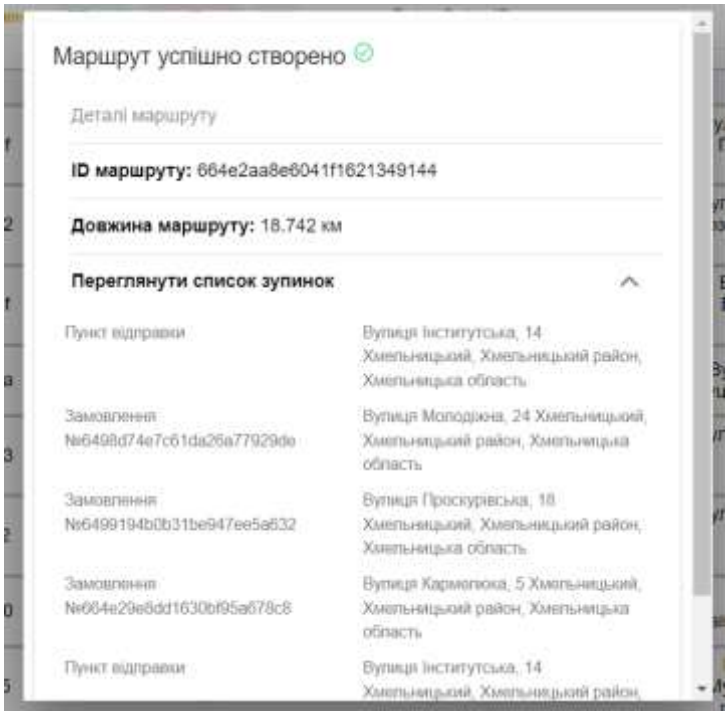


Рисунок 3.27 – Побудований маршрут за методом доставлення

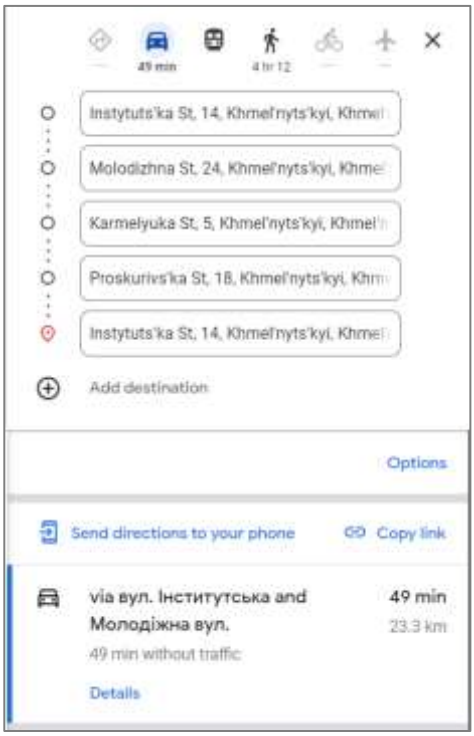


Рисунок 3.28 – Побудований маршрут імітованим водієм

Після цього було імітовано побудову маршруту водієм транспорту з використанням сервісу GoogleMaps. Маршрут із вхідних даних таблиці 3.2 може мати 6 різних варіантів побудови. Водій обрав найкращий, на його думку. Послідовність зупинок у цьому маршруті така: пункт відправки, пункт 2, пункт 3, пункт 1, пункт відправки. Увівши зупинки для нового маршруту у застосунку GoogleMaps в такому порядку, як зазначено, водій отримав результат, зображений на рисунку 3.28. Загальна довжина такого маршруту складатиме 23,3 км.

Отож, порівнюючи загальні довжини двох маршрутів (18, 742 км та 23,3 км), можна з впевненістю сказати, що створений метод за мурашиним алгоритмом підвищує ефективність доставлення допомоги малозабезпеченим. Різниця між довжинами маршрутів сягає більше 4,5 км. Метод за лічені секунди знаходить найоптимальніший маршрут, а водій, через людський фактор, може помилково обрати довший маршрут. Чим більше замовлень потрібно доставити, тим вищий ризик помилки для людини (для трьох замовлень ймовірність правильного вибору 1/6, для п’яти 1/120). Проте для методу доставлення така варіація кількості зупинок не грає великої ролі.

Ще однієї перевагою методу доставлення є те, що він видає номер замовлення, закріплений за адресою доставки. Ця дрібниця також зменшує ризик видачі замовлення на неправильній адресі.

3.8 Висновки до розділу 3

У третьому розділі КРБ спочатку було розглянуто загальну архітектуру вебсистеми допомоги малозабезпеченим. Вона складається із трьох основних компонентів: база даних (MongoDB), сервер (ASP.NET Core WEB API) та клієнт (React-Ionic App). Взаємодія між частинами виконується через мережу Інтернет.

Далі було розглянуто структуру та особливості методу доставлення допомоги за мурашиним алгоритмом. Його реалізація була основним завданням вебсистеми, тому для цього використовується багато класів та сторонніх сервісів таких як, Goooogle Maps Distance Matrix API та AntColony бібліотека. Хоч

реалізація непроста, однак для користувача вебсайту (адміністратора) робота виконується досить швидко, легко та зрозуміло.

Було розглянуто структуру сервера, що зберіг первинну структуру проектів ASP.NET Core WEB API, трішки розширивши її. Однією з особливостей реалізації сервера є окремий проект GetAidBackend.Auth, що повністю відповідає за авторизацією користувача та всі дії, що пов’язані із цим. Клієнтська частина також зберегла первинну структуру проекту React-Ionic App, додавши модулі domain та api, оскільки тут здійснюється комунікація із сервером.

