МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра прикладної математики

Магістерська робота

Розробка веб-сайту для транспортної компанії (реалізація задачі маршрутизації транспортних засобів).

Виконав: студент групи ПМпМ-22с
спеціальності 113 – прикладна математика
(шифр і назва спеціальності)

_____ Мустафаєва Д. А.
(підпис) (прізвище та ініціали)

Керівник _____ Щербатий М. В.
(підпис) (прізвище та ініціали)

Рецензент _____ Вовк В.Д.
(підпис) (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет_____прикладної математики та інформатики_____
Кафедра_____прикладної математики_____
Спеціальність_____113 – прикладна математика_____
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач
кафедри
_____

" ___ " _____ 20 __ року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

_____Мустафаєвій Дінарі Акімівні_____
(прізвище, ім'я, по батькові)

1 .Тема роботи
Розробка веб-сайту для транспортної компанії (реалізація задачі маршрутизації транспортних засобів).
Website development for a transport company (implementation vehicle routing tasks).

керівник роботи _____доцент Щербатий Михайло Васильович_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від "22" вересня 2021 року № 6

2. Строк подання студентом роботи 16 травня 2022 р.

3. Вихідні дані до роботи
1) Clarke, G. and Wright, J., “Scheduling of vehicles from a central depot to a number of delivery points,” Operations Research, vol. 12(4), pp. 5
2) Gillett, B. E. and Miller, L. R., “A heuristic algorithm for the vehicle dispatch problem,” Operations Research, vol. 22, pp. 340–349, 1974.
3) Braysy, O. and Gendreau, M., “Vehicle routing problem wiht time windows, part i: route construction and local search algorithms,” Transportation Science, vol. 39(1), pp. 104–118, 2005.
4) Breedam, A. V., “Improvement heuristics for the vehicle routing problem based on simulated annealing,” European Journal of Operational Research, vol. 86, pp. 480–490, 1995.

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)
1) Огляд літератури. Ознайомлення із поняттями задача маршрутизації транспорту.
2) Програмна реалізація алгоритму задачі маршрутизації транспорту.
3) Програмна реалізація веб-сайту для транспортної компанії.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
1) Графічні зображення схем
2) Ілюстрації вигляду програми, побудованої у системі Visual Studio.
3) Ілюстрації результатів програми – вигляд веб-сайту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 02.09.2021р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1)	Проведення теоретичних досліджень та узгодження алгоритму		
	для реалізації ЗМТ.	24.10.21	Виконано
2)	Написання програми, яка реалізовує алгоритм Кларка та Райта.	15.11.21	Виконано
3)	Створення фронтенд частини сайту.	10.02.22	Виконано
4)	Візуалізація розв'язків алгоритму на мапі.	15.03.22	Виконано
5)	Оформлення роботи.	30.04.22	Виконано

Студент _____ Мустафаєва Д. А.
(підпис) **(прізвище та ініціали)**

Керівник роботи _____ Щербатий М. В.
(підпис) (прізвище та ініціали)

Зміст

Перелік умовних скорочень

1. ЗМТ – задача маршрутизації транспорту.
2. ЗМТЧВ – задача маршрутизації транспорту.
3. Алгоритм КР – Алгоритм Кларка та Райта.
4. EF Core – Entity Framework Core.
5. СКБД – Система керування базами даних.
6. DOM – Document Object Model.

Вступ

Визначення оптимального маршруту для доставки вантажів є одним із найголовніших завдань, що належать до транспортної логістики компанії. В умовах мегаполісу на доставку вантажів впливає безліч різних факторів, таких як розосередженість клієнтів по всьому місту, завантаженість автошляхів, час очікування приймання клієнта, які можуть сильно вплинути на ефективність бізнесу. Для того, щоб врахувати всі чинники, які впливають доставку вантажів, необхідно попередньо здійснювати маршрутизацію перевезень, тобто визначати оптимальні маршрути перевезення вантажів. Так маршрутизація дозволить мінімізувати пройдену відстань автотранспортним засобом, зменшити час проходження всього маршруту, скоротити час знаходження автотранспортного засобу на завантаженні чи розвантаженні. Все це призведе до збільшення продуктивності автотранспортного засобу у рази. Основною вигодою від маршрутизації перевезень є мінімізація транспортних витрат компанії. Транспортна логістика є одним із найбільших центрів витрат, поряд з виробництвом, тому будь-якій компанії просто необхідно контролювати та мінімізувати витрати, що виробляються даними підрозділами.

Тема маршрутизації перевезень є актуальною нині, бо компаніям необхідно у стислий термін формувати оптимальні маршрути для швидкої та якісної доставки вантажів велику кількість клієнтів, розташованих по всьому місту. Велика кількість вітчизняних та зарубіжних наукових праць присвячена розробці різних методик визначення оптимальних маршрутів для доставки вантажів. Багато компаній розробляють різне програмне забезпечення, що дозволяє автоматизувати процес маршрутизації перевезень. Програмне забезпечення також здатне в реальному часі стежити та координувати пересування автотранспортного засобу маршрутом.

У зв'язку з цим метою даної роботи є розробка методики маршрутизації перевезень з метою підвищення ефективності роботи доставок та мінімізації витрат компанії.

Задача маршрутизації транспорту є NP-повною, отже, для її вирішення потрібні оптимізаційні, евристичні чи метаевристичні алгоритми. Новизна роботи полягає у застосуванні алгоритму Кларка та Райта до задачі маршрутизації транспорту.

Завдання випускної кваліфікаційної роботи:

• Визначення поточної ситуації у транспортній логістиці, а саме виявлення методів та процедур, що дозволяють маршрутизувати автоперевезення.

• Пошук різних методів маршрутизації перевезень у науковій літературі.

• Застосування алгоритму Кларка та Райта до задачі маршрутизації транспорту.

• Створення веб-сайту, який надасть користувачеві можливість скласти оптимальний маршрут для доставлення або перевезення товарів.

У даній роботі описаний алгоритм розробки веб-сайту за допомогою бібліотеки React.js та описані підходи взаємодії веб-сайту з додатком Google Maps. Для обрахунку реальних масивів даних у роботі використано карту міста Львова, на якій програма будує маршрути за допомогою додатку Google Maps.

Проект представляє собою веб-сайт деякої транспортної компанії. Він створений з метою спростити роботу адміністратора – програма буде створювати маршрут на кожен день для кожної машини конкретного депо. Задача користувача сайту – прив’язати до вже існуючих замовлень автомобіль, який буде розвозити доставки, та додати їх на карту. Адміністратор зможе одразу поділитися готовою таблицею оптимальних шляхів з водіями.

Розділ 1. Задача маршрутизації транспорту та алгоритми її розв’язання

1.1. Задача маршрутизації транспорту

Задача маршрутизації транспортних засобів (ЗМТ) визначається як задача розробки оптимальної доставки або маршрутів від одного або кількох складів до ряду територіально розкиданих міст або клієнтів, з урахуванням побічних обмежень. Через введення широкого спектру обмежень щодо пропускної здатності, довжини маршруту, часових вікон та відносин пріоритету між клієнтами; досить важко знайти загальноприйняте визначення ЗМТ [2].

ЗМТ можна описати як комбінацію двох добре вивчених задач, а саме проблеми комівояжера і задачі упаковки контейнера, оскільки вона концептуально лежить на перетині двох добре вивчених проблем. Як було досліджено раніше, розв’язки ЗМТ іноді перетворюються в задачу комівояжера шляхом реплікації депо [2].

ЗМТ відноситься до категорії NP-складності. Найскладніші точні алгоритми для ЗМТ здатні вирішити лише екземпляри до 100 клієнтів із різною послідовністю. Здебільшого тому дослідники змушені докладати зусиль до евристичних алгоритмів, крім того, що вони дають гнучкість роботи з різноманітністю варіантів, що виникають на практиці [16].

Задача маршрутизації транспорту вирішує питання складання оптимального плану перевезень вантажу транспортними засобами зі складу (депо) в пункти розподілу (клієнтам). Депо – це деяка особлива вершина, початок і кінець маршрутів всіх транспортних засобів. При цьому мета завдання – мінімізувати загальну вартість маршруту.

У ЗМТ присутнє таке поняття, як сумарна вартість об'їзду послідовності вершин транспортним засобом. Це ключовий мінімізуємий параметр. Насправді він являє собою будь-які витрати, пов'язані з відвідуванням клієнтів. Для вирішення задачі і дослідження ефективності алгоритмів не має значення, що за

ним стоїть насправді. Ми приймаємо його як узагальнення всіх видів витрат на пересування. В роботі для простоти розглянемо цей параметр як відстань між клієнтами. Єдине обмеження пов'язане з тим, що вартість проїзду (відстань) не може бути негативною [13].

