МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів рішення транспортних задач лінійного програмування для підвищення ефективності доставки питної води
(тема)

Виконав:
Студент 2 курсу, групи ІПЗм-19-1
Батюченко В. А.
(прізвище, ініціали)

Спеціальність 121-Інженерія програмного забезпечення
(код і повна назва спеціальності)

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

Керівник проф. Дудар З. В.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 26 » березня 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Батюченка Віталія Андрійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів рішення транспортних задач лінійного програмування для підвищення ефективності доставки питної води
затверджена наказом університету від 26.03.2021 № 385
2. Термін подання роботи до екзаменаційної комісії 21 травня 2021р.
3. Вихідні дані до роботи *електронні ресурси за обраною тематикою, мінімальні вимоги до функціональності програми, загальні вимоги до архітектури системи*
4. Перелік питань, що потрібно опрацювати в роботі *вступ, аналіз предметної області постановка задачі, перелік вимог до програмної системи, опис дослідження, об'єктних моделей, дослідження методів та алгоритмів, опис розробленої програмної реалізації, аналіз можливих застосувань та експлуатації системи.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів) слайди презентації

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	Проф. Дудар З.В.		20.05.21

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз проблемної області дослідження	25.01.21 – 11.02.21	виконано
2	Аналіз аналогів	7.02.21 – 11.02.21	виконано
3	Розробка постановки задачі	11.02.21 – 14.02.21	виконано
4	Дослідження існуючих методів	14.02.21 – 20.02.21	виконано
5	Моделювання предметної області	15.02.21 – 21.02.21	виконано
6	Планування експериментальної частини дослідження	20.02.21 – 05.03.21	виконано
7	Розробка обраних методів	05.03.21 – 25.03.21	виконано
8	Проведення експериментів	25.03.21 – 10.04.21	виконано
9	Тестування програми	05.04.21 – 14.04.21	виконано
10	Оформлення статті	10.04.21 – 21.04.21	виконано
11	Підготовка пояснювальної записки	01.04.21 – 30.04.21	виконано
12	Підготовка презентації та доповіді	30.04.21 – 03.05.21	виконано
13	Нормоконтроль	07.05.21 – 15.05.21	виконано
14	Рецензування	07.05.21 – 15.05.21	виконано
15	Занесення диплома в електронний архів	17.05.21	виконано
16	Попередній захист	21.05.21	виконано
17	Допуск до захисту у зав. кафедри	24.05.21	виконано

Дата видачі завдання 25 січня 2021р.

Студент _____
(підпис)

Керівник роботи _____ Проф. Дудар З.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 84 с., 22 рис., 10 табл., 24 джер.

ЛІНІЙНЕ ПРОГРАМУВАННЯ, ПИТНА ВОДА, ПОСТАЧАННЯ, ОПТИМІЗАЦІЙНА ЗАДАЧА, СИМПЛЕКС МЕТОД, ASP.NET MVC, SQL SERVER, WEB-САЙТ.

Об’єктом дослідження є оптимізація розподілу кур’єрів між замовленнями в системі доставки питної води.

Метою дослідження є порівняння методів рішення транспортних задач лінійного програмування.

Методи розробки базуються на платформі ASP.NET MVC для серверного додатку з використанням мови програмування C#, на ASP.NET Razor, HTML 5, CSS 3, Boostrap 4, Vue.js та JQuery для клієнтського веб-додтку, на Microsoft SQL Server 2017 для бази даних.

У результаті роботи побудовано математичну модель доставки води кур’єрами, проаналізовано та досліджено методи рішення задач лінійного програмування, розроблена програмна система доставки питної води з автоматичним розподілом кур’єрів між замовленнями.

LINEAR PROGRAMMING, DRINKING WATER, SUPPLY, MATHEMATICAL OPTIMIZATION, SIMPLEX ALGORITHM, ASP.NET MVC, SQL SERVER, WEB-SITE.

The object of the study is to optimize the distribution of couriers between orders in the drinking water delivery system.

The aim of the study is to compare methods for solving transport problems of linear programming.

Development methods are based on the ASP.NET MVC platform for the server application using the C# programming language, on ASP.NET Razor, HTML 5, CSS 3,

Boostrap 4, Vue.js and JQuery for the client web application, on Microsoft SQL Server 2017 for databases.

As a result, a mathematical model of water delivery by couriers was built, methods of solving linear programming problems were analyzed and researched, a software system of drinking water delivery with automatic distribution of couriers between orders was developed.

Я, Батюченко Віталій Андрійович, студент гр. ІПЗм-19-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів рішення транспортних задач лінійного програмування для підвищення ефективності доставки питної води», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Транспортування - одна з найбільших галузей у світі. Це відноситься до переміщення товарів і сировини з одного пункту призначення в інший. Цей процес починається з ланцюжка поставок до доставки готової продукції споживачеві.

Транспортування є ключовим елементом в логістичному ланцюжку. Він об'єднує ті компоненти, які вважаються розділеними. Транспортні системи пов'язують компоненти ланцюжка поставок і повинні належним чином управлятися і контролюватися для успішної спільної роботи транспорту і логістики.

В ручну розробляти ланцюжки доставки не зручно та займає багато часу, то потрібно цей процес автоматизувати. Ця проблема називається — завдання маршрутизації транспортних засобів (VRP). Завдання маршрутизації транспортних засобів (VRP) — це комбінаторна задача оптимізації та цілочисельного програмування, яка задає питання: «Який оптимальний набір маршрутів для парку транспортних засобів, який необхідно перетнути, щоб доставити їх заданому набору клієнтів?». Він узагальнює відому задачу комівояжера (TSP).

Дані задачі можливо промоделювати, привести до задач лінійного програмування та вирішити завдяки методам лінійного програмування.

Транспортна система — це система взаємопов’язаних складових (людей, які задіяні в транспортному процесі; інфраструктури; транспортних засобів тощо), яка призначена для транспортування будь-кого (будь-чого). Для кращої роботи транспортних систем, використовуються програмні забезпечення. Програмне забезпечення дозволяє:

- отримувати інформацію о водії, інформацію о замовленні в режимі реального часу;
- зберігати, змінювати інформацію о користувачеві та продивлятися її;

- робити автоматичний розподіл замовлень між кур’єрами;
- повідомляти о нових задачах водіїв;
- повідомляти о статусі замовлення користувача;
- обчислювати час доставки замовлення;
- спілкуватися з користувачем через чат.

В даній роботі був проведений аналіз проблемної області створення систем маршрутизації транспортних засобів на основі публікацій світових та вітчизняних фахівців в проблемній області (див. додаток А), проаналізовані аналоги систем доставки, проведений аналіз існуючих оптимізаційних моделей та методів, промодельована предметна область доставки води, розроблена математична модель оптимізаційної задачі розподілу замовлень між кур’єрами, спроектована база даних програмної системи доставки питної води, розроблена архітектура програмної системи доставки питної води «WaterDelivery», проведено порівняння обраних методів, реалізована програмна система доставки питної води з оптимізацією розподілу кур’єрів між замовленнями, а також протестована дана система.

Робота пройшла успішну перевірку на академічну доброчесність (див. додаток Б). Також робота відповідає всім критеріям нормоконтролю (див. додаток В).

На основі плану проведення дослідження та його результатів було розроблено презентацію (див. додаток Г). Також за результатами роботи до міжнародної наукової конференції «Проблеми та перспективи науки в Україні» подано до опублікування статтю «Дослідження методів рішення транспортних задач лінійного програмування» (див. додаток Д).

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Аналіз проблемної області створення систем маршрутизації транспортних засобів

На сьогоднішній день все більш і більш люди використовують доставку питної води в офіси, заводи, університети і т.п. Якщо вбити в пошукову систему запит «доставка питної води», то ви побачити багато варіантів. Це показує, що сервіси доставки води, дуже популярні. Чиста вода швидко стає все більш цінним товаром, а потреба і доступність завжди ведуть до підвищення цін. Таким чином, потенціал зростання досить широкий. На ринку завжди існує небезпека перенасичення, але, оскільки вода життєва необхідна для виживання, це здається малоймовірним в найближчому майбутньому.

Гарною ідеєю буде спростити дії користувачів та організації для коректної роботи сервісу. Це можливо завдяки додавання алгоритмів для автоматизації. Наприклад можливо додати автоматичне формування маршрутів для доставки води користувачеві, тобто вирішити транспортну задачу.

Транспортна задача — це задача, в якій потрібно розробити оптимальну схему доставки продукту із пунктів відправлення до пункту призначення.

В доставці води користувачу потрібно, щоб йому доставили потрібну кількість води в назначений час. Тому під оптимальність мається на увазі доставити всім користувача потрібну кількість води в назначений час, щоб транспорт затратив менше ресурсів, тобто палива. В даній задачі пунктами призначення являються користувачі системи.

Транспортні засоби невеликі і їм потрібно повертатися на склад щоб взяти нові бутилі з водою для користувачів. Тобто пунктом відправлення являються склади з балонами води.

Математичне програмування — це область математики, що розробляє теорію, чисельні методи рішення багатовимірних задач з обмеженнями. На відміну від класичної математики, математичне програмування займається

математичними методами вирішення завдань знаходження найкращих варіантів з усіх можливих [1]. Методи математичного програмування використовуються в економічних, організаційних, військових та інших системах для вирішення так званих розподільчих завдань. Розподільні завдання (РЗ) виникають при необхідності розподілити обмежені ресурси по роботах відповідно до обраного критерію оптимальності.

Лінійне програмування (ЛП) є найбільш простим і найкраще вивченим розділом математичного програмування [2].

Для вирішення оптимізаційної задачі потрібно змоделювати задачу розподілу кур’єрів між заказами, так як алгоритми працюють з моделями [3].

Транспортну задачу представляють у вигляді лінійної задачі. Де у нас є функція розрахунку затрачених ресурсів для доставки продукту користувачеві, а також обмеження, такі як час доставки, вантажопідйомність транспорту та інші.

Транспортну задачу можливо привести к оптимізаційній задачі. Тому що оптимізаційна задача — це задача знаходження екстремуму певної функції в деякій області векторного простору, обмеженою набором лінійних та/або нелінійних рівності та/або нерівностей.

Один із популярних методів рішення лінійних задач являється симплекс метод. Симплекс-метод — метод розв'язання задачі лінійного програмування, в якому здійснюється скерований рух по опорних планах до знаходження оптимального розв'язку; симплекс-метод також називають методом поступового покращення плану.

Альтернатива симплекс методу являється алгоритм Кармаркара.

Алгоритм Кармаркара — це алгоритм, представлений Нарендра Кармаркаром в 1984 для вирішення задач лінійного програмування. Поточне допустиме рішення не пересувається по межі області допустимих рішень як в симплекс-методі, а рухається по внутрішнім точкам області допустимих значень, покращуючи з кожною ітерацією апроксимацію оптимального вирішення певним дробом і приводячи до оптимального рішення з раціональними даними.

Також для рішення алгоритмічних задач, можливо використовувати

жадібний алгоритм.

Жадібний алгоритм — простий і прямолінійний евристичний алгоритм, який приймає найкраще рішення, виходячи з наявних на поточному етапі даних, не турбуючись про можливі наслідки, сподіваючись врешті-решт отримати оптимальне рішення. Легкий в реалізації і часто дуже ефективний за часом виконання. Багато задач не можуть бути розв'язані з його допомогою.

Щоб зрозуміти який із алгоритмів більш підходить для вирішення транспортних задач, потрібно порівняти їх роботу та результати один з одним на певних тестових вхідних даних.

Для цього потрібно написати програми, які зможуть приймати вхідні дані (кількість машин, карту місцевості, задачі на доставку) та знаходити оптимальне рішення чи приблизно оптимальне рішення за допомогою одного з трьох алгоритмів на вибір. Також програми повинні вимірювати час виконання алгоритмів.