База даних вебсистеми є досить простою. Завдяки використанню документ-орієнтованої бази MongoDB, вона складається лише з чотирьох колекцій: users, orders, routes, refreshTokens.

Після завершення написання коду вебсистему було протестовано юніт-тестами та тест-кейсом. Результати тестування були успішними, що свідчить про правильність роботи вебсистеми допомоги малозабезпеченим та методу доставлення допомоги за мурашиним алгоритмом.

Було проаналізовано та наведено які вимоги потрібні для розгортання вебсистеми, проведено аналіз функціональності системи, а також складена покрокова інструкція для користувачів.

На основі порівняння довжин побудованих маршрутів (методом доставлення, створеного в процесі виконання КРБ, та без нього) з однаковими вхідними даними, стало зрозуміло, що метод значно підвищує ефективність доставлення допомоги малозабезпеченим.

Загальні висновки

Метою кваліфікаційної роботи бакалавра є підвищення ефективності доставлення допомоги у вебсистемах для малозабезпечених людей. Для досягнення поставленої мети розроблено та програмно реалізовано метод доставлення з мурашиним алгоритмом та вебсистему, що буде його використовувати.

Під час виконання КРБ було проведено аналіз методів планування маршрутів та оглянуто існуючі вебреалізації систем допомоги. Розглянуто застосування мурашиного алгоритму в задачах планування маршрутів та його специфічні особливості для вирішення проблеми надання допомоги малозабезпеченим. Було розроблено метод доставлення з мурашиними алгоритмом. Спроектовано серверну та клієнтську частини вебсистеми допомоги малозабезпеченим, а також спроектовано базу даних для неї.

У результаті виконання КРБ отримано метод доставлення (планування маршруту) за мурашиним алгоритмом, що формує та віддає як результат оптимальний маршрут для почергового відвідування кожного пункту призначення. Методу доставлення можна задати початковий пункт (пункт відправки).

Також було створено вебсистему допомоги малозабезпеченим, яка підтримує різні ролі користувачів (адміністратор та споживач) та виконує наступні функції: авторизація та редагування персональних даних користувачів, перегляд користувачами історії своїх заявок, оформлення заявки на допомогу, можливість анонімної фінансової підтримки фонду. Для адміністраторів сайт додатково надає такі функції: показ усіх заявок на допомогу та їх фільтрація за станом готовності до відправки, можливість відбору заявок для створення нового маршруту, побудова нового маршруту, перегляд історії побудованих маршрутів.

Для створення вебсистеми було використано різні технології: мови програмування C# та JS, фреймворк ASP.NET Core WEB API, бібліотеки React та

Ionic, систему керування базами даних MongoDB, а також бібліотеку реалізації алгоритму мурахи AntColony.

Метод доставлення за мурашиним алгоритмом та вебсистему допомоги було відтестовано юніт-тестами та тест-кейсами, в результаті якого, було визначено, що всі необхідні функції працюють правильно. Це свідчить про те, що результат роботи виконання КРБ повністю відповідає поставленому завданню.

Проведено оцінку підвищення ефективності доставлення допомоги за запропонованим методом. Результати аналізу показали, що метод значно підвищує ефективність доставки, формуючи оптимальні маршрути та зменшуючи ризик помилок, допущених людиною.

Перспектива впровадження розробленого методу доставлення та вебсистеми допомоги малозабезпеченим є досить високою, адже, на жаль, актуальність проблеми низького рівня життя в нашій країні існують і будуть існувати ще довго навіть після омріяного закінчення війни. Тому ефективність доставки такої допомоги необхідно підвищувати, так само, як і надати доступні сервіси для швидкого оформлення заявки на допомогу.

Хоч реалізований метод та вебсистема повністю відповідають поставленому завдання та чудово виконують поставлені їм завдання, їх завжди можна покращити. Наприклад, до методу доставлення додати можливість рахувати загальний час поїздки. Вебсистему допомоги малозабезпеченим можна покращити адаптацією до мобільних пристроїв, а також додати нову категорію користувачів – водіїв, що зможуть бачити призначені їм маршрути для доставлення та слідувати за ними по вбудованому навігатору.

Перелік посилань

1) Britannica. The modern road. URL: https://www.britannica.com/technology/road/The-modern-road.

2) Supply Chain Game Changer. The History and State of Vehicle Routing. URL: https://supplychaingamechanger.com/the-history-and-state-of-vehicle-routing/.

3) Upper. What is Route Planning? URL: https://www.upperinc.com/guides/route-planning/.

4) Locate2u. What is Route Planning? URL: https://www.locate2u.com/route-planning/what-is-route-planning/.

5) Wikipedia. Traveling salesman problem. URL: https://en.wikipedia.org/wiki/Travelling_salesman_problem .

6) Gerhard Reinelt. The travelling salesman: Computational solutions for TSP Applications. Heidelberg: Springer-Verlag, 1994. С. 223.

7) Neha Makariye. Towards shortest path computation using Dijkstra algorithm. 2017 International Conference on IoT and Application (ICIOT). 2017. URL: https://ieeexplore.ieee.org/abstract/document/8073641 .

8) Nitin Gupta, Kapil Mangla,Anand Kumar Jha ,Md. Umar. Applying Dijkstra’s Algorithm in Routing Process. International Journal of New Technology and Research (IJNTR). 2016. №5. С. 122-124.

9) UA5.ORG. Алгоритм Дейкстри. URL: https://ua5.org/algorithm/1970-algorytm-dejkstry.html .

