МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЯКУБЕНКО ДМИТРО ОЛЕКСАНДРОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій
к.т.н., доцент
_____ Т.В.Нескородєва

«___» червня 2021 р.

ІНФОРМАЦІЙНА СИСТЕМА МАРШРУТИЗАЦІЇ ВАНТАЖІВОК. Ч. 2 РОЗРОБКА ІНТЕРФЕЙСІВ

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Штовба С.Д.,
професор кафедри інформаційних технологій
д.т.н., професор

Оцінка:____/ ____/ _____
(бали за шкалою ЄКТС /за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2021

АНОТАЦІЯ

Якубенко Д.О. Інформаційна система маршрутизації вантажівок. Ч. 2 Розробка інтерфейсів. Спеціальність 122 «Комп'ютерні науки», освітня програма «Сучасні інформаційні технології та програмування», Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній (бакалаврській) роботі досліджена проблема маршрутизації вантажівок. Показані методи та підходи її вирішення. На основі цих підходів був встановлений оптимальний алгоритм для рішення і розроблена система, що буде вирішувати поставлену задачу. Створенно сучасний, реагуючо-адаптивний, кросбраузерний UI з використанням мови програмування JS та бібліотеки React. Були використані декілька API для побудови фронт-енду.

51 с., 17 рис., 53 джерел.

Ключові слова: VRP, React, JavaScript, Google Map API, маршрутизація, Directions API.

Abstract

Yakubenko DO Truck routing information system. Part 2 Interface development. Specialty 122 "Computer Science", educational program "Modern Information Technologies and Programming", Vasyl Stus Donetsk National University, Vinnytsia, 2021**.**

In the qualification (bachelor's) work the problem of truck routing is investigated. Methods and approaches to its solution are shown. Based on these approaches, the optimal algorithm for the solution was established and a system was developed that will solve the problem. Created a modern, responsive-adaptive, cross-browser UI using the JS programming language and the React library. Several APIs were used to build the front end.51 pp., 17 figs., 53 sources.Keywords: VRP, React, JavaScript, Google Map API, Routing, Directions API

ЗМІСТ

ВСТУП

Проблема маршрутизації транспортних вантажівок (VRP – Vehicle Routing Problem) – це комбінаторна задача оптимізації та цілочисельного програмування, яка задається питанням: “Який оптимальний набір маршрутів для проїзду вантажівок для доставки товару певному набору клієнтів?”.

Знаходження оптимального рішення VRP є задача NP складності, тому розмір проблем, які можна вирішити оптимально за допомогою математичного програмування або комбінаторної оптимізації, може бути обмеженим. Тому комерційні вирішувачі, як правило, використовують евристику через розмір реальних VRP, які їм потрібно вирішити.

Задача та її вирішення широко розповсюджені у промисловості, тому що використовування оптимізації маршруту може зберегти компаніям багато грошей, так як доставка товарів це одна з найважливіших задач яка також є доволі великою частиною від ціни продукту, зазвичай приблизно 10% [1], та і транспортний сектор приносить приблизно 10% від ВВП Європейського Союзу.

Актуальність проблеми зумовлена тим, що майже всі існуючі вирішувачі можуть вирішувати лише список «стандартних» задач, без їх ускладнення та поєднання, комбінування з іншими типами, тому ця частина є недостатньо вивченої та реалізованою.

Метою цієї роботи є розробка інформаційної системи, що буде будувати шлях вантажівок по точкам-замовникам, щоб мінімізувати час та витрати на перевезення товару, а також кількість вантажівок які потрібні для перевезень.

Актуальність теми дослідження. Грамотна маршрутизація доставки вантажів - основа успішної роботи кожного інтернет-магазину. Для своєчасного виконання замовлень клієнтів необхідно оперативно перевозити товари від постачальників на склад і зі складу до покупців. У багатьох великих компаній, що займаються інтернет-продажами є власний транспорт, інші користуються послугами кур'єрів або наймають сторонніх перевізників. І в тому і в іншому випадку потрібно своєчасно розрахувати потребу в транспортуванні вантажів, оцінити витрати на перевезення, швидко обробити заявки на транспорт, сформувати план доставки, побудувати маршрути, видати завдання водіям та експедиторам, проконтролювати виконання цих завдань. Саме тому маршрутизація вантажних перевезень я назвичайно актуальною на сьогоднішній день.

Мета. Розробка веб-системи для оперативного планування перевезень вантажів, що допоможе вибрати оптимальні способи доставки, побудова

ланцюжків поставок із залученням різних видів транспортних засобів, оптимізація завантаження транспорту.

Завдання дослідження. Реалізація Front-End частини для візуалізації рішення з бєк-енду на карті з допомогою даних інструментів:JS, Google API, Git, GitHub, WebStorm, React, Firebase,Firebase Hosting, Google Maps API, Places API, Directions API,Google Maps Platform,Google Cloud Platform.

Об'єкт дослідження. задача маршрутизації вантажівок з часовими вікнами (Vehicle Routing Problem with Time Windows).

Предмет дослідження. Веб-інтерфеси для задачі маршрутизації вантажівок.

Досягнення поставленої мети потребує вирішення таких задач дослідження **:**

- Розглянути поняття веб додатку.
- Обрати інструменти для створення front-end веб додатку.
- Розробити дизайн веб додатку.
- Створити front-end веб додаток, що буде задовольняти мету роботи.

Практична значущість роботи полягає в тому, що її рекомендації можуть

бути використані при вивчені мови програмування JS, React, а розроблений веб додаток знайти конкретну практичну реалізацію в своїй галузі.

Робота складається з 3 розділів, 15 пунктів, 17 рисунків та списку літератури, що містить 37 джерел, загальний обсяг роботи 51 сторінки.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-САЙТУ ТА ДЕТАЛІЗАЦЯ ЗАДАЧ ПРОЕКТУВАННЯ

1.1. Аналіз веб-інфтефейсів як предмету дослідження

Вебса́йт або сайт – це сукупність вебсторінок та залежного вмісту, який доступний у мережі Інтернет, які об'єднані як за змістом та за навігацією під єдиним доменним ім'ям. Фізично сайт може знаходитись як на одному, так і на декількох серверах.

Сайтом також можно називати вузол мережі Інтернет, комп'ютер, та будь-який пристрій за яким закріплена унікальна IP-адреса, і взагалі будь-який об'єкт в Інтернеті, за яким закріплена адреса, що ідентифікує його в мережі (FTP-site, WWW-site тощо).

1.2. Класифікація веб-сайти

Сайти класифікуються за доступністю сервісів, фізичним розташуванням та призначенням. Їх поділяють на:

• Відкриті – сервіси,які повністю доступні для відвідувачів і користувачів;

• Напіввідкриті – ті сервіси,для доступу до яких потрібно зареєструватися (зазвичай безкоштовно);

• Закриті – повністю закриті службові сайти (наприклад, корпоративні сайти,сайти державних структур та інш.), особисті сайти приватних осіб. Такі сайти доступні для приватного кола людей. Доступ нових людей можливий через запрошення.

За фізичним розташуванням,якщо сайт доступний усім користувачам з Інтернету, він вважається зовнішнім, з іншої сторони сайт, до якого доступ можуть здійснювати користувачі локальної мережі - є внутрішнім. Прикладами внутрішнього сайту можуть бути корпоративний сайт

підприємства,сайт державної структури,охоронної служби або сайт приватної особи в локальній мережі провайдера.

За різновидом сайти поділяють на:

• Бізнес-сайти – сайти, які містять інформацію про компанії та їх послуги, виконують функцію торгівлі;

• Інформаційні сайти – призначені для інформування відвідувачів, поширення новин, тематичні сайти, енциклопедії, словники та інші;

• Сайти соц. мереж – багатокористувацькі сайти, які наповнюються учасниками. Сайт являє собою автоматизоване соціальне середовище, яке дозволяє спілкуватися групі користувачів, об'єднаних різними інтересами;

• Веб-портал – універсальний сайт, через який можна вийти на інші ресурси Інтернету;

• Сайти сервісів – службові сайти, які існують у мережі, зокрема, сайти пошукових служб (Google, Bing,Yandex,Yahoo), поштові веб-сайти, форуми, он-лайн сховищ даних (Skydrive,Google Drive,iCloud,DropBox), сайти служб онлайнового документообігу (Google Docs), зберігання та обробки фотографій (Picnik, ImageShack, Panoramio, Photobucket), зберігання відео (YouTube,Tiktok), площадки для онлайн трансляцій(Youtube,Twitch,Livestream,Periscope,Mixer,Smashcast та інші).

1.3. Структура веб-сайтів

Зовнішній вигляд сайту має бути унікальним, проте в кожному сайті можна знайти спільні частини. На будь-якому сайті першою відкривається головна сторінка. На її розробку присвячують найбільше уваги, оскільки дослідження показали, що люди не читають інформацію, що відображається на моніторі, так уважно, як фізичний ресурс(книга,газета або журнал). Вони зазвичай поверхово переглядають її, наприклад, як рекламу. Якщо головна сторінка містить інформацію, яку шукав користувач, він читає її далі, а якщо ні — переходить до інших ресурсів, яких в Інтернеті безліч. У верхній частині головної сторінки розташована так звана шапка(хедер), яка дублюється на інших сторінках сайту. Це роблять для того щоб користувач мав можливість переходити по іншим сторінкам сайту,також через те,що ця частина відображається у вікні браузера першою і відвідувач насамперед звертає увагу на неї. Горизонтальне меню зазвичай розташовують у шапці, іноді дублюючи

його в нижній частині сторінки, а вертикальне — переважно в лівій частині сторінки, у місці, звідки відвідувач починає її переглядати. Меню є одним із найважливіших компонентів сайту, оскільки користувач постійно звертає на нього увагу, і тому вимоги до нього дуже високі. Меню має бути зручним, помітним і зрозумілим, інакше користувач не знайде, як перейти до потрібного розділу, і покине сайт. Пункти верхнього меню мають бути відділені один від одного. Гіперпосилання, розміщені в тексті чи у вигляді графічних об'єктів, дозволяють переходити на різні сторінки сайту або навіть на інші сайти. На сайтах із дуже великим обсягом інформації є розділення сторінок на різні рівні: третього рівня, а якщо необхідно — то й четвертого, п'ятого тощо. Загалом виділяють три різновиди структур веб-сайтів — лінійну, деревоподібну та довільну. Рухаючись по сайту із лінійною структурою, з головної сторінки ви перейдете на другу сторінку, з неї — на третю тощо. На сайті з деревоподібною структурою з головної сторінки ми можемо потрапити на одну зі сторінок другого рівня, звідти – на одну зі сторінок третього рівня і т.д.. Сайт із довільною структурою зовсім неорганізований, але в цьому й полягає принцип його створення. Подорожуючи таким сайтом, ви можете переходити з однієї його сторінки на інші в різні способи, і ваш шлях не обов'язково має бути таким самим. Вибір структури сайту визначається особливостями, які збираються розв'язувати за допомогою веб-сайту.

Також можна навести ще чотири приклади структур сайту. Кожна із цих структур має свої недоліки та переваги у проектуванні веб-сайту.

Стандартна. Основна сторінка містить посилання на різні сторінки вебсайту, а дані сторінки містять посилання, відповідно, на основну веб-сторінку. Це простий та дуже поширений спосіб організації сайту.

Каскад. Посилання в данному випадку в сторінках задані таким чином, що існує тільки один шлях обходу сторінок вебсайту. За каскадного способу організації сторінок відвідувач сайту може переміщуватись тільки в одному з напрямків — вперед або назад.

Хмарочос. В данній моделі відвідувач може опинитись на деяких сторінках, якщо вони йдуть правильним шляхом. Це нагадує підйом до потрібної кімнати у великому хмарочосі. У цьому випадку всі сторінки сайту містять посилання на інші сторінки, і користувач може легко перейти з будь-якої сторінки на іншу. Вона популярна в тих випадках, коли посиланнями на документи користуються не надто часто.

Фронтенда (англ. Front-end) називається клієнтська сторона,що призначена для користувача інтерфейсу по відношенню до програмно-апаратної частини будь-якого сервісу.

Просто кажучи, frontend - це розробка функціональності і призначеного для користувача інтерфейсу, який працює на стороні клієнтського додатку або веб-сайту. Сюди відносять все, що користувач бачить, відкриваючи веб-сторінку. Для створення зручного і бажанного продукту фронтенд-розробники співпрацюють з програмістами, дизайнерами, UX-аналітиками та інш..

Відкривши сторінку будь-якого сайту ви побачите перед собою його інтерфейс. Далі клацнувши правою кнопкою миші і натиснувши «Подивитися код сторінки». Цей код і є наш фронтенд. Він описує все, що ви бачите, починаючи від верстки сторніки і всієї палітри використаних кольорів, завершуючи шрифтами, графічними елементами і т. п.

1.4. Google Maps API

Google Maps API - це набір різних інструментів для додавання карт Google в мобільні та веб-додатки для створення на їх основі багатофункціональних рішень для внутрішнього використання.

Google постійно поновлює та вдосконалює свої дані, тому їх карти - одні з найбільш точних у світі.

Google Maps API дозволяє розробникам використовувати карти, високодетальні знімки, панорами вулиць, інформацію про завантаженість доріг та їх напрямок, місцях інтересу і багато іншого, без необхідності на фінансові затрати для створення і оновлення власних базових карт і даних.

Головні можливості API Google Карти

Для використання API Google Карти потрібно мати певний рівень знань в мові з об'єктно-орієнтованим програмуванням. Для цієї мети нам відмінно підійде JavaScript. Для повноцінної роботи потрібно отримати Google Api ключ .За допомогою унікального ключа компанія Google може не тільки відстежувати додатки працюють з сервісом API, але і у разі потреби зв'язатися з власником додатку. Зручно і практично для роботи з картами використовують накладення (шари), за рахунок яких за координатами можна будувати геометричні фігури - примітиви, з їх допомогою відбувається візуалізація на карті.

Основні переваги API Google Maps

Цей сервіс має свої переваги, які зацікавлюють не тільки програмістів, але і звичайних користувачей:

• Простота. Сервіс простий і зрозумілий для використання, користувачеві потрібно всього лише вказати стартове місце розташування і вписати дані, які потрібно відобразити на карті. На вказаному місці з'явиться помітка(маркер), при натисканні на який відобразиться вся інформація об'єкта;

• Стабільна візуалізація. Користувачі сайту відразу побачать можливі шляхи та як краще проїхати, або пройти(в останньому оновлені була додана можливість побудови шляхів для єко-транспорту);

• Доступний функціонал. Сервіс можна оформити під загальну або кастомну тематику , додатково розширивши його можливості під потреби вашого додатку;

• Допомагає донести посил додатку до користувачів сайту;

• Можна додавати та поновлювати актуальну інформацію про всі філіали та магазини компанії, а також про інші підприємства.

ВИСНОВОК ДО РОЗДІЛУ 1.

В цьому розділі було розглянуто загальне поняття про front-end web додатків. Визначено необхідність використання фреймворку та API, висвітлено їх переваги та недоліки.

РОЗДІЛ 2

ДЕТАЛІЗАЦІЯ ЗАВДАННЯ ТА ОБГРУНТУВАННЯ ІНСТРУМЕНТАРІЮ

2.1 Варіанти Front-end бібліотек та Map API для проектних рішень

Бібліотеки для Front-end.

Для вирішення поставленної задачі, існує декілька підходів. Для JavaScript існує три основних, великих фрейморвка ,які надають розробнику інструменти для розробки та конфігурування всього проекту, і їх назви це React.js ,Angular.js та Vue.js.

React – лідер серед існуючих бібліотек для JavaScript для розробки юзер інтерфейсу (UI). React пропонує відповідь на вхідні дані користувача, використовуючи новий метод візуалізації веб-сайтів.

Компоненти данного фреймворку були розроблені компанією Facebook. React був запущений як інструмент JavaScript з відкритим вхідним кодом в 2013 році. На даний час React випереджає своїх конкурентів, таких як Angular,Vue.js і Bootstrap, три найбільш використаних бібліотек JavaScript нашого часу.

React - це JavaScript бібліотека GUI з відкритим вхідним кодом, яка зосереджена на конкретній меті - ефективному виконанні завдань в рамках розробки інтерфейсу. Його можна віднести до категорії "V" в архітектурному шаблоні MVC (модель-вид-контролер).

Як розробник JavaScript, ви легко зрозумієте основи React. І вже за кілька днів зможете почати розробляти свій перший додаток.

Щоб розширити і закріпити ваші знання, продовжуйте читати. Прекрасне джерело інформації - керівництва на офіційному сайті. Там ви також знайдете багато корисного про те, як використовувати інструмент: відео, навчальні посібники та інші важливі дані.

