Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

ДИПЛОМНА РОБОТА
МАГІСТРА

на тему *Інформаційна система доставки онлайн замовлень в мережі закладів швидкого харчування*

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

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

Виконав: студент *2* курсу, група *КНм-19-1* _____ *Є.А. Ткачук*
Підпис Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КНІТ* _____ *Р.О. Багрій*
Підпис Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* _____ *Р.О. Багрій*
Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор _____ *О.В. Бармак*
Підпис Ініціали, прізвище

7 *12* 2020 р.

Хмельницький 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*
Кафедра *комп'ютерних наук та інформаційних технологій*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

_____
(підпис)
д.т.н., професор О.В. Бармак
«7» 12 *2020* року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ МАГІСТРА

1. Тема дипломної роботи магістра: *«Інформаційна система доставки онлайн замовлень в мережі закладів швидкого харчування»*

2. Завдання видано студенту *Ткачуку Євгенію Андрійовичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *к.т.н., доцент Багрій Руслан Олександрович*
(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «9» 9 *2020* р. № 22

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

Мета роботи – створення інформаційної системи, для оптимізації доставки замовлень мережі закладів швидкого харчування, яка сприятиме зменшенню кількості витрачених ресурсів та часу на доставку. Для досягнення мети необхідно дослідити існуючі підходи розв'язку подібних задач, проаналізувати їх та створити відповідну інформаційну систему.

Реферат

Дипломна робота магістра присвячена розробці інформаційної системи для оптимізації доставки замовлень в мережі закладів швидкого харчування.

Актуальність теми. З популяризацією та розповсюдженням мережі інтернет, багато сервісів мігрувало в мережу. Все більше користувачів надають перевагу різноманітним Інтернет сервісам в мережі, взамін звичним стаціонарним точкам. Інтернет сервіси різного роду діяльності набувають свою популярність з кожним днем через свою зручність та простоту використання.

Безліч власників різноманітних сервісів вчасно підхватили тенденцію і відкрили свої віртуальні магазини в мережі, що дозволило зайняти досить прибуткову нішу, яка отримала популярність ще до пандемії, не винятком стали і різноманітні заклади харчування.

Доставка замовлень стала дуже актуальна в період пандемії COVID-19. ВООЗ рекомендувало впровадження карантину, самоізоляції та наполегливо не рекомендувало залишати житло навіть для покупки продуктів, щоб мінімізують контакти між людьми та знизити навантаження на систему охорони здоров’я. З введенням карантинних заходів всі заклади харчування мусили припинити прийняття клієнтів та закритись на період карантину, що спричинило не аби які збитки. Єдиним способом продовження роботи закладу, була наявність доставки замовлень клієнту додому.

Запропонована інформаційна система дозволяє оптимізувати маршрути доставки замовлень в мережах закладів швидкого харчування.

Кінцевою метою застосування розробленої інформаційної системи оптимізації маршрутів доставки замовлень, є використання її в мережах закладів швидкого харчування, що забезпечило би економію ресурсів та часу на доставку замовлень та позитивно вплинуло б на економічне становище закладів. Вхідними даними інформаційної системи є замовлення закладу які оброблюються системою, а на виході отримується оптимізований маршрут доставки замовлень.

Мета і задачі роботи. Метою магістерської роботи є розробка інформаційної системи оптимізації маршрутів доставки онлайн замовлень в мережі закладів швидкого харчування, яка б дала можливість скоротити час доставки замовлень під час підвищеного попиту в зв'язку з пандемією.

Для досягнення поставленої мети визначені наступні задачі:

– дослідити методи оптимізації доставки замовлень;

– дослідити способи отримання відстаней між точками доставки та навігацію по ним;

– вдосконалити метод оптимізації доставки з врахування динамічної зміни точок доставки;

– спроектувати та розробити інформаційну систему оптимізації маршрутів доставки.

Програмна реалізація інформаційної системи також повинна забезпечити функціональні вимоги для здійснення та обробки онлайн-замовлень.

Об'єкт дослідження – оптимізація маршрутів доставки.

Предмет дослідження – інформаційна система оптимізації маршрутів доставки, методи розв'язку задачі комівояжера.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані такі результати:

1. Досліджено практичну ефективність методів розв'язку задачі комівояжера, що дало можливість оптимізувати маршрути доставки замовлень;

2. Розроблено інформаційну систему оптимізації маршрутів доставки замовлень в мережі закладів швидкого харчування, що дозволило скоротити час доставки замовлень під час підвищеного попиту в зв'язку з пандемією.