10) Brilliant. Dijkstra's Shortest Path Algorithm. URL: https://brilliant.org/wiki/dijkstras-short-path-finder/ .

11) Wikipedia. Dijkstra's algorithm. URL: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm .

12) Noraini Mohd Razali, John Geraghty. Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. Proceedings of the World Congress on Engineering. 2011. №11. С. 5-10.

13) SpringerLink. A review on genetic algorithm: past, present, and future. URL: https://link.springer.com/article/10.1007/s11042-020-10139-6.

14) IET. Effective hybrid genetic algorithm for removing salt and pepper noise. URL: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-ipr.2019.0566.

15) Thomas Back, David B Fogel ¨ and Zbigniew Michalewicz. Evolution computation 1: Basic Algorithms and Operators. New York. Taylor & Francis Group. 2017. С. 331.

16) Lance Chambers. GENETIC ALGORITHMS: New Frontiers Volume II. Boca Raton. 1995 С. 429.

17) Лазуренко Владислав. Швидкий пошук за допомогою мурашиного алгоритму. Збірник наукових праць студентів, аспірантів і молодих вчених «Молода наука-2019». №5. С. 206 – 207.

18) П.П. Штогрин. Використання мурашиного алгоритму для пошуку близького до оптимального маршруту. Міжнародна наукова інтернет-конференція "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення". Тернопіль. 2022. №67. С. 59 – 61.

19) Wu Deng, Junjie Xu, Yingjie Song and Huimin Zhao. An effective improved co-evolution ant colony optimisation algorithm with multi-strategies and its application. Internation Journal of Bio-Inspired Computation. 2020. №3. С. 158 – 170.

20) Marie-Pierre Meurville, Adria C. LeBoeuf. Trophallaxis: the functions and evolution of social fluid exchange in ant colonies (Hymenoptera: Formicidae). Myrmecol. News. 2021. С. 1-30.

21) Juhriyansyah Dalle, Dwi Hastuti, Muhammad Riko Anshori Prasetya. The Use of an Application Running on the Ant Colony Algorithm in Determining the Nearest Path between Two Points. Journal of Advances in Information Technology. 2021. № 3. С. 206-213.

22) Stefka Fidanova. Ant Colony Optimization and Applications. Cham: Springer. 2021. С. 947.

23) Платформа допомоги врятованим. URL: https://www.help-platform.in.ua/gumanitarna-dopomoga/ .

24) Карітас України. URL: https://caritas.ua/about/ .

25) ЄДопомога. URL: https://social.edopomoga.gov.ua/uk/ .

26) De Gruyter. A novel travel route planning method based on an ant colony optimization algorithm. URL: https://www.degruyter.com/document/doi/10.1515/geo-2022-0541/html?lang=en.

27) Дія. Державна допомога малозабезпеченим сім'ям. URL: https://guide.diia.gov.ua/view/pryznachennia-derzhavnoi-sotsialnoi-dopomohy-malozabezpechenym-simiam-2fafff55-1193-488f-b6cd-3cc4dae9faef.

28) GitHub. AntColony. URL: https://github.com/st1lson/AntColony.

29) ProgressTelerik. ASP.NET Core for Beginners: Web APIs. URL: https://www.telerik.com/blogs/aspnet-core-beginners-web-apis#:~:text=An%20ASP.NET%20Core%20Web,for%20working%20with%20HTTP%20requests.

30) Medium. Understanding JWT. URL: https://medium.com/@extio/understanding-json-web-tokens-jwt-a-secure-approach-to-web-authentication-f551e8d66deb.

31) React. URL: https://uk.legacy.reactjs.org/tutorial/tutorial.html.