React ефективно оновлює процес DOM (об'єктна модель документа). Можливо, ви вже й самі знаєте, що цей процес може викликати багато розчарувань під час розробки веб-додатків. На щастя, React використовує віртуальні DOM, тому ви можете уникнути багатьох проблем.Інструмент дозволяє створювати віртуальні DOM-дерево та розміщувати їх в пам'яті. В результаті кожен раз, коли відбувається зміна в реальному DOM, віртуальний змінюється негайно.Ця система буде перешкоджати тому, щоб фактичний

DOM не форсувати постійні оновлення. Як результат, не страждатиме швидкість вашої програми.

Angular - фреймворк для JavaScript. Бібліотека допомагає створювати веб-додатки - сайти, завантажуються тільки один раз. У даній технології досить низький поріг входження, тому вона часто стає наступним і цілком логічним кроком після вивчення JS. Цінність таких фахівців сильно підвищується - у людини розширюються можливості і закріплюється розуміння того, що програмування - це про постійне навчання та розвиток.

Angular більше підходить для великих проектів з жорсткою структурою. У ньому є досить багато готових рішень, більш продумана система збору та зберігання інформації. Це спрощує конструювання великих веб-сайтів і забезпечує надійне їх функціонування.

Можна використовувати Angular як для гібридних, так і для SPA-додатків. Останні є лише однієї сторінки сайтами, створеними таким чином, що переходячи на псевдостраніцу користувач не завантажує нову інформацію. Оновлюються виключно динамічні дані. Виходить, завантаження необхідна тільки один раз, незалежно від того, на яку псевдостраніцу користувач потрапив першої. На стандартних сайтах кожна нова сторінка підтягується окремо - люди з неякісним інтернет-з'єднанням змушені чекати, спостерігаючи за порожнім екраном, скелетон або прелоадера.

Нове покоління SPA-додатків, PWA, створюють, щоб сайт міг працювати і оффлайн. Їх можна досить легко завантажити на смартфон. Подібне дійсно можливо і для цього використовується Angular. Варто враховувати, що динамічні дані такий продукт довантажувати не зможе. Користувач, буде вільно працювати з контентом, вже присутнім на пристрої. Але SPA і PWA - далеко не все, що можна зробити на Angular.

Vue - це веб-фреймворк для створення призначених для користувача інтерфейсів на мові програмування JavaScript. Vue створений для поступового внесення в існуючу програму. Він вирішує різні завдання рівня виду (view), спрощує роботу з іншими бібліотеками та дозволяє створювати складні односторінкові сайти (SPA, Single-Page Applications,Landing-Page).Для роботи з фреймворком, вам потрібно знати HTML, CSS і звичайно ж JavaScript на базовому рівні.

Vue дозволяє розділяти весь код програми на компоненти і збирати їх в єдине додаток. Компоненти можна використовувати повторно в будь-яких інших додатках.

Map API рішення.

Для даної задачі було підібрано два готових рішення:Google Maps та Microsoft Bing Maps.Обидві карти мають відкриту документацію та API,з яким можна працювати.

Вибір був не складний,оскільки Google Maps має значну перевагу завдяки великій кількості реалізованих прикладів,детальній документації та значної кількості додаткових API для різних задач,тому я використовую Google Maps API.

2.2.Інструменти для розробки веб додатку.

Name Duration for...	vue-v2.6.2-keyed	angular-v8.2.14-keyed	react-v16.8.6-keyed
create rows	160.9 ± 2.4	157.3 ± 3.0	176.2 ± 3.9
replace all rows	133.1 ± 1.2	137.5 ± 1.4	145.7 ± 1.3
partial update	223.8 ± 7.9	152.3 ± 3.4	218.7 ± 2.8
select row	305.9 ± 10.7	74.8 ± 2.7	124.6 ± 5.1
swap rows	69.9 ± 2.8	441.7 ± 2.4	445.5 ± 2.1
remove row	29.0 ± 0.5	27.7 ± 0.8	27.6 ± 2.1
create many rows	1,300.6 ± 15.1	1,406.4 ± 36.5	1,827.0 ± 75.1
append rows to large table	310.2 ± 4.5	303.3 ± 13.0	345.9 ± 6.8
clear rows	156.3 ± 1.1	248.6 ± 3.3	149.2 ± 2.1
slowdown geometric mean	1.24	1.32	1.46

Рисунок 2.1 - JS Framework Benchmark: Angular vs React vs Vue

Як ви можете бачити вище, Vue при виборі рядків значно повільніший, ніж Angular і React. З іншого боку, Angular та React не дуже ефективні при обміні рядків.

Це єдині суттєві відмінності в показниках відтворення - і в більшості випадків вони не дадуть помітних результатів. Оскільки вибір рядків є більш розповсюдженою функцією, ніж обмін рядками, я б сказав, що цей орієнтир

ставить Vue на третє місце після Angular та React, які поділяють найвищу позицію.

Що стосується пам'яті та часу завантаження, то React та Vue оцінюють дуже добре, але Angular трохи повільніший. Для завантаження базового сценарію Angular може зайняти 150 мс, для його запуску потрібно більше пам'яті.

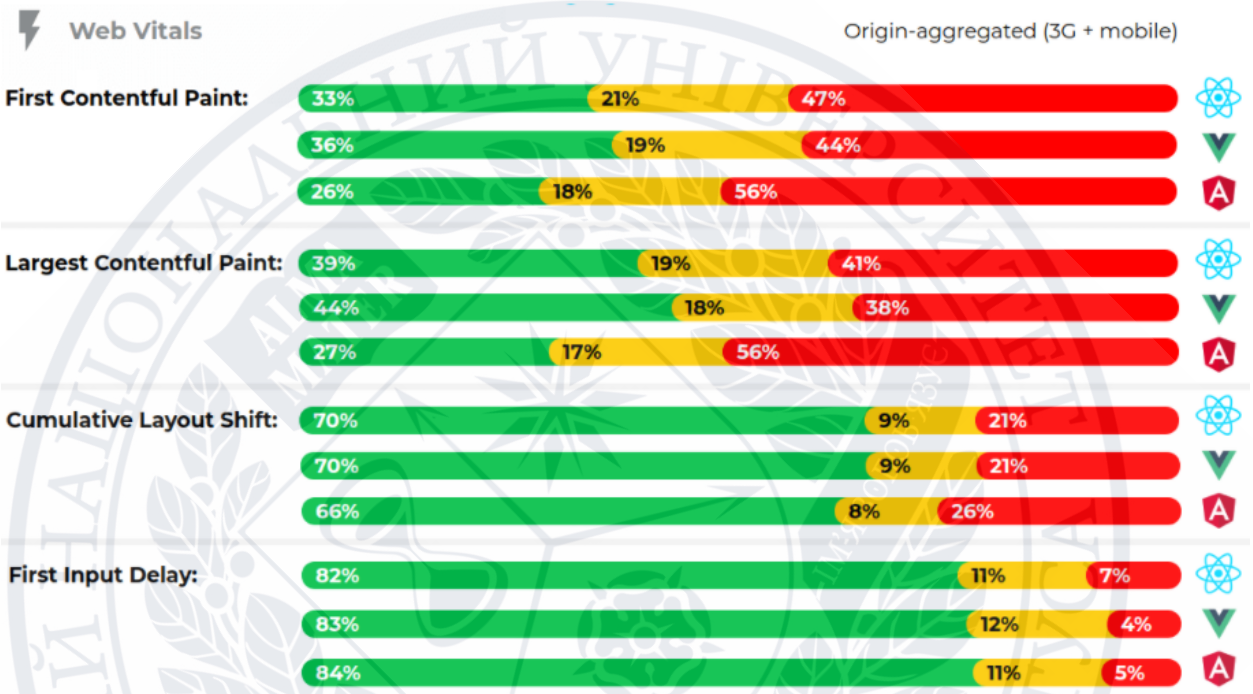


Рисунок 2.2 - Perf Track: Angular vs React vs Vue

Perf Track від Google Chrome Labs показує виробничі дані з тисяч веб-сайтів. На цю статистику впливає багато іншого, і не лише рамки вибору, але давайте розглянемо цифри.

Віртуальний DOM

Віртуальний DOM (VDOM) - це одна із відомих концепцій програмування, в якій ідеальне або «віртуальне» уявлення призначеного для юзера інтерфейсу зберігається в пам'яті та синхронізується з «реальним» DOM за допомогою бібліотеки, такий як ReactDOM. Даний процес називається узгодженням.

Даний підхід забезпечується декларативним React API: ви говорите React, в якому стані повинен перебувати UI, і він гарантує, що DOM відповідає цьому стану. Це відволікає маніпуляції з атрибутами, обробку подій і ручне

оновлення DOM, які в іншому випадку вам довелося б використовувати для створення вашого застосування.

«Віртуальний DOM» є швидше патерном, ніж конкретної технологією. У світі React термін «віртуальний DOM» зазвичай асоціюється з елементами React, оскільки вони є об'єктами, що представляють призначений для користувача інтерфейс. React, однак, також використовує внутрішні об'єкти, звані «волокнами» («fibers»), для зберігання додаткової інформації про дерево компонентів. Вони також можуть вважатися частиною реалізації «віртуального DOM» в React.

Після того, як ми розглянули, що таке VDOM, настав час поговорити про те, як він використовується в React.

1. React використовує патерн проектування «Спостерігач» (observer) і реагує на зміни стану

У React кожна частина UI є компонентом і майже кожен компонент має стан (state). При зміні стану компонента, React оновлює VDOM. Після поновлення VDOM, React порівнює його поточну версію з попередньою. Цей процес називається «пошуком відмінностей» (diffing).

Після виявлення об'єктів, що змінилися в VDOM, React оновлює відповідні об'єкти в RDOM. Це істотно підвищує продуктивність в порівнянні з прямими маніпуляціями DOM. Саме це робить React високопродуктивної бібліотекою JavaScript.

2. React використовує механізм пакетного (batch) поновлення RDOM

Це також позитивно впливає на продуктивність. Названий механізм передбачає відправку оновлень у вигляді пакетів (набору, серії) замість відправки оновлення окремих програмного забезпечення при кожній зміні стану.

Повторна отрисовка UI - найбільш витратна частина, React забезпечує точкову і групову перерисовку RDOM.

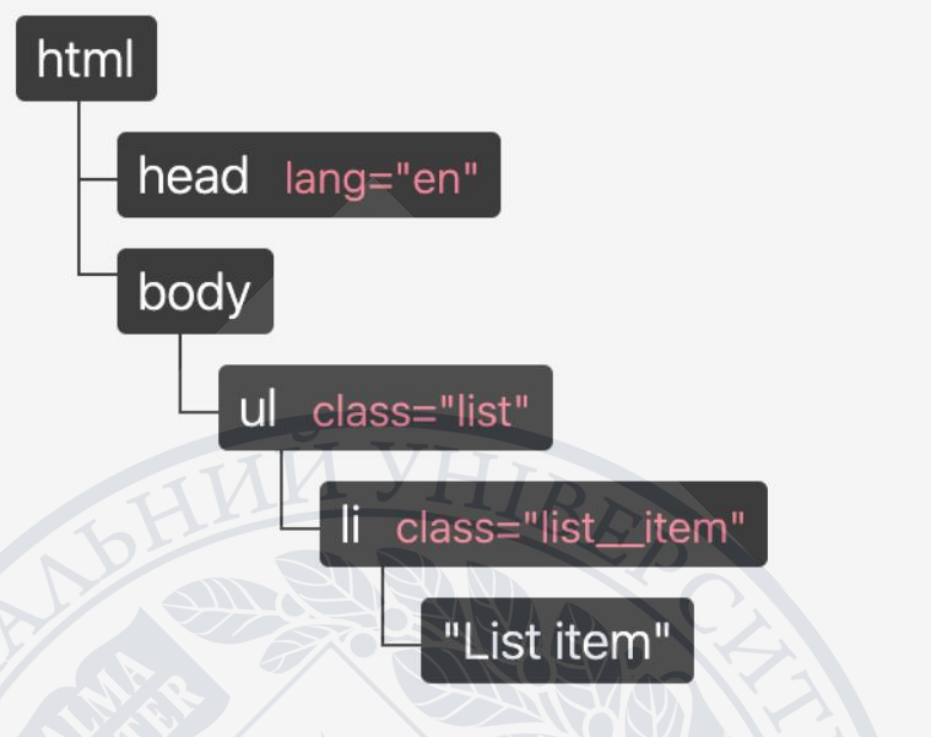


Рисунок 2.3 – Вигляд DOM-дерева

JSX

Компонент React зазвичай написаний на JSX. Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники мають можливість писати на чистому JavaScript. JSX чимось схожий на PHP,XHP.

Методи життєвого циклу

Методи життєвого циклу — це різновиди методів, які вбудовуються за допомогою ReactJS. Вони дозволяють розробнику обробляти дані в різних точках життєвого циклу програми React. Наприклад:

shouldComponentUpdate — це метод життєвого циклу відповідає за оновлення компоненту.

componentWillMount — це метод життєвого циклу, який каже Javascript налаштувати певні дані на вставлення у віртуальний DOM.

componentDidMount — це метод життєвого циклу, використовується для додавання JSON-даних, а також для визначення властивостей та станів.

render є найважливішим методом життєвого циклу, необхідним у будь-якому компоненті. Метод render — потрібен для з'єднання з JSX та відображення власного JSX.

Вкладені елементи

Декілька елементів на одному рівні загортаються в один елемент контейнера, наприклад елемент , або повернутий як масив.

Атрибути

JSX надає ряд аналогічних атрибутів,які призначені для відображення тих, що надаються у форматі HTML. Атрибути користувача також можуть бути передані компоненту. Всі атрибути будуть отримані компонентом як реквізит.

Directions API - API Directions - це веб-сервіс, який використовує запит HTTP для повернення напрямків руху у форматі JSON або XML між точками на карті. Ви можете отримати вказівки щодо транспорту, таких як транзит, їзда, піші прогулянки або їзда на еко-транспорті.

Для розрахунків напрямків, які реагують у режимі реального часу на введення користувачем (наприклад, всередині елемента користувальницького інтерфейсу), ви можете використовувати API напрямків або, якщо ви використовуєте API JavaScript Maps Maps, скористатися службою напрямків. Для використання на сервері ви можете використовувати клієнти різних мов для служб Карт Google.

Places API - це сервіс, який повертає інформацію про місця, що використовують наші запити. Місця визначаються в API як заклади, географічне розташування або визначні пам'ятки.

API Places використовує ідентифікатор місця, щоб однозначно ідентифікувати місце. Докладніше про формат та використання цього ідентифікатора в API Places та інших API див. У документації ідентифікаторів місць

Maps Javascript API - дозволяє налаштовувати карту за власним вмістом та зображеннями на веб-сторінках та мобільних пристроях. API JavaScript Maps пропонує чотири основних типи карт (дорожню карту, супутникову, гібридну та карту місцевості), які ви можете змінити, використовуючи шари стилей, елементи керування та події, а також різні служби та додаткові бібліотеки.

Завантаження API JavaScript Maps.

API JavaScript Maps Maps завантажується за допомогою тегу сценарію, який можна додати вбудовано у ваш файл HTML або динамічно за допомогою окремого файлу JavaScript. Ми рекомендуємо вам переглянути обидва підходи та вибрати той, який найбільш підходить для структури коду у вашому проекті.

Roads API – визначає для користувача дороги, по яких рухався транспортний засіб, та надає додаткові метадані дороги, такі як обмеження швидкості.

Перш ніж розпочати розробку за допомогою Roads API, перегляньте вимоги до автентифікації (вам потрібен ключ API), а також інформацію про використання та платіжну інформацію API.

The Roads API дозволяє зіставити координати GPS з геометрією дороги та визначити обмеження швидкості на цих відрізках дороги. API доступний через простий інтерфейс HTTPS і надає такі послуги:

Прив’язка до доріг Ця послуга повертає найкращу геометрію дороги для заданого набору координат GPS. Ця послуга займає до 100 GPS-точок, зібраних по маршруту, і повертає подібний набір даних із пунктами, прив'язаними до найбільш вірогідних доріг, по яких рухався транспортний засіб. За бажанням ви можете попросити інтерполяцію точок, в результаті чого вийде шлях, який плавно слідує геометрії дороги.

Найближчі дороги Ця послуга повертає окремі сегменти доріг для заданого набору координат GPS. Ця послуга займає до 100 точок GPS і повертає найближчий відрізок дороги для кожної точки. Передані бали не повинні бути частиною безперервного шляху.

Обмеження швидкості Ця послуга повертає встановлене обмеження швидкості для сегмента дороги. Послуга "Обмеження швидкості" доступна для всіх клієнтів, які мають ліцензію на відстеження активів. Для клієнтів Google Maps Platform Premium Plan, які перейшли на ціноутворення на виплату, ця функція залишається активною.

2.3. Огляд дизайнерського інструменту Figma

Figma - векторний сервіс для розробки інтерфейсів з можливістю організації спільної роботи. Працює у двох форматах: у браузері та як додаток на десктопі. Зберігає онлайн-версії файлів, з якими працював користувач.