Практичне значення одержаних результатів. На основі розробленої інформаційної системи оптимізації маршрутів було створено автоматизовану систему з можливістю здійснення замовлень та оптимізацією маршрутів подальшої доставки.

При розробці системи використовувалось картографічне API сервісу Google Maps.

Апробація результатів дипломної роботи магістра та публікації.:

Доповідь на тему «Методи оптимізації доставки замовлень» на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет).

За темою дипломної роботи магістра автором виконана *наукова публікація* [27].

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 28 найменувань. Загальний обсяг дипломної роботи магістра становить 97 сторінки, з них 76 сторінок основного тексту та 17 сторінок додатків. У роботі наведено 37 рисунків та 15 таблиць.

Ключові слова: методи оптимізація, оптимальний маршрут, задача комівояжера.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ОС	Операційна система
ПЗ	Програмне забезпечення
ІС	Інформаційна система
СКБД	Система керування базами даних

Вступ

Дипломна робота магістра присвячена розробці інформаційної системи для оптимізації доставки замовлень в мережі закладів швидкого харчування.

Актуальність теми. З популяризацією та розповсюдженням мережі інтернет, багато сервісів мігрувало в мережу. Все більше користувачів надають перевагу різноманітним Інтернет сервісам в мережі, взамін звичним стаціонарним точкам. Інтернет сервіси різного роду діяльності набувають свою популярність з кожним днем через свою зручність та простоту використання.

Безліч власників різноманітних сервісів вчасно підхватили тенденцію і відкрили свої віртуальні магазини в мережі, що дозволило зайняти досить прибуткову нішу, яка отримала популярність ще до пандемії, не винятком стали і різноманітні заклади харчування.

Доставка замовлень стала дуже актуальна в період пандемії COVID-19. ВООЗ рекомендувало впровадження карантину, самоізоляції та наполегливо не рекомендувало залишати житло навіть для покупки продуктів, щоб мінімізують контакти між людьми та знизити навантаження на систему охорони здоров’я. З введенням карантинних заходів всі заклади харчування мусили припинити прийняття клієнтів та закритись на період карантину, що спричинило не аби які збитки. Єдиним способом продовження роботи закладу, була наявність доставки замовлень клієнту додому.

Запропонована інформаційна система дозволяє оптимізувати маршрути доставки замовлень в мережах закладів швидкого харчування.

Кінцевою метою застосування розробленої інформаційної системи оптимізації маршрутів доставки замовлень, є використання її в мережах закладів швидкого харчування, що забезпечило би економію ресурсів та часу на доставку замовлень та позитивно вплинуло б на економічне становище закладів. Вхідними даними інформаційної системи є замовлення закладу які оброблюються системою, а на виході отримується оптимізований маршрут доставки замовлень.

Мета і задачі роботи. Метою магістерської роботи є розробка інформаційної системи оптимізації маршрутів доставки онлайн замовлень в мережі закладів швидкого харчування, яка б дала можливість скоротити час доставки замовлень під час підвищеного попиту в зв’язку з пандемією.

Для досягнення поставленої мети визначені наступні задачі:

– дослідити методи оптимізації доставки замовлень;

– дослідити способи отримання відстаней між точками доставки та навігацію по ним;

– вдосконалити метод оптимізації доставки з врахування динамічної зміни точок доставки;

– спроектувати та розробити інформаційну систему оптимізації маршрутів доставки.

Програмна реалізація інформаційної системи також повинна забезпечити функціональні вимоги для здійснення та обробки онлайн-замовлень.

Об’єкт дослідження – оптимізація маршрутів доставки.

Предмет дослідження – інформаційна система оптимізації маршрутів доставки, методи розв’язку задачі комівояжера.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані такі результати:

3. Досліджено практичну ефективність методів розв’язку задачі комівояжера, що дало можливість оптимізувати маршрути доставки замовлень;

4. Розроблено інформаційну систему оптимізації маршрутів доставки замовлень в мережі закладів швидкого харчування, що дозволило скоротити час доставки замовлень під час підвищеного попиту в зв’язку з пандемією.

Практичне значення одержаних результатів. На основі розробленої інформаційної системи оптимізації маршрутів було створено автоматизовану систему з можливістю здійснення замовлень та оптимізацією маршрутів подальшої доставки.

При розробці системи використовувалось картографічне API сервісу Google Maps.