1.2. Задача маршрутизації транспорту з часовими вікнами

Проблема маршрутизації транспортних засобів із часовими вікнами (ЗМТЧВ) — це проблема проектування маршрутів з найменшою вартістю від одного депо до набору географічно розкиданих точок таким чином, щоб кожен пункт відвідував лише один транспортний засіб протягом заданого інтервалу часу. Усі маршрути починаються і закінчуються в депо, і загальна потреба всіх пунктів на маршруті не повинна перевищувати місткості транспортного засобу. Іншими словами, потреби кожного клієнта мають бути задоволені [18].

Цей варіант ЗМТ, ЗМТЧВ, також є NP-складним. Савелсберг дослідив, що навіть отримати можливе рішення ЗМТЧВ, коли розмір депо фіксований, є NP-складною проблемою. Тому евристика знову є найбільш можливим підходом до вирішення [18].

Часове вікно клієнта може характеризуватися раннім і пізнім часом, протягом якого обслуговування повинно розпочатися. Концепція часових вікон існує в багатьох реальних життєвих ситуаціях, таких як послуги набору номера, маршрути шкільного автобуса та банківські доставки. Планування має бути розроблене таким чином, щоб врахувати питання наявності персоналу для завантаження транспортних засобів, правил дорожнього руху та умов, а також деяких інших попередньо визначених уподобань клієнтів.

Соломон (1987) вивчав ряд евристик, включаючи «алгоритм економії», «алгоритм найближчого сусіда, орієнтований на час», «алгоритм вставки» та «алгоритм розгортки, орієнтований на час» для ЗМТЧВ, і додатково сконструював набір контрольних задач для легкого порівняння різноманітних

процедур вирішення. Численні дослідники використовували набори тестів Соломона, коли вони перевіряли свої евристики або точні алгоритми [1].

Ван Ландегем (1988) представив свою двокритеріальну евристику на основі евристики економії для ЗМТЧВ і обговорив, що взаємодія між просторовими та часовими проблемами ускладнює розуміння основної динаміки проблеми.

Колен та ін. (1987) представили оптимальний алгоритм, заснований на концепції розгалужень і обмежень для ЗМТЧВ, але не зміг реалізувати його в умовах великих проблем. Алгоритм здатний вирішувати проблеми до чотирьох транспортних засобів, які обслуговують чотирнадцять клієнтів [17].

1.3. Постановка задачі та її математична модель

ЗМТ визначається графом $G = (V, A)$, де V – це депо, яке представлено вузлами 0 і $n + 1$, A – це множини дуг, які представляють зв'язок між клієнтами та депо. Усі можливі маршрути транспортних засобів відповідають шляхам у G, які починаються з вузла 0 і закінчується у вузлі $n + 1$. Часове вікно також пов'язане з вузлами 0 і $n + 1$, тобто $[a_0, b_0] = [a_{n+1}, b_{n+1}] = [E, L]$, де E і L представляють якнайшвидше можливе відправлення з депо і найпізніше можливе прибуття в депо, відповідно. Крім того, для цих двох вузлів визначено нульові вимоги та час обслуговування, тобто $d_0 = d_{n+1} = s_0 = s_{n+1} = 0$. Можливі рішення існують лише за умови $a_0 = E \leq \min_{i \in V \setminus \{0\}} b_i - t_{0i}$ та $b_{n+1} = L \geq \min_{i \in V \setminus \{0\}} a_i + s_i + t_{i0}$. Зауважте також, що дугу $(i, j) \in A$ можна усунути через тимчасові міркування, якщо $a_i + s_i + t_{ij} > b_j$, або обмеження ємності, якщо $d_i + d_j > C$, або через інші фактори. Нарешті, згадаємо, що коли транспортним засобам дозволяється залишатися на депо, особливо у випадку, коли основна мета полягає в мінімізації кількості використовуваних транспортних засобів, дуга $(0, n + 1)$ з $c_{0,n+1} = t_{0,n+1} = 0$ необхідно додати до множини дуг A.

Далі ми представляємо формулювання математичного програмування для ЗМТЧВ, що включає два типи змінних: змінні потоку x_{ijk}, $(i,j) \in A, k \in K$, що дорівнює 1, якщо дуга (i, j) використовується транспортним засобом k, і 0 інакше; і змінні часу w_{ik}, $i \in V, k \in K$, що вказують початок обслуговування у вузлі i при обслуговуванні транспортним засобом k.

Потім ЗМТЧВ можна формально описати як таку модель потоку багатотоварної мережі з часовим вікном і обмеженнями пропускної здатності:

$$min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{1.1}$$

$$\sum_{k \in K} \sum_{j \in \Delta+(i)} x_{ijk} = 1\,, \qquad \forall i \in N \tag{1.2}$$

$$\sum_{j \in \Delta+(0)} x_{0jk} = 1, \qquad \forall k \in K \tag{1.3}$$

$$\sum_{i \in \Delta-(j)} x_{ijk} - \sum_{i \in \Delta+(j)} x_{ijk} = 0, \qquad \forall k \in K, \qquad \forall j \in N \tag{1.4}$$

$$\sum_{i \in \Delta-(n+1)} x_{i,n+1,k} = 1, \qquad \forall k \in K \tag{1.5}$$

$$x_{ijk}\left(w_{ik} + s_i + t_{ij} - w_{jk}\right) \leq 0, \qquad \forall k \in K, \qquad \forall (i,j) \in A \tag{1.6}$$

$$a_i \left(\sum_{j \in \Delta+(i)} x_{ijk} \right) \leq w_{ik} \leq b_i \left(\sum_{j \in \Delta+(i)} x_{ijk} \right), \qquad \forall k \in K, \forall i \in N \tag{1.7}$$

$$E \leq w_{ik} \leq L, \qquad \forall k \in K, \qquad \forall i \in \{0, n+1\} \tag{1.8}$$

$$\sum_{i \in N} d_i \sum_{i \in \Delta+(i)} x_{ijk} \leq C, \qquad \forall k \in K \tag{1.9}$$

$$x_{ijk} \geq 0, \qquad \forall k \in K, \qquad \forall (i,j) \in A$$

$$x_{ijk} \in \{0,1\}, \qquad \forall k \in K, \qquad \forall (i,j) \in A \tag{1.10}$$

Цільова функція (1.1) цього нелінійного формулювання виражає загальну вартість. Враховуючи, що $N = V \setminus \{0, n+1\}$ представляє набір клієнтів, обмеження (1.2) обмежують призначення кожного клієнта точно одним маршрутом транспортного засобу. Далі, обмеження (1.3)–(1.5) характеризують потік на шляху, яким слідуватиме транспортний засіб k. Крім того, обмеження (1.6) – (1.8) та (1.9) гарантують виконання графіка відповідно до міркувань часу та можливостей. Зауважте, що для заданого k обмеження (1.7) змушують $w_{ik} = 0$, коли клієнт і не відвідує транспортний засіб k. Нарешті, (1.10) накладає бінарні умови на змінні потоку.

Бінарні умови (1.10) дозволяють лінеаризувати обмеження (1.6) як:

$$w_{ik} + s_i + t_{ij} - w_{jk} \leq (1 - x_{ijk}) M_{ij}, \qquad \forall k \in K, \qquad \forall (i,j) \in A \tag{1.11}$$

де M_{ij} — великі константи. Крім того, M_{ij} можна замінити на $max\{b_i + s_i + t_{ij} - a_j, 0\}$, $(i,j) \in A$, а обмеження (1.6) або (1.11) потрібно застосовувати лише для дуг $(i,j) \in A$, такий, що $M_{ij} > 0$; інакше, коли $max\{b_i + s_i + t_{ij} - a_j, 0\} = 0$, ці обмеження задовольняються для всіх значень w_{ik}, w_{jk} та x_{ijk}.

1.4. Алгоритми розв'язання та їх класифікація

Існує багато методів вирішення проблем маршрутизації. Деякі з цих методів є точними, вони гарантують знайти оптимальне точне та найкраще рішення за достатнього часу, а інші є методами наближення (евристика та метаевристика), які дають хороші рішення за розумний проміжок часу, хоча немає гарантії досягнення оптимальності [4].

Ці методи можна розділити на три основні типи: точні, на основі евристики та на основі метаевристики.