Даний редактор підходить як для створення простих прототипів і дизайн-систем, так і складних проектів (мобільні додатки, веб-сайти). У 2018

році платформа стала одним із тих інструментів для розробників і дизайнерів, що найбільш швидко розвиваються.

Переваги Figma для роботи

Вихідні тексти документів зберігаються в хмарі.

Не потрібно пересилати макети, викладати їх в хмару і контролювати версії. Просто заходиш в акаунт Figma і бачиш оригінал. Якщо співробітника немає на роботі, не доведеться шукати макет на його комп'ютері - все в командному доступі.

Командна робота над макетами

Як і в Google Docs, в Figma можна працювати над документом разом: давати доступ на перегляд і редагування, паралельно працювати над макетом - на екрані буде видно курсори різного кольору.

Завдяки цьому всі учасники проекту краще розуміють контекст. Наприклад, дизайнер розробляє прототип додатка, а UX копірайтер прямо в Figma пише тексти для інтерфейсу.

2.4.Огляд мови програмування JavaScript.

JavaScript — мова програмування для розробки динамічних веб-додатків. Найчастіше використовується для створення вебсторінок, що надає можливість на стороні клієнта взаємодіяти з користувачем, керувати браузером, обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

-написання сценаріїв вебсторінок для надання їм інтерактивності

-створення односторінкових та прогресивних вебзастосунків (React, AngularJS, Vue.js)

-програмування на боці сервера (Node.js(Express.js))

-стаціонарних застосунків (Electron, NW.js)

-мобільних застосунків (React Native, Cordova)

-ценаріїв в прикладних програмах (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter)

-всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

2.5. Мова розмітки гіпертексту Html

Html - HTML (англ. HyperText Markup Language — мова розмітки гіпертексту) — це мова тегів, засобами якої здійснюється розмічання вебсторінок для мережі Інтернет. Браузери отримують HTML-документи з вебсервера або з локальної пам'яті й передають документи в мультимедійні вебсторінки. HTML описує структуру вебсторінки семантично і спочатку включені сигнали для зовнішнього вигляду документа.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами, написаними з використанням кутових дужок. Теги, такі як і безпосередньо вводять вміст на сторінку. Інші теги, такі як<img /> <input /> <p>оточують і надають інформацію про текст документа і можуть включати інші теги як піделементи. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

HTML може вбудовувати програми, написані на мові сценаріїв, наприклад JavaScript, що впливає на поведінку та вміст вебсторінок. Включення CSS визначає вигляд і компонування вмісту. World Wide Web Consortium (W3C), який супроводжує стандарти HTML та CSS, заохочує використання CSS над явним презентаційним HTML з 1997 року.

HTML був винайдений Тімом Бернерс-Лі, фізиком з дослідницького інституту ЦЕРН в Швейцарії. Він придумав ідею інтернет-гіпертекстової системи.

Hypertext означає текст, що містить посилання на інші тексти, які глядачі можуть отримати негайно. Він опублікував першу версію HTML в 1991 році,

складається з 18 тегів HTML. З тих пір кожна нова версія мови HTML з'явилася з розміткою нових тегів і атрибутів (модифікаторів тегів).

Згідно із Довідником HTML Element Reference від Mozilla Developer Network, в даний час існує 140 тегів HTML, хоча деякі з них вже застаріли (не підтримуються сучасними браузерами).

Через швидке зростання популярності HTML тепер вважається офіційним веб-стандартом. Специфікації HTML підтримуються і розробляються консорціумом World Wide Web (W3C). Ви можете перевірити останній стан мови в будь-який час на веб-сайті W3C (англ).

Найбільшим оновленням мови стало впровадження HTML5 в 2014 році. Було додано кілька нових семантичних тегів до розмітки, які показують зміст їх власного контенту, наприклад <article>, <header> і <footer>.

HTML-документи - це файли, які закінчуються розширенням .html або .htm. Ви можете переглядати його за допомогою будь-якого веб-браузера (наприклад, Google Chrome, Safari або Mozilla Firefox). Браузер читає HTML-файл і відображає його вміст, щоб користувачі інтернету могли його переглядати.

Зазвичай середній веб-сайт включає кілька різних HTML-сторінок (англ). Наприклад: домашні сторінки, звичайні сторінки, сторінки контактів матимуть окремі HTML-документи.

Кожна HTML-сторінка складається з набору тегів (також званих елементами), які ви можете назвати будівельними блоками веб-сторінок. Вони створюють ієрархію, яка структурує контент за розділами, параграфами, заголовкам і іншим блокам контенту.

Більшість елементів HTML мають відкриття і закриття, в яких використовується синтаксис <tag> </ tag>.

2.6. CSS

Каскадні таблиці стилів, які залюбки називають CSS, - це проста мова дизайну, призначена для спрощення процесу зробити веб-сторінки презентабельними.CSS обробляє зовнішній вигляд частини веб-сторінки. За допомогою CSS ви можете керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром та розміщенням стовпців, які фонові зображення чи кольори використовуються, дизайном макета, варіаціями відображення для різних пристроїв та розмірами екрану а також безліч інших ефектів.CSS легко вивчити і зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Найчастіше CSS поєднується з мовами розмітки HTML або XHTML.

Переваги CSSCSS економить час - Ви можете написати CSS один раз, а потім повторно використовувати той самий аркуш на декількох HTML-сторінках. Ви можете визначити стиль для кожного елемента HTML і застосувати його до скільки завгодно веб-сторінок.Сторінки завантажуються швидше - якщо ви використовуєте CSS, вам не потрібно кожного разу писати атрибути тегів HTML. Просто напишіть одне правило CSS тегу та застосуйте його до всіх випадків входження цього тегу. Тож менше коду означає швидший час завантаження.Простота обслуговування - Щоб внести глобальні зміни, просто змініть стиль, і всі елементи на всіх веб-сторінках будуть оновлені автоматично.Покращені стилі HTML - CSS має набагато ширший набір атрибутів, ніж HTML, тому ви можете набагато краще виглядати на своїй HTML-сторінці в порівнянні з атрибутами HTML.Сумісність декількох пристроїв - таблиці стилів дозволяють оптимізувати вміст для більш ніж одного типу пристроїв. Використовуючи один і той же документ HTML, різні версії веб-сайту можуть бути представлені для портативних пристроїв, таких як КПК та мобільні телефони, або для друку.Глобальні веб-стандарти - Зараз атрибути HTML застаріли, і рекомендується використовувати CSS. Тож непогано почати використовувати CSS на всіх HTML-сторінках, щоб зробити їх сумісними з майбутніми браузерами.

2.7. JetBrains WebStorm

JetBrains WebStorm —середовище веб-розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm це спеціалізована версія PhpStorm, пропонуюча підмножину з його можливостей. WebStorm постачається з додатковими плаґінами JavaScript (такими як для Node.js), котрі доступні безкоштовно.

WebStorm підтримує такі веб мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду, навігацію по коду, рефакторинг, зневадження та інтеграцію з GIT(Github,GitLab та інші). Одна з переваг середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується рефакторинг.

Основні можливості.

Інтеграція з GIT,такими як: Subversion, Git, GitHub, Perforce, Mercurial, CVS.Дані системи контролю версій підтримуються зі старту з можливостями побудови списку змін і відкладених змін

Інтеграція з системами відстеження помилок

Модифікація файлів .css, html, .js з одночасним переглядом результатів (Live Edit, в деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»)

Віддалене розгортання за протоколами FTP, SFTP, на монтованих мережевих дисках тощо з можливістю автоматичної синхронізації

Можливості Zen Coding і Emmet

Підтримка:

Web, Angular, React, Vue.js, Server, Node.js, Meteor, Mobile, Ionic, Cordova, React, Native, Desktop, Electron, Live Edit.

LiveEdit — можливість WebStorm, котра з'явилася з версії 5 і дозволяє одночасно редагувати код html, css або javascript і бачити, як результат відображається в браузері . Для цього потрібна підтримка такої можливості з боку браузера, тому WebStorm при установці ставить плагін для Google Chrome.

Підтримка node.js

WebStorm підтримує зневадження застосунків у node.js. Також підтримується повний набір функцій редагування застосунків на javascript — як для виконання на сервері, так і в браузері: автодоповнення, навігація по коду, рефакторінг і перевірка на помилки.

Для node.js підтримується також виведення повідомлень node.js на окрему вкладку в IDE.

LESS, Sass, SCSS

Мови стилів LESS, Sass і SCSS, які розширюють можливості описів стилів у CSS, повністю підтримуються в WebStorm, зокрема, підтримується рефакторинг коду для них, коли треба змінити вираз (наприклад, #a9a9a9) на змінну (наприклад @grey), для того, щоб зробити код читанішим і простіше перевизначати параметри (наприклад, шляхом присвоєння їм значення @grey: #a9a9a9)

Підтримка CoffeeScript

У версіях від WebStorm 5 для CoffeeScript є навігація за кодом, автодоповнення, рефакторинг, підсвічування синтаксису і перевірка на помилки.

Підтримка JavaScript, HTML, CSS в IntelliJ IDEA

JetBrains також розробляє і підтримує інше середовище розробки — IntelliJ IDEA з аналогічними можливостями підтримки JavaScript, HTML і CSS.

Firebase Hosting забезпечує швидкий і безпечний хостинг для вашого веб-додатки, статичного і динамічного контенту і мікросервісов.

Firebase Hosting - це хостинг веб-контенту виробничого рівня для розробників. За допомогою однієї команди ви можете швидко розгорнути веб-додатки і обслуговувати як статичний, так і динамічний контент в глобальній CDN (мережі доставки контенту). Ви також можете пов'язати хостинг Firebase з хмарними функціями або Cloud Run для створення і розміщення мікросервісов в Firebase.

Хостинг Firebase створений для сучасного веб-розробника. Веб-сайти та додатки стали більш потужними, ніж будь-коли, з появою інтерфейсних JavaScript-фреймворків, таких як Angular, і інструментів статичних генераторів, таких як Jekyll. Незалежно від того, розгортаєте ви просту цільову сторінку додатка або складне прогресивне веб-додаток (PWA), хостинг надає вам інфраструктуру, функції та інструменти, адаптовані для розгортання та управління веб-сайтами та додатками.

Використовуючи інтерфейс командного рядка Firebase, ви розгортаєте файли з локальних каталогів на своєму комп'ютері на наші сервери хостингу. Крім обслуговування статичного контенту, ви можете використовувати хмарні функції для Firebase або Cloud Run для обслуговування динамічного контенту і розміщення мікросервісов на своїх сайтах. Весь контент обслуговується через SSL-з'єднання з найближчого прикордонного сервера в нашій глобальній мережі CDN.

Ви також можете переглянути та протестувати свої зміни перед запуском. Використовуючи Firebase Local Emulator Suite, ви можете емулювати свій додаток і внутрішні ресурси по локально розміщеному URL-адресою. Ви також можете поділитися своїми змінами по тимчасовому URL-адресою попереднього перегляду і налаштувати інтеграцію GitHub для спрощення ітерацій під час розробки.

Firebase Hosting пропонує полегшені варіанти конфігурації хостингу для створення складних PWA. Ви можете легко переписати URL-адреси для маршрутизації на стороні клієнта, налаштувати призначені для користувача заголовки і навіть обслуговувати локалізований контент.

Для обслуговування вашого контенту Firebase пропонує кілька варіантів домену та субдомена:

За замовчуванням, кожен проект Firebase має вільні домени на web.app і firebaseapp.com доменів. Ці два сайти обслуговують один і той же розгорнутий контент і конфігурацію.

Ви можете створити кілька сайтів, якщо у вас є пов'язані сайти і додатки, які обслуговують різний контент, але при цьому використовують одні і ті ж ресурси проекту Firebase (наприклад, якщо у вас є блог, панель адміністратора і загальнодоступне додаток).

Ви можете підключити власне доменне ім'я до сайту, розміщеному на Firebase.

Firebase автоматично надає SSL-сертифікати для всіх ваших доменів, щоб весь ваш контент передавався безпечно.

Firebase - американська компанія, постачальник клауд послуг, була заснована в 2011 році Ендрю Лі і Джеймсом Темпліном, і придбана в 2014 році корпорацією Google.

Основний сервіс - хмарна Система Управління Базами Даних класу NoSQL, яка дозволяє розробникам будь-яких додатків зберігати і синхронізувати дані між декількома додатками. Підтримана особливість інтеграції з додатками під операційні системи Android та iOS, реалізована API для додатків на JavaScript, Java, Objective-C і Node.js, також можливо працювати з базою даних в стилі REST з ряду JavaScript-фреймворків, включаючи AngularJS, ReactJS, Vue.js, Ember.js і Backbone.js. Передбачено API для шифрування даних.

Серед різних послуг, що надавалися компанією - запущений 13 травня 2014 року хостинг для зберігання статичних файлів (таких як CSS, HTML, JavaScript), що забезпечує доставку через CDN і сервіс аутентифікації клієнта з використанням коду тільки на стороні клієнта з підтримкою входу через Facebook, GitHub, Twitter і Google (FirebaseSimpleLogin).

Firebase надає такі функції, як аналітика, бази даних, обмін повідомленнями та звіти про аварійне, тому можна швидко працювати і зосередитися на своїх користувачів.

Firebase побудований на інфраструктурі Google і масштабується автоматично навіть для великих програм.

Продукти Firebase добре працюють індивідуально і разом, вони чудово обмінюються даними.

Firebase зберігає та синхронізує дані, додані у глобальному масштабі, швидко і безпечно доставляє ресурси додатків та має просту і безпечну аутентифікацію користувачів.

Що таке Git

Git – це система контролю версій, яка надає можливість розробникам відстежувати зміни в файлах та працювати над одним проектом командою.

Підхід Git до зберігання даних схожий на набір знімків мініатюрної файлової системи. Кожен раз, коли ви зберігаєте стан свого проекту в Git, система запам'ятовує, як виглядає кожен файл в цей момент, і зберігає посилання на цей знімок.

Переваги Git:

-Безкоштовний та відкритий вихідний код. Можна безкоштовно скачати і вносити будь-які зміни у вихідний код;

-Невеликий і швидкий. Виконує всі операції локально, що збільшує його швидкість. Крім того, Git локально зберігає увесь репозиторій в невеликий файл без втрати даних;

-Резервне копіювання. Git ефективний для зберігання бекапів, тому відомо мало випадків, коли хтось втрачав дані при використанні Git;

-Просте розгалуження. В інших системах контролю версій створення веток- втомлива і трудомістка задача, так як весь код копіюється в нову гілку. У Git управління гілками реалізовано набагато простіше і ефективніше.

GitHub – найбільш популярний веб-сервіс для хостингу IT-проектів та їх спільної розробки. Даний сервіс заснований на системі контролю версій Git і був розроблений на Ruby on Rails і Erlang компанією GitHub. Сервіс безкоштовний для проектів з відкритим кодом і невеликих приватних проектів, надаючи їм всі можливості (включаючи SSL), а для великих корпоративних проектів пропонуються різні платні тарифні плани. Творці сайту називають GitHub «соціальною мережею для розробників». Крім розміщення коду, учасники сервісу можуть спілкуватися, коментувати правки один одного та допомагати у рішені питань , а також стежити за новинами. За допомогою можливостей Git програмісти всього світу можуть об'єднувати свої репозиторії.

GitHub пропонує зручний інтерфейс для цього і може відображати внесок кожного учасника у вигляді дерева. Для проектів є особисті сторінки,

невеликі Вікі і система стеження за вадами. Прямо на сайті можна переглянути файли проектів з підсвічуванням синтаксису для більшості мов програмування. Можна створювати приватні репозиторії, які будуть видні тільки вам і вибраним вами людям. Раніше можливість створювати приватні репозиторії була платною. Є можливість прямого додавання нових файлів в свій репозиторій через веб-інтерфейс сервісу. Код проектів можна не тільки скопіювати через Git, а й скачати у вигляді звичайних архівів з сайту. Крім Git, сервіс підтримує отримання і редагування коду через SVN і Mercurial. На сайті є pastebin-сервіс gist.github.com для швидкої публікації фрагментів коду.

Також для цієї мови існує велика кількість фрейморків для написання рішень для будь яких задач. І рішення поставленої задачі буде будуватися на стеку фреймворка Spring. Spring сильно спрощує розробку веб-додатків на базі стеку JavaEE (Java Enterprise Edition), що є набором специфікацій, що розширюють звичайну Java для створення веб-сервісів. Spring має величезну спільноту користувачів, що означає, що можна безкоштовно знайти навчальні матеріали та курси по ньому. Більше того, фреймворк допомагає автоматично налаштовувати всі компоненти додатку, сприяє створенню тестів та взагалі тестування додатків надаючи за замовчуванням інструменти для юніт тестів та інтеграційних тестів, допомагає уникати написання анотацій, складних XML конфігурувань, має вбудовані HTTP сервери (Jetty, Tomcat) тестування веб-додатків, надає доступ до дуже зручних інструментів таких як Spring Data, Spring Security, Spring JDBC. Також фреймворк надає безліч плагінів для роботи з вбудованими базами даних, та базами, що знаходяться в пам’яті машини, дозволяю легко підключатися до всіх популярних баз даних, таких як Oracle, PostgreSQL, MySQL, MongoDB та інші. Як вже зрозуміло, надає всі потрібні будь якому розробнику інструменти, та навіть більше.