Апробація результатів дипломної роботи магістра та публікації.:

Доповідь на тему «Методи оптимізації доставки замовлень» на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет).

За темою дипломної роботи магістра автором виконана *наукова публікація* [27].

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 28 найменувань. Загальний обсяг дипломної роботи магістра становить 97 сторінки, з них 76 сторінок основного тексту та 17 сторінок додатків. У роботі наведено 37 рисунків та 15 таблиць.

Ключові слова: методи оптимізація, оптимальний маршрут, задача комівояжера.

Розділ 1

Аналіз сучасного стану використання інформаційних систем онлайн замовлень мереж закладів швидкого харчування

1.1 Аналіз предметної області

З популяризацією та розповсюдженням мережі інтернет, багато сервісів мігрувало в мережу. Все більше користувачів надають перевагу різноманітним Інтернет сервісам в мережі, взамін звичним стаціонарним точкам. Інтернет сервіси різного роду діяльності набувають свою популярність з кожним днем через свою зручність та простоту використання.

Безліч власників різноманітних сервісів вчасно підхватили тенденцію і відкрили свої віртуальні магазини в мережі, що дозволило зайняти досить прибуткову нішу яка отримала популярність ще до пандемії, не винятком стали і різноманітні заклади харчування.

Значних темпів розвитку область отримала в період пандемії COVID-19 – інфекційна хвороба, яка вперше виявлена у людини в грудні 2019 року в місті Ухань, Центральний Китай. Хвороба почалася як спалах, що розвинулась у пандемію [1].

ВООЗ рекомендувало впровадження карантину, самоізоляції та наполегливо не рекомендувало залишати житло навіть для покупки продуктів, щоб мінімізують контакти між людьми та знизити навантаження на систему охорони здоров’я. З введенням карантинних заходів всі заклади харчування мусили припинити прийняття клієнтів та закритись на період карантину, що спричинило не аби які збитки.

Умовою продовження роботи стала наявність доставки замовлення покупцю до дому, що дало змогу людям поласувати стравами з своїх улюблених закладів під час карантину.

Аналізуючи способи отримання послуг в закладах харчування, можна виділити три основні напрямки:

Безпосередня присутність людини в закладі для отримання послуги що стало не можливим в період карантину.

Замовлення в закладі в телефонному режимі, але такий спосіб не є оптимальним тому що часто в популярних закладах лінія зайнята через великий потік замовлень.

Автономна система оформлення замовлень, часто вебзастосунок, де можна оформити замовлення на бажанні страви без черг та телефонних дзвінків і з доставкою в зазначене місце.

Аналізуючи інформаційні системи (ІС) онлайн замовлень в мережі, можна зробити висновок, що найрозповсюдженішою системою - ІС онлайн замовлень реалізована у вигляді веб-додатку.

1.2 Аналіз видів вебдодатків

Щоб дослідити види вебдодатків, потрібно спочатку дослідити що собою являє сам вебдодаток.

Вебдодаток або вебзастосунок – розподілений застосунок, в якому клієнтом виступає браузер, а сервером – вебсервер [2].

Логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому вебзастосунки є міжплатформовими сервісами [2], тобто сервісами які працюють на різних системах.

Найрозповсюдженими типами мобільних вебзастосунків: вебсайт з адаптивним дизайном та мобільний додаток, розглянемо їх детальніше.

Вебсайт з адаптивним дизайном

Вебсайт – це сукупність логічно зв'язаної гіпертекстової інформації, оформленої у вигляді окремих сторінок і доступної в мережі Інтернет.

Подібне визначення веб-сайту було правильним на початку існування Інтернету, коли Мережа і веб-сайти використовувалися в основному як розважальна система.

До кінця 90-х років веб-сайти дійсно були в основному статичними сторінками. Для створення веб-сайту було потрібне лише знання мови гіпертекстової розмітки HTML. Якщо ж сторінка надавала якісь програмні засоби це були виключно засоби, що міг надати сервер, на якому розташований веб-сайт.

Про зручність і красу тогочасних веб-сайтів взагалі особливо не доводилося говорити. Час спливає, розвиваються мови програмування, розширюються канали передачі інформації... Зараз Інтернет вже є самодостатньою галуззю економіки, а веб-сайти стали повноправними представництвами фірм в Інтернеті [3].