Для задання вхідних даних та роботи з даними користувачів, краще всього підходить база даних , тому що це нам дасть: компактність - інформація зберігається в БД, немає необхідності зберігати багатотомні паперові картотеки; швидкість - швидкість обробки інформації (пошук, внесення змін) комп'ютером набагато вище ручної обробки; низькі трудовитрати - немає необхідності в стомлюючої ручній роботі над даними; застосовність - завжди доступна свіжа інформація. Додаткові переваги з'являються при використанні БД в середовищі з багатьма користувачами, оскільки стає можливим здійснювати централізоване управління даними. Інформаційна система, що працює на основі БД, повинна включати в свій склад засоби, що забезпечують потреби користувачів різних категорій на всіх етапах життєвого циклу систем БД: проектування, створення, експлуатації. Інформаційна система дозволяює вирішувати цілий комплекс завдань, серед яких найбільш важливими є: зберігання інформації, швидкий пошук інформації за ознаками, систематизація інформації, обробка інформації, синтез нової інформації на підставі інформації бази [4].

Для гарною роботи сервісу та підтримки його роботи потрібно розробити

гарну архітектуру. Архітектура програмного забезпечення — сукупність найважливіших рішень про організацію програмної системи. Архітектура включає: вибір структурних елементів і їх інтерфейсів, за допомогою яких складена система, а також їх поведінки в рамках співпраці структурних елементів; з'єднання обраних елементів структури і поведінки у все більш крупні системи; архітектурний стиль, який направляє всю організацію - все елементи, їх інтерфейси, їх співпраця і їх з'єднання. Документування архітектури програмного забезпечення спрощує процес комунікації між розробниками, дозволяє зафіксувати прийняті проектні рішення і надати інформацію про них експлуатаційного персоналу системи, повторно використовувати компоненти і шаблони проекту в інших [5]. Одна із популярних та перевірених часом архітектура являється багаторівнева архітектура, яка дозволяє створювати гнучкі і повторно-використовуванні програми. Поділяючи додаток на рівні абстракції, розробники набувають можливість внесення змін в якийсь певний шар, замість того, щоб переробляти все додаток цілком [6].

Доставка води, чи чогось іншого, — це процес де організація маючи якийсь-то товар, який потрібен групі людей, і за допомогою засобів доставки, доставляють цей продукт людям. Щоб розвити свій сервіс, компанії створюють рекламу, для цього гарно підходить сайт. Тому що багато людей витрачають багато часу в інтернеті і за допомогою покупки реклами у пошукових систем, можливо просувати свій бізнес.

Вже одне наявність корпоративного сайту характеризує компанію як сучасну, яка розуміє тенденції світової економіки і йде в ногу з часом. Але мало просто мати сайт, потрібно зробити так, щоб він створював і підтримував найбільш вірний спосіб товару або послуги. Будь аспект сайту - дизайн, функціональність, текст - все має велике значення для формування позитивного образу марки [7].

Сайт компанії можна створювати і розглядати покладаючись на різні аспекти. Для кого-то це засіб спілкування з користувачами, для кого-то нова сходинка в розширенні своєї аудиторії, а для кого-то сайт виступає черговий

осередком у розвитку бізнесу, якимсь засобом інтернет-маркетингу.

Розглядати сайт, як інструмент ефективної реклами можна. Але на одному створенні сайту ваша рекламна інтернет-компанія не закінчується. Підходи до створення сайту можуть відрізнятися і якщо ви вибрали саме цей, то варто задуматися, як витягти зі свого сайту максимальну користь.

Вплив сайту на інтернет-аудиторію незаперечно, але може принести як корисні плоди, так і неприємні опади. Важливо не забувати про те, що створюючи свою рекламну площадку в інтернеті ви не можете впасти в бруд обличчям. На вас ляжуть нелегкі турботи полягають в розкручуванні сайту, наповнення та оновлення його новим унікальним і цікавим контентом. Для реклами в інтернеті одного створення сайту недостатньо. Щоб вийти на вершину своєї популярності і затребуваності потрібно докласти чимало зусиль і вирішити не один десяток проблем [8].

1.2 Аналіз існуючих аналогів

Існує безліч сервісів доставки води по всьому світі. Переважно ці сервіси обслуговують одне місто, країну, штат та так далі. Найчастіше, вони не відрізняються один від одного, доставляють певну води з якогось джерела, переважно в бутлях об’ємом 18,9 літрів (5 галонів).

DS Services - історично, основна діяльність була побудована на забезпеченні безпечної, великої дегустації бутильованої води в будинки та офіси по всій території США. Крім цієї основи їхнього бізнесу, можуть запропонувати власні кавові напої під двома домашніми брендами: Standard Coffee® та Javarama. А під брендом Relyant® вони також пропонують системи фільтрації води, обладнання та послуги. Також, вони забезпечують безпечну питну воду під час надзвичайних ситуацій та стихійних лих [9].

DS Services надає чашки, мішалки, кришки, рукава, столові прилади,

соломинки і серветки. А також, продає установи для заварювання кави (бутель ставиться знизу установи, а кофе чи чай засипається зверху, чашка також зверху, висота установи приблизно 1 метр), для подачі води (бутель ставиться знизу, його не видно, тому в установі присутня лампочка, яка повідомляє мало чи ні води в ємкості, висота 1 метр), для подачі води з нагріванням чи охолодженням її (бутель ставиться зверху в перевернутому стані, висота установи приблизно 1 метр + висота бутля) [9].

Також сервіси доставки води є і в Україні. Один із них — My Water. My Water — офіційний інтернет-магазин групи компаній IDS Borjomi Ukraine, яка є оператором на ринку видобутку та доставки води. Більше 20 років пропонує покупцям воду, розлиту по європейським стандартам якості. На сьогоднішній день представляє 6 власних торгових марок, в тому числі Моршинська, Миргородська та Borjomi. Має більше 200 000 клієнтів по всій Україні [10].

В Харкові є сервіс доставки питної води Роганська. Доставка проводиться у офіси, дома, квартири. Питна вода Роганська — природна питна вода, яку видобувають із водоносного горизонту глибиною 687 метрів, фільтрована за технологіями з обробкою ультрафіолетом. Питна вода "жива" і відповідає екологічним вимогам Євросоюзу [11].

Аналіз та порівняння програмних аналогів з «WaterDelivery» наведено у таблиці 1.1.

1.3 Постановка задачі

Під час виконання кваліфікаційної роботи потрібно провести дослідження методів рішення транспортних задач для підвищення ефективності доставки питної води.

В ході виконання роботи потрібно вирішити наступні завдання:

– провести аналіз існуючих методів рішення транспортних задач лінійного програмування та обрати методи для подальшого дослідження;

– побудувати математичну модель транспортної задачі лінійного програмування в області доставки питної води;

– провести аналіз та моделювання (UML, концептуальне) предметної області поставки питної води;

– спроектувати і розробити базу даних для системи;

– спроектувати і розробити архітектуру програмної системи;

– реалізувати методи рішення оптимізаційних задач;

– програмно реалізувати систему доставки питної води;

– побудувати план експерименту (підготувати вибірки даних, вибрати показники для оцінки ефективності методів), провести експеримент та розробити рекомендації щодо використання методів.

Таблиця 1.1 — Аналіз та порівняння програмних аналогів з «WaterDelivery»

Критерії	DS Services	My Water	Роганська	WaterDelivery
Приємний і зрозумілий інтерфейс	+	-	-	+
Замовлення на певний час	-(певний графік доставки)	- (з 8-19)	-	+
Повідомлення про доставку	+	+	+	+
Доставка окрім великих тар води	+	-	+	+
Автоматичний розподіл завдань між кур’єрами	?	?	?	+

На основі результатів дослідження необхідно змоделювати та розробити програмну систему доставки питної води.

Програмна система повинна мати три категорій користувачів:

– користувач – може налаштовувати датчик, зробити заказ води, перевірити наявність води в тарі;

– адміністратор – може добавляти продукт на склад, добавляти кур’єра, назначити кур’єра на заказ, добавити користувачеві датчик;

– кур’єр – може продивлятися задачі, доставити продукт користувачеві.

Для коректної роботи програмною системи, повинно бути реалізоване наступне:

– можливість реєстрації нових користувачів;

– можливість роботи замовлення;

– автоматичний розподіл замовлень для кур’єрів;

– відображення заказів для кур’єрів.

Для реалізації програмної системи були обрані наступні технології. Для реалізації back-end буде використовуватися фреймворк ASP .NET MVC 5. Для доступу до реляційних даних буде використовуватися СКБД MS SQL Server 2019, ORM Entity Framework. Для створення сторінок для веб-додатку буде використовуватися ASP.NET Razor, HTML 5, CSS 3, Boostrap 4, Vue.js та JQuery.

2 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

2.1 Аналіз проблемної області моделювання оптимізаційних задач

Математичне програмування - це математична дисципліна, в якій розробляються методи відшукання екстремальних значень цільової функції серед безлічі її можливих значень, що визначаються обмеженнями. Наявність обмежень робить завдання математичного програмування принципово відмінними від класичних задач математичного аналізу з відшукання екстремальних значень функції. Методи математичного аналізу для пошуку екстремуму функції в задачах математичного програмування виявляються непридатними. Для вирішення завдань математичного програмування розроблені і розробляються спеціальні методи і теорії. Так як при вирішенні цих завдань доводиться виконувати значний обсяг обчислень, то при порівняльній оцінці методів велике значення надається ефективності і зручності їх реалізації на ЕОМ. Математичне програмування можна розглядати як сукупність самостійних розділів, які займаються вивченням і розробкою методів вирішення певних класів задач.

Залежно від властивостей цільової функції і функції обмежень всі завдання математичного програмування діляться на певні класи:

- задачі лінійного програмування;
- задачі нелінійного програмування (які, в свою чергу, складаються з задач опуклого програмування, сепарабельного програмування, квадратичного програмування, цілочисельного програмування та стохастичного програмування);
- задачі динамічного програмування;
- тощо.

Якщо цільова функція і функції обмежень - лінійні функції, то відповідна задача пошуку екстремуму є задачею лінійного програмування. Якщо хоча б одна із зазначених функцій нелінійна, то відповідна задача пошуку екстремуму є завданням нелінійного програмування [12].

Лінійне програмування — це один із методів досягнення найкращого варіанту у математичній моделі чиї вимоги представлені через лінійні відношення. Лінійне програмування входить до математичного програмування (математичної оптимізації).

Більш формально, лінійне програмування це інструмент для оптимізації лінійної цільової функції, яка обмежена лінійними рівняннями і лінійними нерівностями. Її допустима множина є опуклим політопом, який є множиною визначеною як перетин скінченної кількості півпростірів, кожен з яких визначає лінійна нерівність. Її цільова функція є дійсно-значима афінна функція визначена на цьому багатограннику. Алгоритм лінійного програмування знаходить точку на багатограннику де ця функція набуває найбільшого чи найменшого значення якщо така точка існує [13].

Лінійне програмування застосовується при вирішенні економічних задач, в таких завданнях як управління і планування виробництва; в задачах визначення оптимального розміщення устаткування на морських судах, в цехах; в задачах визначення оптимального плану перевезень вантажу (транспортна задача); в задачах оптимального розподілу кадрів і т.д [12].

Задачі лінійного програмування (ЗЛП) модна поділити на класи:

- загальні розподільчі задачі;
- задачі про призначення;
- задачі про суміш;
- транспортні задачі.

Незалежно від способу розв'язання транспортної задачі, мета рішення — знайти оптимальний план перевезення вантажу (план з найменшими транспортними витратами), якщо відомі запаси вантажу на складах, кількість вантажу, необхідного кожному споживачеві і транспортні витрати на перевезення одиниці вантажу з кожного складу будь-якому споживачеві [14].

На таблиці 2.1 представлений загальний опис оптимізаційної задачі. На ній представленні ресурси, які повинні виконати певні роботи. На роботи та ресурси накладаються певні обмеження. Тобто нам потрібно в оптимізаційній задачі

добитися мінімальних затрат на виконання робіт нашими ресурсами, так щоб всі обмеження були дотриманні.