Точні методи

Як правило, точні методи намагаються отримати точне, а отже, найкраще рішення даної проблеми. Оскільки і задача комівояжера, і ЗМТ є NP-складними проблемами, не існує точного методу, який би добре працював для вирішення кожного з їх екземплярів. Іншими словами, для отримання оптимального рішення їм може знадобитися як завгодно багато часу [15].

Найбільш відомим точним методом є метод Branch and Bound – гілок та меж. По суті, цей метод є варіацією повного перебору з виключенням підмножин допустимих рішень, які свідомо не містять оптимальні рішення. Метод гілок і меж показує хороші результати для задач невеликої розмірності. Оскільки час обчислень зростає надто швидко, даний алгоритм неможливо застосовувати для задач з більш ніж з 25-30 вершинами [1].

Загалом, як точні методи пошуку серед усіх можливих рішень проблеми, вони зазвичай є непосильними для великих задач, що обмежує їх застосовність у порівнянні з евристичними та метаевристичними методами.

Евристичні методи

Евристичний алгоритм – це алгоритм рішення задач, який для більшості випадків дає прийнятне, але не обов'язково «правильне» рішення [1]. Евристичний алгоритм здійснює відносно обмежений пошук в просторі рішень, при цьому дає хороші рішення за задовільний час [3].

Хоча евристичні методи не гарантують отримання найкращого рішення, вони здатні знайти гарне рішення за розумний час. Більшість із тих, що використовуються для вирішення ЗМТ, походять від задачі комівояжера.

Евристичні методи можна класифікувати на три основні категорії, які залежать від їх природи.

1. Алгоритми побудови – це ті, в яких рішення починається з порожнього маршруту, який потім будується шляхом включення клієнтів, зазвичай по одному, до тих пір, поки не буде розроблено повний тур, намагаючись зберегти

загальну вартість якомога нижчою. Найвідомішими алгоритмами цієї категорії є вставка найближчого сусіда і алгоритм заощадження Кларка і Райта.

2. Алгоритми покращення – це ті, які створюють рішення, а потім намагаються його покращити шляхом пошуку більш ефективного. Найвідомішими алгоритмами в цій категорії є локальний пошук r-Opt (2-Opt і 3-Opt) і ланцюжковий Lin-Kernighan (ILK).

3. Двофазова евристика — це ті алгоритми, які працюють, щоб знайти рішення за допомогою двох різних фаз, таких як кластеризація та маршрутизація.

Методи на основі метаевристики

Оскільки класичні евристичні методи, коротко згадані в попередньому підрозділі, виконують обмежений пошук; вони можуть застрягти в місцевих оптимальних умовах. Метаевристика, навпаки, використовується для мінімізації ймовірності цього, і тому намагається досягти глобального оптимального рішення.

Метаевристика — це набір концепцій, за допомогою яких інші евристики можуть знайти вихід із локального оптимального рішення, продовжуючи пошук кращих областей простору рішень і таким чином намагаючись досягти глобального оптимального рішення [5]. У літературі було запропоновано багато критеріїв для класифікації метаевристики, корисним з яких є розгляд кількості рішень, які використовуються одночасно. Таким чином, метаевристики можна класифікувати на «одноточкові» та «на основі населення»; іншими словами, в будь-який конкретний момент часу, незалежно від того, чи працює алгоритм з одним рішенням або сукупністю.

1. Одноточковий пошук

Метаевристичні алгоритми, які працюють над єдиним рішенням і які також називаються методами траєкторії, охоплюють метаевристики на основі локального пошуку, такі як пошук за табу і імітований відпал.

2. Популяційний пошук

Метаевристичні алгоритми, які виконують процеси пошуку, описуючи одночасно еволюцію набору точок у просторі пошуку, такі як оптимізація колоній мурах і генетичний алгоритм.

1.5. Алгоритм Кларка та Райта для ЗМТ

Алгоритм Кларка та Райта є найпопулярнішим евристичним алгоритмом для задачі маршрутизації транспортних засобів. Алгоритм обчислює всю економію S_{ij} між клієнтами i та j. Припускаючи, що c_{i0} — це вартість подорожі від клієнта i до j. Нижче наведено опис алгоритму Кларка і Райта для вирішення ЗМТ:

- Крок 1: Обчисліть заощадження

$$S_{ij} = c_{i0} + c_{0j} - c_{ij} \text{ для } i,j = 1,\dots,n \text{ і } i \neq j$$

Проранжуйте заощадження S_{ij} і перерахуйте їх у порядку спадання.

- Крок 2: Створіть «список заощаджень», обробляйте список заощаджень, починаючи з самого верхнього запису в списку (найбільшого S_{ij}). Для економії, що розглядається (S_{ij}), включіть посилання (i,j) до маршруту, якщо жодні обмеження маршруту не будуть порушені через включення (i,j). Необхідно розглянути наступні три випадки.
  - Випадок 1: якщо ні i, ні j вже не були призначені для маршруту, то ініціюється новий маршрут, що включає i та j.
  - Випадок 2: якщо саме одна з двох точок (i або j) вже була включена в існуючий маршрут, і ця точка не є внутрішньою частиною цього маршруту (точка є внутрішньою частиною маршруту, якщо вона не примикає до депо в порядок обходу точки), то посилання (i,j) додається до того самого маршруту. Якщо точка внутрішня і не порушує пропускну здатність, додайте (i,j) до того ж маршруту. Якщо це порушує пропускну здатність, створіть новий маршрут з точкою (замовником) i.

- Випадок 3: Якщо обидва маршрути i, j вже були включені до двох різних існуючих маршрутів, і жодна точка не є внутрішньою для маршруту, тоді обидва маршрути об'єднуються шляхом з'єднання i та j. Якщо вони внутрішні, то об'єднання неможливо здійснити.

- Крок 3 : Якщо ощадний список S_{ij} не вичерпано, поверніться до кроку 2, обробляючи наступний запис у списку; інакше зупинитись [14].

Розділ 2. Опис програмних засобів та середовища розробки реалізації сайту

2.1. Microsoft SQL Server

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

SQL Server має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані будь-якою мовою платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework.

На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність у порівнянні з загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server.

2.2. Entity Framework core

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану технологію від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping – відображення даних на реальні об'єкти). Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-якої СКБД, якщо для неї є потрібний провайдер.

За замовчуванням на цей час Microsoft надає ряд вбудованих провайдерів: для роботи з MS SQL Server, для SQLite, для PostgreSQL. Також є провайдери від

сторонніх постачальників, наприклад, для MySQL. Як технологія доступу до даних Entity Framework Core може використовуватися на різних платформах стека .NET.

Відмінною рисою Entity Framework Core, як технології ORM, є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконанні запиту транслює вираження LINQ в вирази, зрозумілі для конкретної СКБД (як правило, в вирази SQL).

2.3. ASP .Net Core

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів.

З одного боку, ASP.NET Core є продовженням розвитку платформи ASP.NET. Але з іншого боку, це не просто черговий реліз. Вихід ASP.NET Core фактично означає революцію всієї платформи, її якісна зміна.

ASP.NET Core тепер повністю є opensource-фреймворком. Всі вихідні файли фреймворку доступні на GitHub. Завдяки модульності фреймворку всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget [12].

ASP.NET Core містить в собі фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версіях платформи дані технології реалізувалися окремо і тому містили багато функціональності, що дублювалась. Зараз же вони об'єднані в одну програмну модель ASP.NET Core MVC. А Web Forms повністю пішли в минуле.

ASP.NET Core характеризується розширюваністю. Фреймворк побудований з набору незалежних компонентів. І ми можемо або використовувати вбудовану реалізацію цих компонентів, або розширити їх за

допомогою механізму спадкування, або створити і застосовувати свої компоненти зі своїм функціоналом.

Також було спрощено управління залежностями і конфігурація проекту. Фреймворк тепер має свій легкий контейнер для впровадження залежностей, і більше нема потреби застосовувати сторонні контейнери, такі як Autofac, Ninject. Хоча за бажанням їх також можна продовжувати використовувати.

Як інструментарій розробки ми можемо використовувати останні випуски Visual Studio, починаючи з версії Visual Studio 2015.

2.4. Мова програмування C#

C# — об'єктно-орієнтована мова програмування, вона є основною мовою розробки програм на платформі .NET корпорації Microsoft. У ній вдало поєднуються випробувані засоби програмування з останніми нововведеннями і надається можливість для ефективного і дуже практичного написання програм, призначених для обчислювального середовища сучасних підприємств [6].