Більше того, ця мова є досить популярною, про це говорить статистика від GitHub за 2020 рік. Незважаючи на те що ця мова трішки загубила свою популярність за останній рік, вона все ще знаходиться у числі перших трьох мов. Це надає змогу отримувати для цієї мові актуальні знання, приклади коду та інше.

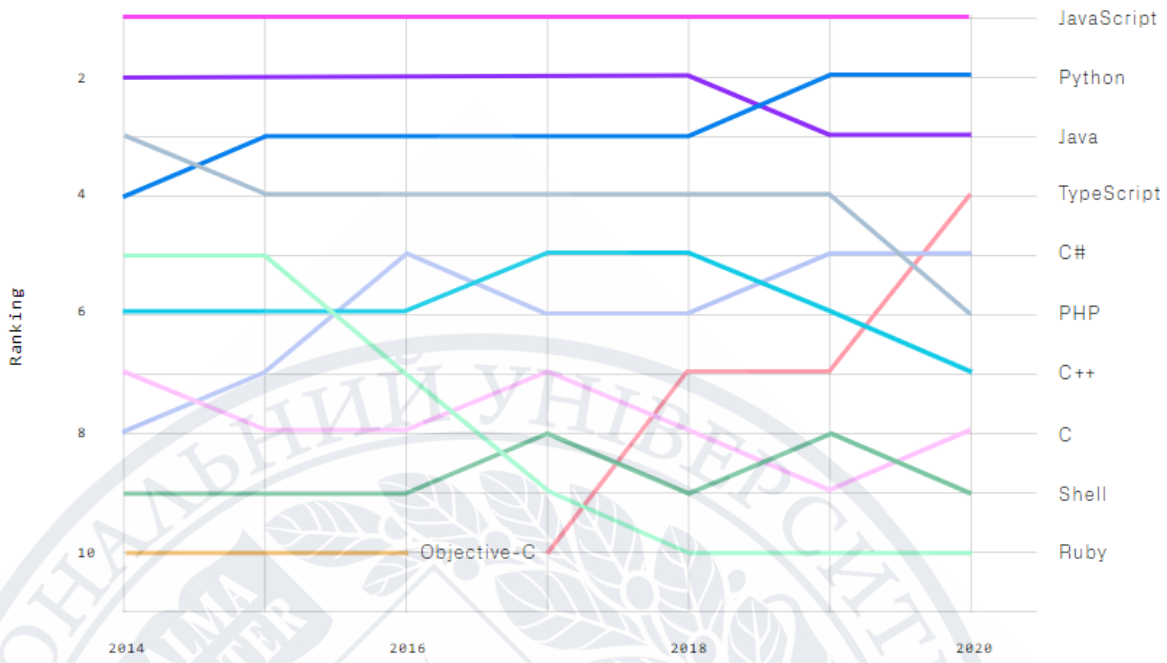


Рисунок 2.4 – Найпопулярніші мови програмування на GitHub

Все це надає змогу швидко розпочати розробку, мінімізувати час який потрібний для написання розповсюджених задач та спростити повсякденні проблеми.

2.8. Формат JSON, метод toJSON

JSON (JavaScript Object Notation) - це загальний формат для представлення значень і об'єктів. Його опис задокументовано в стандарті RFC 4627. Спочатку він був створений для JavaScript, але багато інших мов також мають бібліотеки, які можуть працювати з ним. Таким чином, JSON легко використовувати для обміну даними, коли клієнт використовує JavaScript, а сервер написаний на Ruby / PHP / Java або будь-якому іншому мовою.

JavaScript надає методи:

JSON.stringify для перетворення об'єктів в JSON.

JSON.parse для перетворення JSON назад в об'єкт.

Метод JSON.stringify (student) бере об'єкт і перетворює його в рядок.

Отримана рядок json називається JSON-форматованим або серіалізовані об'єктом. Ми можемо відправити його по мережі або помістити в звичайне сховище даних.

Зверніть увагу, що об'єкт в форматі JSON має кілька важливих відмінностей від об'єктного литерала:

Рядки використовують подвійні лапки. Ніяких одинарних лапок або зворотних лапок в JSON. Так 'John' стає "John".

Імена властивостей об'єкта також полягають в подвійні лапки. Це обов'язково. Так age: 30 стає "age": 30.

JSON.stringify може бути застосований і до примітивів.

JSON підтримує такі типи даних:

- Об'єкти {...}

- Масиви [...]

- примітиви:

- рядки,

- числа,

- логічні значення true / false,

- null.

JSON - це формат, який зберігає структуровану інформацію і в основному використовується для передачі даних між сервером і клієнтом.

Файл JSON являє собою більш просту і легку альтернативу розширенню з аналогічними функціями XML (Extensive Markup Language).

Розробники використовують JSON для роботи з AJAX (асинхронний JavaScript і XML). Ці формати добре працюють разом для досягнення асинхронної завантаження збережених даних, а це означає, що веб-сайт може оновлювати свою інформацію без оновлення сторінки.

Цей процес легше реалізувати з JSON, ніж з XML / RSS. І сьогодні, коли багато сайтів використовують AJAX, файл .json став дуже популярним.

Крім того, він дозволяє користувачам запитувати дані з іншого домену методом, який називається JSONP, застосовуючи теги <script>. Ви і не зможете передавати дані між доменами будь-яким іншим чином через правила обмеження домену.

У нього є багато переваг:

- Він може асинхронно завантажувати інформацію, щоб ваш сайт був більш чуйним і краще обробляв потік даних.

- Ви також можете використовувати його для подолання междоменной проблем при обміні даними з іншого сайту.

- -JSON більш простий і важить менше, ніж XML.

ВИСНОВОКИ ДО РОЗДІЛУ 2

Були проаналізовані основні front-end фреймворки,карти та їх аналоги.

Дані рішення є найбільш зручними та гнучкими для вирішення цієї задачі.

Середовище розробки вибрано найбільш зручне для вузьконаправленної розробки front-end частини.

РОЗДІЛ 3

Розробка фронт-енду для веб додатку маршрутизації вантажівок

3.1. Розробка макета і прототипу front-end додатку.

UI (User Interface Design) - Дизайн юзер інтерфейсу (UI) зосереджений на тому, що може потрібно зробити користувачам, та забезпеченню того, щоб інтерфейс мав елементи, до яких легко отримати доступ, зрозуміти та використовувати для спрощення цих дій. Юзер інтерфейс об'єднує концепції взаємодії дизайну, візуального дизайну та інформаційної архітектури.

Вибір елементів інтерфейсу

Користувачі ознайомилися з елементами інтерфейсу, що діють певним чином, тому намагайтеся бути послідовними та передбачуваними у виборі та їх макеті. Це допоможе у виконанні завдання, ефективності та задоволенні.

Елементи інтерфейсу включають, але не обмежуючись ними:

Елементи керування введенням: кнопки, текстові поля, прапорці, перемикачі, випадаючі списки, списки, перемикачі, поле дати

Навігаційні компоненти: хлібний малюнок, повзунок, поле пошуку, пагінація, повзунок, теги, піктограми

Інформаційні компоненти: підказки, піктограми, панель виконання, сповіщення, вікна повідомлень, модальні вікна

Тара: гармошка

Бувають випадки, коли для відображення вмісту може бути доречним кілька елементів. Коли це трапляється, важливо враховувати компроміси. Наприклад, іноді елементи, які можуть допомогти вам заощадити простір, покладають більший тягар на користувача розумово, примушуючи його здогадуватися про те, що знаходиться в випадаючому списку або який елемент може бути.

Кращі практики з проектування інтерфейсу

Все походить від знання ваших користувачів, включаючи розуміння їх цілей, навичок, уподобань та тенденцій. Дізнавшись про свого користувача, обов’язково враховуйте наступне при розробці інтерфейсу:

Зберігайте інтерфейс простим. Найкращі інтерфейси майже непомітні для користувача. Вони уникають зайвих елементів і чітко розуміють мову, якою вони користуються на етикетках та у повідомленнях.

Створюйте узгодженість та використовуйте загальні елементи інтерфейсу. Використовуючи загальні елементи у своєму інтерфейсі, користувачі почуваються комфортніше і можуть швидше робити справи. Також важливо створити шаблони мови, верстки та дизайну на всьому сайті, щоб сприяти підвищенню ефективності. Як тільки користувач дізнається, як

щось робити, він повинен мати можливість перенести цю майстерність в інші частини сайту.

Будьте цілеспрямованими в макеті сторінки. Розгляньте просторові зв'язки між елементами на сторінці та структуруйте сторінку на основі важливості. Ретельне розміщення предметів може допомогти привернути увагу до найважливіших частин інформації та допоможе сканувати та читати.

Стратегічно використовувати колір і текстуру. Ви можете спрямовувати увагу на або перенаправляти увагу з предметів, використовуючи колір, світло, контраст і текстуру на свою користь.

Використовуйте типографіку для створення ієрархії та чіткості. Уважно розгляньте, як ви використовуєте гарнітуру. Різні розміри, шрифти та розташування тексту сприяють збільшенню масштабованості, читабельності та читабельності.

Переконайтеся, що система повідомляє, що відбувається. Завжди повідомляйте своїх користувачів про місцезнаходження, дії, зміни стану чи помилки. Використання різних елементів інтерфейсу для повідомлення стану та, якщо потрібно, наступних кроків може зменшити розчарування вашого користувача.

Подумайте про за замовчуванням. Ретельно продумуючи та передбачаючи цілі, які люди приносять на ваш сайт, ви можете створити стандартні налаштування, які зменшують навантаження на користувача. Це стає особливо важливим, коли мова йде про дизайн форми, де у вас може бути можливість попередньо вибрати або заповнити деякі поля.

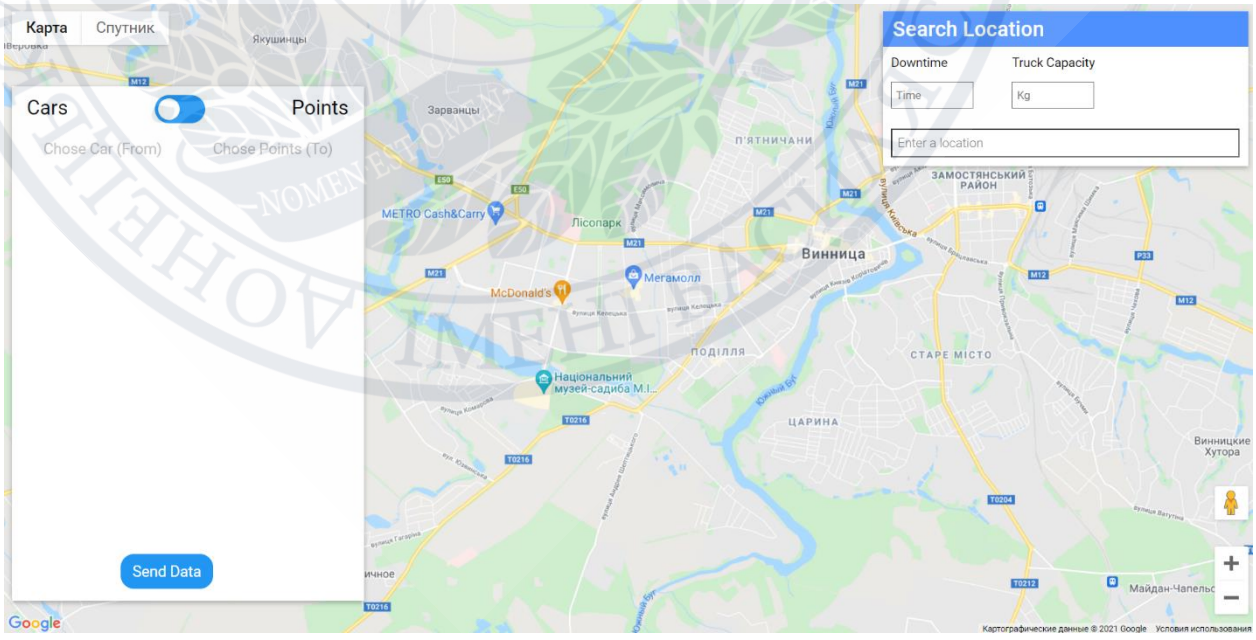


Рисунок 3.1 – Стартовий екран

На даному екрані ми маємо можливість вибрати вантажівку,яку вагу вона буде перевозити,час затримки та вибір локації з якої вона буде їхати.

В лівій частині екрану у нас є перемикач для вибору Вантажівки та Точок по яким вона буде проїжджати.

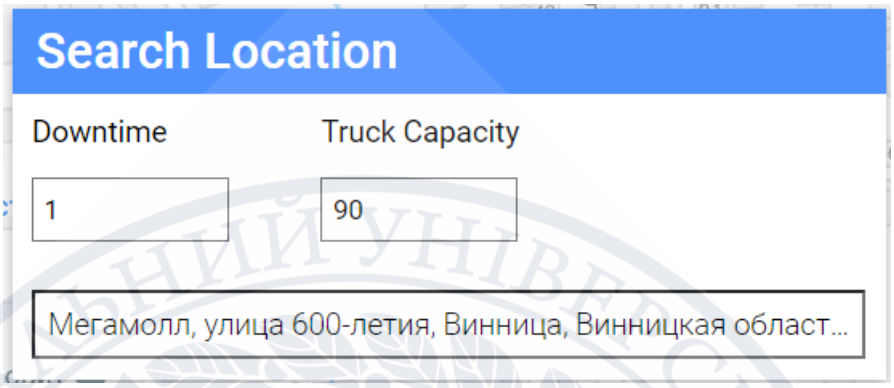


Рисунок 3.2 – Приклад заповнених даних для вантажівок

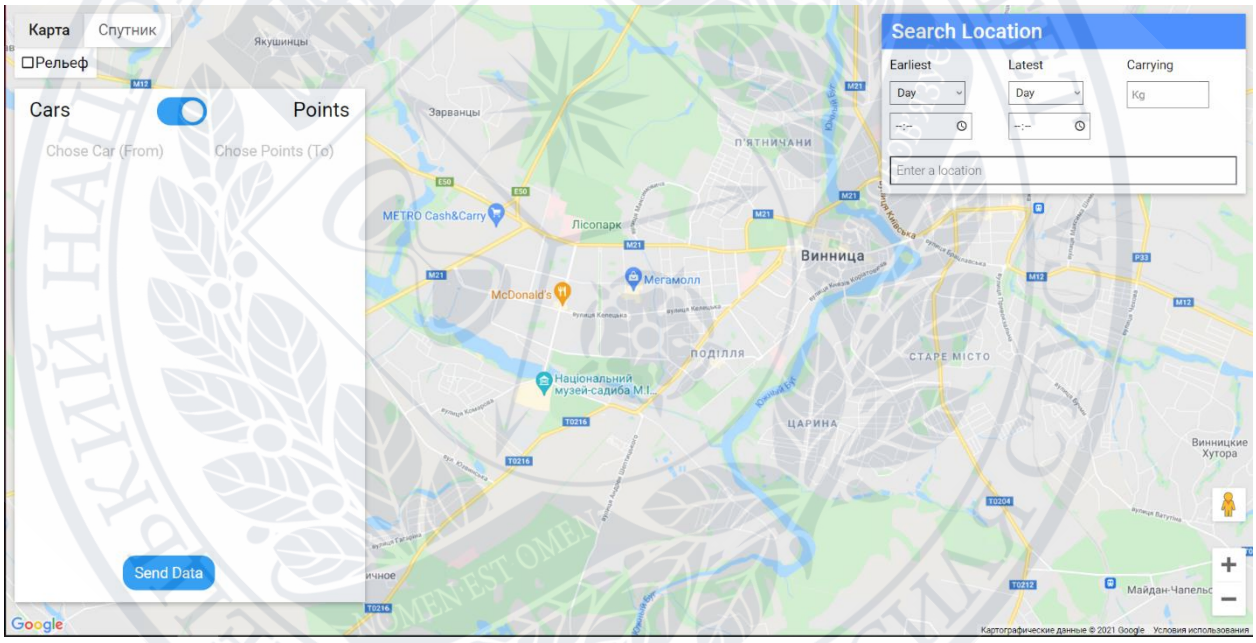


Рисунок 3.3 – Екран додавання точок проїзду вантажівкою

Для точок ми задаємо такі параметри:

-Мінімальна дата доставки

-Максимальна дата доставки

-Кількість товару(у кг)