Таблиця 2.1 Табличний вигляд оптимізаційної задачі

	Роботи			Обмеження на ресурси
Ресурси	Робота 1	…	Робота N	
Ресурс 1	Затрати 1 1	…	Затрати 1 N	Обмеження 1
…	…	…	…	…
Ресурс M	Затрати M 1	…	Затрати M N	Обмеження M
Обмеження на роботи	Обмеження 1	…	Обмеження M	

У процесі дослідження об'єкта часто буває недоцільно або навіть неможливо мати справу безпосередньо з досліджуваним об'єктом. Зручніше буває замінити його іншим об'єктом, подібним даним у тих аспектах, які важливі в цьому дослідженні. У загальному вигляді модель можна визначити як умовний образ (спрощене зображення) реального об'єкта, який створюється для більш глибокого вивчення дійсності. Метод дослідження, що базується на розробці і використанні моделей, називається моделюванням. Необхідність моделювання обумовлена складністю, а часом і неможливістю прямого вивчення реального об'єкта. Значно доступніше створювати і вивчати прообрази реальних об'єктів, тобто моделі. Можна сказати, що теоретичне знання про що-небудь, як правило, являє собою сукупність різних моделей. Ці моделі відображають істотні властивості реального об'єкта, хоча насправді дійсність значно набагато змістовніші і багатше.

Модель - фізична система або символьне опис, що відображає істотні, з точки зору мети дослідження, властивості або характеристики досліджуваного об'єкта. Модель, відображаючи або відтворюючи об'єкт дослідження, здатна заміщати його так, що її вивчення дає нову інформацію про цей об'єкт.

Характерні особливості завдань лінійного програмування (ЛП) наступні:

– критерій оптимальності (цільова функція) f(X) являє собою лінійну функцію від рішення X = (x1, x2, ..., xn);

– обмежувальні умови, які накладаються на можливі рішення, мають вигляд лінійних рівностей або нерівностей.

Загальна задача лінійного програмування (ЛП) математично може бути сформульована таким способом:

Цільова функція (ЦФ)

$$f(X) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow \max\ (\min), \tag{2.1}$$

де c_i задані коефіцієнти,

x_j - шукані змінні

при обмеженнях

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n\ (\leq, =, \geq,)\ b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n\ (\leq, =, \geq)\ b_2, \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n\ (\leq, =, \geq)\ b_m, \\ x_j \geq 0,\ \left(j = \overline{1, n}\right). \end{cases} \tag{2.2}$$

Кожна строчка представляє якесь обмеження для наших шуканих змінних, де a_{ij} задані коефіцієнти, а b_i представляє значення обмеження.

У компактному вигляді, використовуючи знак суми Σ, завдання (2.1) та (2.2) можна записати так:

$$f = \sum_{j=1}^{n} c_j x_j \rightarrow \max\ (\min), \tag{2.3}$$

$$\begin{gathered} \sum_{j=1}^{n} a_{ij} x_j\ (\leq, =, \geq) b_i\ ,\ \left(i = \overline{1, m}\right), \\ x_j \geq 0, \quad \left(j = \overline{1, n}\right). \end{gathered} \tag{2.4}$$

Тут x_j - шукані змінні; вираз (2.3) задає функцію мети; вираз (2.4) - обмеження задачі ЛП [2].

Таким чином, побудувавши вище описану математичну модель транспортної задачі, можна переходити до її вирішення одним з існуючих методів.

2.2 Аналіз та вибір методів рішення транспортної задачі лінійного програмування

Задача розподілу заказів між кур'єрами відноситься до лінійного програмування. Лінійне програмування — метод досягнення найліпшого виходу (такого як найбільший прибуток або найменша вартість) у математичній моделі чиї вимоги представлені через лінійні відношення. Лінійне програмування є особливим випадком математичного програмування (математичної оптимізації).

Більш формально, лінійне програмування є технікою для оптимізації лінійної цільової функції, що обмежена лінійними рівняннями і лінійними нерівностями. Її допустима множина є опуклим політопом, який є множиною визначеною як перетин скінченної кількості півпросторів, кожен з яких визначає лінійна нерівність. Її цільова функція є дійсно-значима афінна функція визначена на цьому багатограннику. Алгоритм лінійного програмування знаходить точку на багатограннику де ця функція набуває найбільшого чи найменшого значення якщо така точка існує.

З різноманітних методів рішення лінійного завдань можливо виділити:

- жадібний алгоритм;
- графічний метод;
- метод еліпсоїдів;
- симплекс метод;
- алгебраїчний метод;

– алгоритм Кармаркара.

Жадібний алгоритм має такі переваги, як легкість розробки, найчастіше алгоритм легко придумати та реалізувати, швидко виконується порівняно з другими алгоритмами. Але жадібний алгоритм не завжди видає 100% оптимальний варіант рішення задачі, також інколи не знаходить вірне рішення задачі лінійного програмування, тобто деякі обмеження не були дотримані.

Графічним методом можна вирішувати, в основному, завдання лінійного програмування, які мають дві змінні. У разі трьох змінних графічний метод стає менш наочним, а при більшому числі змінних - неможливим. Він заснований на геометричному поданні допустимих рішень і ЦФ завдання.

Метод еліпсоїдів - алгоритм знаходження точки, що лежить в перетині опуклих множин.

Якщо в задачі лінійного програмування вдалося побудувати кулю, що містить шукане рішення, то вона може бути вирішена методом еліпсоїдів. Для цього спочатку знаходимо якусь точку u всередині кулі, що задовольняє обмеженням завдання. Проводимо через неї гіперплоскість f (x) <f (u), де f - цільова функція, і знаходимо точку в перетині вхідних і нової гіперплоскостей (починаючи з поточного еліпсоїда) [15].

Симплекс-метод — це спосіб вирішення задачі лінійного програмування. Ідея цього метода полягає в тому, що здійснюється послідовний рух по опорних планах, поки не буде знайдено оптимального розв'язку; симплекс-метод також називають методом поступового покращення плану [16].

Симплекс метод отримує набір лінійних нерівностей або лінійні цільову функцію і знаходить оптимальну допустиму точку.

Лінійне програмування це максимізація або мінімізація деякого лінійного функціоналу на багатовимірному просторі при заданих лінійних обмеженнях.

Зауважимо, що кожне з лінійних нерівностей на змінні обмежує півпростір у відповідному лінійному просторі. В результаті всі нерівності обмежують деякий багатогранник (можливо, нескінченний), званий також багатогранний комплексом. Рівняння W (x) = c, де W (x) – максимізує (або мінімізує) лінійний

функціонал, породжує гіперплощину L (c). Залежність від c породжує сімейство паралельних гіперплощин. Тоді екстремальна задача набуває наступне формулювання - потрібно знайти таке найбільше c, що гіперплощина L (c) перетинає багатогранник хоча б в одній точці. Зауважимо, що перетин оптимальної гіперплощини і багатогранника буде містити хоча б одну вершину, причому, їх буде більше однієї, якщо перетин містить ребро або k-мірну грань. Тому максимум функціоналу можна шукати в вершинах багатогранника. Принцип симплекс-методу полягає в тому, що вибирається одна з вершин багатогранника, після чого починається рух по його ребрах від вершини до вершини в сторону збільшення значення функціоналу. Коли перехід по ребру з поточної вершини в іншу вершину з більш високим значенням функціоналу неможливий, вважається, що оптимальне значення c знайдено.

Послідовність обчислень симплекс-методом можна розділити на дві основні фази [17]:

- знаходження вихідної вершини безлічі допустимих рішень;
- послідовний перехід від однієї вершини до іншої, що веде до оптимізації значення цільової функції.

Алгоритм Кармаркара — це алгоритм, представлений Нарендра Кармаркаром в 1984 для вирішення задач лінійного програмування. Поточне допустиме рішення не пересувається по межі області допустимих рішень як в симплекс-методі, а рухається по внутрішнім точкам області допустимих значень, покращуючи з кожною ітерацією апроксимацію оптимального вирішення певним дробом і приводячи до оптимального рішення з раціональними даними.

Алгоритм Кармаркара належить класу методів внутрішньої точки - поточний допустиме рішення не пересувається по межі області допустимих рішень як в симплекс-методі, а рухається по внутрішнім точкам області допустимих значень, покращуючи з кожної ітерації апроксимацію оптимального вирішення певної дробом і приводячи до оптимального рішення з раціональними даними [18].

Так як цільова функція матиме більше 2-х змінних, то графічний метод не підійде для знаходження рішення. Також так як не завжди можливо побудувати шар для метода еліпсоїдів, то він також не підходить для знаходження рішення. Тобто для експерименту підходять на ступні методи:

- жадібний алгоритм;
- симплекс метод;
- алгоритм Кармаркара.

2.3 Аналіз та UML-моделювання предметної області доставки води

На сьогоднішній день все більш і більш люди використовують доставку питної води в офіси, заводи, університети і т.п. Якщо вбити в пошукову систему запит «доставка питної води», то ви побачити багато варіантів. Це показує, що сервіси доставки води, дуже популярні. Чиста вода швидко стає все більш цінним товаром, а потреба і доступність завжди ведуть до підвищення цін. Таким чином, потенціал зростання досить широкий. На ринку завжди існує небезпека перенасичення, але, оскільки вода життєва необхідна для виживання, це здається малоймовірним в найближчому майбутньому [19].

Доставляти можна ємкості з водою різної форми та об’єму. Такі як пластикові пляшки об’ємом 0.2 л., 0.5 л., 1 л. та 2 л., а також бутлі з водою. Вони всі витягнутої форми.

Певну кількість деяких ємності з водою, користувач хоче, щоб йому доставили в певний період часу до певної адреси. Користувач буде не задоволений якщо, кількість, тип ємкості не буде співпадати зі списком замовлення, або не доставлять в назначений час або не на правильну адресу. Найчастіше користувач може почекати 1 день після замовлення, так як слідкувати за кількістю води легко та можливо розпланувати цю затримку доставки питної води.

Кур’єр керує транспортним засобом, який потребляє ресурс для поїздки. За цей ресурс потрібно платити і кількість його споживання залежить від пройденої відстані. Заправляти транспорт ресурсом легко і не займає багато часу, тобто не сильно впливає на час поїздки. Транспортний засіб має певну вантажопідйомність. Тому він може взяти не більш певного об’єму (кількості) тар з водою. Це значить, що можливо кур’єрам потрібно буде повертатися на склад за новими тарами.

Адміністратор слідкує за кур’єрами та за системою. Перевіряє, що всі близькі за часом доставки замовлення доставляються, а також наступні замовлення розподіленні між кур’єрами, щоб вони могли планувати план поїздки. Може сам розподіляти замовлення між кур’єрами. Слідкує, що кур’єри вчасно отримують інформацію о їх замовленнях, аналізує відгуки користувачів. Адміністратор, може добавляти нових користувачів, кур’єрів, а також редагувати їх інформацію.

UML – уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

На рисунку 2.1 наведено діаграму варіантів використання (Use Case), яка має трьох акторів – користувач програмної системи, адміністратор та кур’єр.

Користувач може зареєструватися, після авторизації він може зробити замовлення на доставку йому води.

Адміністратор може зареєструватися, після авторизації він може додати до системи кур’єра, додати продукт до складу, призначити кур’єра на замовлення, запустити автоматичний розподіл кур’єрів між замовленнями.

Кур’єр може авторизуватися, після цього він може переглянути список своїх завдань, відмітить факт того, що він взяв продукт зі складу, відзначити у системі вдалу чи невдалу доставку продукту користувачеві.