Незважаючи на те, що C# є самодостатньою мовою програмування, у нього є особливий взаємозв'язок із середовищем виконання .NET Framework. Наявність такого взаємозв'язку пояснюється двома причинами. По-перше, C# спочатку призначався для створення коду, який повинен виконуватися в середовищі .NET Framework. І по-друге, використовувані в C# бібліотеки визначені в середовищі .NET Framework. На практиці це означає, що C# і .NET Framework тісно пов'язані один з одним, хоча теоретично C# можна відокремити від середовища .NET Framework.

На сьогодні C# є флагманською мовою Microsoft, оскільки вона повною мірою використовує нові можливості .NET. Інші мови програмування, хоча і підтримуються, вони мають застарілі прогалини у використанні .NET.

2.5. HTML/CSS

HTML (HyperText Markup Language) — основана на SGML текстова мова розмітки, призначена для маркування документів, що містять текст,

зображення, гіперпосилання, тощо. HTML-документи лежать в основі Веб, і відображаються із допомогою веб-браузерів. Разом із видимою інформацією, HTML-документи містять додаткові метадані, такі як, наприклад, мова тексту, автор документа, стислий підсумок [8].

Дані у форматі HTML нагадують звичайні текстові файли за винятком того, що деякі символи в них (так звані теги (tag)) інтерпретуються як розмітка. Розмітка надає документу деяку, визначену тегами, структуру: параграфи, розділи, абзаци, списки, малюнки, таблиці, колонтитули, індекси, зміст тощо. Всередині кожного блоку можна змінювати шрифт, розмір символів, колір тексту, виділяти текст курсивом та/або робити його напівжирним.

Головною особливістю HTML є спроможність використовувати гіперзв'язки (links), завдяки яким можливі посилання та переходи з поточної веб-сторінки на інші документи, як локальні (документи поточного сервера), так і такі, що знаходяться на серверах в найвіддаленіших регіонах земної кулі. Мова HTML, також дозволяє вставляти в документи зображення, звук, відео та ін. Перегляд HTML-документу здійснюється за допомогою веб-оглядача (таких програм як Internet Explorer, Netscape та ін.).

CSS (Cascading Style Sheets) - мова таблиць стилів, який дозволяє прикріплювати стиль (наприклад, шрифти і колір) до структурованих документів (наприклад, документами HTML і додатків XML). Зазвичай CSS-стилі використовуються для створення і зміни стилю елементів веб-сторінок і призначених для користувача інтерфейсів, написаних на мовах HTML і XHTML, але також можуть бути застосовані до будь-якого виду XML-документа. Відокремлюючи стиль подання документів від вмісту документів, CSS спрощує створення веб-сторінок і обслуговування сайтів.

Каскадні таблиці стилів описують правила форматування елементів за допомогою властивостей і допустимих значень цих властивостей. Для кожного

елемента можна використовувати обмежений набір властивостей, інші властивості не будуть чинити на нього ніякого впливу.

2.6. React.js

React – це бібліотека JavaScript, яка використовується для створення призначеного для користувача інтерфейсу. React був створений компанією Facebook, а перший реліз бібліотеки побачив світ у березні 2013 року.

React представляє ідеальний інструмент для створення масштабованих веб-додатків (у цьому випадку мова йде про фронтенд), особливо в тих ситуаціях, коли додаток являє SPA (односторінковий додаток) [9].

Віртуальний DOM

Вся структура веб-сторінки може бути представлена за допомогою DOM (Document Object Model) - організація елементів html, якими ми можемо маніпулювати, змінювати, видаляти або додавати нові. Для взаємодії з DOM застосовується мова JavaScript. Однак коли ми намагаємося маніпулювати html-елементами за допомогою JavaScript, то ми можемо зіткнутися зі зниженням продуктивності. А операції над елементами можуть зайняти деякий час. Однак якби ми працювали з коду js з об'єктами JavaScript, то операції проводилися б швидше.

Для розв’язання проблеми продуктивності якраз і з'явилася концепція віртуального DOM. Віртуальний DOM представляє легковажну копію звичайного DOM. І відмінною рисою React є те, що дана бібліотека працює саме з віртуальним DOM, а не звичайним.

Якщо з додатком потрібно дізнатися інформацію про стан елементів, то відбувається звернення до віртуального DOM.

Якщо необхідно змінити елементи веб-сторінки, то зміни спочатку вносяться в віртуальний DOM. Потім новий стан віртуального DOM порівнюється з поточним станом. І якщо ці стани розрізняються, то React

знаходить мінімальну кількість маніпуляцій, які необхідні до поновлення реального DOM до нового стану і виробляє їх.

У підсумку така схема взаємодії з елементами веб-сторінки працює набагато швидше і ефективніше, ніж якби ми працювали з JavaScript з DOM безпосередньо.

2.7. Redux+React.js

Ще одну форму побудови архітектури додатку на React представляє Redux. Redux є контейнером для управління станом додатка і багато в чому нагадує Flux. Redux не прив'язаний безпосередньо до React.js і може також використовуватися з іншими js-бібліотеками і фреймворками.

Ключові моменти Redux:

- Сховище (store): зберігає стан додатка.
- Дії (actions): деякий набір інформації, який виходить від додатка до сховища і який вказує, що саме потрібно зробити. Для передачі цієї інформації у сховища викликається метод dispatch () .
- Творці дій (action creators): функції, які створюють дії
- Reducer : функція (або кілька функцій), яка отримує дію і відповідно до цього дію змінює стан сховища

Загальну схему взаємодії елементів архітектури Redux можна виразити як на Рис.1:

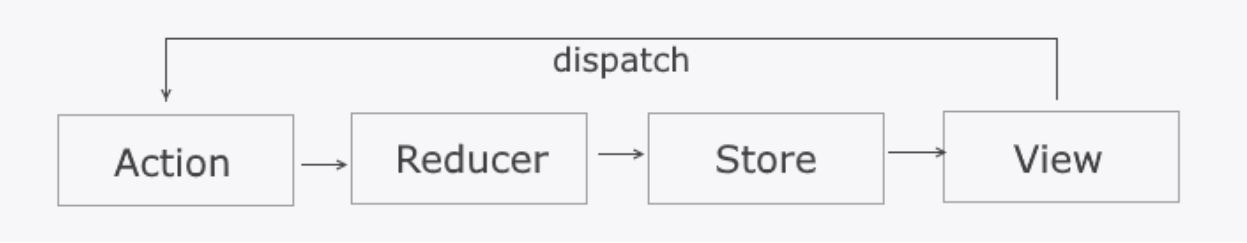

Рис. 2.7.1 – Схема взаємодії елементів архітектури Redux

З View (тобто з компонентів React) ми посилаємо дію, це дія отримує функція reducer, яка відповідно до дії оновлює стан сховища. Потім компоненти React застосовують оновлений стан зі сховища.

Розділ 3. Програмна реалізація проекту

3.1. Основна концепція та зміст веб-сайту

Проект представляє собою приватний веб-сайта деякої транспортної компанії. Він створений з метою спростити роботу адміністратора. Програма створює маршрут на кожен день для кожної машини конкретного депо. Задача користувача програми – прив’язати до вже існуючих замовлень автомобіль, який буде розвозити доставки, та додати адреса цих замовлень на мапу. Адміністратор зможе одразу поділитися готовою таблицею оптимальних шляхів з водіями.

Реалізовано такий основний функціонал, який задовольнятиме потреби компанії:

- перегляд усіх замовлень компанії
- створення нових складів
- створення нових машин з прив’язкою до наявних складів
- додавання нового клієнта до основної бази
- оформлення нового замовлення
- можливість створити оптимальний маршрут для кожної задіяної машини

Головна сторінка сайту – «Створити нове замовлення» містить наступні поля, потрібні для подальших математичних обрахунків:

- вага
- дата створення
- зручний час доставки
- ім’я замовника
- адреса доставки
- склад
- повідомлення з особливою приміткою від замовника.

Дизайн веб-сайту

Дизайн грає важливу роль в створенні сайту, тому він повинен відповідати наступним вимогам:

- кольорове і гармонійне оформлення
- текстова інформація зручна для читання
- зовнішній вигляд сайту повинен відповідати обраній тематиці
- навігація по сайту має бути інтуїтивною для спрощення у користуванні.

Вгорі нашого сайту є навігаційне меню, яке дозволяє зручно переміщуватись між сторінками. Внизу нашого сайту – футер і навігаційне меню у ньому.

Створена зручна і мінімалістична сторінка з процедурою логування та реєстрації, за допомогою якої можна здійснити вхід на даний веб-сайт і отримати доступ до повного функціонала.

У цьому пункті були виділені основні риси, які, на мою думку, є найважливішими при початковому плануванні веб-сайту.