Зараз існує величезна кількість пристроїв які можуть відвідувати вебсайти починаючи з стаціонарних комп’ютерів і закінчуючи смартфонами, але для коректної роботи на різних пристроях з різною роздільною здатністю дисплея потрібен відповідний дизайн та розширення тут на допомогу і приходить адаптивний дизайн.

Адаптивний дизайн – це такий підхід в дизайні, що «адаптується» до поведінки користувача та його системи грунтуючись на розмірі екрану, платформи, орієнтації екрану і т.д. Такий підхід поєднує в собі суміш різних сіток і верств, зображень і правильного використання CSS.

Наприклад, якщо користувач заходить на сайт з iPad замість звичного лептопа, сайт повинен «підлаштуватися» під розмір екрану, зменшити розміри зображень і, скажімо, прибрати флеш елементи(рисунок 1.1) [4].

Рисунок 1.1 – Приклад адаптивного дизайну на різних пристроях [4].

Але адаптивний дизайн це не тільки «підстроювання» під розмір екрану. Це, все-таки, новий підхід у самій розробці дизайну. Давайте детальніше зупинимося на ключових елементах адаптивного дизайну.

На даний момент існує безліч розмірів екранів і їх орієнтації книжкової або альбомної. Мало того, тепер мобільні пристрої можуть змінювати орієнтацію сторінки від книжкової до альбомної при повороті. А серед власників великих моніторів поширеним є «вільний» розмір, тобто не на повний екран, а як кожному зручно, тобто фактично розмір браузера може бути яким завгодно. Адаптивний дизайн розробляється таким чином, щоб вірно та красиво виглядати при будь-якому розмірі екрану.

Також не забуваймо за тачскріни які зараз досягли піку популярності. Вони дуже популярні на мобільних пристроях не тільки невеликого розміру – багато нетбуків і лептопів, а навіть і декстопні комп’ютери підтримують сенсорний ввід.

Хороший адаптивний дизайн забезпечує приємний для користувача вигляд контенту та правильну функціональність на будь яких пристроях.

Мобільні додатки

Мобільний додаток це програмне забезпечення (ПЗ) , призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних застосунків встановлені на самому пристрої або можуть бути завантажені на нього з онлайн магазинів мобільних застосунків, таких як App Store, Google Play, Windows Phone Store та інших, безкоштовно або за плату [5].

Спочатку мобільні застосунки використовувалися для швидкої перевірки електронної пошти, але їх високий попит призвів до розширення їх призначень і в інших областях, таких як ігри для мобільних телефонів, GPS, спілкування, перегляд відео та користування Інтернетом [5].

В наш час двома найпопулярнішими операційними системами для мобільних додатків є Android та IOS. Операційна система (ОС) – це базовий комплекс програм, що виконує керування апаратною складовою комп'ютера або віртуальної машини; забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем (рисунок 1.2) [6].

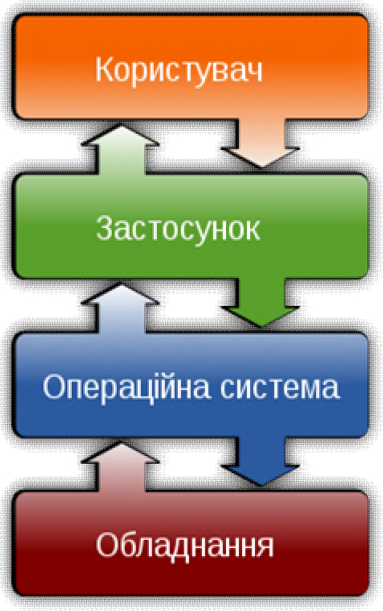

Рисунок 1.2 – Принцип роботи ОС

Проаналізуймо двох лідерів ринку в сфері мобільних ОС під системи яких розробники виготовляють свої додатки.

Android – це операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux [7]. Наразі найпопулярніша мобільна ОС в світі зі своєю галереєю додатків Google Play (рисунок 1.3) [5].