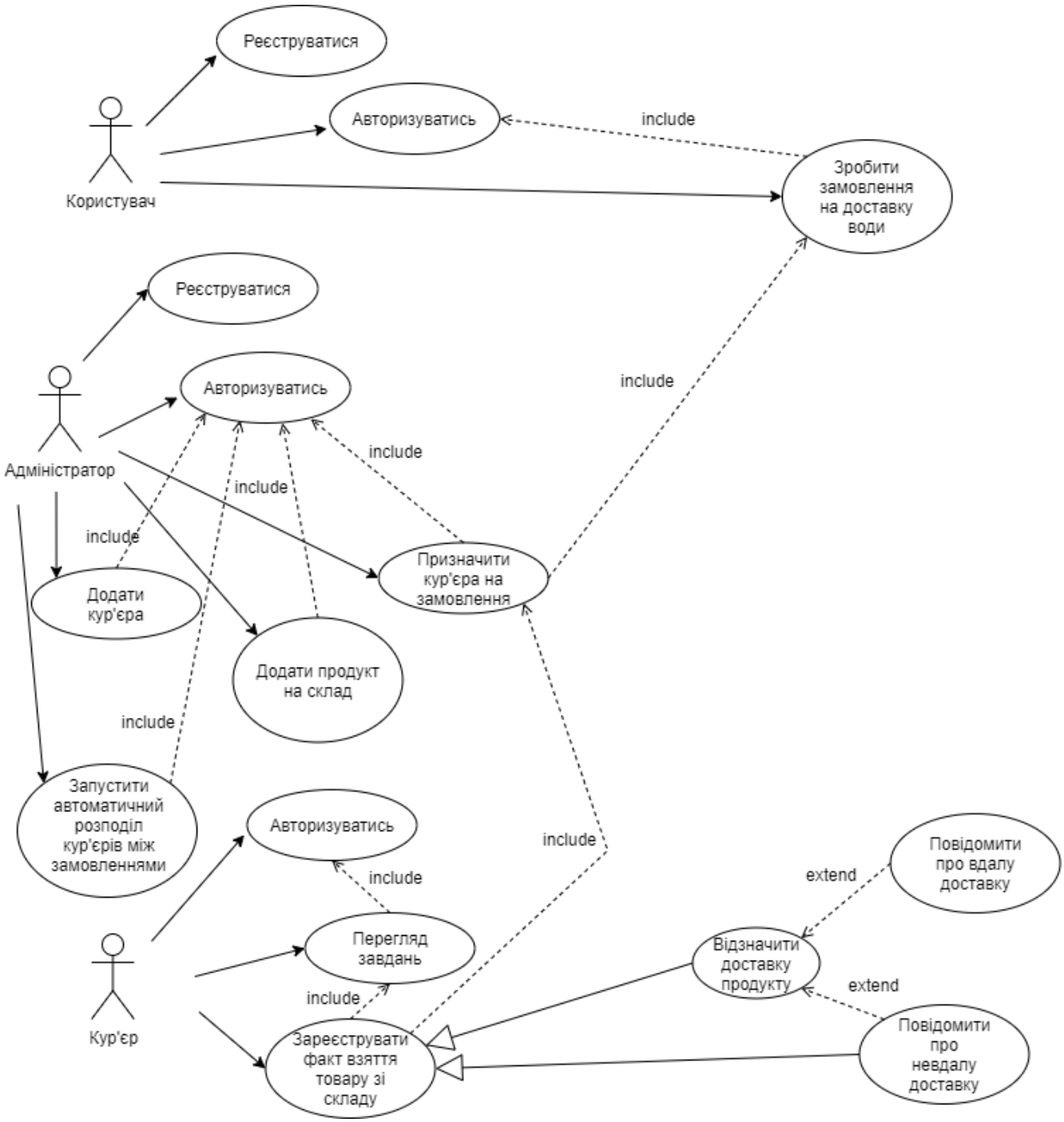

Рисунок 2.1 – UseCase-діаграма

Схема взаємозв’язку об’єктів предметної області доставки питної води приведена на рисунку 2.2 у вигляді загальної діаграми класів.

На підставі концептуального моделювання предметної області доставки питної води отримані наступні сутності:

– сутність Адреса з атрибутами: ID адреси, повна адреса;

– сутність Склад з атрибутами: ID складу, ID адреси;

– сутність Тип тари з атрибутами: ID типу тари, ціна тари з водою, об’єм, кількість на складі;

– сутність Тара для води з атрибутами: ID тари для води, ID складу, ID типу тари;

– сутність Замовлення з атрибутами: ID замовлення, ID адреси, ID користувача, ID кур’єра;

– сутність Тара_для_води_замовлення з атрибутами: ID Тара_для_води_замовлення, ID тари з водою, ID замовлення;

– сутність Користувач з атрибутами: ID користувача, ім’я, прізвище, телефонний номер;

– сутність Адміністратор з атрибутами: ID адміністратора, ім’я, прізвище, телефонний номер;

– сутність Кур’єр з атрибутами: ID кур’єра, ім’я, прізвище, телефонний номер, ID Адміністратора.

Рисунок 2.2 – Загальна діаграма класів

2.4 Розробка математичної моделі оптимізаційної задачі

Користувач хоче отримати свою воду в повній кількості та вчасно. Тому засіб повинен приїхати до користувача х потрібною кількістю товару та в певний проміжок часу. Кожний користувач має свою адресу, тому можливо визначити відстань та приблизний час поїздки між складом з водою та користувачем, а також між двома користувачами. Кожин засіб має свою вантажопідйомність.

В задачі доставки питної води, ресурсами являються кур'єри з їх транспортами для доставки води, а роботами являються доставки певному користувачеві замовлення. Існують такі обмеження, як максимальний розмір вантажу, який може вмістити транспорт, інтервал часу, коли користувач може прийняти замовлення, також є час за який може транспорт проїхати від одного користувача (або складу) до іншого користувача (або складу).

Для подальшого математичного моделювання введено наступні позначення:

- C_{ij} — вартість поїздки для транспорту з пункту i до пункту j;
- t_{ij} — час необхідний для обслуговування клієнта i та час поїздки від пункту i до пункту j;
- a_i — час, починаючи з якого можливо почати обслуговувати користувача i;
- b_i — час, до якого треба почати обслуговувати користувача i;
- d_i — кількість води, яку потрібно доставити користувачеві i;
- X_{ijk} — ознака того, що засіб k доставив користувачеві j доставку, при цьому попередній користувач (або складом), який обслуговувався даним транспортом, був користувач i;
- q_k — вантажопідйомність транспорту k;
- S_{ik}— час початку обслуговування транспорту k користувача i.

Дана інформація приведена у таблицях 2.2 та 2.3.

Таблиця 2.2 – Табличне надання оптимізаційної транспортної задачі

	Роботи, які необхідно виконати						
Ресурси	Клієнт 1			…	Клієнт n		
Транспорт 1	C_{01}	…	C_{n1}	…	C_{0n}	…	C_{n-1n}
…	…	…	…	…	…	…	…
Транспорт m	C_{01}	…	C_{n1}	…	C_{0n}	…	C_{n-1n}
Кількість транспортів для доставки користувачеві	1			…	1		
Вага заказу	d_1			…	d_n		
Час, починаючи з якого можливо почати обслуговувати користувача	a_1			…	a_n		
Час, до якого треба почати обслуговувати користувача	b_1			…	b_n		
Час обслуговування користувача	ts_1			…	ts_n		
Час поїздки від одного користувача до другого	tv_{21}	…	tv_{n1}	…	tv_{1n}	…	tv_{n-1n}

Таблиця 2.3 – Продовження табличного надання задачі

	Роботи, які необхідно виконати			
Ресурси	Депо (Склад)			Максимальна вага вантажу
Транспорт 1	C_{1n+1}	…	C_{nn+1}	q_1
…	…			…
Транспорт m	C_{1n+1}	…	C_{nn+1}	q_m
Кількість транспортів для доставки користувачеві	m			
Вага заказу	0			

Алгоритм маршрутизації повинен найти маршрути, які будуть займати менше всього часу на доставку та задовольняти певним обмеженням.

Задача лінійного програмування може бути описана наступною математичною моделлю:

$$\sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=1}^{n+1} c_{ij}x_{ijk} \to min, i \neq j$$

Система обмежень даної функції представлена нерівностями:

$$\sum_{k=1}^{m}\sum_{j=1}^{n} x_{ijk} = 1, \forall\, i = \overline{0,n} \tag{2.5}$$

$$\sum_{i=1}^{n} d_i \sum_{j=1}^{n+1} x_{ijk} \leq q_k, \forall\, k = \overline{1,m} \tag{2.6}$$

$$\sum_{j=1}^{n} x_{0jk} = 1, \forall\, k = \overline{1,m} \tag{2.7}$$

$$\sum_{i=1}^{n} x_{in+1k} = 1, \forall\, k = \overline{1,m} \tag{2.8}$$

$$\sum_{i=0}^{n} x_{ihk} - \sum_{j=1}^{n+1} x_{hjk} = 0, \forall\, h = \overline{1,n}, \forall\, k = \overline{1,m} \tag{2.9}$$

$$s_i + \left(BigValue + ts_i + tv_{ij}\right) * \sum_{k=1}^{m} x_{ijk} \leq s_j + BigValue, \forall\, i = \overline{1,n}, j = \overline{1,n} \tag{2.10}$$

$$a_i \leq s_i \leq b_i, \forall\, i = \overline{1,n} \tag{2.11}$$

$$x_{ijk} \in \{0,1\}, x_{0n+1k} = 0, \forall\, i = \overline{0,n}, j = \overline{1,n+1}, \forall k = \overline{1,m} \tag{2.12}$$

Умова (2.5) гарантує, що кожен клієнт має бути відвіданий один раз, умова (2.6) означає, що не буде перевищено вантажність жодного транспортного засобу. Наступні три рівності (2.7), (2.8) і (2.9) гарантують, що кожен

транспортний засіб починає маршрут в депо 0, обслуговує клієнтів і закінчує маршрут у депо $n + 1$(або 0). Нерівність (2.10) стверджує, що транспортний засіб k не може прийти до j раніше ніж $s_{ik} + t_{ij}$, якщо він їде з вузла i до вузла j. Нарешті обмеження (2.11) гарантує, що часові вікна враховані, а (2.12) вказує можливі значення змінних, які використовуються.

Таким чином була побудована математична модель оптимізаційної задачі з розподілу замовлень між кур'єрами. Дана модель буде використовуватися в методах лінійного програмування для вирішення задачі розподілу замовлень між кур'єрами.

2.5 Проектування бази даних

Так як можливо замовити різні ємкості з водою, а ємкості з водою можуть брати участь у багатьох замовленнях, то створимо окрему сутність частина замовлення.

Оскільки користувач системи з роллю користувач може зробити безліч замовлень та користувач системи з роллю кур'єр може доставити безліч замовлень, необхідно створити сутність інформація замовлення певного користувача системи.

В поїздці між двома користувачами системи з роллю користувач або адміністратор (склад), беруть учать адреси двох користувачів системи, також користувач системи з роллю кур'єр повинен проїхати від першого користувача системи до другого користувача системи. Також адреса одного користувача системи з роллю користувач або адміністратор (склад) та користувача системи з роллю кур'єр можуть брати участь у безліч поїздках. Із-за цього потрібно створити окрему сутність інформація поїздки певного користувача системи.

Завдяки сутностям, які убрали залежності багато до багатьох , дана схема база даних була приведена до третьої нормальної форми.

Схема бази даних зображена на рисунку 2.3.