3.2. Опис головних компонентів системи

Для зручного доступу клієнтів та працівників до сервісу, мною було обрано шлях створення веб-сайту, оскільки в наш час ми маємо можливість безперервного доступу до мережі Інтернет і можемо користуватись сервісом з будь-якої точки у світі. Для створення серверної частини я використала Microsoft Asp .Net Core, а для самого сайту бібліотеку React.js.

Отже створюємо порожній проект ASP .NET Core в середовищі Microsoft Visual Studio. Також, створимо папку «Contoller» де будемо зберігати наші контролери, які оброблятимуть запити до сервера. Додамо клас ValuesController в якому визначимо відповіді на HTTP запити. Потрібно створити 4 методи для обробки «GET», «PUT», «PUSH», «DELETE» запитів.

```
[Route("api/[controller]")]
[ApiController]
0 references
public class ValuesController : ControllerBase
{

    // GET api/values
    [HttpGet]
    0 references | 0 requests | 0 exceptions
    public ActionResult<string> Get()
    {
        VRP_Method vrp = new VRP_Method();
        var result = vrp.Calculate();
        result.ToString();
        return Ok(result);
    }
```

Рис. 3.2.1 – Приклад реалізації методу обробки «GET» запиту

Також створимо клас VRP, який і буде виконувати всі обрахунки і передавати їх серверу. Визначимо метод, який повертатиме коректні дані із сервера GoogleMaps. Даний метод використовує спеціальне посилання і ключ доступу для того, щоб отримувати найновіші дані стосовно найкоротших відстаней до певних адрес.

```
public string ReturnDistance(string start, string finish)
{

    WebRequest request = WebRequest.Create("https://maps.googleapis.com/maps/api/directions/json?origin=Львів," + start
     + "&destination=Львів," + finish + "&key=AIzaSyDGFlSfb-W4ZtI9DcE6B11DUuim-YDWcAA");
    // If required by the server, set the credentials.
    request.Credentials = CredentialCache.DefaultCredentials;
    request.Headers.Add("Method", "GET");

    // Get the response.
    WebResponse response = request.GetResponse();
    // Display the status.
    Console.WriteLine(((HttpWebResponse)response).StatusDescription);

    // Get the stream containing content returned by the server.
    // The using block ensures the stream is automatically closed.
    string responseFromServer = null;
    using (Stream dataStream = response.GetResponseStream())
    {
        // Open the stream using a StreamReader for easy access.
        StreamReader reader = new StreamReader(dataStream);
        // Read the content.
        responseFromServer = reader.ReadToEnd();
    }

    // Close the response.
    response.Close();
    return responseFromServer;
}
```

Рис. 3.2.2 – Реалізація методу для доступу до сервісу GoogleMaps

Щоб будь-який користувач міг без проблем користуватись даним сервісом створимо сайт за допомогою бібліотеки React.js для цього визначимо файл index.html із заданою розміткою, а також створимо певну структуру папок. Також для роботи із даною бібліотекою встановимо розширення Node.js, та запишемо певні конфігурації у файл package.json.

Далі ми створюємо реактівські компоненти, в яких і будемо відображати контент на нашому сайті, всі компоненти матимуть схожу структуру, але різні дані для відображень.

```
import React from "react";
import Title from "../Default components/Title";
import ContactForm from "./ContactForm";
import ContactCard from "../Default components/ContactCard";

class Contact extends React.Component {
  render() {
    return (
      <div className="contact page-content">
        <Title title="Contact" />
        <div className="contact-form">
          <div className="col-xs-12 col-sm-6 contact-information">
            <h2>Contact information</h2>
            <ContactCard
              Name="John Smith"
              Position="General contact manager"
              Email="john.smith@company.com"
              Phone="+38(099)-999-99-99"
              Address="Lviv, Levytskogo 27 street"
            />
          </div>
          <div className="col-xs-12 col-sm-6 form">
            <ContactForm />
          </div>
        </div>
      </div>
    );
  }
}

export default Contact;
```

Рис. 3.2.3 – Приклад створення компоненти «Contact»

На рис. 3.2.3 ми бачимо створення компоненти «Contact», що відповідатиме нашій «Contact» сторінці на сайті, дану сторінку використаємо для збору відгуків від клієнтів. За таким самим принципом створимо і наступні компоненти для нашого сайту.

Далі створимо основну компоненту, яка і буде робити запити до нашого серверу, та малюватиме карту Львова із обрахованими маршрутами.

```
import React from "react";
import Title from "../Default components/Title";
import { bindActionCreators } from "redux";
import { connect } from "react-redux";
import { actionCreators } from "./action";
import MapContainer from "../Default components/MapContainer";

class MyOrders extends React.Component {
  componentWillMount(){
      this.props.requestGetOrders();
  }
  render() {
    return (
      <div className="my-orders page-content">
        <Title title="My orders" />
        <div style={{height:"1000px"}}>{this.props.orders.length !== 0 ?
        <MapContainer myPoints={this.props.orders}/> :
        <div></div>}</div>
      </div>
    );
  }
}

export default connect(
    state => state.orderReducer,
    dispatch => bindActionCreators(actionCreators, dispatch)
  )  (MyOrders);
```

Рис. 3.2.3 – Створення компоненти MyOrders

Як бачимо з рисунку вище в даній компоненті викликається інша компонента MapContainer, яка і є нашою картою. Даний функціонал React.js дозволяє дуже зручно розробляти веб-сайт, оскільки сторінки можна створювати блоками із різним контентом, та вставляти в них інші блоки.

Компонента MapContainer повертає карту Львова, а також має підкомпоненти такі як Marker, Polyline. Marker – компонента яка малює позначку на карті, а компоненти Polyline – малює шляхи між заданими точками.

3.3. Графічний інтерфейс

Для того, щоб адміністраторам було зручно користуватися основним функціоналом сервісу було прийняте рішення створити веб-сайт. Для створення серверної частини я використала Microsoft Asp .Net Core, а для самого сайту бібліотеку React.js.

Запустимо сервер натиснувши на кнопку «RUN». Нам одразу відкриється URL посилання до сервера у веб-браузері. Наш запит виконується сервером, отже, ми можемо запустити сайт і спробувати робити запити до сервера. Запустимо сайт командою в командному рядку, як показано на Рис.3.3.1.

```
C:\Users\Dinara>cd C:\Users\Dinara\.all Docs\LNU\6 курс\diploma\VRP_frontend

C:\Users\Dinara\.all Docs\LNU\6 курс\diploma\VRP_frontend>npm start
```

Рис.3.3.1 – Приклад команди, за допомогою якої запускаємо наш веб-сайт

Далі у браузері відкривається посилання http://localhost:3000/, «Home» сторінка на створеному веб-ресурсі. До проекту був доданий файл «styles.css» з визначеною розміткою за замовчуванням. Завдяки цьому розмітка на сайті має сучасний вигляд, а також дозволяє зручно доступатись до повного функціонала веб-сайту.

Домашня сторінка має містити в собі основну суть сайту. Функціоналом сайту можуть користуватися лише залоговоні, тому єдина кнопка «Lets start» перенапрвіить користувача на сторінку логінації. Домашня сторінка показана на Рис. 3.3.2.