-Проміжок часу

-Вибір локації на карті(за назвою)

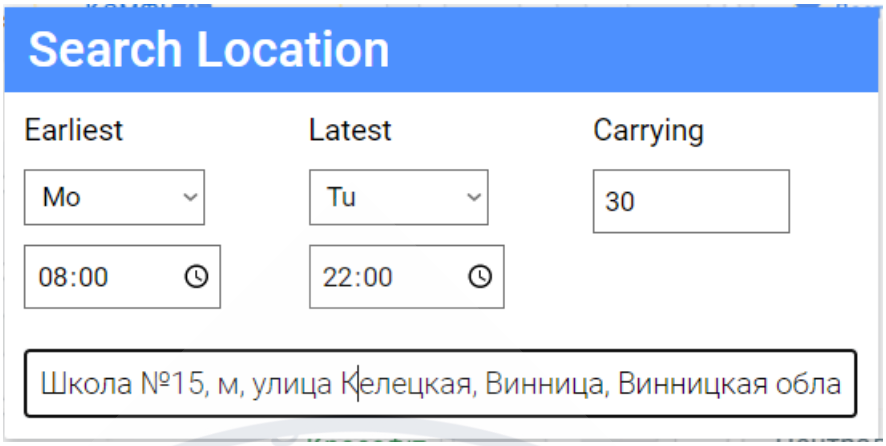
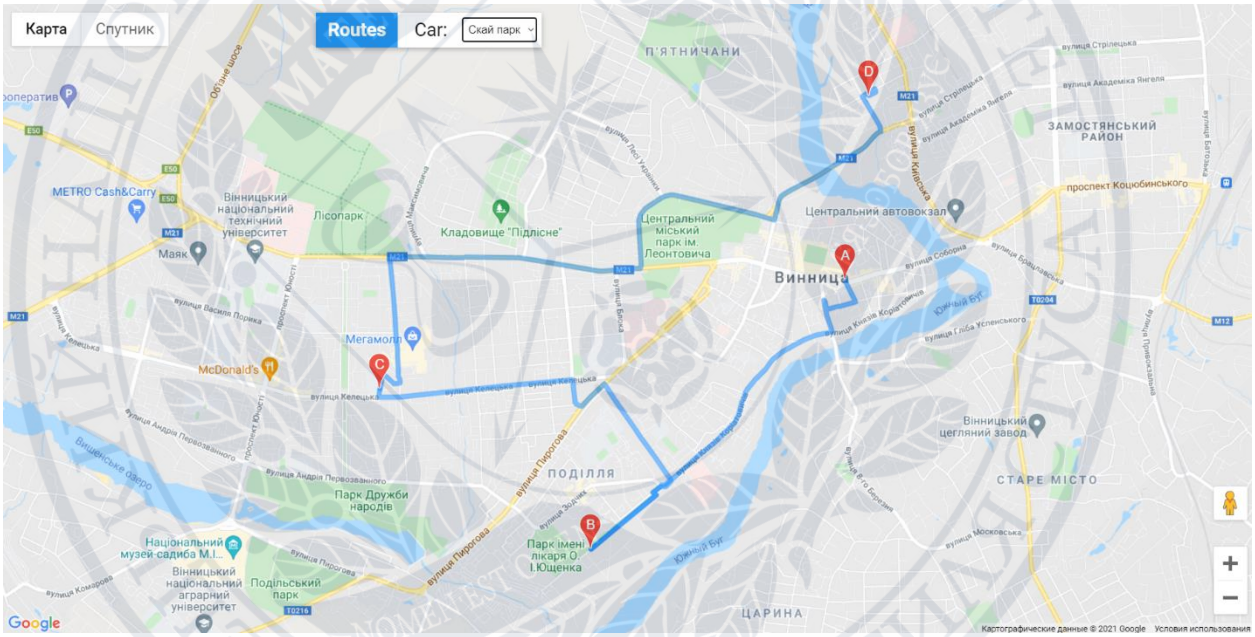


Рисунок 3.4 – Приклад заповнених даних для точки

Коли ми задали дані вантажівок та точок,ми можемо віправляти наші дані до бєк-енду,де вони пройдуть по алгоритму та поверне нам рішення,яке буде відображено на карті.



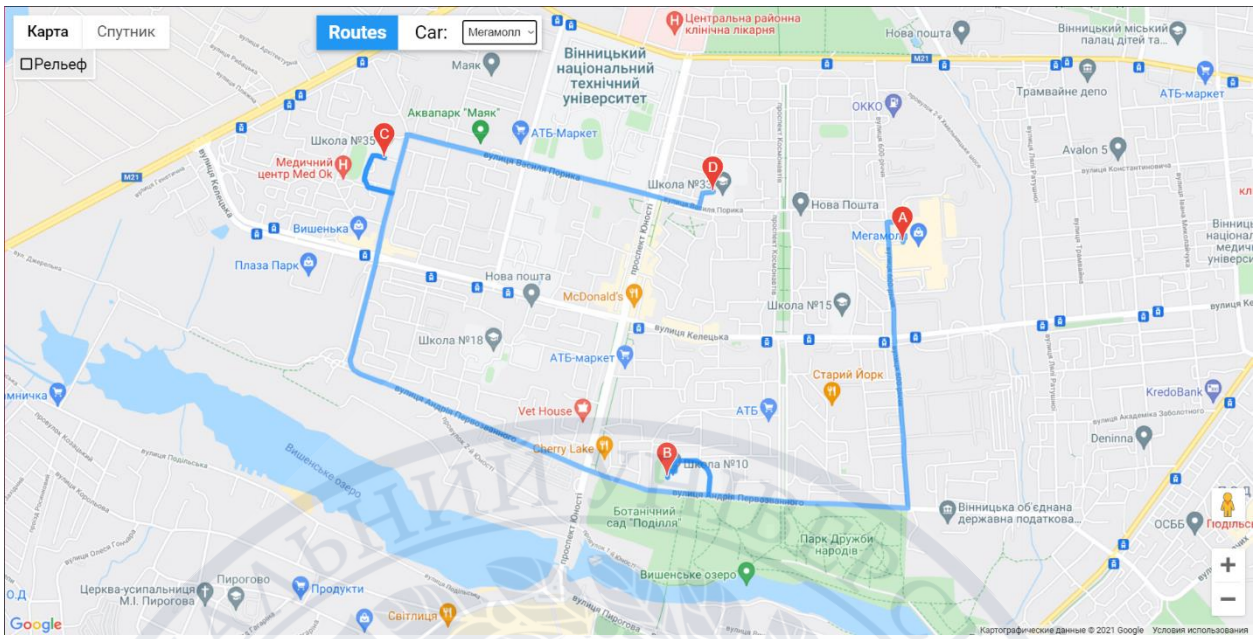
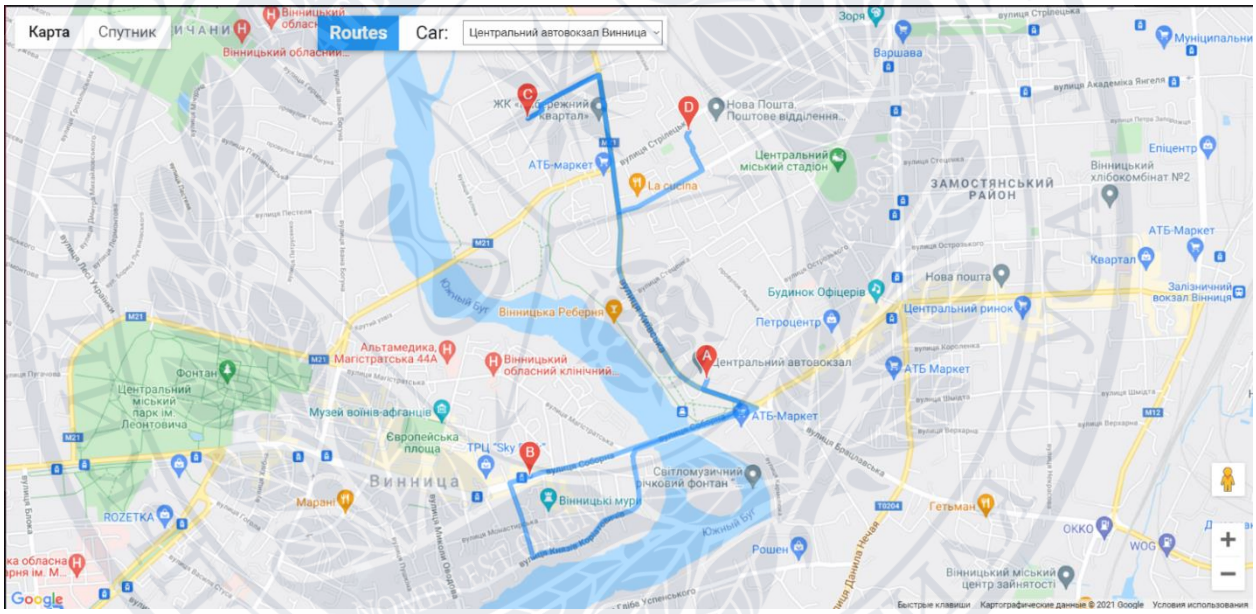


Рисунок 3.5,3.6 – Приклад(а) виведення шляхів для вантажівок



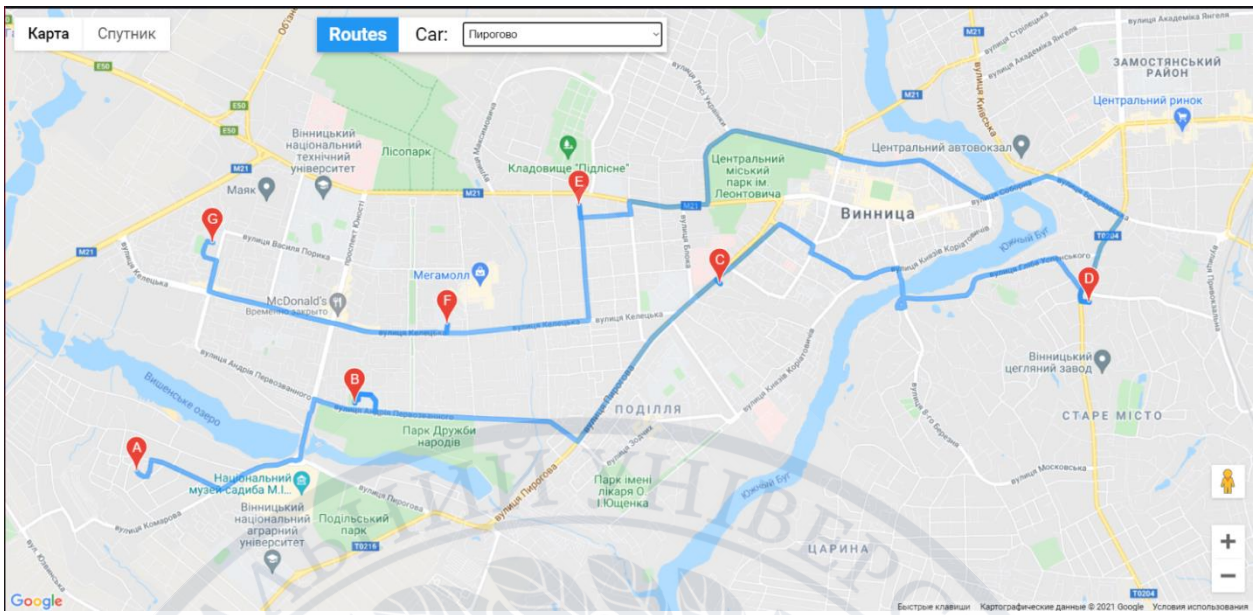


Рисунок 3.7,3.8 – Приклад(b) виведення шляхів для вантажівок

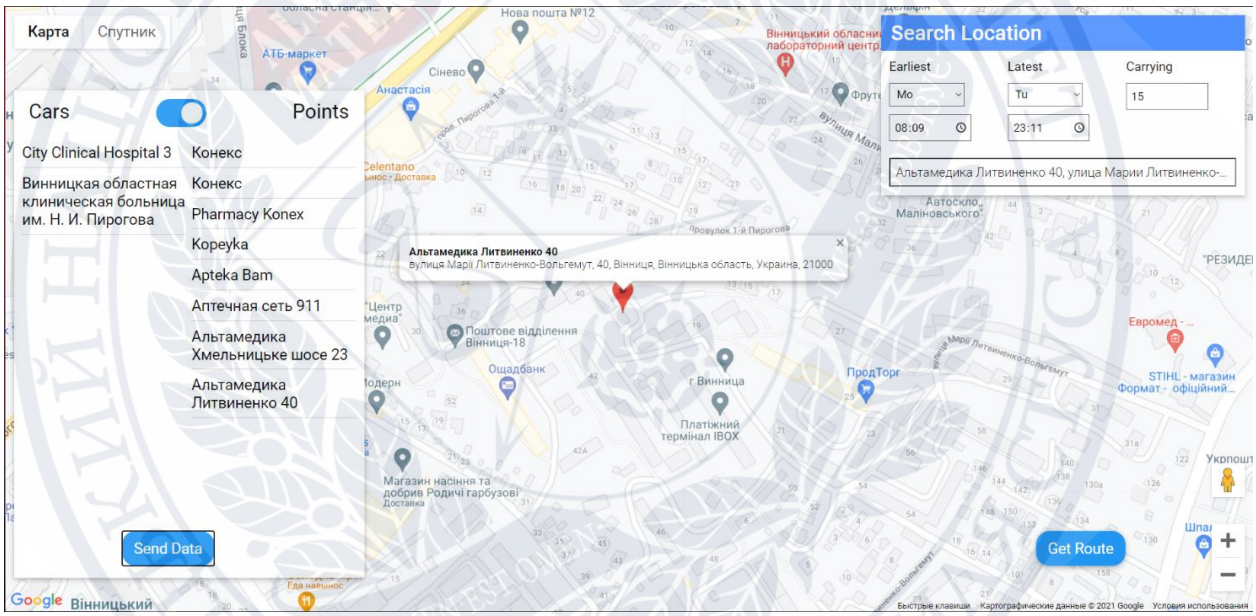


Рисунок 3.9 – Приклад заповнених данних

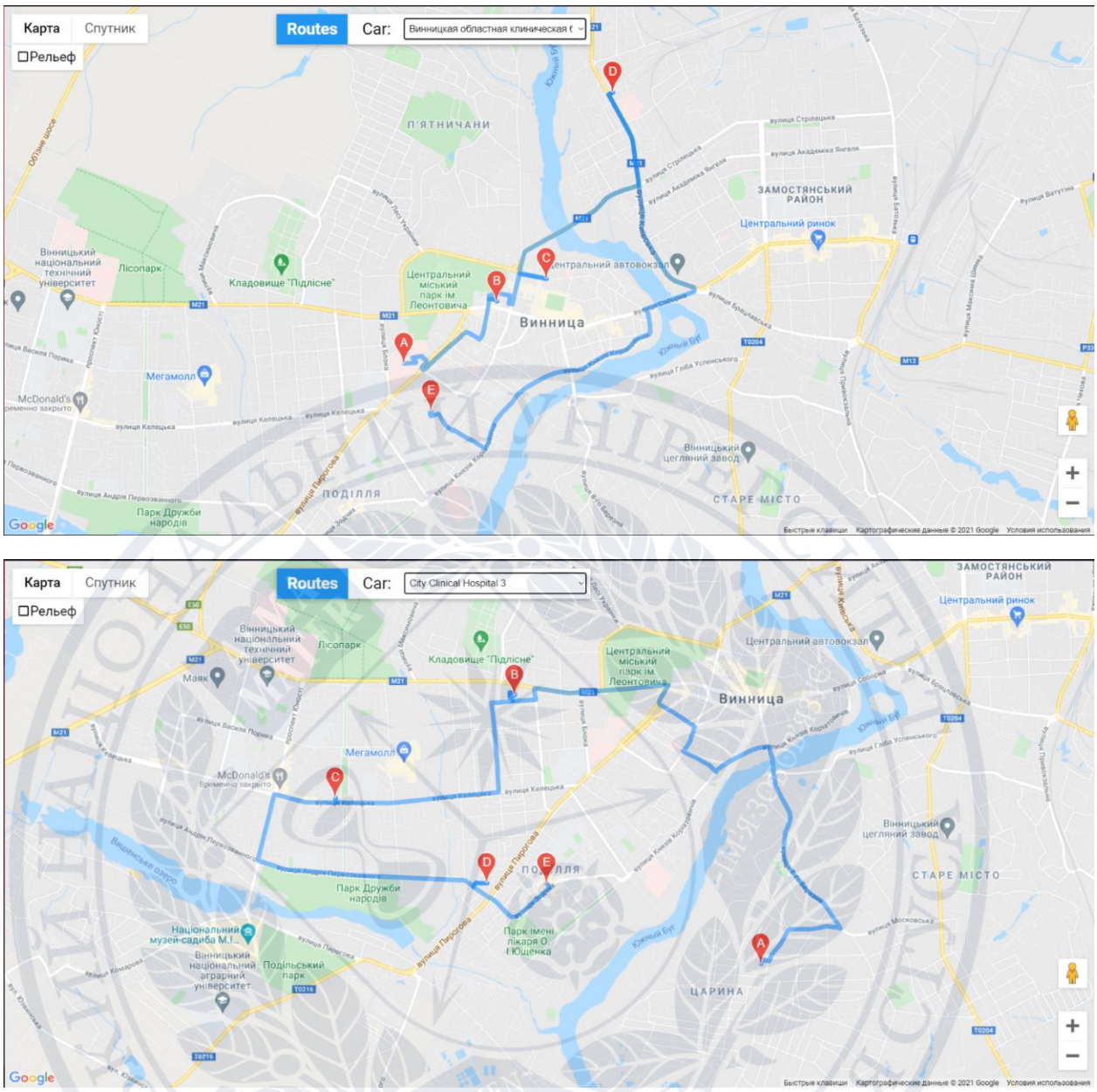


Рисунок 3.10,3.11 – Приклад(с) виведення шляхів для вантажівок

Ми отримуємо шляхи для вантажівок(помітка А – наша вантажівка),інші точки – точки,по яким буде проїжджати вантажівка.