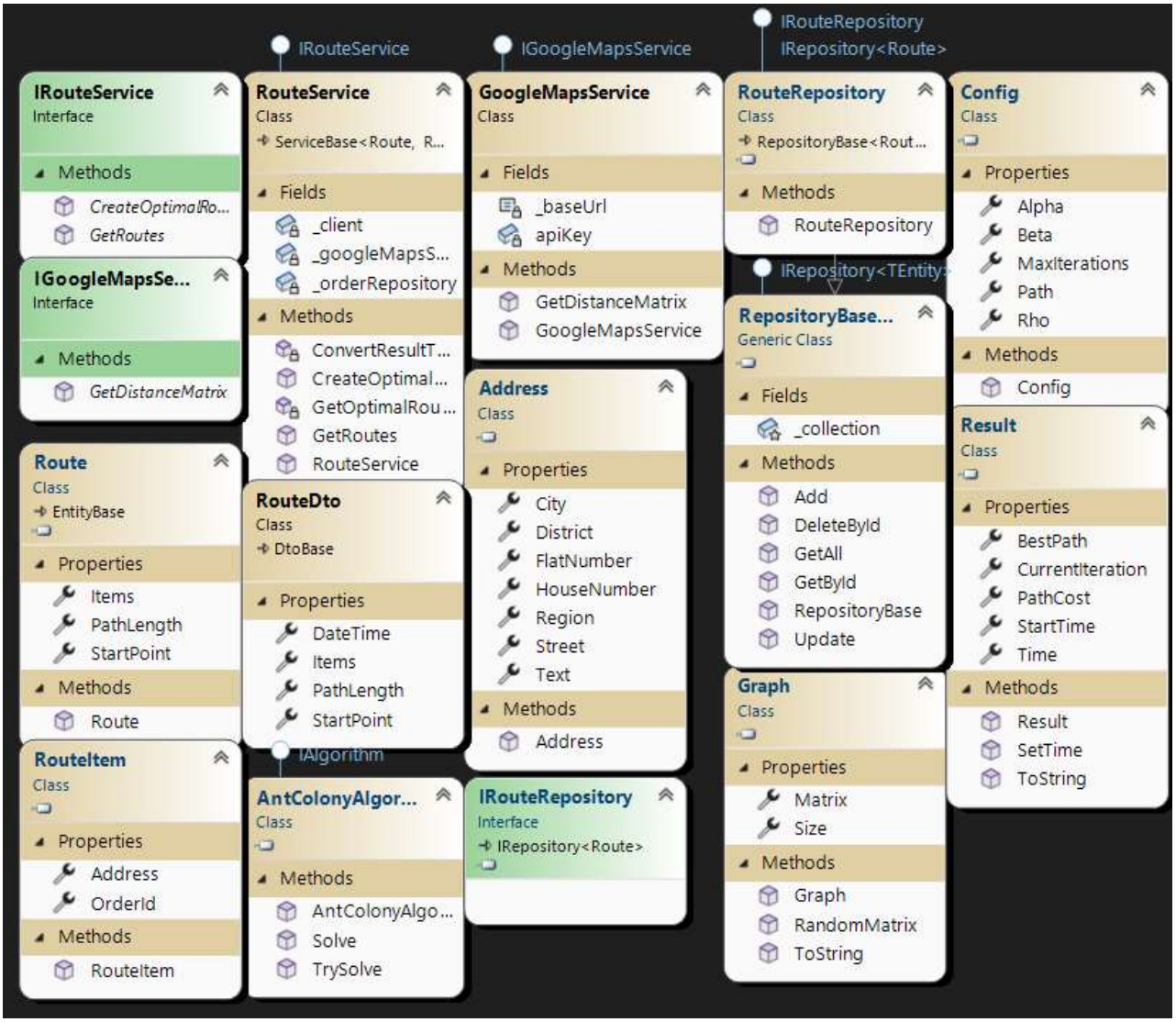
32) Ionic. URL: https://ionicframework.com/.

33) MongoDB. Documentation. URL: https://www.mongodb.com/docs/.

ДОДАТКИ

Додаток А

Розгорнута структура класів методу доставлення з використанням алгоритму мурахи



Додаток Б

Програмні коди

Лістинг GoogleMapsService.cs:

```
using GetAidBackend.Services.Abstractionas;
using Newtonsoft.Json.Linq;

namespace GetAidBackend.Services.Implementations
{
    public class GoogleMapsService : IGoogleMapsService
    {
        private const string _baseUrl = "https://maps.googleapis.com/maps/api";
        private readonly string apiKey;

        public GoogleMapsService()
        {
            apiKey = Environment.GetEnvironmentVariable("GOOGLE_API_KEY");
        }

        public async Task<int[,]> GetDistanceMatrix(string[] addresses)
        {
            int[,] distanceMatrix = new int[addresses.Length, addresses.Length];

            for (int i = 0; i < addresses.Length; i++)
            {
                for (int j = i + 1; j < addresses.Length; j++)
                {
                    string url = $"{_baseUrl}/distancematrix/json?" +
                        $"origins={Uri.EscapeDataString(addresses[i])}&" +
                        $"destinations={Uri.EscapeDataString(addresses[j])}&" +
                        $"key={apiKey}";

                    using HttpClient client = new HttpClient();
                    string json = await client.GetStringAsync(url);

                    JObject response = JObject.Parse(json);
                    JToken element = response["rows"][0]["elements"][0];

                    distanceMatrix[i, j] = (int)element["distance"]["value"];
                    distanceMatrix[j, i] = (int)element["distance"]["value"];
                }
            }

            return distanceMatrix;
        }
    }
}
```

Лістинг RouteService.cs:

```
using AntColony.Algorithm;
using AntColony.Core;
using AntColony.Core.Graphs;
using AutoMapper;
using GetAidBackend.Domain;
using GetAidBackend.Services.Abstractionas;
using GetAidBackend.Services.Dtos;
using GetAidBackend.Storage.Abstractions;
using MongoDB.Driver;

namespace GetAidBackend.Services.Implementations
{
```

```
    public class RouteService : ServiceBase<Route, RouteDto, IRouteRepository>,
IRouteService
    {
        private readonly IOrderRepository _orderRepository;
        private readonly IGoogleMapsService _googleMapsService;
        private readonly IMongoClient _client;

        public RouteService(
            IMongoClient client,
            IRouteRepository repository,
            IOrderRepository orderRepository,
            IGoogleMapsService googleMapsService,
            IMapper mapper)
            : base(repository, mapper)
        {
            _client = client;
            _googleMapsService = googleMapsService;
            _orderRepository = orderRepository;
        }

        public async Task<List<RouteDto>> GetRoutes()
        {
            var result = await _repository.GetAll();
            return GetDtosFromEntities(result);
        }

        public async Task<RouteDto> CreateOptimalRoute(string[] ordersId, Address
startPoint)
        {
            var orders = await _orderRepository.GetByIds(ordersId);
            string[] addresses = new string[] { startPoint.Text };
            addresses = addresses.Concat(orders.Select(_ =>
_.Address.Text).ToArray()).ToArray();

            var distanceMatrix = await _googleMapsService.GetDistanceMatrix(addresses);
            Result result = GetOptimalRoute(addresses.Length, distanceMatrix);

            var route = ConvertResultToRoute(orders, result, startPoint);
            using var session = await _client.StartSessionAsync();
            session.StartTransaction();

            route = await _repository.Add(route, session);
            await _orderRepository.DeliverOrders(ordersId, session);
            await session.CommitTransactionAsync();

            return GetDtoFromEntity(route);
        }

        private Result GetOptimalRoute(int count, int[,] distanceMatrix)
        {
            var random = new Random();
            var config = new Config();
            var graph = new Graph(count, distanceMatrix);

            IAlgorithm antAlgorithm = new AntColonyAlgorithm(graph, config, random);

            return antAlgorithm.Solve();
        }

        private Route ConvertResultToRoute(List<Order> items, Result result, Address
startPoint)
        {
            var orderedItems = new List<Order>();

            for (int i = 1; i < result.BestPath.Count - 1; i++)
            {
```

```
                int index = result.BestPath[i];
                orderedItems.Add(items[index - 1]);
            }

            var route = new Route()
            {
                StartPoint = startPoint,
                Items = orderedItems.Select(_ => new RouteItem() { Address = _.Address,
OrderId = _.Id }).ToList(),
                PathLength = result.PathCost
            };

            return route;
        }
    }
}
```