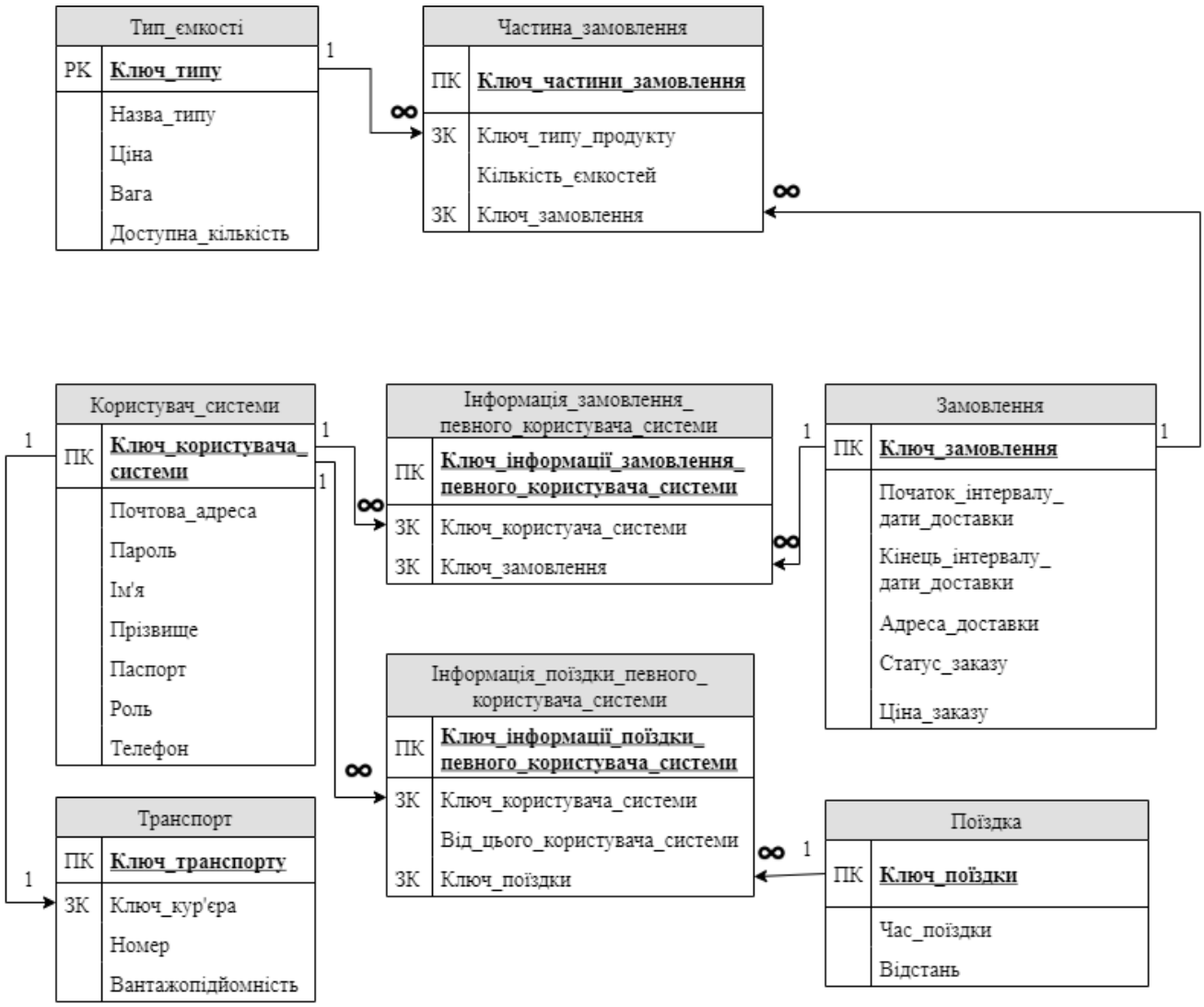

Рисунок 2.3 – Схема бази даних

2.6 Планування експериментального дослідження

Для порівняння вибраних раніше методів рішення лінійних завдань потрібно провести тести алгоритмів на різних конфігураціях обладнання, на різних розмірах тестових даних та на різних обмеженнях лінійного рівняння. Для

кожної конфігурації буде вимірюватися швидкість роботи алгоритму, навантаження на систему, точність алгоритму.

Кожний алгоритм буде тестуватися на однакових вхідних даних та конфігураціях обладнання. Кожний набір даних буде протестований на всіх наборах конфігураціях обладнання. Опис тестових наборів даних наведений нижче (див. табл. 2.4 – 2.7).

Таблиця 2.4 — Опис набору тестових даних для експерименту № 1

Кількість адрес доставок	24
Кількість кур’єрів	2
Середній час поїздки від точки доставки до найближчої точки доставки	10 хвилин
Найбільший час поїздки від точки доставки до найближчої точки доставки	15 хвилин
Середня вага вантажа замовлення	30 кг.
Середній час прийняття замовлення користувачем	15 хвилин
Мінімальний час прийняття замовлення користувачем	10 хвилин

Таблиця 2.5 — Опис набору тестових даних для експерименту № 2

Кількість адрес доставок	40
Кількість кур’єрів	4
Середній час поїздки від точки доставки до найближчої точки доставки	10 хвилин
Найбільший час поїздки від точки доставки до найближчої точки доставки	15 хвилин
Середня вага вантажа замовлення	30 кг.
Середній час прийняття замовлення користувачем	15 хвилин
Мінімальний час прийняття замовлення користувачем	10 хвилин

Таблиця 2.6 — Опис набору тестових даних для експерименту № 3

Кількість адрес доставок	40
Кількість кур’єрів	4
Середній час поїздки від точки доставки до найближчої точки доставки	30 хвилин
Найбільший час поїздки від точки доставки до найближчої точки доставки	40 хвилин
Середня вага вантажа замовлення	40 кг.
Середній час прийняття замовлення користувачем	20 хвилин
Мінімальний час прийняття замовлення користувачем	10 хвилин

Таблиця 2.7 — Опис набору тестових даних для експерименту № 4

Кількість адрес доставок	72
Кількість кур’єрів	6
Середній час поїздки від точки доставки до найближчої точки доставки	30 хвилин
Найбільший час поїздки від точки доставки до найближчої точки доставки	40 хвилин
Середня вага вантажа замовлення	40 кг.
Середній час прийняття замовлення користувачем	20 хвилин
Мінімальний час прийняття замовлення користувачем	10 хвилин

Опис конфігурацій обладнання наведений нижче (див. табл. 2.8 – 2.9).

Таблиця 2.8 — Опис конфігурації обладнання

RAM	8GB
Кількість потоків	1
Частота процесора	2.6 ГГц

Таблиця 2.9 — Опис конфігурації обладнання

RAM	8GB
Кількість потоків	6
Частота процесора	2.6 ГГц

Результати виконання вибраних алгоритмів на тестових даних будуть порівнюватися з правильним рішенням для цих же тестових даних. Буде вимірюватися завантаженість RAM, потоків процесору. Також буде вимірюватись час виконання алгоритмів на тестах.

2.7 Розробка архітектури системи

Дана програмна система буде складатися з таких компонентів: сервер бази даних, який буде зберігати дані користувачів системи; веб-сервер, який буде обробляти дані, проводи логічні операції, та спілкуватися з іншими компонентами; користувач браузера, який буде взаємодіяти з веб-сервером.

В даній програмній системі буде використовуватися багаторівнева архітектура для веб-сервера, яка буде складатися з наступних рівнів:

Рівень доступу до даних, в якому буде реалізовані методи та моделі для зв’язку з джерелом даних та відповідати на запит рівня бізнес-логіки.

Рівень бізнес-логіки, в якому буде реалізована бізнес логіка, реалізовані алгоритми для автоматизації. Також буде спілкуватися за допомогою моделей та методів з рівнем доступу до даних та рівнем уявлень.

Рівень уявлень, в якому будуть формуватися веб-сторінки для користувачів.

Для відображення загальної структури системи побудуємо діаграму розгортання — діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Наприклад, щоб описати веб-сайт діаграма розгортання повинна показувати, які

апаратні компоненти («вузли») існують (наприклад, веб-сервер, сервер бази даних, сервер додатки), які програмні компоненти («артефакти») працюють на кожному вузлі (наприклад , веб-додаток, база даних), і як різні частини цього комплексу з'єднуються один з одним (наприклад, JDBC, REST, RMI). [20] Діаграма розгортання представлена на рисунку 2.4.

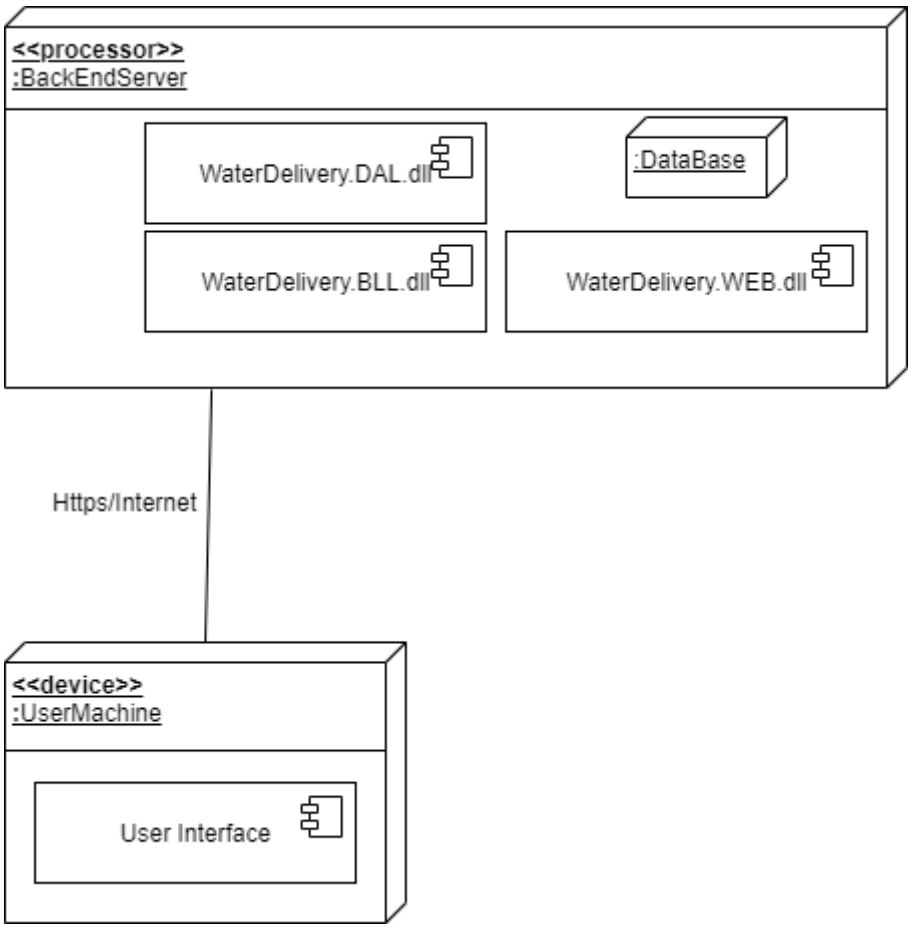

Рисунок 2.4 – Діаграма розгортання

Таким чином, на базі розроблених схеми БД, архітектури програмної системи, UseCase-діаграми, концептуальної моделі, математичної моделі можна перейти до програмної реалізації програмної системи доставки питної води з функцією автоматичного розподілу кур’єрів між замовленнями на базі вибраних методів рішення задач лінійного програмування.

3 ОПИС ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ

3.1 Підготовка вхідних даних для дослідження

Для порівняння обраних алгоритмів, потрібно сформувати данні для математичної моделі, тобто нам потрібні наступні дані:

- кількість заказів;
- кількість кур’єрів (транспортних засобів);
- об’єм кожного замовлення;
- відстані між кожними точками замовлень;
- відстань від складу до кожної точки замовлення;
- час поїздки між кожними точками замовлень;
- час поїздки від складу до точок замовлень;
- вантажопідйомність кожного транспортного засобу;
- часові рамки кожного замовлення.

Для формування даних, була побудована програма. Так як вище було описані обмеження по даним моделі для кожного експерименту, програма приймає наступні вхідні дані:

- кількість замовлень;
- кількість транспортних засобів;
- середній час поїздки від точки доставки (або складу) до найближчої точки доставки (або складу);
- найбільший час поїздки від точки доставки (або складу) до найближчої точки доставки (або складу);
- середня вага вантажа замовлення;
- середній час прийняття замовлення користувачем;
- мінімальний час прийняття замовлення користувачем.

В тестовій програмі транспортний засіб проїжджає 100 метрів за 1 хвилину транспортний засіб.