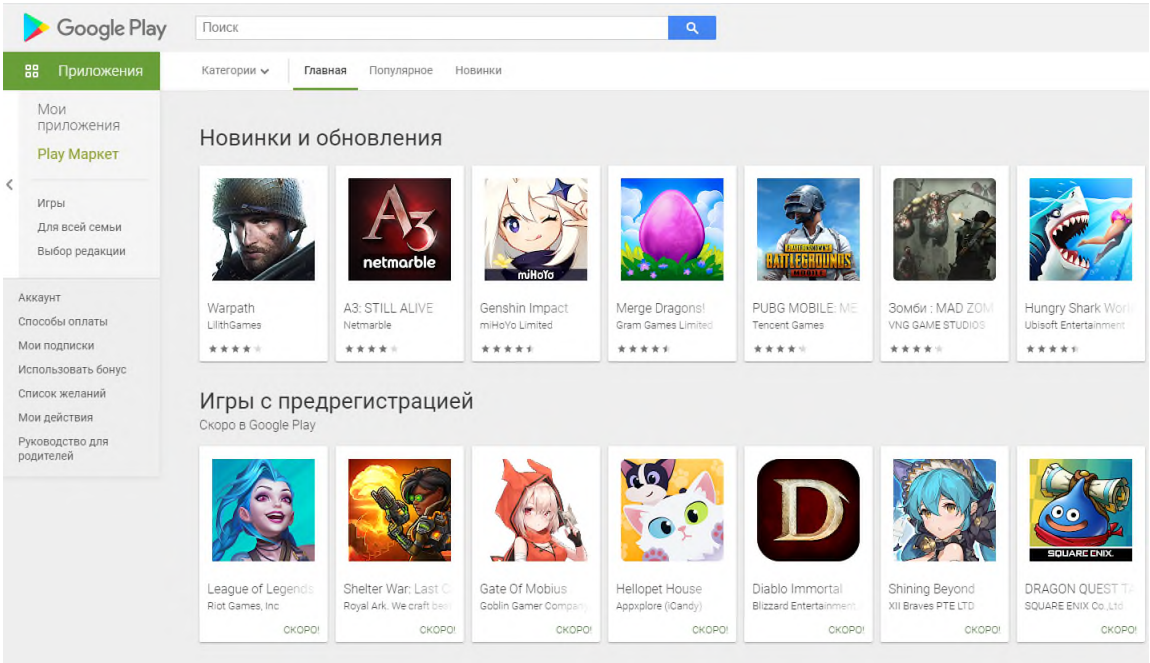

Рисунок 1.3 – Галерея додатків Google Play

Використовується як ОС на безлічі різних мобільних телефонів зручна та легка в використанні, але важка в розробці тому що у мобільних пристроях в рази менші можливості в порівнянні з стаціонарними комп’ютерами.

Мобільні додатки мають ураховувати різноманіття розмірів дисплею робити правки для оптимізації на менш потужні процесори та інші більш слабкі в порівнянні з стаціонарними машинами компоненти, а ще основні зміни в ПЗ треба робити на всіх актуальних версія ОС.

IOS – це мобільна операційна система від Apple розроблена спочатку спеціально iPhone, а потім впроваджена на більшість пристроїв компаннії. На відміну від Android використовується виключно на пристроях компанії Apple. Також має свою галерею додатків під назвою Apple App Stope (рисунок 1.4).