Лістинг route.ts:

```
import { AxiosRequestConfig } from "axios";
import { sendRequest } from ".";
import { OrderAddress } from "../domain/order";

  export const createRoute = async (orders: string[], startPoint: OrderAddress):
Promise<any> => {
    const request: AxiosRequestConfig = {
      method: "post",
        url:  "admin/routes",
        data: {
            ordersId: orders,
            startPoint: startPoint
        }
        ,
    };
   return await sendRequest(request);
  };

  export const getRoutes = async (): Promise<any> => {
    const request: AxiosRequestConfig = {
      method: "get",
        url:  "admin/routes",
    };
   return await sendRequest(request);
  };
```

Додаток В

Презентаційний матеріал

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим

Виконала:
студентка групи КН-20-2
Зоряна Боревіч

Керівник:
зав. каф. КН, д.т.н., професор
Олександр Бармак

2024

Актуальність

«Справжня людяність полягає в здатності допомагати іншим без очікування на винагороду» – Конфуцій

- В умовах сучасних викликів, таких як економічна нестабільність та соціальні кризи, забезпечення надання основних ресурсів малозабезпеченому населенню стало особливо важливим. В Україні існує значна потреба в ефективних і надійних системах розподілу гуманітарної допомоги. Одним з основних питань, з яким стикаються благодійні організації, є оптимізація процесу доставки, що тягне за собою мінімізацію часу і витрат при забезпеченні своєчасної і точної доставки.
- Військові дії в нашій країні призвели до масового переміщення населення, руйнування інфраструктури та значного збільшення кількості вразливих осіб, які гостро потребують допомоги. У таких умовах ефективна система доставки допомоги є не тільки питанням соціальної відповідальності, але й життєво важливим елементом забезпечення виживання постраждалих та збереження населення України.

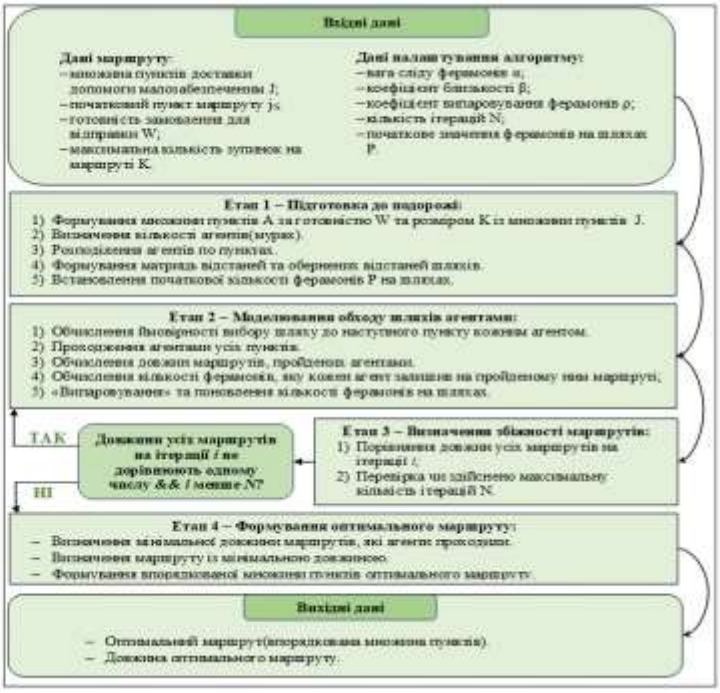
Об'єкт дослідження
Процес доставлення допомоги у веб системі для малозабезпечених людей

Предмет дослідження
Методи оптимізації, еволюційні алгоритми, мурашиний алгоритм у системах планування маршрутів

Мета КРБ
Підвищення ефективності доставлення допомоги у веб системах для малозабезпечених людей

Завдання КРБ
1. Оглянути метод планування маршрутів та існуючих веб реалізацій систем допомоги.
2. Дослідити використання алгоритму мурахи в задачах планування маршрутів та його особливості для вирішення проблеми допомоги малозабезпеченим.
3. Розробити метод доставлення (планування маршруту) за мурашиним алгоритмом, що буде формувати та віддавати як результат оптимальний маршрут для почергового відвідування кожного пункту призначення.
4. Створити вебсистему допомоги малозабезпеченим для користувачів із різними ролями (адміністратор та споживач).
5. Протестувати метод доставлення допомоги за мурашиним алгоритмом та вебсистему, що використовує його.
6. Оцінити покращення ефективності доставлення допомоги за запропонованим методом.