Алгоритм генерації відстань між користувачами наступний:

– береться квадратна матриця, в комірках будуть розташовані замовлення та склад;

– відстань (в метрах) між комірками це сума модуля різниці між індексами стовпців та модуля різниці між індексами рядків помножена на 2800;

– склад розташовується в центр матриці;

– матриця розбивається на 4 рівні частини;

– замовлення розбиваються на 4 рівні групи;

– кожна з груп розташовується в одній із частин;

– замовлення по черзі починаються розташуватися рядом зі складом і далі наближаються до границь матриці, при чому відстань між двома сусідніми в черзі замовленнями дорівнює цілому випадковому значенню яке більше або дорівнює найбільший час поїздки мінус середній час поїдки від точки до точки поділити на 5 та менше або дорівнює найбільшому часу поїздки поділити на 5;

Код алгоритму представлений на рисунку 3.1.

Інші зміні модель формуються наступним чином:

– вага кожного замовлення дорівнює випадковому цілому числу, яке більше або дорівнює 5 та менше або дорівнює середня вага вантажа замовлення помножена на 2 мінус 5;

– прийняття замовлення користувачем дорівнює випадковому цілому числу, яке більше або дорівнює мінімальному часу прийняття замовлення користувачем та менше або дорівнює середній час прийняття замовлення користувачем помножений на 2 мінус мінімальний час прийняття замовлення користувачем;

– вантажопідйомність кожного транспортного засобу дорівнює 920 кг.

Всі данні записуються у текстовий файл у форматі ключ значення. NoSql бази дані не використовуються для цього, тому що виконується завжди один тип запису і зчитування [21].

```
var rand = new Random();
for (var i = 0; i < userCount / 4; ++i)
{
    var distance = rand.Next((maxTravelTime - avTravelTime) / 5, maxTravelTime / 5 + 1);
    var x = rand.Next(distance + 1);
    if (i == 0)
    {
        if (x == 0)
        {
            x = 1;
        }
        if (x == distance)
        {
            --x;
        }
    }
    var y = distance - x;
    curX += xMultiplier * x;
    curY += yMultiplier * y;
    usersXY.Add(curX);
    usersXY.Add(curY);
}
```

Рисунок 3.1– Код розміщення замовлення в матриці

3.2 Розробка жадібного алгоритму

Розглянемо жадібний алгоритм, створений для оптимізаційної задачі відповідно до математичної моделі (див. рис. 3.2).

Одна з умов математичної моделі – вдовольнити всіх клієнтів, отже виконуємо алгоритм доки не відвідуємо всіх користувачів. Знаходимо для кожного не відвідуваного користувача найближчий транспорт, який може приїхати від останнього пункту призначення вчасно та додати до поточного багажу ємкості з водою для цього користувача.

Після цього знаходимо користувача у якого найменше значення різниці між лівою границею та часом приїзду обраного транспорту до цього користувача та записуємо, що наступним транспорт буде обслуговувати обраного користувача.

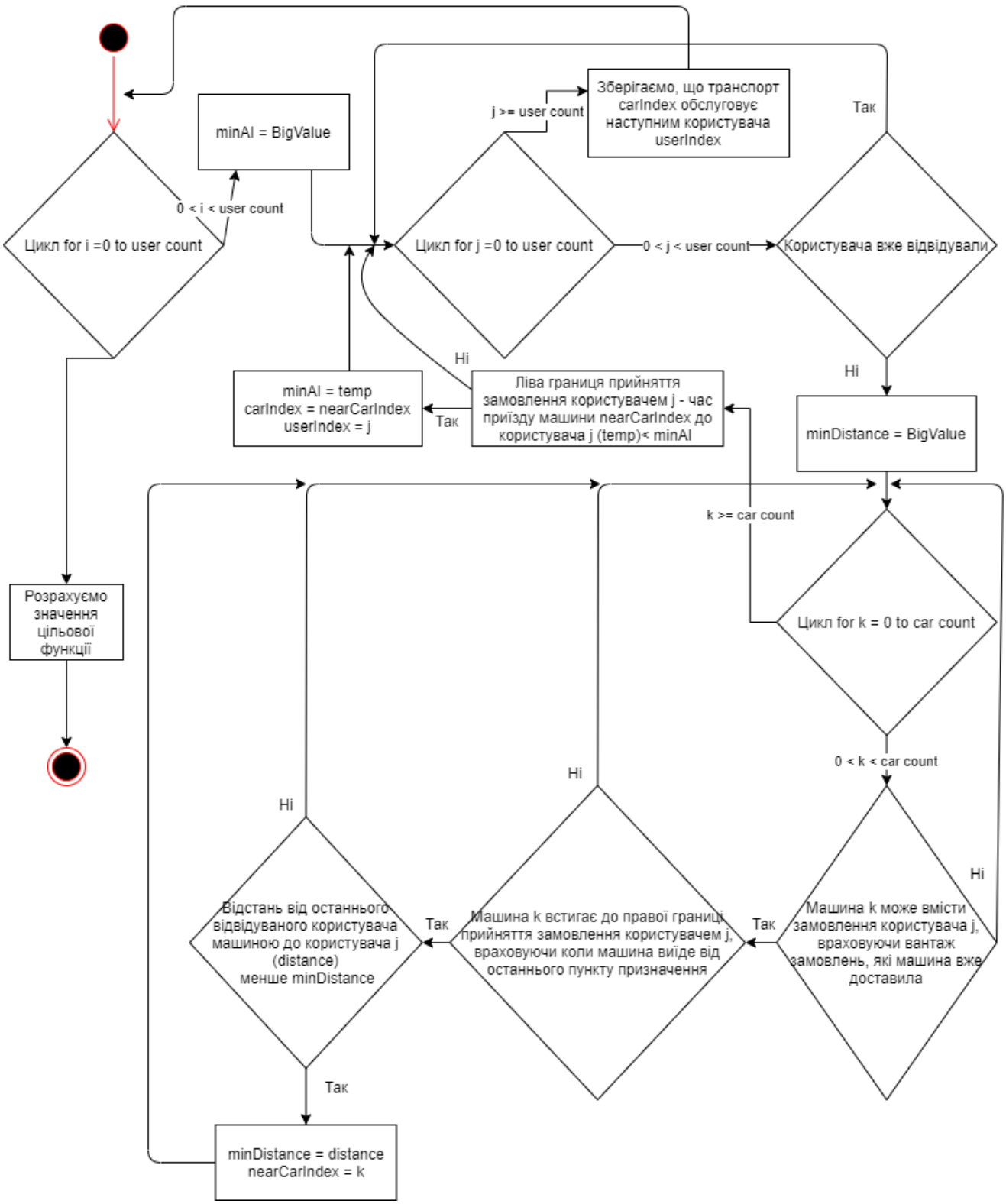

Рисунок 3.2 – Блок-схема жадібного алгоритму

Код жадібного алгоритму представлений на рисунку 3.3.

```
var minAI = double.MaxValue;
var carIndex = -1;
var userIndex = -1;
for (var i = 0; i < userCount; ++i)
{
    if (!visitedUsers[i])
    {
        var nearCar = -1;
        var minV = double.MaxValue;
        for (var j = 0; j < carCount; ++j)
        {
            if (currentCarUser[j] == -1)
            {
                if (minV > funcVars[i])
                {
                    nearCar = j;
                    minV = funcVars[i];
                }
            }
            else
            {
                t1 = 0;
                if (i > currentCarUser[j])
                {
                    t1 = 1;
                }
                fIndex = userCount + currentCarUser[j] * (userCount - 1) + i - t1;
                if (minV > funcVars[fIndex] && qk[j] - di[i] >= 0 && curCarTime[j] + tv[fIndex - userCount] <= bi[i])
                {
                    minV = funcVars[fIndex];
                    nearCar = j;
                }
            }
        }
```

Рисунок 3.3 – Код жадібного алгоритму

3.3 Проведення експериментального дослідження

Для порівняння були взяті наступні методи:

– жадібний алгоритм (власна реалізація);

– симплекс-метод (реалізація взята з ресурсу https://github.com/tymofiidolenko/simplex та доповнена власною реалізацією).

Була розроблена програма для виконання алгоритмів. В програмі спочатку зчитуються підготовленні вхідні дані з текстового файлу. Потім формуються обмеження у вигляді двомірного масиву, коефіцієнти цільової функції у вигляді одномірного масиву. Потім запускається таймер в програмі та обраний алгоритм. Результатом алгоритму являється значення змінних цільової функції (порядок

поїздки кур’єрів між замовленнями) та значення цільової функції. Ці результати та час виконання алгоритму записуються у текстовий файл.

Завдяки Windows інструменту (Task Manager), було замірено навантаження на центральний процесор та використання RAM при виконанні програми (алгоритмів).

Результати експериментів наведено в таблиці 3.1.

У першому експерименті симплекс метод використовував не більше 0.5GB RAM. У другому та третьому експерименті симплекс метод використовував не більш 2GB RAM. Симплекс методу у четвертому експерименті не хватило 8GB RAM. У першому, другому та третьому експерименті симплекс метод використовував ЦП на 100%.

Таблиця 3.1 — Результат експериментів

№ експери-менту	№ конфігурації	Симплекс-метод		Жадібний алгоритм	
		Результат цільової функції	Час обчислення (мс)	Результат цільової функції	Час обчислення(мс)
1	1	786800	6886	936500	31
2	1	1814400	930879	2867200	396
3	1	5182800	2136556	7851300	405
4	1	-	-	16163200	8970
1	2	786800	1529	936500	31
2	2	1814400	199915	2867200	396
3	2	51828006	456869	7851300	405
4	2	-	-	16163200	8970

У першому експерименті жадібний алгоритм використовував не більше 100MB RAM. У другому та третьому експерименті жадібний алгоритм використовував не більш 0.5GB RAM. У четвертому експерименті жадібний алгоритм використовував не більш ніж 3.5GB RAM. У першому експерименті

жадібний алгоритм використовував ЦП на 10%. У другому, третьому та четвертому експерименті жадібний алгоритм використовував ЦП на 100%.

Як бачимо симплекс-метод дає значний виграш в оптимізації, але витрачає значний, але цілком припустимий в предметній області, час на розрахунки при великій розмірності задачі. А при максимальному, але цілком реальному, навантаженні (72 клієнти) рішення взагалі не було отримано. Використання симплекс-методу є більш доцільним для розрахунків бізнес-задач, але для великої розмірності рекомендується розбивати задачу на декілька, проводячи кластеризацію клієнтів за територіальним принципом.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Реалізований симплекс метод для розподілу кур’єрів між заказами був впроваджений в програмну систему доставки питної води, розроблену в результаті бакалаврської роботи. Була змінені та доповнені все шари програмної системи.