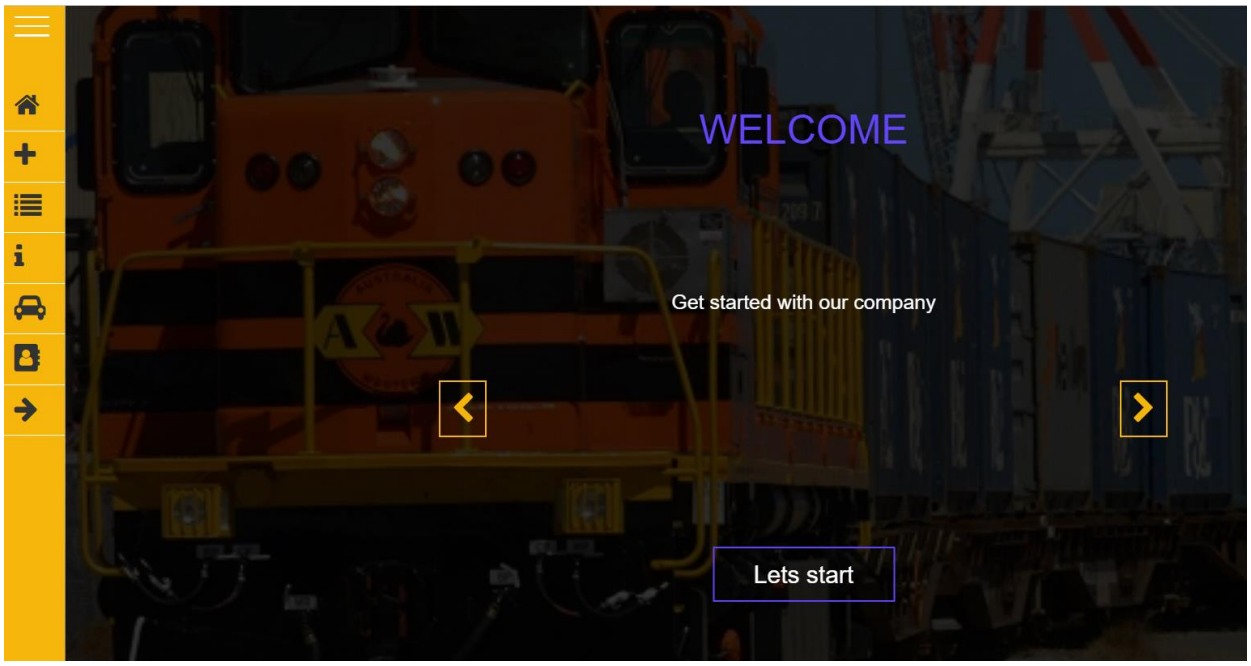

Рис.3.3.2 – Домашня сторінка сайту та навігаційне бокове меню

На рис.3.3.3 ми бачимо навігаційне меню, яке дозволяє зручно переміщуватись між сторінками, бачимо адаптивний та сучасний дизайн, який дозволяє зручно користуватись сайтом.

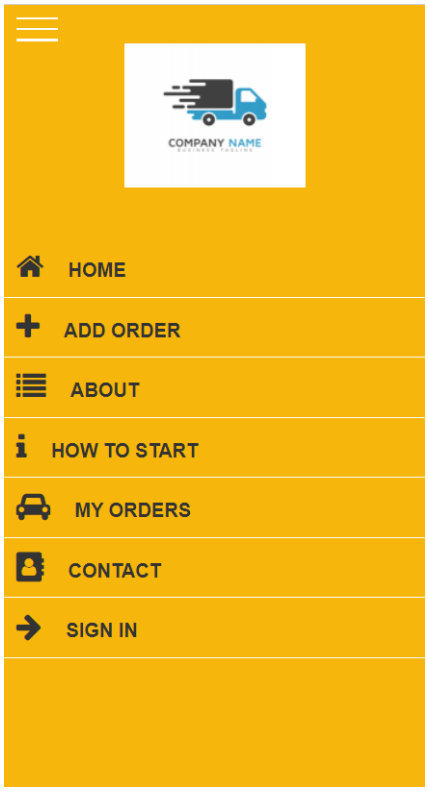

Рис.3.3.3 – Side navigation menu

Внизу сайту (Рис.3.3.4) представлений футер з короткою інформацією про сайт, навігаційним меню, яке також дозволяє зручно переміщуватись між сторінками даного веб-сайту, та посиланнями на соціальні мережі, де нас можна знайти.

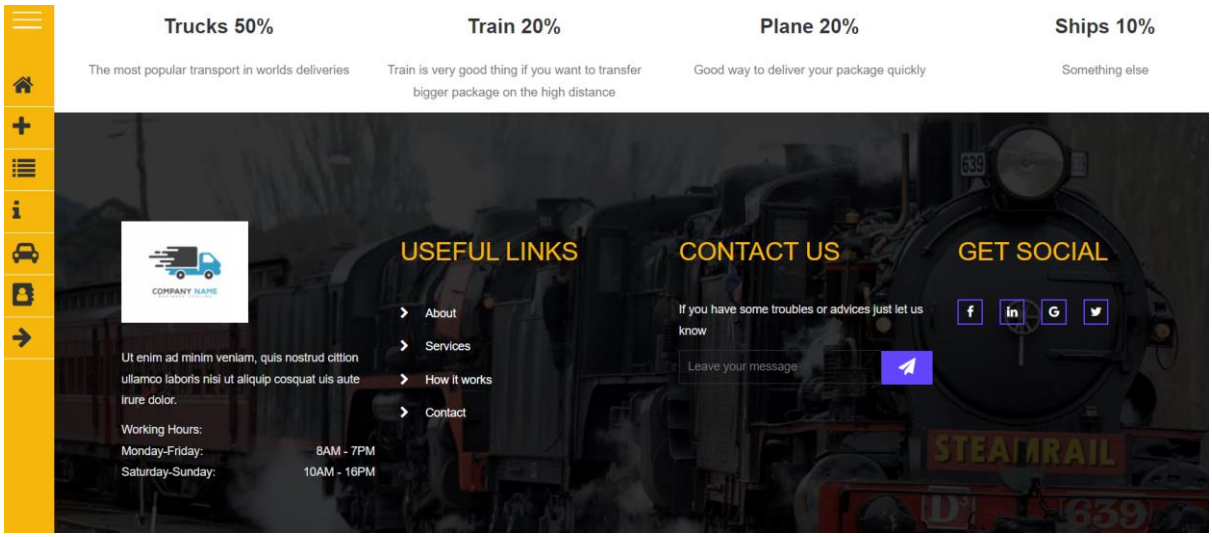

Рис.3.3.4 – Футер сайту

На рис.3.3.5 представлена сторінка за допомогою якої користувач може залишати відгук чи звернення до адміністраторів. Якщо користувач залогований, то дані про нього заповнюються автоматично і залишається набрати лише текст.

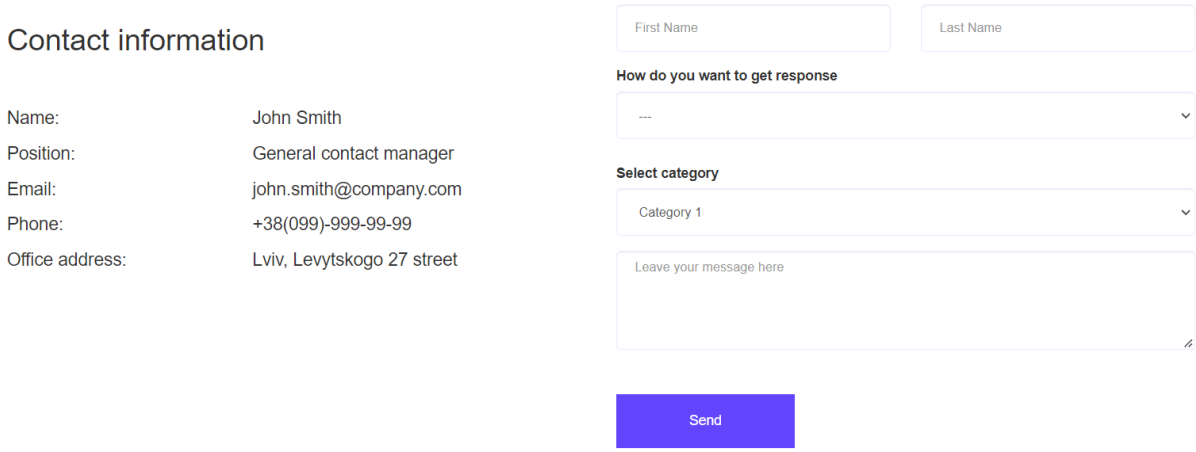

Рис.3.3.5 – Зворотній контакт

Реєстрація та вхід в систему

Користувач має можливість зареєструватися в системі, ввівши наступні обов'язкові дані:

- дійсний e-mail
- пароль(мінімум 10 символів) та його підтвердження.

Однією з переваг мого сайту є миттєва реєстрація користувача, саме тому обов'язкові дані мінімальні. Приклад логування наведено на Рис.3.3.6.