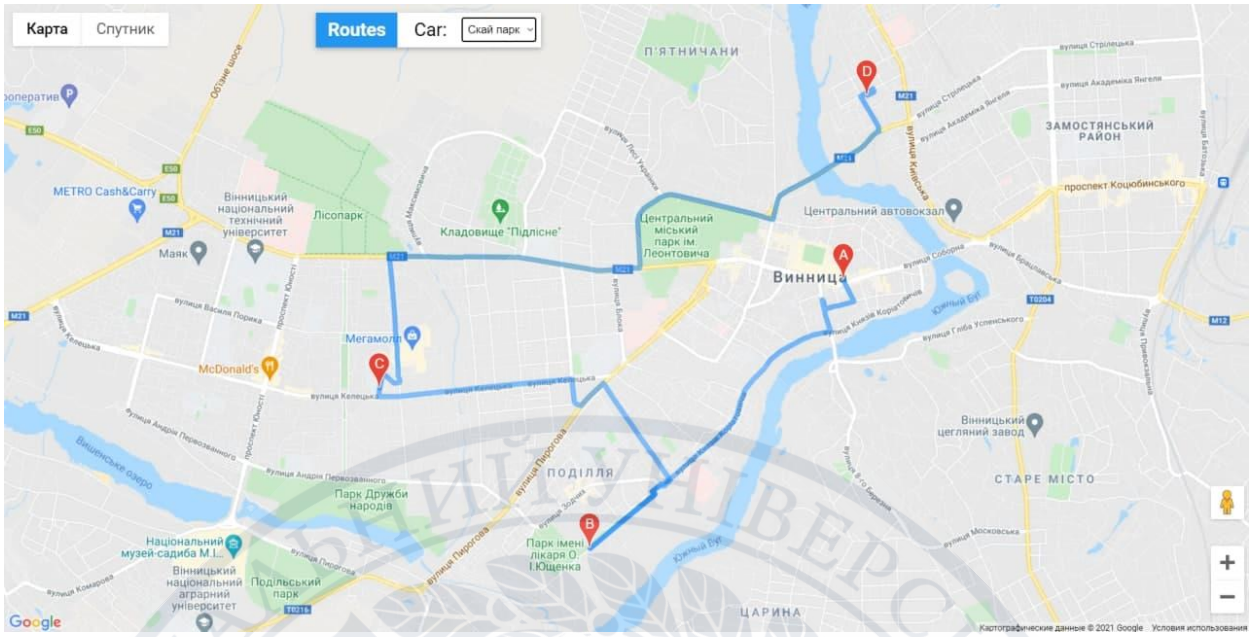


Рисунок 3.12 - Directions API

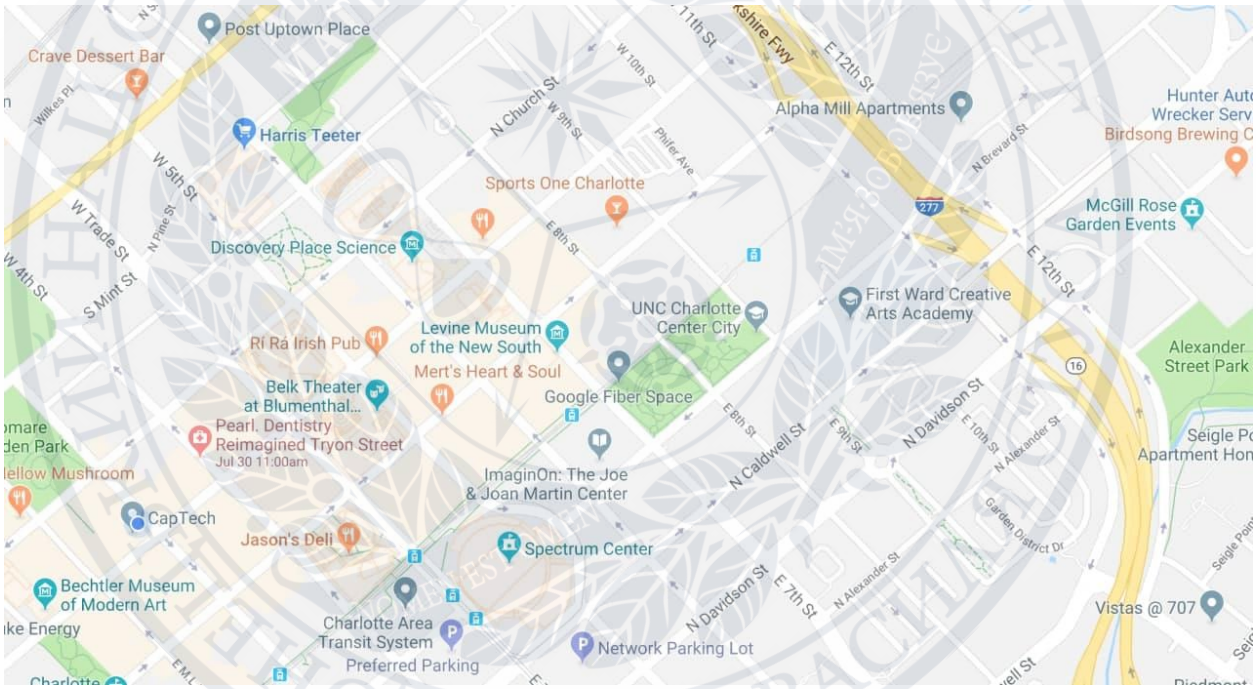


Рисунок 3.13 - Google Map API

3.2.Опис роботи фронт енд частини

1.Підключення Google Map API

Для підключення Google Map API ми використовуємо заготовлену лінку та вставляємо до неї свій сгенерований API ключ.Далі підключаємо її у тезі <script>.

2.Вибираємо точки для вантажівки та точок роз’їзду.

3.Відправляємо точки до бєк-енду.Після додавання усіх точок,ми їх надсилаємо до бєк-енду у форматі JSON.

4.Приймаємо рішення з бєк-енду.На бєк-енді алгоритм пробігається по нашим точкам та параметрам,які ми задавали та знаходиться найкращі маршрути.

5.Перебудовуємо фронт-енд частину.Коли ми отримуємо рішення з бєк-енду,ми перебудовуємо екран на новий,на якому вже відображаємо наше рішення за допомогою Directions API.

ВИСНОВКИ ЗА РОЗДІЛОМ 3

Були пояснені технічні сторони реалізації фронт-енд частини, пояснено як працює та які дані ми відправляємо та отримаємо.Було продемонстровано візуальне рішення для декількох вантажівок.

ВИСНОВОКИ

В результаті виконання цієї роботи було реалізовано фронт-енд частину для даної роботи.

Після цього було розглянуті варіанти карт для поставленої задачі,такі як

Google Maps API та Bing Maps.З цих двох варіантів було вибрано Google Maps API,оскільки він має кращу документація та має велику кількість додаткових сумістних API для виконання цієї задачі.

Після оцінки існуючих рішень була розпочата розробка фронт-енд частини для сервісу рішення VRPTW, яка може масштабуватись до MDVRPTW. Для розробки були використані така бібліотека як React, для якістної структури проекту, Google Maps API, для реалізування карти,Directions API та Places API для можливості побудови шляхів. Розробка була на мові програмування JavaScript. Це все дало змогу швидко написати проект, мінімізувати роботу с файлами, дещо зменшити об’єм потрібного коду та мати можливість розгорнути систему у будь якому середовищі без потреби додаткових адаптацій.

У результаті роботи були розроблені: фронт-енд який вирішує поставлену задачу, запроси для відправлення наших точок до бєк-енду у форматі JSON, валідатор даних та обробник помилок. Веб-додаток був запущений на Firebase Hosting та стабільно працює.

СПИСОК ЛІТЕРАТУРИ

1. Metropolis, N., Rosenbluth, A., Rosenbluth M., Teller, A., Teller, E. Equations of state calculations by fast computing machines. // Chemical Physics, Vol. 21, P.1087-1091, 1983.
2. Davis, A. Handbook of Genetic Algorithms, New York, 1991.
3. Or, I. Traveling salesman type combinatorial optimization problems and their relation to the logistics of blood banking. Evanston, 1976.
4. Bramle, J. Simchi-Levi, D. The Logic of Logistics: theory, algorithms, and applications for logistics management, New York, 1997.
5. L. Schrage. Formulation and structure of more complex/realistic routing and scheduling problems. // Networks, Vol.11, P. 229–232, 1981.
6. Z. Feng. Multi-period vehicle routing problem with recurring dynamic time windows. Конференція Digtial Object Identifier, с. 1–6, 2011.
7. M. L. Fisher. Optimal solution of vehicle routing problems using minimum K-trees. // Operations Research, Vol. 42, P. 626–642, 1994.
8. A. W. J. Kolen, A. H. G. R. Kan, та H. W. J. M. Trienekens. Vehicle routing with time windows. // Operations Research, Vol. 35, номер 2, P. 266–273, 1987.
9. L. Bodin, L. Berman. Routing and Scheduling of School Buses by Computer. // Transportation Science, Vol. 13, P. 113-129, 1979.
10. F. Alonso, M.J. Alvarez, J.E. Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. // Operations Research, Vol. 59, P. 963–976, 2008.
11. R. Ayadi, A.E. Elldrissi, Y. Benadada, El Hilali Alaou. Evolutionary algorithm for a green vehicle routing problem with multiple trips. Міжнародна конференція з питань логістики та управління операціями, с. 148–154, 2014.
12. N. Azi, M. Gendreau, Potvin J-Y. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. // Computers & Operations Research Vol. 41, P.167–173, 2014.

13. N. Bianchessi, G. Righini. Heuristic Algorithms for the Vehicle Routing Problem with Simultaneous Pick-up and Delivery. // Computers & Operations Research, Vol. 34, P. 578–594, 2007.
14. J. Brandão. A New Tabu Search Algorithm for the Vehicle Routing Problem with Backhauls. // European Journal of Operational Research, Vol. 173, P. 540-555, 2006.
15. J.-F. Chen, T.-H. Wu. Vehicle Routing Problem with Simultaneous Deliveries and Pickups. // Journal of the Operational Research Society, Vol. 57, P. 579-587, 2006.
16. T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. // Operational Research, Vol. 60, P. 611-624, 2013.
17. J. Renaud, G. Laporte, F.F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. Comput. // Computers & Operations Research, Vol. 23, P. 229-235, 1996.
18. J. Lysgaard, A.N. Letchford, R.W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. // Mathematical Programming, Vol. 100, P. 423-445, 2004.
19. M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. Журнал Operational Research, Vol. 42, P. 977-978, 1994.
20. M. Desrochers, J. Desrosiers, M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. // Operational Research, Vol. 40, P. 342-354, 1992.
21. J. Brandão, A.Mercer. A tabu search algorithm for the multi-trip vehicle routing problem. // European Journal of Operational Research Vol. 100, P.180–191, 1997.
22. Laporte, G., Gendreau, M. Classical and modern heuristics for the vehicle routing problem. Журнал International Transactions in Operational Research, випуск 7, с. 285–300, 2000.

23. Golden, B., Wasil, E. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. // Computers and Operations Research Vol. 34, P. 2198-2930, 2007.
24. Sariklis, D., Powell, S. A Heuristic Method for the Open Vehicle Routing Problem. // Journal of the Operational Research Society, Vol. 51, P. 564–573, 2000.
25. Brandao, J. A Tabu Search Algorithm for the Open Vehicle Routing Problem. // European Journal of Operational Research, Vol. 157, P. 552–564 2004.
26. A. Archer, M. Bateni, M. Hajiaghayi, H. Karloff. Improved approximation algorithms for prize-collecting steiner tree and tsp. // SIAM Journal on Computing, Vol. 40, P. 309–332, 2011.
27. https://developers.google.com/maps/documentation/directions/quickstart
28. https://developers.google.com/maps/documentation/places/web-service/overview
29. https://developers.google.com/maps/documentation/geocoding/overview
30. https://klondike-studio.ru/wiki/front-end/
31. https://ru.reactjs.org - React
32. https://uk.wikipedia.org/wiki/Firebase - Firebase
33. https://firebase.google.com/docs/hosting - Firebase Hosting
34. https://uk.wikipedia.org/wiki/WebStorm - Webstorm
35. https://developers.google.com/maps?hl=ru – Google Maps
36. https://firebase.google.com - Firebase
37. https://github.com - Github
38. https://www.jetbrains.com/ru-ru/webstorm/ - Webstorm IDE
39. https://wiki.rookee.ru/css/
40. https://itob.ru/solutions/planirovanie-dostavki/
41. https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics
42. https://htmlacademy.ru/courses/basic-html-css
43. https://developer.mozilla.org/ru/docs/Web/HTML

44.https://learn.javascript.ru/json - JSON
45.https://tproger.ru/translations/difference-between-git-and-github/
46.https://sendpulse.com/ru/blog/figma
47.https://tilda.education/articles-figma
48.https://www.geoapify.com/places-api
49.https://learn-reactjs.ru/faq/virtual-dom-and-internals
50.https://habr.com/ru/company/macloud/blog/558682/
51.https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON
52.https://timeweb.com/ru/community/articles/format-json
53.https://webkyrs.info/page/chto-takoe-json

ДОДАТОК А

Код стилів CSS

Loader.css

```
#page_loader {
    display: none;
    position: fixed;
    width: 100%;
    height: 100%;
    text-align: center;
    background-color: rgba(0, 0, 0, 0.2);
    z-index: 40000;
    padding-top: 50vh;
}

.loader_screen {
    position: relative;
    margin: 0 auto;
    width: 100px;
}

.loader_screen:before {
    content: '';
    display: block;
    padding-top: 100%;
}

.circular {
    -webkit-animation: rotate 2s linear infinite;
```

```
    animation: rotate 2s linear infinite;
    height: 100%;
    -webkit-transform-origin: center center;
    -ms-transform-origin: center center;
    transform-origin: center center;
    width: 100%;
    position: absolute;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    margin: auto;
}

.path {
    stroke-dasharray: 1, 200;
    stroke-dashoffset: 0;
    -webkit-animation: dash 1.5s ease-in-out infinite, color 6s ease-in-out infinite;
    animation: dash 1.5s ease-in-out infinite, color 6s ease-in-out infinite;
    stroke-linecap: round;
}

@-webkit-keyframes
rotate {  100% {
    -webkit-transform: rotate(360deg);
    transform: rotate(360deg);
}
```

```
}

@keyframes
rotate {  100% {
   -webkit-transform: rotate(360deg);
   transform: rotate(360deg);
}
}

@-webkit-keyframes
dash {  0% {
   stroke-dasharray: 1, 200;
   stroke-dashoffset: 0;
}
   50% {
      stroke-dasharray: 89, 200;
      stroke-dashoffset: -35;
   }
   100% {
      stroke-dasharray: 89, 200;
      stroke-dashoffset: -124;
   }
}

@keyframes
dash {  0% {
   stroke-dasharray: 1, 200;
```

```
    stroke-dashoffset: 0;
}
    50% {
        stroke-dasharray: 89, 200;
        stroke-dashoffset: -35;
    }
    100% {
        stroke-dasharray: 89, 200;
        stroke-dashoffset: -124;
    }
}

@-webkit-keyframes
color {  100%, 0% {
    stroke: #d62d20;
}
    40% {
        stroke: #0057e7;
    }
    66% {
        stroke: #008744;
    }
    80%, 90% {
        stroke: #ffa700;
    }
}
```

```
@keyframes
color {  100%, 0% {
    stroke: #d62d20;
}
    40% {
        stroke: #0057e7;
    }
    66% {
        stroke: #008744;
    }
    80%, 90% {
        stroke: #ffa700;
    }
}
```