4.1 Опис впровадження алгоритму розподілення кур’єрів між замовленнями

Для підготовки даних алгоритму, був розроблений метод для отримання всій потрібної інформації метод, а саме інформація заказів які потрібно доставити в день, для якого розраховується розподіл кур’єрів, відстані та час поїздки між адресами та складом (див. рис. 4.1).

```
private PredicateModel GetDataForPredicate(DateTime date)
{
    var model = new PredicateModel();
    var allOrdersPerDay = Database.Orders.Find(i => i.StartInterval.
    foreach(var order in allOrdersPerDay)
    {
        var preOrder = new PreOrder();
        preOrder.Order = order;
        var weight = 0;
        foreach(var orderPart in order.OrderParts)
        {
            weight += orderPart.Count * orderPart.BottleType.Weight;
        }
        preOrder.Weight = weight;
        foreach(var o in model.PreOrders)
        {
            var t = getTimeAndDistance(order, o.Order);
            preOrder.TimeToUsers[o.Order.Id] = t.Time;
            preOrder.DistanceToUsers[o.Order.Id] = t.Distance;
            o.TimeToUsers[order.Id] = t.Time;
            o.DistanceToUsers[order.Id] = t.Distance;
        }
        model.PreOrders.Add(preOrder);
    }
    model.Cars = Database.Cars.GetAll().ToList();
    return model;
}
```

Рисунок 4.1 – Метод отримання даних для моделі алгоритму

Далі був написаний метод, який перетворює дані в коефіцієнти цільової функції та обмеження (див. рис. 4.2).

```
for (var i = 0; i < userCount; ++i)
{
    var t = new double[rowLength];
    var tt = new double[rowLength];

    t[i + oneUserlength * carCount] = 1;
    tt[i + oneUserlength * carCount] = 1;
    consMatrx.Add(t);
    freeVars.Add(ai[i]);
    signs.Add(">=");
    consMatrx.Add(tt);
    freeVars.Add(bi[i]);
    signs.Add("<=");

}

problems.Add(new Problem(consMatrx.ToArray(), signs.ToArray(), freeVars.ToArray(), funcVars.ToArray(), 0, false));
```

Рисунок 4.2 – Метод отримання моделі оптимізаційної задачі

Також був добавлений симплекс метод, який розподіляє кур'єрів між замовленнями (див. рис. 4.3).

```
public Tuple<List<SimplexSnap>, SimplexResult> GetResult()
{
    List<SimplexSnap> snaps = new List<SimplexSnap>();
    snaps.Add(new SimplexSnap(b, matrix, M, F, C, functionVariables, isMDone, m));
    SimplexIndexResult result = nextStep();
    int i = 0;
    while (result.result == SimplexResult.NotYetFound || i == 400)
    {
        calculateSimplexTableau(result.index);

        result = nextStep();

        if (result.result != SimplexResult.NotYetFound || CheckIfDone() || i == 400)
        {
            if (result.result == SimplexResult.NotYetFound && i != 400)
            {
                result.result = SimplexResult.Found;
            }
            snaps[0] = new SimplexSnap(b, matrix, M, F, C, functionVariables, isMDone, m);
        }
        i++;
    }

    return new Tuple<List<SimplexSnap>, SimplexResult>(snaps, result.result);
}
```

Рисунок 4.3 –Виклик симплекс методу

Також був написаний метод для збереження результатів алгоритму в базу даних (див. рис. 4.4).

```
public void SavePredicateToDB(SimplexSnap snap, PredicateModel model)
{
    var userCount = model.PreOrders.Count;
    var carCount = model.Cars.Count;
    var oneUserlength = userCount + (userCount - 1) * userCount + userCount;
    for (var i = 0; i < userCount; ++i)
    {
        for (var k = 0; k < carCount; ++k)
        {
            if (snap.funcVars[i + oneUserlength * k] == 1)
            {
                CreateOrderInfo(model.Cars[k].Id, -1, model.PreOrders[i].Order.Id);
            }
            if (snap.funcVars[i + oneUserlength * (k + 1) - userCount] == 1)
            {
                CreateOrderInfo(model.Cars[k].Id, -1, model.PreOrders[i].Order.Id);
            }
        }
        for (var j = 0; j < userCount; ++j)
        {
            if (i != j)
            {
                var t1 = 0;
                if (i > j)
                {
                    t1 = 1;
                }
                for (var k = 0; k < carCount; ++k)
                {
                    if (snap.funcVars[userCount + j * (userCount - 1) + i - t1 + oneUserlength * k] == 1)
                    {
                        CreateOrderInfo(model.Cars[k].Id, model.PreOrders[j].Order.Id, model.PreOrders[i].Order.Id);
                    }
                }
            }
        }
    }
}
```

Рисунок 4.4 – Збереження результату симплекс метода в базу даних

4.2 Опис додаткової логіки додатку

Для коректної роботи нового функціоналу була створена додаткова логіка в усіх основних шарах програмної системи.

Для роботи з новою схемою бази даних був змінений контекст бази даних (див. рис. 4.5), були додані нові репозиторії (див. рис. 4.6).

```
public DbSet<User> Users { get; set; }
public DbSet<Travel> Travels { get; set; }
public DbSet<UserTravelInfo> UserTravelInfos { get; set; }
public DbSet<Car> Cars { get; set; }
public DbSet<Order> Orders { get; set; }
public DbSet<UserOrderInfo> UserOrderInfos { get; set; }
public DbSet<OrderPart> OrderParts { get; set; }
public DbSet<BottleType> BottleTypes { get; set; }
```

Рисунок 4.5 – Контекст бази даних

```
public class TravelRepository : IRepository<Travel>
{
    private WaterControlContext db;

    public TravelRepository(WaterControlContext context)
    {
        this.db = context;
    }

    public IEnumerable<Travel> GetAll()
    {
        return db.Travels;
    }

    public Travel Get(int id)
    {
        return db.Travels.Find(id);
    }

    public int Create(Travel order)
    {
       return db.Travels.Add(order).Id;
    }

    public void Update(Travel order)
    {
        db.Entry(order).State = EntityState.Modified;
    }

    public IEnumerable<Travel> Find(Func<Travel, Boolean> predicate)
    {
        return db.Travels.Where(predicate).ToList();
    }

    public void Delete(int id)
    {
        Travel travel = db.Travels.Find(id);
        if (travel != null)
            db.Travels.Remove(travel);
```

Рисунок 4.6 – Репозиторій поїздки

Для роботи з даними використовуються репозиторії, які виконують додавання, зміну, видалення та пошук даних. Загальний інтерфейс репозиторію

представлений на рисунку 4.7, а також приклад репозиторію замовлення розташований на рисунку 4.8.

```
public interface IRepository<T> where T : class
{
    IEnumerable<T> GetAll();
    T Get(int id);
    IEnumerable<T> Find(Func<T, Boolean> predicate);
    void Create(T item);
    void Update(T item);
    void Delete(int id);
}
```

Рисунок 4.7 — Інтерфейс репозиторію

```
public class OrderRepository : IRepository<Order>
{
    private WaterControlContext db;

    public OrderRepository(WaterControlContext context)
    {
        this.db = context;
    }

    public IEnumerable<Order> GetAll()
    {
        return db.Orders;
    }

    public Order Get(int id)
    {
        return db.Orders.Find(id);
    }

    public int Create(Order order)
    {
       return db.Orders.Add(order).Id;
    }

    public void Update(Order order)
    {
        db.Entry(order).State = EntityState.Modified;
    }

    public IEnumerable<Order> Find(Func<Order, Boolean> predicate)
    {
        return db.Orders.Where(predicate).ToList();
    }
```

Рисунок 4.8 — Репозиторій замовлення

При створенні репозиторію ми зберігаю контекст бази даних, який описаний вище. При виконанні операції, система звертається до класу контексту, який відповідає за таблицю в базі даних, та робить операцію, яка відповідає SQL запиту. Наприклад методу Create відповідає SQL запит Insert. Також після операції зміни бази даних, потрібно визвати метод у контексту, який зберігає зміни в базі даних. Це потрібно для того, щоб при комплексних змінах часто не звертатися до бази даних.

Також були змінені моделі користувача та замовлення (див. рис. 4.9), а також додані нові моделі.

```
public class User
{

    public int Id { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
    public string FirstName { get; set; }
    public string SecondName { get; set; }
    public string Passport { get; set; }
    public string Role { get; set; }
    public string Phone { get; set; }
    public ICollection<UserOrderInfo> UserOrderInfos { get; set; }
    public ICollection<UserTravelInfo> UserTravelInfos { get; set; }
    public ICollection<Car> Cars { get; set; }
}
```

Рисунок 4.9 – Нова модель користувача

В шарі бізнес логіки були обновленні методи для отримання інформації користувача, замовлення та добавленні нові методи для отримання інформації нових моделей системи.

Шар бізнес логіки лежить між шаром уявлень і шаром доступу к базі даним. Він обробляє їх запити та виконує дії. Для зв’язку з іншими шарами використовуються об’єкти передачі даних. Приклад цього об’єкту для передачі даних користувача представлений на рисунку 4.10.

```
public class UserDTO
{
    public int Id { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
    public string FirstName { get; set; }
    public string SecondName { get; set; }
    public string Passport { get; set; }
    public string Role { get; set; }
    public string Phone { get; set; }
}
```

Рисунок 4.10 — Об’єкт передачі даних користувача

Дані об’єкти мають поля з тими же типами та назвами, що і моделі в шарі бази даних та в шарі уявлень. Це зроблено для того, щоб використати бібліотеку, яка копіює дані з одного об’єкта в інший, типи яких відрізняються.

Для опису дій, які повинні виконуватися, використовуються інтерфейси сервісів, а також для реалізації цих дій використовуються самі сервіси. Приклад інтерфейсу сервісу користувача на рисунку 4.11, а приклад сервісу датчика на рисунку 4.12.

```
public interface IUserService
{
    void CreateUser(UserDTO userDTO);
    UserDTO GetUser(string email);
    IEnumerable<UserDTO> GetCouriers();
    string GetRole(string email, string password);
    void Dispose();
}
```

Рисунок 4.11 — Інтерфейс сервісу користувача

```
public class UserService : IUserService
{
    IUnitOfWork Database { get; set; }

    public UserService(IUnitOfWork uow)
    {
        Database = uow;
    }
    public void CreateUser(UserDTO userDTO)
    {
        var emptyUser = Database.Users.Find(i => i.Email == userDTO.Email).FirstOrDefault();
        if (emptyUser != null)
        {
            throw new ValidationException("User is exist with this email", "");
        }
        if (userDTO.Role == null) {
            userDTO.Role = "User";
        }
        var mapper = new MapperConfiguration(cfg => cfg.CreateMap<UserDTO, User>()).CreateMapper();
        User user = mapper.Map<UserDTO, User>(userDTO);
        Database.Users.Create(user);
        Database.Save();
    }

    public UserDTO GetUser(string email)
    {
        var user = Database.Users.Find(u => u.Email == email).FirstOrDefault();
        if (user == null)
            throw new ValidationException("User isn`t exist with this email", "");
        var mapper = new MapperConfiguration(cfg => cfg.CreateMap<User, UserDTO>()).CreateMapper();
        return mapper.Map<User, UserDTO>(user);
    }

    public IEnumerable<UserDTO> GetCouriers()
```

Рисунок 4.12 — Сервіс користувача

Для зв’язку з шаром бізнес логіки використовуються сервіси, які були реалізовані в шарі бізнес логіки (див. рис. 4.13).

При створенні контролери, зберігаються потрібні сервіси для роботи контролера.

```
public class HomeController : Controller
{
    IUserService userService;

    public HomeController(IUserService serv)
    {
        userService = serv;
    }
    public ActionResult Index()
    {
```

Рисунок 4.13 — Зв’язок з шаром бізнес логіки

Було обновлено відображення списку замовлень для кур’єрів (див. рис. 4.14).