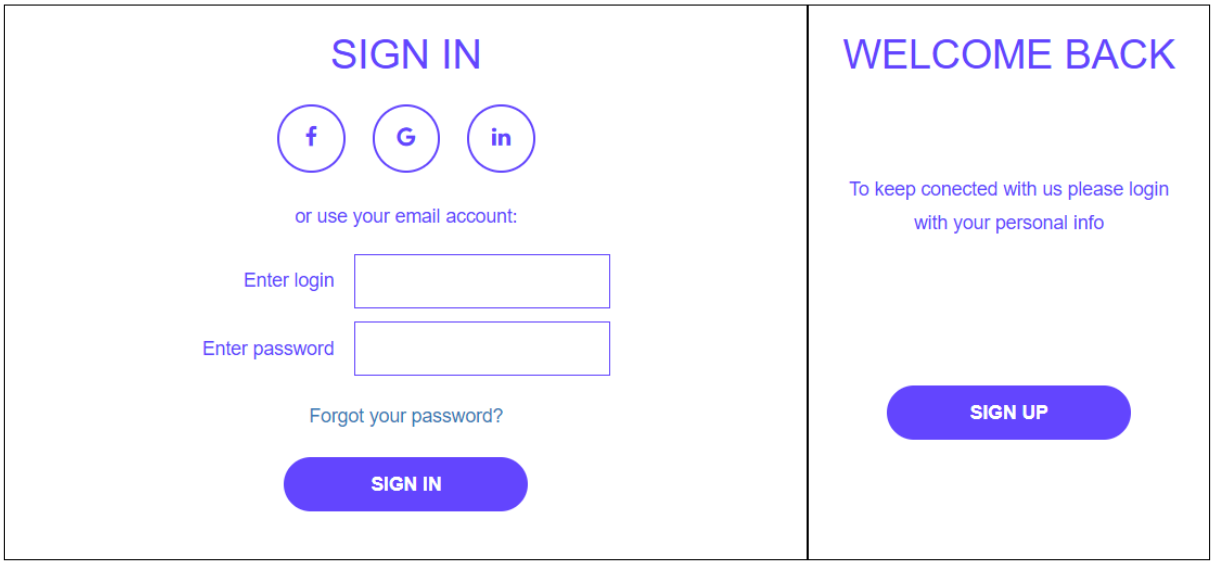

Рис.3.3.6 – Сторінка логування/реєстрації з вибраною опцією логування

Зареєстрований користувач має ввести свої дані та переходить до використання можливостей сайту в повному обсязі. Якщо користувач має права адміністратора, то для нього відкривається можливість додавати замовлення та привязувати його до машини, яка буде займатися доставкою вантажів. На рис. 3.3.7 зображена форма додавання нового замовлення. На ній відображено головні поля, у яких треба заповнити різну інформацію щодо замовлення:

- Information about customer – дані замовника, до якого прямуватиме доставка.

- Address – заповнити дані про адресу доставки замовлення та час, зручний для отримання замовлення.
- General order information – заповнити загальну інформацію про пакунок (вага, склад, зручний час доставки, особливі побажання клієнта).

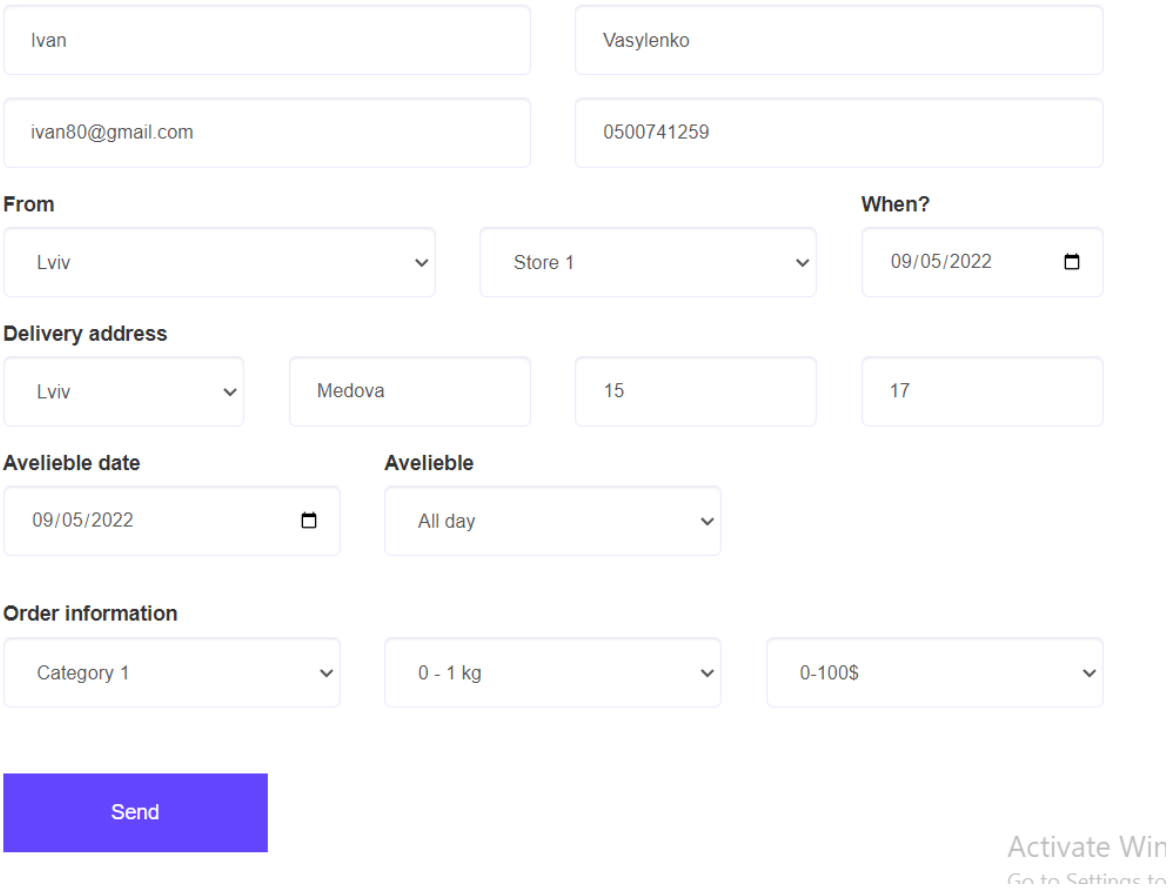

Рис.3.3.7 – Форма створення нової доставки

3.4. Реалізація задачі маршрутизації транспорту

Перейдемо до головної задачі проекту, а саме розв'язку задачі маршрутизації транспортних засобів. Після логінації в якості співробітника компанії, користувач, перейшовши на сторінку «My orders», водій отримує «сьогоднішній» маршрут, який він повинен виконати, починаючи та закінчуючи складом. Він отримає навігаційну карту із послідовністю проходження адрес

клієнтів, які обрахувала програма за допомогою GoogleMaps API. На рис. 3.4.1 та 3.4.2 зеленим маркером позначені клієнти компанії (адреси замовлень), а червоним – склад, з якого водій буде відправлятися і куди прибуває.

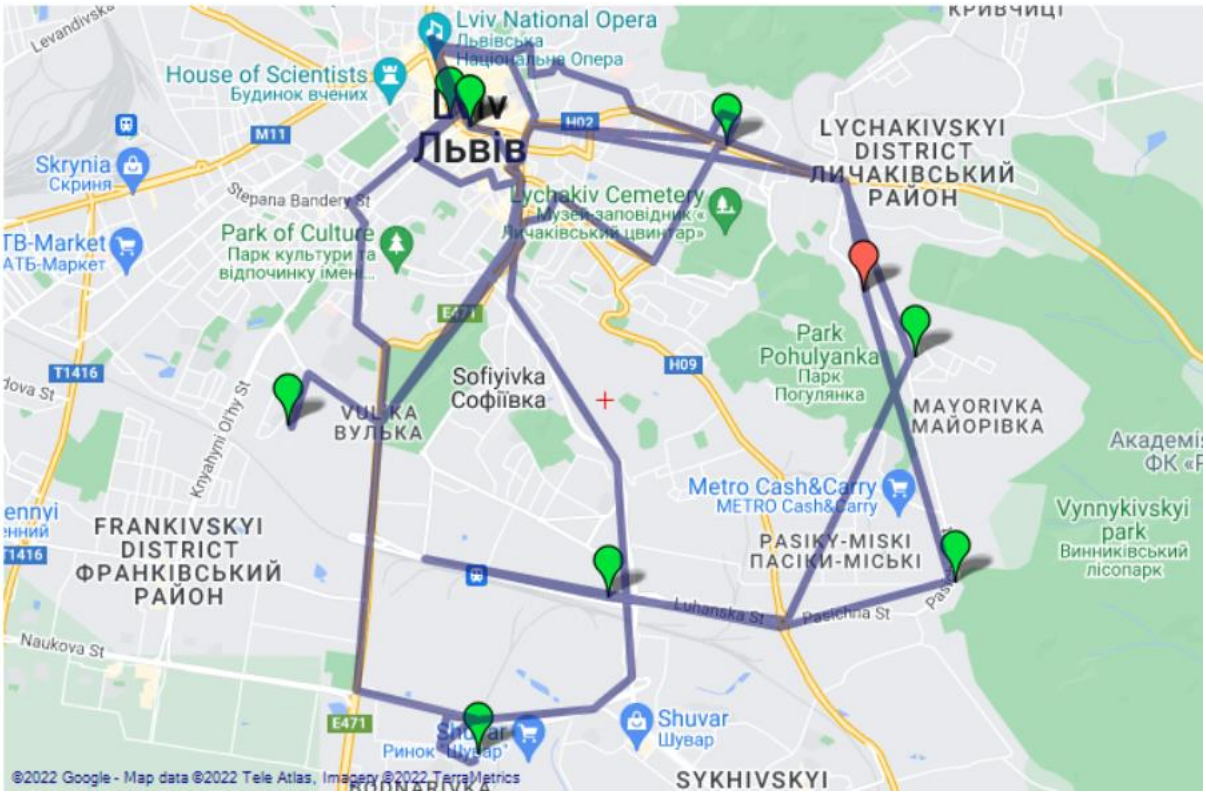

Рис.3.4.1 – Побудований маршрут 1

VRPService – головний клас проекту, який виконує всі обрахунки задачі маршрутизації транспорту. У ньому реалізовано метод GetRoute, який містить алгоритм пошуку оптимального шляху. Визначимо метод, який повертатиме коректні дані із сервера GoogleMaps.