Метод доставлення за мурашиним алгоритмом



Метод доставлення за мурашиним алгоритмом з використанням бібліотеки AntColony

Вхідні дані

- множина пунктів доставки соціальної допомоги J;
- початковий пункт маршруту j₀;
- готовність замовлення для відправки W;
- максимальна кількість зупинок для маршруту K.

Обробка вхідних даних

1) Формування множини пунктів A за готовністю W та розміром K із множини пунктів J.
2) Створення матриці відстаней як екземпляр типу **AntColony.Graph**.
3) Створення об'єкту типу **AntColony.Config** із налаштуваннями алгоритму мурахи.
4) Ініціювання об'єкта класу Random.
5) Створення екземпляру класу new **AntColony.AntColonyAlgorithm**(Graph, Config, Random)

Виконання алгоритму мурахи

Виклик методу **antColonyAlgorithm.Solve()**, що виконує наступні кроки:

- Моделювання обходу шляхів агентами.
- Визначення збіжності маршрутів.
- Формування оптимального маршруту.

Вихідні дані

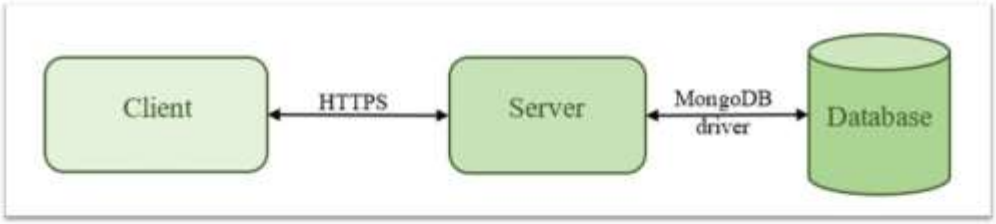
Об'єкт типу **AntColony.Result**, що містить такі поля: оптимальний маршрут, його довжина, час пошуку маршруту

Діаграма класів бібліотеки AntColony

Загальна архітектура вебсистеми та застовані технології

Google Distance Matrix API

Web API with ASP.NET Core 6.0



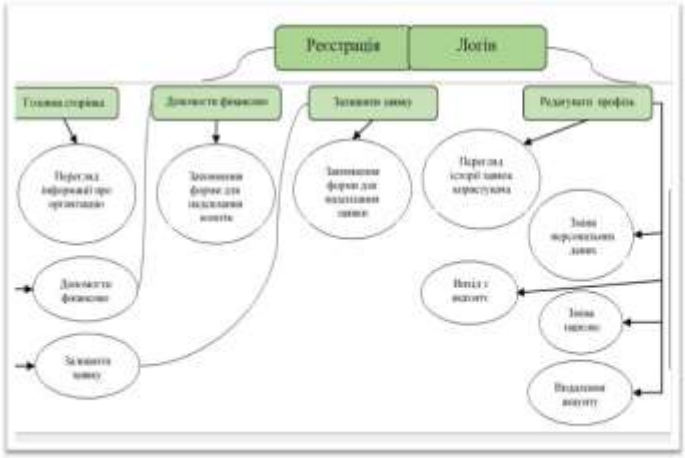
mongoDB.

AXIOS

AntColony

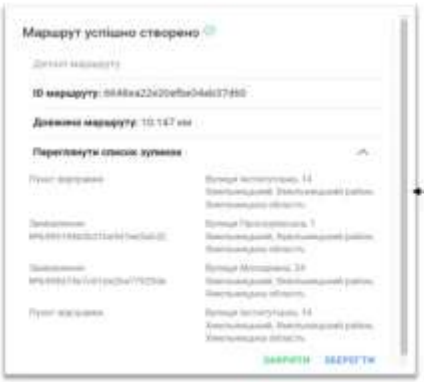
Функціональна структура веб-сайту для користувачів різних ролей

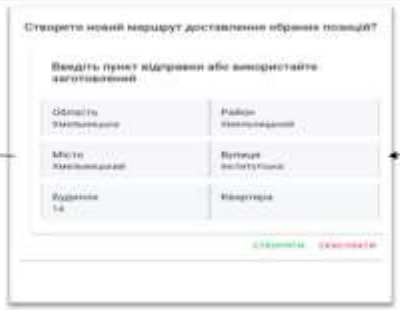
Функції вебсистеми лише для адміністраторів



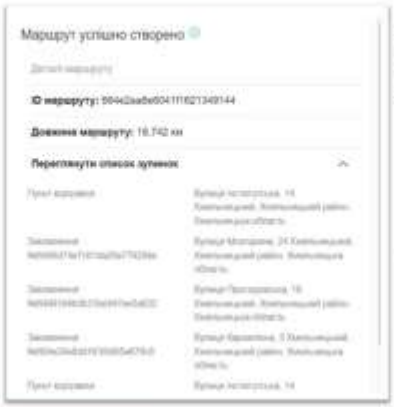
Діаграма функцій вебсайту для споживачів