Styles.css

```
#site_body {
    width: 100%;
    height: 100%;
}

#map {
    height: 100%;
}

html, body {
    height: 100%;
```

```
    margin: 0;
    padding: 0;
}

#description {
    font-family: Roboto, sans-serif;
    font-size: 15px;
    font-weight: 300;
}

#infowindow-content .title {
    font-weight: bold;
}

#infowindow-content {
    display: none;
}

#map #infowindow-content {
    width: calc(100% - 400px);
    display: inline;
}

.gm-fullscreen-control {
    display: none;
}
```

```
.pac-card {
    margin: 10px 10px 0 0;
    border-radius: 2px 0 0 2px;
    box-sizing: border-box;
    -moz-box-sizing: border-box;
    outline: none;
    background-color: #fff;
    font-family: Roboto, sans-serif;
}

.pac-controls {
    display: inline-block;
    padding: 5px 11px;
}

.pac-controls label {
    font-family: Roboto, sans-serif;
    font-size: 13px;
    font-weight: 300;
}

#pac-input {
    background-color: #fff;
    font-family: Roboto, sans-serif;
    font-size: 15px;
    font-weight: 300;
    margin-left: 12px;
```

```
    padding: 0 11px 0 13px;
    text-overflow: ellipsis;
    width: 400px;
}

#pac-input:focus {
    border-color: #4d90fe;
}

#title {
    color: #fff;
    background-color: #4d90fe;
    font-size: 25px;
    font-weight: 500;
    padding: 6px 12px;
}

/*switch*/
.switch {
    position: relative;
    display: inline-block;
    width: 60px;
    height: 34px;
}

/* Hide default HTML checkbox */
.switch input {
```

```
    display: none;
}

/* The slider */
.slider {
    position: absolute;
    cursor: pointer;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: #2196F3;
    -webkit-transition: .4s;
    transition: .4s;
}

.slider:before {
    position: absolute;
    content: "";
    height: 26px;
    width: 26px;
    left: 4px;
    bottom: 4px;
    background-color: white;
    -webkit-transition: .4s;
    transition: .4s;
}
```

```
input:checked + .slider:before {
    -webkit-transform: translateX(26px);
    -ms-transform: translateX(26px);
    transform: translateX(26px);
}

p {
    display: block;
    margin-block-start: 0;
    margin-block-end: 0;
    margin-inline-start: 0;
    margin-inline-end: 0;
}

.slider.round {
    border-radius: 34px;
}

.slider.round:before {
    border-radius: 50%;
}

.list {
    width: 197px;
}
```

```
.from_to_lists {
    flex-direction: row;
    display: flex;
    justify-content: space-between;
}

.position_list {
    margin: 100px 0 10px 10px;
    padding: 10px;
    width: 400px;
    height: calc(100% - 160px);
    position: fixed;
    background-color: #ffffff;
    z-index: 1000;
}

.check_switch {
    margin-bottom: 12px;
    padding: 0 8px;
    flex-direction: row;
    display: flex;
    justify-content: space-between;
}

.switch_text_item {
    font-family: Roboto, sans-serif;
    font-size: 24px;
```

```
    font-weight: 400;
    text-align: center;
}

.chose_text {
    padding-top: 8px;
    text-align: center;
    font-family: Roboto, sans-serif;
    font-size: 18px;
    font-weight: 400;
    color: #bfbfbf;
}

#pac-input {
    margin: 10px;
    padding: 6px;
}

.chosen_point {
    margin: 8px 0;
    font-family: Roboto, sans-serif;
    font-size: 18px;
    font-weight: 400;
    padding-bottom: 6px;
    border-bottom: 1px solid #d7d7d7;
}
```

```
.shadow {
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
}

.button_style {
    border: none;
    color: white;
    background-color: #2196F3;
    padding: 10px 14px;
    text-align: center;
    font-family: Roboto, sans-serif;
    font-size: 18px;
    text-decoration: none;
    position: fixed;
    cursor: pointer;
    border-radius: 16px;
    z-index: 2000;
}

.button_style:hover {
    background-color: #4DAAFF;
}

.button_style:active {
    background-color: #73BBFF;
}
```

```
.send_button {
    margin: calc(100vh - 96px) 0 0 140px;
}

.route_button {
    margin: calc(100vh - 96px) 0 0 calc(100vw - 260px);
}

/* 2 window */

#floating-panel {
    position: absolute;
    top: 10px;
    left: 25%;
    z-index: 5;
    background-color: #fff;
    text-align: center;
    font-family: Roboto, sans-serif;
    line-height: 30px;
    vertical-align: text-top;
    display: flex;
    flex-direction: row;
}

.main_select {
    max-width: 250px;
    padding: 4px 2px;
```

```
    margin: 6px;
}

.text_style {
    font-family: Roboto, sans-serif;
    font-size: 20px;
    color: #000;
}

.select_text_style {
    font-family: Roboto, sans-serif;
    font-size: 14px;
    color: #000;
}

option {
    padding: 4px 2px;
}

.route_title {
    font-family: Roboto, sans-serif;
    font-size: 22px;
    font-weight: bold;
    color: #fff;
    background-color: #2196F3;
    padding: 0 6px;
    margin-right: 8px;
```

```
}

.p_option {
    margin: 6px 8px;
}

.b_option {
    margin: 6px;
    font-family: Roboto, sans-serif;
    font-size: 22px;
    color: #000;
    padding: 0 6px;
}

#is_get_route {
    display: none;
}

.input_sup_data {
    width: 90px;
    height: 28px;
    margin: 0;
    padding: 0 0 0 4px;
    font-family: Roboto, sans-serif;
}

.sup_p {
```

```
    font-family: Roboto, sans-serif;
    font-size: 15px;
    font-weight: 400;
}

#carrying_p { grid-area: carrying_p; }
#kg_point { grid-area: kg_point; }
#earliest_p { grid-area: earliest_p; }
#latest_p { grid-area: latest_p; }
#day_latest { grid-area: day_latest; }
#time_earliest { grid-area: time_earliest; }
#time_latest { grid-area: time_latest; }

.point-grid-container {
    display: grid;
    grid-template-areas:
    'earliest_p latest_p carrying_p'
    'day_earliest day_latest kg_point'
    'time_earliest time_latest kg_point';
    grid-gap: 10px;
    padding: 10px;
}

.point-grid-container > div {
    text-align: center;
    padding: 20px 0;
    font-size: 30px;
```

```
}

#downtime_p { grid-area: downtime_p; }
#downtime_input { grid-area: downtime_input; }
#truck_capacity_p { grid-area: truck_capacity_p; }
#truck_capacity_input { grid-area: truck_capacity_input; }
.spacer { grid-area: spacer; }



.car-grid-container {
    display: grid;
    grid-template-areas:
    'downtime_p truck_capacity_p spacer'
    'downtime_input truck_capacity_input spacer';
    grid-gap: 10px;
    padding: 10px;
}

.car-grid-container > div {
    text-align: center;
    padding: 20px 0;
    font-size: 30px;
}
```