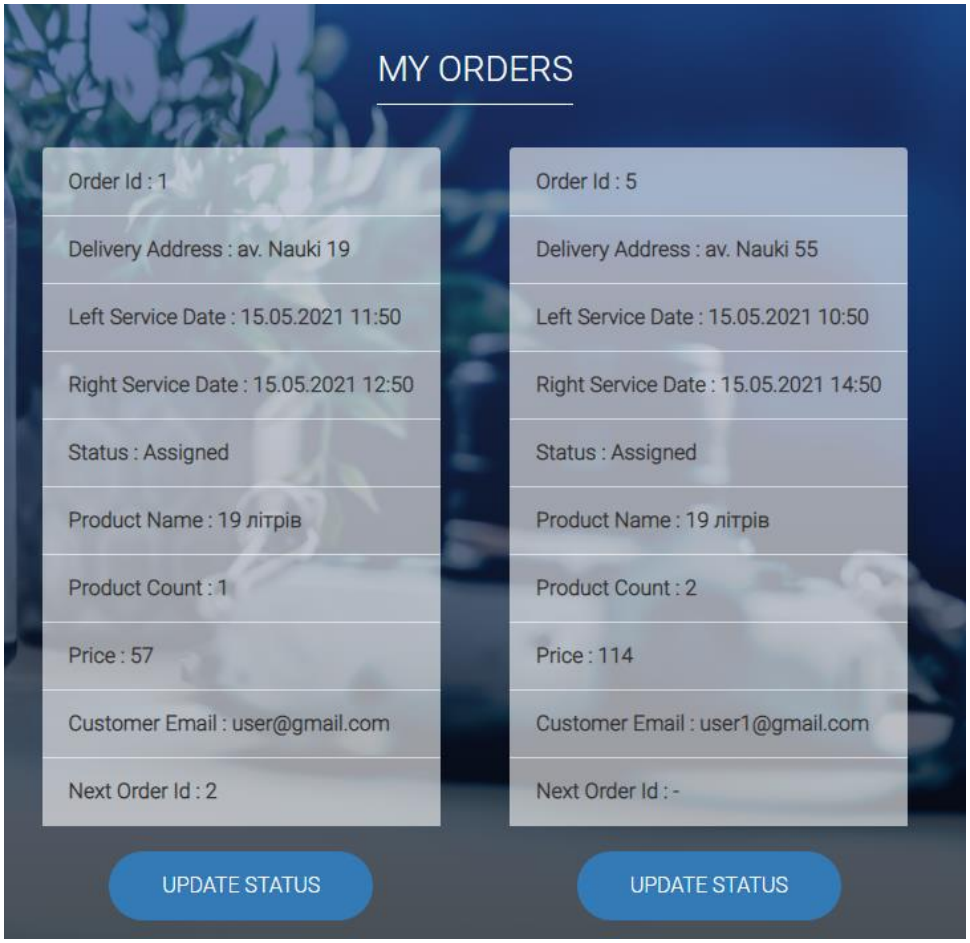

Рисунок 4.14 – Відображення списку замовлень для кур’єра

Так, як було добавлено новий функціонал, треба перевертіти його на працездатність, а також перевірити не зламався чи старий функціонал програмної системи.

5 АНАЛІЗ ДОСЛІДНИЦЬКОЇ ЕКСПЛУАТАЦІЇ ТА МОЖЛИВИХ ЗАСТОСУВАНЬ

5.1 Аналіз можливих застосувань

В ході виконання магістерської кваліфікаційної роботи були порівнянні методи рішення лінійного програмування в рамках оптимізації розподілу кур’єрів між замовленнями в сервісі доставки питної води. Реалізацію симплекс метода було впроваджено в систему доставки води для розподілу кур’єрів між замовленнями. Дану програмну систему можливо використовувати в сервісах доставки води для зменшення довжини поїздки машинами, а також для дотримання часових рамок доставки води користувачеві.

У подальшому можливо порівняти інші методи лінійного програмування та інші підходи для вирішення розподілу кур’єрів між замовленнями. Також можна переписати програму з реалізації під віртуальну машину на контейнер, так як це дасть виграш у продуктивності програмного забезпечення [22].

5.2 Опис тестування системи

Щоб провірити якість продукту та отримати інформацію про якість в межах контексту, в якому продукт має використовуватись, використовують тестування програмного забезпечення. Тестування також включає в себе пошук помилок, дефектів, випробування програмних складових з метою оцінки якості. Тестування проводиться в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином, для перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності.

Для тестування розробленої системи було використано модульне і функціональне тестування.

Модульне тестування – це тип тестування програмного забезпечення, де тестуються окремі блоки або компоненти програми. Метою є перевірка того факту, що кожна одиниця програмного коду працює належним чином. Модульне тестування проводиться під час розробки (фази кодування) програми розробниками. Модульні тести ізолюють розділ коду та перевіряють його коректність. Одиницею може бути окрема функція, метод, процедура, модуль або об’єкт.

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат програмне забезпечення тестується стандартним виконанням програми з метою виявлення баґів (помилок або інших дефектів).

Процес розробки зазвичай передбачає, коли та як буде відбуватися тестування. Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно [23].

Для тестування програмної системи, а точніше back-end частини, використовувалося модульне тестування. Модульне тестування — це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. У процедурному програмуванні модулем вважають окрему функцію або процедуру. В об'єктно-орієнтованому програмуванні — інтерфейс, клас. Метою модульного тестування є ізоляція кожної частини програми та впевненість у тому, що кожна окрема частина є коректною. Модульний тест забезпечує жорсткий «контракт», за яким має працювати тестований код. Як результат, це надає деякі переваги. Модульне тестування допомагає знайти помилки раніше в циклі розробки ПЗ, що робить розробку дешевшою та швидшою [24]. Для модульних тестувань використовувався MSTest, тому що він гарно працює з .NET Framework.

Також для модульних тестувань використовувались Moq для імітування функцій та створення Mock-об'єктів.

Для тестування був створений проект, в якому є посилання на другі проекти. Для шару бізнес логіки були протестовані сервіси. Приклад для тестування змінення статусу заказу, зі гарними даними, представлений на рисунку 5.1. В даному методі передаються всі можливі пари, що було, що стало. І перевіряється, чи правильно змінився статус.

```
[TestMethod]
[DataRow(StatusesEnum.Open, StatusesEnum.Assigne)]
[DataRow(StatusesEnum.Assigne, StatusesEnum.InProgress)]
[DataRow(StatusesEnum.InProgress, StatusesEnum.Close)]
public void UpdateOrderStatusWithoutError(string oldStatus, string expectedNewStatus)
{
    var moqDb = new Mock<IUnitOfWork>();
    moqDb.Setup(i => i.Orders.Get(It.IsAny<int>())).Returns(new Order() { Status = oldStatus });
    var orderService = new OrderService(moqDb.Object);
    Assert.AreEqual(expectedNewStatus, orderService.UpdateOrderStatus(1));
}
```

Рисунок 5.1 — Метод тестування для зміни статусу замовлення

Критичні моменти програмного засобу були протестовані завдяки модульному тестуванню та мануальному тестуванню. Тести показали, що програмна система працює без критичних помилок, тобто може використовуватись справжніми користувачами.

ВИСНОВКИ

В ході даної роботи було проведено дослідження методів рішення транспортних задач для підвищення ефективності доставки питної води.

Був проведений аналіз проблемної області створення систем маршрутизації транспортних засобів доставки питної води. Виявлено, що ринок доставки води не зникне і там можливо автоматизувати розподіл замовлень на доставку води між кур’єрами.

Були визначені вимоги до програмного продукту.

Був проведений аналіз існуючих оптимізаційних моделей та методів. Після цього були вибрані методи для рішення оптимізаційної задачі. А саме: жадібний алгоритм, симплекс метод, алгоритм Кармаркара.

Був проведений аналіз та UML-моделювання предметної області. Де були виявлені основні сутності системи, описали їх та зв’язки між сутностями.

На основі описаних функцій системи доставки води з автоматичним розподілом кур’єрів між замовленнями та обмеженнях системи, була побудована математична модель оптимізаційної задачі.

На основі даних системи, було описані тестові данні та конфігурація обладнання на якому будуть запускатися вибрані алгоритми.

Були виявлені інформаційні потреби. На основі котрих була розроблена схема бази даних, були визначені сутності, їх атрибути та зв’язки між сутностями, та архітектура веб-сервера та суспільна схеми системи. Для архітектури веб-сервера було визначено, що буде використовуватися трьох-рівнева архітектура (доступ до даних, бізнес-логіка, уявлення).

Був проведений експеримент, в якому порівнювався симплекс метод з жадібним алгоритмом для вирішення оптимізаційної задачі розподілу заказів між кур’єрами в системі доставки води. Було виявлено, що жадібний алгоритм не підходить для вирішення задачі, тому що знайденні відповіді набагато відрізняються від оптимального. Також доведено, що використання математичних

моделей для оптимізації маршрутів та призначень на доставку, а також грамотне, з урахуванням розмірів оптимізаційних задач, використання методів рішення таких задач дозволяє підвищити ефективність процесу доставки питної води.

Була доповнена програмна система доставки питної води функціоналом розподілу кур’єрів між замовленнями на основі симплекс методу.

Була протестована програмна система. За результатами дослідження опубліковано матеріали в збірці публікацій наукової конференції (див. додаток Д).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Оптимизация (математика) [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) (дата звернення: 07.03.2021)

2. Ю.И. Бурименко, И.Ю. Лебедева, А.Ю. Щуровска. 2016 Оптимизационные методы и модели с решением задач на компьютере. Одесса, 2016, с.7-10, 32-34, 45 (дата звернення: 07.03.2021)

3. Oleksandr Topchii, Oleksandr Samantsov, Oksana Mazurova, Mariia Shirokopetleva. A Study of Optimization Models for Creation of Artificial Intelligence for The Computer Game in The Tower Defense Genre. // Problem of Infocommunications. Science and Technology (PIC S&T'2020), Kharkiv, Ukraine. 2020. (дата звернення: 07.03.2021)

4. Преимущества использования БД [Електронний ресурс] – Режим доступу: https://lektsii.com/2-109219.html (дата звернення: 07.03.2021)

5. Архитектура программного обеспечения [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F (дата звернення: 07.03.2021)

6. Многоуровневая архитектура [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0 (дата звернення: 07.03.2021)

7. Корпоративный сайт как средство рекламной коммуникации [Електронний ресурс] – Режим доступу: https://www.marketing.spb.ru/lib-comm/internet/corpsite_howto.htm (дата звернення: 07.03.2021)

8. Сайт как инструмент рекламы [Електронний ресурс] – Режим доступу: https://uh.ua/kb/otvety/prodazhi/sajt-kak-instrument-reklamy.html (дата звернення: 07.03.2021)

9. Bottled Water Delivery Service [Електронний ресурс] – Режим доступу: https://www.water.com/bottled-water-delivery-service (дата звернення: 16.03.2021)

10. My Water shop [Електронний ресурс] – Режим доступу: https://mywatershop.ua/about/ (дата звернення: 16.03.2021)

11. Симплекс-метод [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4 (дата звернення: 16.03.2021)

12. Введение в математическое программирование [Електронний ресурс] – Режим доступу: https://intuit.ru/studies/courses/1020/188/lecture/4917 (дата звернення: 16.03.2021)

13. Лінійне програмування [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F (дата звернення: 16.03.2021)

14. Решение транспортной задачи распределительным методом на примерах [Електронний ресурс] – Режим доступу: https://function-x.ru/zadacha_transportnaja_raspredelitelnyi_metod.html (дата звернення: 16.03.2021)

15. Метод эллипсоидов [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%8D%D0%BB%D0%BB%D0%B8%D0%BF%D1%81%D0%BE%D0%B8%D0%B4%D0%BE%D0%B2 (дата звернення: 16.03.2021)

16. Bottled Water Delivery Service [Електронний ресурс] – Режим доступу: https://www.water.com/bottled-water-delivery-service (дата звернення: 22.03.2021)

17. My Water shop [Електронний ресурс] – Режим доступу: https://mywatershop.ua/about/ (дата звернення: 22.03.2021)

18. Алгоритм Кармаркара [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D1%80%D0%BA%D0%B0%D1%80%D0%B0 (дата звернення: 22.03.2021)

19. How to start a water delivery business [Електронний ресурс] – Режим доступу: https://howtostartanllc.com/business-ideas/water-delivery (дата звернення: 23.03.2021)

20. Диаграмма развёртывания [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D1%80%D0%B0%D0%B7%D0%B2%D1%91%D1%80%D1%82%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F (дата звернення: 23.03.2021)

21. Kuzochkina A., Z. Dudar, Shirokopetleva M. Analyzing and Comparison of NoSQL DBMS. // International Scientific and Practical Conference «Problems of Infocommunications. Science and Technology» (PIC S&T-2018). 2018. P. 560.

22. Natalia Kravets, Tseshkovskyi N., Liutova K. S. Containers and virtual machinesin microsoft Azure. // Science, society, education: topical issues and development prospects. Abstracts of the 7th International scientific and practical conference. 2020. P. 278.

23. Тестування програмного забезпечення [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення (дата звернення: 05.05.2019)

24. Модульне тестування [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Модульне_тестування (дата звернення: 06.05.2019)