Етапи створення нового маршруту





Оцінка покращення ефективності доставлення допомоги

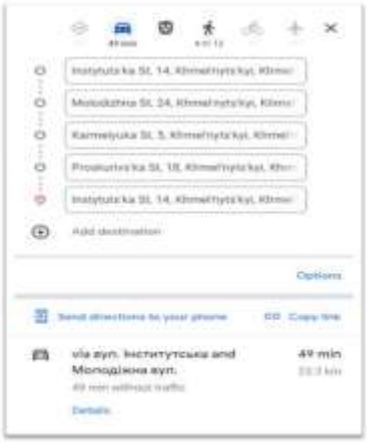


Побудований маршрут методом доставлення

Вхідні дані для побудови маршруту доставлення

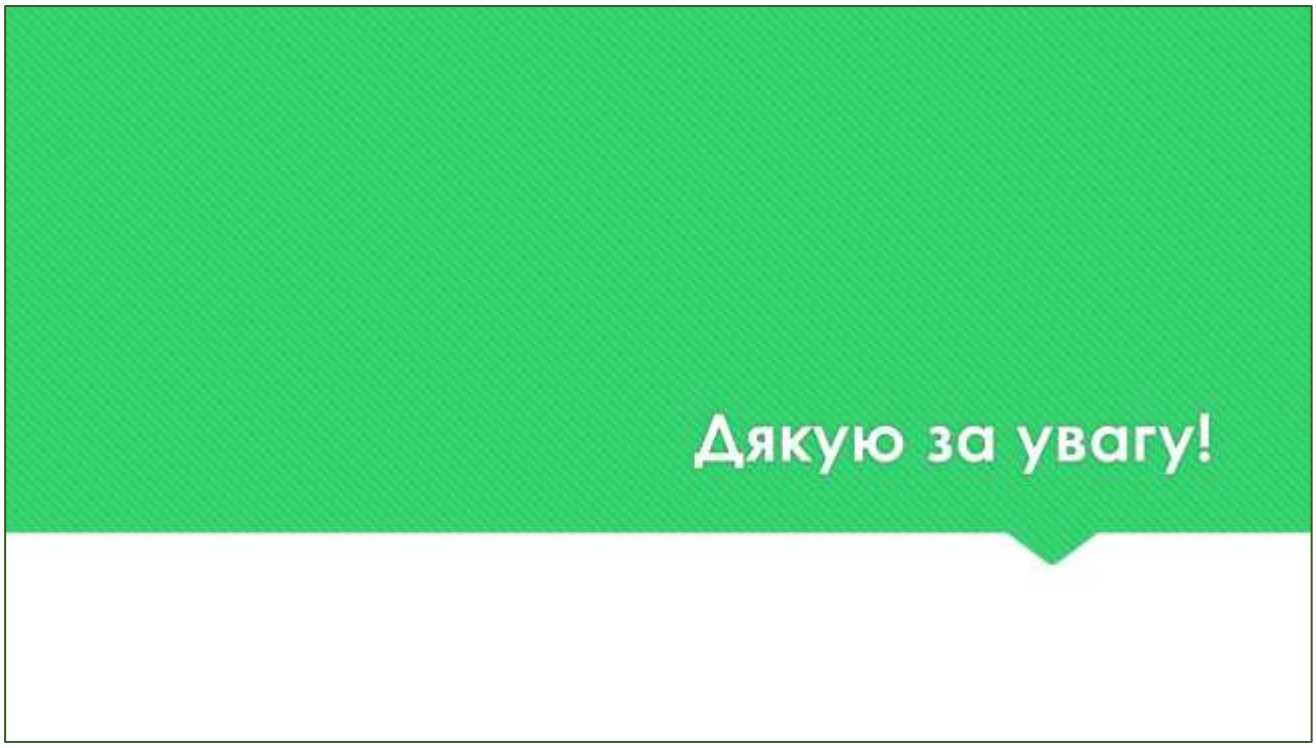
Назва пункту	№Замовлення	Адреса
Пункт відправки	-	Вулиця Інститутська, 14 Хмельницький
Пункт 1	6499194b0b31be947ee5a632	Вулиця Проскурівська, 18 Хмельницький
Пункт 2	6498d74e7c61da26a77929de	Вулиця Молодіжна, 24 Хмельницький
Пункт 3	664e29e8dd1630bf95a678c8	Вулиця Кармелюка, 5 Хмельницький

Побудований маршрут імітованим водієм



Висновки

- У результаті виконання КРБ отримано метод доставлення за мурашиним алгоритмом, що формує та віддає як результат оптимальний маршрут для почергового відвідування кожного пункту призначення. Також було створено вебсистему допомоги малозабезпеченим, що використовує метод доставлення. Метод доставлення та вебсистема успішно пройшли тести.
- Проведено оцінку підвищення ефективності доставлення допомоги за запропонованим методом. Результати аналізу показали, що метод значно підвищує ефективність доставки, формуючи оптимальні маршрути та зменшуючи ризик помилок, допущених людиною.
- Перспектива впровадження розробленого методу доставлення та вебсистеми допомоги малозабезпеченим є досить високою, адже, на жаль, актуальність проблеми низького рівня життя в нашій країні існують і будуть існувати ще довго навіть після омріяного закінчення війни. Тому ефективність доставки такої допомоги необхідно підвищувати, так само, як і надавати доступні сервіси для швидкого оформлення та опрацювання заявки на допомогу.


Дякую за увагу!