ДОДАТОК Б

Код Script файлу

```
let isPointFrom = true
let isStartPage = true

let listPointsFrom = []
let listPointsTo = []

let listPointsFromNames = []
let listPointsToNames = []

let listToRoute = []

let responseObj
let currentFromPointLL

let dimensionValue = 2
let earliest = 1
let latest = 100

let costPerWaitingTime = 1
let vehicleCapacity = 6

function initMap() {
    document.getElementById("page_loader").innerHTML = `
    <div class="loader_screen">
        <svg class="circular" viewBox="25 25 50 50">
```

```
        <circle class="path" cx="50" cy="50" r="20" fill="none" stroke-width="6" stroke-miterlimit="10"/>
    </svg>
</div>`

if (isStartPage) {
    document.getElementById("site_body").innerHTML = `
    <div class="position_list shadow">
        <div class="check_switch">
            <div class="switch_text_item"><p>Cars</p></div>
            <label class="switch">
                <input type="checkbox" id="get_switch" onchange="getSwitchFunc()">
                <span class="slider round"></span>
            </label>
            <div class="switch_text_item"><p>Points</p></div>
        </div>

        <div class="from_to_lists">
            <div class="list" id="from_list">
                <p class="chose_text">Chose Car (From)</p>
            </div>
            <div class="list" id="to_list">
                <p class="chose_text">Chose Points (To)</p>
            </div>
        </div>
    </div>
```

```
<button type="button" class="send_button button_style" onclick="sendData()">Send Data</button>

<button type="button" id="is_get_route" class="route_button button_style shadow" onclick="getRoute()">Get Route</button>


<div class="pac-card shadow" id="pac-card">
  <div id="title">Search Location</div>
  <div id="car_point_data">
    <div class="car-grid-container">
      <p id="downtime_p" class="sup_p">Downtime</p>
      <input id="downtime_input" class="input_sup_data" type="number" placeholder="Time">
      <p id="truck_capacity_p" class="sup_p">Truck Capacity</p>
      <input id="truck_capacity_input" class="input_sup_data" type="number" placeholder="Kg">
      <div class="spacer input_sup_data"></div>
      <div class="spacer input_sup_data"></div>
    </div>
  </div>

  <div id="pac-container">
    <input id="pac-input" type="text" placeholder="Enter a location">
  </div>
</div>


<div id="map"></div>
<div id="infowindow-content">
  <span id="place-name" class="title"></span><br>
  <span id="place-address"></span>
```

```
</div>`

const map = new google.maps.Map(document.getElementById("map"), {
  center: {lat: 49.224313, lng: 28.4266849},
  zoom: 13,
})

const card = document.getElementById("pac-card")
const input = document.getElementById("pac-input")
const biasInputElement = document.getElementById("use-location-bias")
const strictBoundsInputElement = document.getElementById("use-strict-bounds")
const options = {
  componentRestrictions: {country: "ua"},
  fields: ["formatted_address", "geometry", "name"],
  origin: map.getCenter(),
  strictBounds: false,
  types: ["establishment"],
}

map.controls[google.maps.ControlPosition.TOP_RIGHT].push(card)
const autocomplete = new google.maps.places.Autocomplete(input, options)
autocomplete.bindTo("bounds", map)
const infowindow = new google.maps.InfoWindow()
const infowindowContent = document.getElementById("infowindow-content")
infowindow.setContent(infowindowContent)
```

```
        const marker = new google.maps.Marker({
            map,
            anchorPoint: new google.maps.Point(0, -29),
        })

        autocomplete.addListener("place_changed", () => {
            infowindow.close()
            marker.setVisible(false)
            const place = autocomplete.getPlace()

            if (!place.geometry || !place.geometry.location) {
              window.alert("No details available for input: '" + place.name + "'")
              return
            }

            if (place.geometry.viewport) {
              map.fitBounds(place.geometry.viewport)
              //document.getElementById('textbox_id').value

                if (isPointFrom) {
                    costPerWaitingTime =
document.getElementById('downtime_input').value
                    vehicleCapacity =
document.getElementById('truck_capacity_input').value
                    let pointFrom = {
                        "costPerWaitingTime": costPerWaitingTime,
                        "vehicleCapacity": vehicleCapacity,
                        "vehicleStartCoordinateX": place.geometry.location.lng(),
```

```
                "vehicleStartCoordinateY": place.geometry.location.lat(),
                "vehicleType": ((listPointsFrom.length + 1).toString() +
'_veh').toString(),
            }
            let pointFromName = {
                "name": place.name,
                "lng": place.geometry.location.lng(),
                "lat": place.geometry.location.lat(),
                "id": ((listPointsFrom.length + 1).toString() + '_veh').toString(),
            }
            listPointsFrom.push(pointFrom)
            listPointsFromNames.push(pointFromName)
        } else {
            let dayEar = document.getElementById('day_earliest').value
            let timeEar = document.getElementById('time_earliest').value
            let dayLat = document.getElementById('day_latest').value
            let timeLat = document.getElementById('time_latest').value

            let hourEar = timeEar.split(':')[0].replace(/^0/,'')
            let minutesEar = timeEar.split(':')[1].replace(/^0/,'')
            let hourLat = timeLat.split(':')[0].replace(/^0/,'')
            let minutesLat = timeLat.split(':')[1].replace(/^0/,'')

            earliest = (parseInt(dayEar) * 24 + parseInt(hourEar)) * 3600 +
parseInt(minutesEar) * 60
            latest = (parseInt(dayLat) * 24 + parseInt(hourLat)) * 3600 +
parseInt(minutesLat) * 60
            dimensionValue = document.getElementById('kg_point').value
```

```
            if (earliest > latest) {
                let earliestReverse = latest
                let latestReverse = earliest
                earliest = earliestReverse
                latest = latestReverse
            }

            let pointTo = {
                "dimensionValue": dimensionValue,
                "earliest": earliest,
                "latest": latest,
                "locationY": place.geometry.location.lat(),
                "locationX": place.geometry.location.lng(),
                "serviceId": listPointsTo.length + 1,
            }
            let pointToName = {
                "name": place.name,
                "lng": place.geometry.location.lng(),
                "lat": place.geometry.location.lat(),
            }
            listPointsTo.push(pointTo)
            listPointsToNames.push(pointToName)
        }
        if (isPointFrom) {
            document.getElementById('from_list').innerHTML =
listPointsFromNames.map(e =>
                `<div class="chosen_point"><p>${e.name}</p></div>`
```

```
            ).join('')
        } else {
            document.getElementById('to_list').innerHTML =
listPointsToNames.map(e =>
                `<div class="chosen_point"><p>${e.name}</p></div>`
            ).join('')
        }

    } else {
        map.setCenter(place.geometry.location)
        map.setZoom(17)
    }

    marker.setPosition(place.geometry.location)
    marker.setVisible(true)
    infowindowContent.children["place-name"].textContent = place.name
    infowindowContent.children["place-address"].textContent =
place.formatted_address
    infowindow.open(map, marker)
})

function setupClickListener(id, types) {
    const radioButton = document.getElementById(id)
    radioButton.addEventListener("click", () => {
        autocomplete.setTypes(types)
        input.value = ""
    })
}
```

```
setupClickListener("changetype-all", [])
setupClickListener("changetype-address", ["address"])
setupClickListener("changetype-establishment", ["establishment"])
setupClickListener("changetype-geocode", ["geocode"])

biasInputElement.addEventListener("change", () => {
    if (biasInputElement.checked) {
        autocomplete.bindTo("bounds", map)
    } else {
        autocomplete.unbind("bounds")
        autocomplete.setBounds({east: 180, west: -180, north: 90, south: -90})
        strictBoundsInputElement.checked = biasInputElement.checked
    }
    input.value = ""
})

strictBoundsInputElement.addEventListener("change", () => {
    autocomplete.setOptions({
        strictBounds: strictBoundsInputElement.checked,
    })

    if (strictBoundsInputElement.checked) {
        biasInputElement.checked = strictBoundsInputElement.checked
        autocomplete.bindTo("bounds", map)
    }
    input.value = ""
```

```
        })


    } else {
        document.getElementById("site_body").innerHTML = `
        <!-- TODO: Get New Data -->
        <!--<button type="button" id="next_step" class="route_button button_style shadow" onclick="getNewData()">Get New Route</button>-->
        <div id="floating-panel" class="shadow text_style">
            <div class="route_title">
                <p class="p_option">Routes</p>
            </div>
            <p class="b_option">Car:</p>
            <select id="start" class="main_select">
                <option class="select_text_style" id="start_option" disabled selected value>Select car</option>
                    <!-- TODO: For testing -->
<!--                <option class="select_text_style" id="start_option" value="49.21454293572737|28.406946922955882">Національний музей-садиба М.І. Пирогова</option>-->
<!--                <option class="select_text_style" id="start_option" value="49.23681511279994|28.483030049124384">Центральний автовокзал</option>-->
<!-- TODO: Data 100 -->


<!--                <option class="select_text_style" id="start_option" value="49.24619879999999|28.4738159">1 point</option>-->
<!--                <option class="select_text_style" id="start_option" value="49.2268704|28.4192906">2 point</option>-->
```

```
            <!-- value="49.23681511279994|28.483030049124384" -->  <!-- TODO: Data 3 -->
          </select>
        </div>
        <div id="map"></div>`

        let directionsService = new google.maps.DirectionsService
        let directionsDisplay = new google.maps.DirectionsRenderer
        let map = new google.maps.Map(document.getElementById('map'), {
          zoom: 12,
          center: {lat: 49.224313, lng: 28.4266849},
        })
        directionsDisplay.setMap(map)

        let onChangeHandler = function () {
          calculateAndDisplayRoute(directionsService, directionsDisplay)
        }
        document.getElementById('start').addEventListener('change', onChangeHandler)
    }

    function calculateAndDisplayRoute(directionsService, directionsDisplay) {

        let originValLat = parseFloat(document.getElementById('start').value.split('|')[0])
        let originValLng = parseFloat(document.getElementById('start').value.split('|')[1])
```

```
        currentFromPointLL = document.getElementById('start').value.replaceAll('|',
'')
        let toWaypoints = []
        let lastDestination


        for (let i = 0; i < responseObj.length; i++) {
            if ((responseObj[i].lat.toString() + responseObj[i].lng.toString()) ===
currentFromPointLL) {
                if (Array.isArray(responseObj[i].act)) {
                    listToRoute = responseObj[i].act
                } else {
                    listToRoute = [responseObj[i].act]
                }
            }
        }

        console.log('listToRoute length')
        console.log(listToRoute.length)
        for (let i = 0; i < listToRoute.length; i++) {
            console.log('tyts')
            if(i === listToRoute.length - 1) {
                console.log('tyts1')
                // lastDestination = new
google.maps.LatLng(parseFloat(listToRoute[i].lng), parseFloat(listToRoute[i].lat))
// TODO: If invalid lat/lng
                lastDestination = new
google.maps.LatLng(parseFloat(listToRoute[i].lat), parseFloat(listToRoute[i].lng))
// TODO: If valid lat/lng
            } else {
```

```
            console.log('tyt2')
            toWaypoints.push(
                {
                    // location: new google.maps.LatLng(parseFloat(listToRoute[i].lng), parseFloat(listToRoute[i].lat)), // TODO: If invalid lat/lng
                    location: new google.maps.LatLng(parseFloat(listToRoute[i].lat), parseFloat(listToRoute[i].lng)), // TODO: If valid lat/lng
                    stopover: true,
                }
            )
        }
    }
    // toWaypoints.reverse() // TODO: If invalid list

    console.log('TO Waypoints length')
    console.log(toWaypoints.length)

    directionsService.route({
        origin: new google.maps.LatLng(originValLat, originValLng),
        destination: lastDestination,
        // destination: new google.maps.LatLng(49.23358654810193, 28.47908473590271), // TODO: hardcode for testing
        waypoints: toWaypoints,
        optimizeWaypoints: true,
        travelMode: 'DRIVING'
    }, function (response, status) {
        if (status === 'OK') {
            directionsDisplay.setDirections(response)
```

```
        } else {
            window.alert('Directions request failed due to ' + status)
        }
      })
  }

  if (listPointsFromNames.length > 0) {
      document.getElementById('start').innerHTML = listPointsFromNames.map(e
=>
        `<select class="select_text_style main_select" id="start">
          <option class="select_text_style" value='${e.lat + '|' +
e.lng}'>${e.name}</option>
        </select>`
      ).join('')
  }
}

function getSwitchFunc() {
  isPointFrom = !isPointFrom
  if(isPointFrom) {
      document.getElementById("car_point_data").innerHTML = `
      <div class="car-grid-container">
        <p id="downtime_p" class="sup_p">Downtime</p>
        <input id="downtime_input" class="input_sup_data" type="number"
placeholder="Time">
        <p id="truck_capacity_p" class="sup_p">Truck Capacity</p>
        <input id="truck_capacity_input" class="input_sup_data" type="number"
placeholder="Kg">
        <div class="spacer input_sup_data"></div>
```

```
            <div class="spacer input_sup_data"></div>
        </div>`
    } else {
        document.getElementById("car_point_data").innerHTML = `
        <div class="point-grid-container">
            <p id="earliest_p" class="sup_p">Earliest</p>
            <select class="input_sup_data" id="day_earliest">
                <option disabled selected value>Day</option>
                <option value="0">Mo</option>
                <option value="1">Tu</option>
                <option value="2">We</option>
                <option value="3">Th</option>
                <option value="4">Fr</option>
            </select>
            <input type="time" id="time_earliest" class="input_sup_data" placeholder="Earliest">

            <p id="latest_p" class="sup_p">Latest</p>
            <select class="input_sup_data" id="day_latest">
                <option disabled selected value>Day</option>
                <option value="0">Mo</option>
                <option value="1">Tu</option>
                <option value="2">We</option>
                <option value="3">Th</option>
                <option value="4">Fr</option>
            </select>
            <input type="time" class="input_sup_data" id="time_latest" placeholder="Latest">
```

```
            <p id="carrying_p" class="sup_p">Carrying</p>
            <input id="kg_point" class="input_sup_data" type="number" placeholder="Kg">
        </div>`
    }
}

async function sendData() {
    displayStyle( document.getElementById('page_loader'), 'block')

    // TODO: Release

    let data = JSON.stringify({
        "depots": listPointsFrom,
        "services": listPointsTo,
    })

    // TODO: Testing

    let data2 = JSON.stringify({
        "depots": [
            {
                "costPerWaitingTime": 1,
                "vehicleCapacity": 6,
                "vehicleStartCoordinateX": 28.429281324719984,
                "vehicleStartCoordinateY": 49.2120995932243,
                "vehicleType": '1_veh',
```

```
    },
    {
      "costPerWaitingTime": 1,
      "vehicleCapacity": 6,
      "vehicleStartCoordinateX": 28.483030049124384,
      "vehicleStartCoordinateY": 49.23681511279994,
      "vehicleType": '2_veh',
    }
    // {
    //   "costPerWaitingTime": 1,
    //   "vehicleCapacity": 6,
    //   "vehicleStartCoordinateX": 28.47908473590271,
    //   "vehicleStartCoordinateY": 49.23358654810193,
    //   "vehicleType": '2_veh',
    // }
  ],
  "services": [
    {
      "dimensionValue": 2,
      "earliest": 1,
      "latest": 100,
      "locationY": 28.450279592585897,
      "locationX": 49.22767671205461,
      "serviceId": 1,
    },
    {
      "dimensionValue": 2,
```

```
    "earliest": 1,
    "latest": 100,
    "locationY": 28.45362205526641,
    "locationX": 49.23054700073081,
    "serviceId": 2,
  },
  // {
  //   "dimensionValue": 2,
  //   "earliest": 1,
  //   "latest": 100,
  //   "locationY": 28.460303311438892,
  //   "locationX": 49.2346226860734,
  //   "serviceId": 3,
  // },
  {
    "dimensionValue": 2,
    "earliest": 1,
    "latest": 100,
    "locationY": 28.462819868552238,
    "locationX": 49.23322284318994,
    "serviceId": 3,
  },
  {
    "dimensionValue": 2,
    "earliest": 1,
    "latest": 100,
    "locationY": 28.470867566496207,
```

```
            "locationX": 49.23301501794864,
            "serviceId": 4,
        }
    ],
})

let data3 = JSON.stringify({
    "depots":[
        {
            "costPerWaitingTime":"1",
            "vehicleCapacity":"20",
            "vehicleStartCoordinateX":28.4549972,
            "vehicleStartCoordinateY":49.2301139,
            "vehicleType":"1_veh"
        },
        {
            "costPerWaitingTime":"1",
            "vehicleCapacity":"30",
            "vehicleStartCoordinateX":28.4719065,
            "vehicleStartCoordinateY":49.233089,
            "vehicleType":"2_veh"
        }
    ],
    "services":[
        {
            "dimensionValue":"4",
            "earliest":11580,
```

```
    "latest":184440,
    "locationY":49.235103,
    "locationX":28.4724561,
    "serviceId":1
  },
  {
    "dimensionValue":"3",
    "earliest":170220,
    "latest":289200,
    "locationY":49.2338836,
    "locationX":28.41116349999999,
    "serviceId":2
  },
  {
    "dimensionValue":"4",
    "earliest":83820,
    "latest":289200,
    "locationY":49.22215800000001,
    "locationX":28.41002,
    "serviceId":3
  },
  {
    "dimensionValue":"4",
    "earliest":83820,
    "latest":289200,
    "locationY":49.2218065,
    "locationX":28.4101896,
```

```
    "serviceId":4
},
{
    "dimensionValue":"4",
    "earliest":83820,
    "latest":289200,
    "locationY":49.2208899,
    "locationX":28.41113949999999,
    "serviceId":5
},
{
    "dimensionValue":"1",
    "earliest":83820,
    "latest":393960,
    "locationY":49.2194954,
    "locationX":28.4421841,
    "serviceId":6
},
{
    "dimensionValue":"1",
    "earliest":83820,
    "latest":393960,
    "locationY":49.2278767,
    "locationX":28.4464708,
    "serviceId":7
},
{
```

```
    "dimensionValue":"1",
    "earliest":83820,
    "latest":393960,
    "locationY":49.2270754,
    "locationX":28.43953729999999,
    "serviceId":8
  },
  {
    "dimensionValue":"1",
    "earliest":83820,
    "latest":393960,
    "locationY":49.2236446,
    "locationX":28.4472085,
    "serviceId":9
  },
  {
    "dimensionValue":"1",
    "earliest":83820,
    "latest":393960,
    "locationY":49.2278767,
    "locationX":28.4464708,
    "serviceId":10
  },
  {
    "dimensionValue":"1",
    "earliest":83820,
    "latest":393960,
```

```
            "locationY":49.22306810000001,
            "locationX":28.4249608,
            "serviceId":11
        }
    ]
})

let data4 = JSON.stringify({
    "depots":[
        {
            "costPerWaitingTime":1,
            "vehicleCapacity":6,
            "vehicleStartCoordinateX":28.429281324719984,
            "vehicleStartCoordinateY":49.2120995932243,
            "vehicleType":"1_veh"
        },
        {
            "costPerWaitingTime":1,
            "vehicleCapacity":6,
            "vehicleStartCoordinateX":28.483030049124384,
            "vehicleStartCoordinateY":49.23681511279994,
            "vehicleType":"2_veh"
        }
    ],
    "services":[
        {
            "dimensionValue":2,
```

```
    "earliest":1,
    "latest":100,
    "locationY":28.450279592585897,
    "locationX":49.22767671205461,
    "serviceId":1
},
{
    "dimensionValue":2,
    "earliest":1,
    "latest":100,
    "locationY":28.45362205526641,
    "locationX":49.23054700073081,
    "serviceId":2
},
{
    "dimensionValue":2,
    "earliest":1,
    "latest":100,
    "locationY":28.462819868552238,
    "locationX":49.23322284318994,
    "serviceId":3
},
{
    "dimensionValue":2,
    "earliest":1,
    "latest":100,
    "locationY":28.470867566496207,
```

```
            "locationX":49.23301501794864,
            "serviceId":4
        }
    ]
})

const data100 = JSON.stringify({
    "depots":[
        {
            "costPerWaitingTime":"1",
            "vehicleCapacity":"20",
            "vehicleStartCoordinateX":28.4738159,
            "vehicleStartCoordinateY":49.24619879999999,
            "vehicleType":"1_veh"
        },
        {
            "costPerWaitingTime":"1",
            "vehicleCapacity":"20",
            "vehicleStartCoordinateX":28.4192906,
            "vehicleStartCoordinateY":49.2268704,
            "vehicleType":"2_veh"
        }
    ],
    "services":[
        {
            "dimensionValue":"4",
            "earliest":99540,
```

```
    "latest":228900,
    "locationY":49.22804639999999,
    "locationX":28.4272842,
    "serviceId":1
  },
  {
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.21955209999999,
    "locationX":28.4274278,
    "serviceId":2
  },
  {
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.224313,
    "locationX":28.4266849,
    "serviceId":3
  },
  {
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.21592219999999,
    "locationX":28.4341809,
```

```
    "serviceId":4
},
{
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.233089,
    "locationX":28.4719065,
    "serviceId":5
},
{
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.2301139,
    "locationX":28.4549972,
    "serviceId":6
},
{
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.23239299999999,
    "locationX":28.470819,
    "serviceId":7
},
{
```

```
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.235103,
    "locationX":28.4724561,
    "serviceId":8
  },
  {
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.2338836,
    "locationX":28.41116349999999,
    "serviceId":9
  },
  {
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
    "locationY":49.2303168,
    "locationX":28.3977048,
    "serviceId":10
  },
  {
    "dimensionValue":"3",
    "earliest":99540,
    "latest":228900,
```

```
    "locationY":49.2290906,
    "locationX":28.4605769,
    "serviceId":11
  },
  {
    "dimensionValue":"4",
    "earliest":99540,
    "latest":228900,
    "locationY":49.2334299,
    "locationX":28.46757209999999,
    "serviceId":12
  },
  {
    "dimensionValue":"4",
    "earliest":99540,
    "latest":228900,
    "locationY":49.2333371,
    "locationX":28.4677577,
    "serviceId":13
  },
  {
    "dimensionValue":"4",
    "earliest":99540,
    "latest":228900,
    "locationY":49.23385519999999,
    "locationX":28.4687926,
    "serviceId":14
```

```
        }
    ]
})

const xhr = new XMLHttpRequest()
xhr.withCredentials = false

xhr.addEventListener("readystatechange", function () {
    if (this.readyState === this.DONE) {
        console.log('DONE')
        console.log('Response Text')
        console.log(this.responseText)
        let parseJSON = JSON.parse(this.responseText)
        responseObj = parseJSON.problem.solutions.solution[0].routes.route
        console.log('Parse JSON')
        console.log(parseJSON.problem.solutions.solution[0].routes.route)
        console.log('---------------------------------')
    } else {
        console.log('!DONE')
        console.log(this.responseText)
        console.log('---------------------------------')
    }
})

xhr.open("POST", "https://vrp-solution.herokuapp.com/solve")
xhr.setRequestHeader("Content-Type", "application/json")
xhr.setRequestHeader("Accept", "application/json")
```

```
    xhr.send(data)

    console.log('My data')
    console.log(data)
    console.log('----------------------------------')

    await getRouteBut()
}

// TODO: Get New Data
/*
async function getNewData() {
    displayStyle(document.getElementById('page_loader'), 'block')

    console.log('LOOOOOOOOOOOOOOL')
    console.log(listToRoute.length)

    let updatedOneCarPoints

    for (let i = 0; i < responseObj.length; i++) {
        if ((responseObj[i].lat.toString() + responseObj[i].lng.toString()) ===
currentFromPointLL) {
            currentFromPointLL = listToRoute[0].lat.toString() + '|' +
listToRoute[0].lng.toString();
            updatedOneCarPoints = {
                "costPerWaitingTime": 1,
                "vehicleCapacity": 6,
```

```
            "vehicleStartCoordinateX": parseFloat(responseObj[i].lng),
            "vehicleStartCoordinateY": parseFloat(responseObj[i].lat),
            "vehicleType": ((i + 1).toString() + '_veh').toString(),
        }
        listToRoute.shift()
        // responseObj[i].act = listToRoute
    }
}

console.log('LOOOOOOOOOOOOOL1')
console.log(listToRoute.length)

let updatedData = JSON.stringify({
    "depots": [updatedOneCarPoints],
    "services": listToRoute,
})

const xhr = new XMLHttpRequest()
xhr.withCredentials = false

xhr.addEventListener("readystatechange", function () {
    if (this.readyState === this.DONE) {
        console.log('DONE')
        console.log('Response Text')
        console.log(this.responseText)
        let parseJSON = JSON.parse(this.responseText)
        responseObj = parseJSON.problem.solutions.solution[0].routes.route
```

```
            console.log('Parse JSON')
            console.log(parseJSON.problem.solutions.solution[0].routes.route)
            console.log('---------------------------------')
        } else {
            console.log('!DONE')
            console.log(this.responseText)
            console.log('---------------------------------')
        }
    })

    xhr.open("POST", "https://vrp-solution.herokuapp.com/solve")
    xhr.setRequestHeader("Content-Type", "application/json")
    xhr.setRequestHeader("Accept", "application/json")

    xhr.send(updatedData)

    console.log('My updated Data')
    console.log(updatedData)
    console.log('---------------------------------')

    initMap()
    await getRouteBut()
}
*/
function getRoute() {
    isStartPage = !isStartPage
    initMap()
```

```
}

async function getRouteBut() {
    return new Promise(() =>
        setTimeout(() => {
            displayStyle( document.getElementById('page_loader'), 'none')
            displayStyle( document.getElementById('is_get_route'), 'block')
        }, 1500)
    )
}

function displayStyle(node, style) {
    node.style.display = style
}
```

ДОДАТОК В

Код HTML

```
<!DOCTYPE html>
<html lang="ua">
<head>
   <title>Map App</title>
   <link rel="stylesheet" href="css/styles.css">
   <link rel="stylesheet" href="css/loader.css">
   <link rel="shortcut icon"
href="https://developers.google.com/maps/images/maps-icon.svg">
   <meta name="http-equiv" content="Content-type: text/html; charset=UTF-8">
   <script src="https://polyfill.io/v3/polyfill.min.js?features=default"></script>
   <script type="text/javascript" src="js/script.js" charset="utf-8"></script>
</head>
<body>
   <div id="page_loader"></div>
   <div id="site_body"></div>
   <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBIwzALxUPNbatRBj3
Xi1Uhp0fFzwWNBkE&callback=initMap&libraries=places&v=weekly"
async></script>
</body>
</html>
```