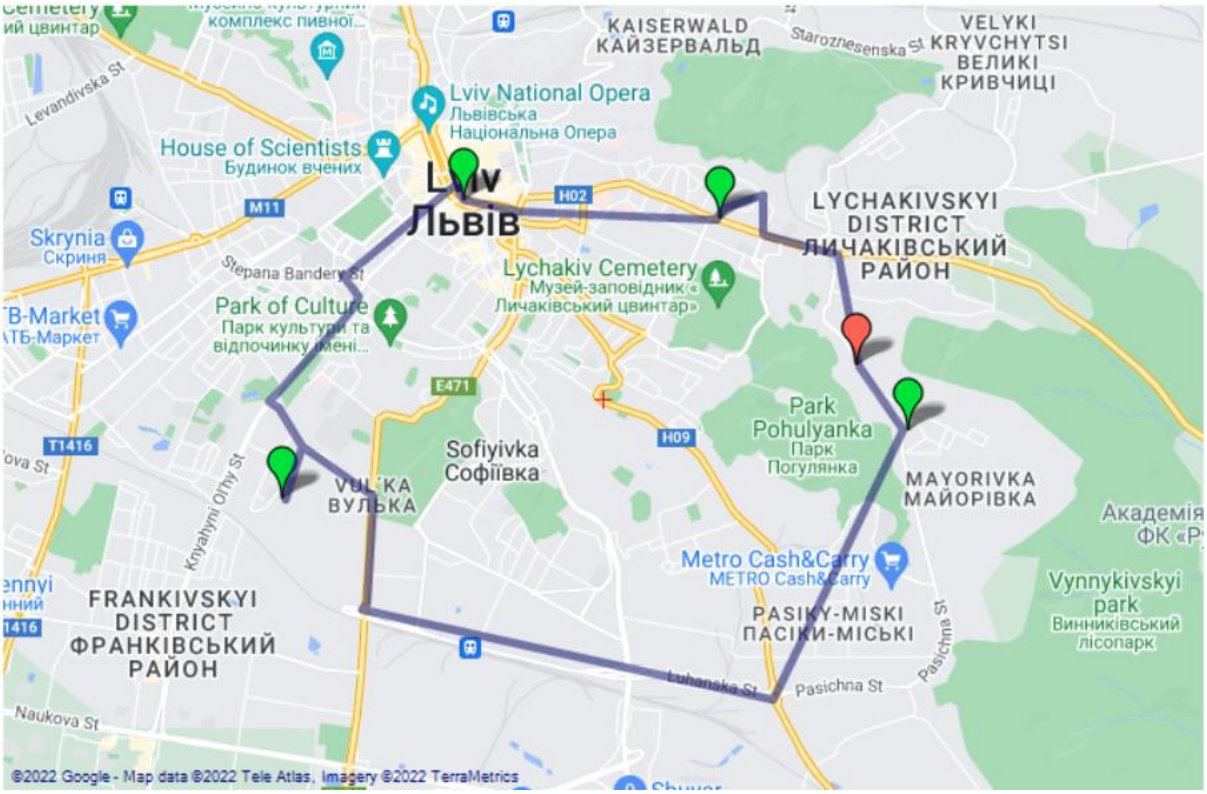

Рис.3.4.2 – Побудований маршрут 2

Дана програма повертає масив координат на карті, які були обраховані за допомогою евристичного методу та GoogleMaps API. Далі цей масив додається до карти, і описує карту шляхів Львова, оскільки саме у цьому місці працює компанія доставки, яка і є власником даного додатку.

Висновки

Моя робота присвячена створенню програми, яка надасть користувачеві можливість скласти оптимальний маршрут для доставлення або перевезення товарів. Під час написання програми було використано чимало передових технологій таких, як Microsoft SQL server, Entity Framework Core, ASP .NET CORE, React.js, Redux + React.js.

При виконанні магістерської роботи, я опрацювала матеріал, що стосується розв'язання задачі маршрутизації транспорту(VRP), та відповідно, на основі даного матеріалу, розробила алгоритм розв’язання задачі маршрутизації транспорту.

Також було створено веб-сайт із зручним користувацьким інтерфейсом, який дозволяє зручно та досить швидко розв’язати проблему маршрутизації транспорту на прикладі реальної карти міста Львів, та з певним набором масиву адрес користувачів. Дана програма: розв’язує задачу маршрутизації транспорту, виводить масив координат на карту за допомогою GoogleMap API, а також прокладає найоптимальніший маршрут між даними координатами з урахування реальних дорожніх шляхів.

Розроблений додаток повністю відповідає поставленим вимогам, які були перераховані вище. Основною його перевагою є сучасний дизайн та можливість використання цього веб-сайту будь-якою транспортною компанією.

Список використаної літератури

1. Кощєєв І. С. Оптимізація доставки вантажу споживачам з урахуванням його розміщення всередині транспортних засобів на основі евристичних методів. – 2016

2. Трофимов Д., Федуков А. Завдання маршрутизації транспорту (http://rain.ifmo.ru/cat/view.php/theory/unsorted/vrp-2006).

3. Буерсокс Д. Дж. Логістика: інтегрований ланцюг поставок. / Д. Дж. Буерсокс – М.: ЗАТ «Олімп-Бізнес», 2005 – 342с.

4. Пожидаєв М. С. Алгоритми рішення задачі маршрутизації транспорту. / М. С. Пожидаєв – 2010 – 245с.

5. Сластніков С. А. (2013) Застосування метаевристичних алгоритмів для задачі маршрутизації транспорту. / С. А. Сластніков – 2013 – 411с.

6. Ріхтер Дж. CLR via C#: (4th Edition) / Дж. Ріхтер – К. Питер, 2013. – 896с.

7. Емельянова Т.С. Об одном генетическом алгоритме решения транспортной задачи с ограничением по времени // Известия Южного федерального университета. Технические науки, Т. 81, № 4, 2008. С. 45-50.

8. Технічна підтримка з HTML5, CSS3 [Електронний ресурс] – http://www.w3schools.com/ .

9. Підримка з JS [Електронний ресурс] – https://learn.javascript.ru/ .

10. .Net підтримка [Електронний ресурс] – https://docs.microsoft.com/.

11. Підтримка під час розробки [Електронний ресурс] – https://stackoverflow.com/ .

12. Підтримка з ASP.NET [Електронний ресурс] – https://metanit.com/ .

13. Скукис А. Е. Оптимизационные задачи в транспортной логистике / А. Е. Скукис // Теорія оптимальних рішень. - 2015. - № 2015. - С. 106-113.

14. Dantzig, G. B. The truck dispatching problem [Текст] / G. B. Dantzig, J. H. Ramser. - Management Science, 1959.

15. Сергеев С.И., Сигал И.Х, Меламед И.И. Задача коммивояжера. Вопросы теории // Автоматика и телемеханика, № 9, 1989. С. 3-33.

16. Lenstra J.K., Kan A.H.G. Complexity of vehicle routing and scheduling problems // Networks, Vol. 11, No. 2, 1981. pp. 221-227.

17. Irnich S., Toth P., Vigo D. The family of vehicle routing problems // In:Vehicle Routing: Problems, Methods, and Applications / Ed. by Toth P., Vigo D. SIAM, 2014. pp. 1-36.

18. M. Savelsbergh. Local search in routing problems with time windows. Annals of Operations Research, 1985. – С. 305.