Tue Jun 11 14:37:33 EEST 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 0.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

	Документ		Сумарний збіг по Базі Даних	
ID: 129740 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим Додано в БД: 2024-06-11 Автора: Зоряна БОРЕВІЧ Керівники: Олександр БАРМАК Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	91827	848	1634 (2%)	37 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КН

ID перевірки:
1016347992

Дата перевірки:
11.06.2024 16:10:53 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2024 16:16:00 EEST

ID користувача:
100005671

Назва документа: КН-20-2 Боревіч_ЗАПИСКА

Кількість сторінок: 80 Кількість слів: 13428 Кількість символів: 106228 Розмір файлу: 5.31 MB ID файлу: 1016149429

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.8%
Схожість

Найбільша схожість: 2.49% з джерелом з Бібліотеки (ID файлу: 1016145563)

5.33% Джерела з Інтернету 780 ..... Сторінка 82

3.43% Джерела з Бібліотеки 138 ..... Сторінка 86

0% Цитат

Включення цитат вимкнене

Включення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 13

Підозріле форматування 18 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим*

Автор: *студент групи КН-20-2 Зоряна БОРЕВІЧ*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *д.т.н., проф. Олександр БАРМАК*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Зоряни БОРЕВІЧ, не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти що не мають авторства і містять поширені конструкції; серед запозичень знаходяться загальновідомі терміни та скорочення.

Обсяг запозичень, визначений системами виявлення збігів/ідентичності/схожості, складає:

- за системою Anti-Plagiarism – 0%;

- за системою Unicheck – 6.8 %.

Керівник роботи *Олександр БАРМАК*

Гарант ОП *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Боревіч Зоряна Євгеніївна*
за темою: *Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим*

1. Актуальність обраної теми

Актуальність обраної студенткою теми є досить високою, зважаючи на військові дії у Україні, що роблять велику кількість населення малозабезпеченими людьми, яким необхідна допомога. Доставка допомоги має бути якомога ефективнішою, тому розробка методу доставлення за мурашиним алгоритмом є справді актуальною проблемою, що потребує вирішення.

2. Повнота розкриття мети та завдань роботи

Під час виконання кваліфікаційної роботи бакалавра студентка Боревіч Зоряна Євгеніївна повністю розкрила мету роботи, а саме підвищення ефективності доставлення допомоги у вебсистемах для малозабезпечених людей. Провівши оцінку покращення ефективності, вона довела, що мету роботи було досягнуто. Усі поставлені завдання кваліфікаційної роботи бакалавра також було виконано повною мірою.

3. Зміст кожного розділу роботи

У першому розділі роботи було ретельно оглянуто різні методи планування маршрутів та вебреалізації систем допомоги. Другий розділ присвячено методу доставлення за мурашиним алгоритмом, де було розроблено даний метод, з використанням бібліотеки алгоритму мурахи, а також спроектовано вебсистему та базу даних. У третьому розділі було спроектовано загальну архітектуру вебсистеми, розроблено структуру методу доставлення та вебсистеми. Також було проведено тестування системи, детально показано функціональність вебсистеми та проведено оцінку покращення доставлення допомоги з використанням створеного методу. Усі розділи роботи є гарно структурованими та повністю розкривають зміст досліджуваної теми у них.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблений метод доставлення є практичним для використання у вебсистемах допомоги малозабезпеченим, оскільки правильно та швидко виконує розрахунки і видає дійсно найоптимальніший маршрут доставлення. Метод є досить легким для розуміння та використання. Розроблена вебсистема допомоги малозабезпеченим має привабливий та зрозумілий для користувачів інтерфейс, має усі необхідні функції, щоб з нею могли працювати як адміністратори, так і споживачі.

5. Якість оформлення кваліфікаційної роботи бакалавра

Кваліфікаційна робота бакалавра відповідає усім вимогам оформлення, повністю розкриває тему дослідження та її мету. Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

6. Недоліки кваліфікаційної роботи бакалавра

Недоліків у роботі студентки Боревіч Зоряни Євгеніївни не виявлено.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

*Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**».*

Рецензент к.т.н., доцент каф. КІІС 12.06.24
Гнатчук Є.Г.

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Боревіч Зоряни Євгеніївни*

за темою *Метод доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим*

1. Актуальність теми

Актуальним завданням, яке потребує аналізу і досліджується у даній роботі, є розробка методу доставлення за мурашиним алгоритмом для вебсистеми допомоги малозабезпеченим. Для ефективного використання методу доставлення у вебсистемах слід забезпечити виконання функцій створення та редагування нової заявки на допомогу, автоматизоване створення нового маршруту доставлення з використанням методу доставлення з передбаченим вибором початкового пункту, перегляд історії замовлень із статусом їх виконання, перегляд історії маршрутів. Розробка такого метода є актуальною задачею комп'ютерних наук.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом, а саме описом предметної області, об'єктами вивчення та діяльності є математичні, інформаційні, імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи і технології отримання, зберігання, обробки, передачі та використання інформації. Метою роботи саме є підвищення ефективності доставлення допомоги у вебсистемах для малозабезпечених людей, що можна досягнути розробкою методу доставлення за мурашиним алгоритмом. При вирішенні поставленої задачі використано математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, що виникають при розробці вказаного методу. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

При роботі над кваліфікаційною роботою бакалавра Боревіч Зоряна Євгеніївна проявила себе кваліфікованим фахівцем та дисциплінованою студенткою, вчасно виконуючи поставлені етапи дослідження. Як в процесі написання пояснювальної записки, так і при розробці прикладного програмного забезпечення проявила достатні для

одержання успішного результату компетентності та результати навчання. Опанувала професійні скіли за напрямком «Комп'ютерні науки» та достатньо значний софт скіл .

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студентки, яка самостійно виконувала всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показала достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та розроблено програмне забезпечення для валідації та верифікації запропонованого метода.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений у роботі метод та його програмна реалізація може бути використана малозабезпеченими людьми, які потребують допомоги, та адміністраторами благодійних фондів задля підвищення ефективності доставки допомоги.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**» .*

Керівник _____ *д.т.н., проф. зав. каф. КН Олександр БАРМАК*