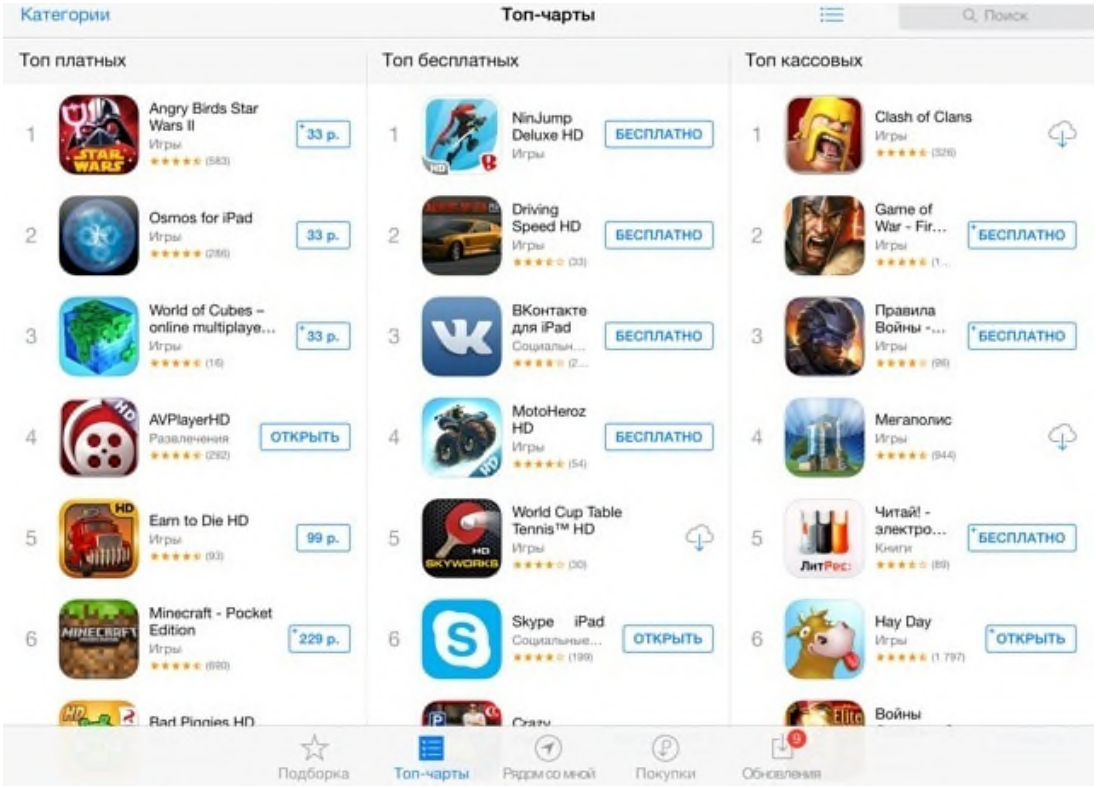

Рисунок 1.4 – Галерея додатків App Store

Але ця ОС використовується тільки однією компанією робить життя розробникам трішки простіше через відсутність такого різноманіття пристроїв як в ситуації з Android.

1.3 Аналіз існуючих сервісів доставки онлайн замовлень

Провівши огляд існуючих сервісів з доставки онлайн замовлень в Україні, можемо виділити декілька з них:

Zakaz.ua – ця компанія працює у восьми містах країни і привозить замовлення з декількох супермаркетів і є одною з популярних сервісів доставки в Україні (рисунок 1.5).

Замовлення оформлюються як за допомогою сайту, так і через мобільний додатоки Android та IOS. Простий та зручний процес вибору товарів дає змогу швидко оформити замовлення.

З мінусів велика завантаженість сервісу та як наслідок довга доставка яка займає день і більше.

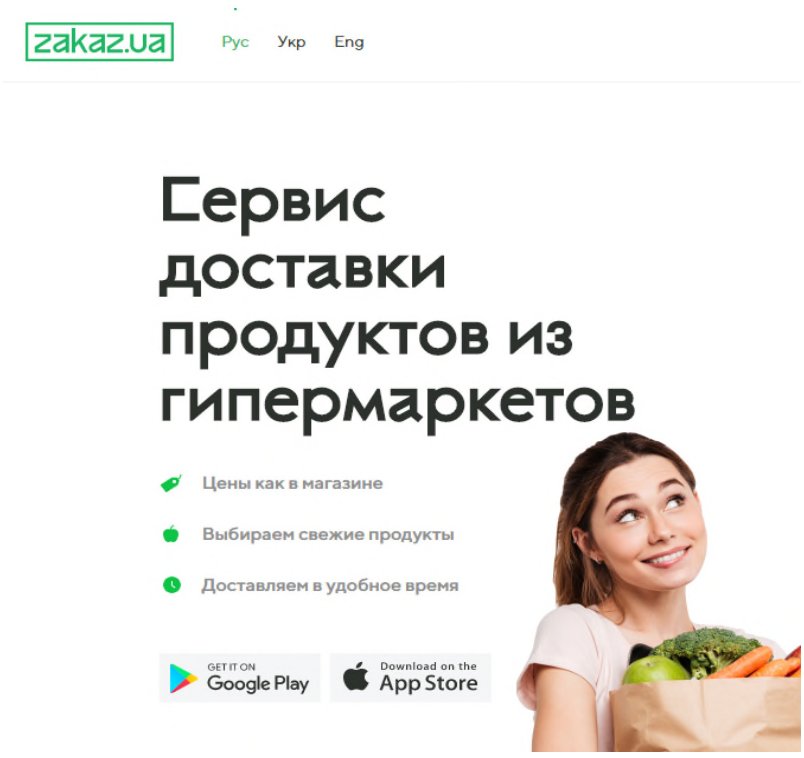

Рисунок 1.5 – Головна сторінка сайту Zakaz.ua

Fozzyshop – це торгова група яка об'єднує такі супермаркети Fozzy, Сільпо, Фора та Trash (рисунок 1.6).

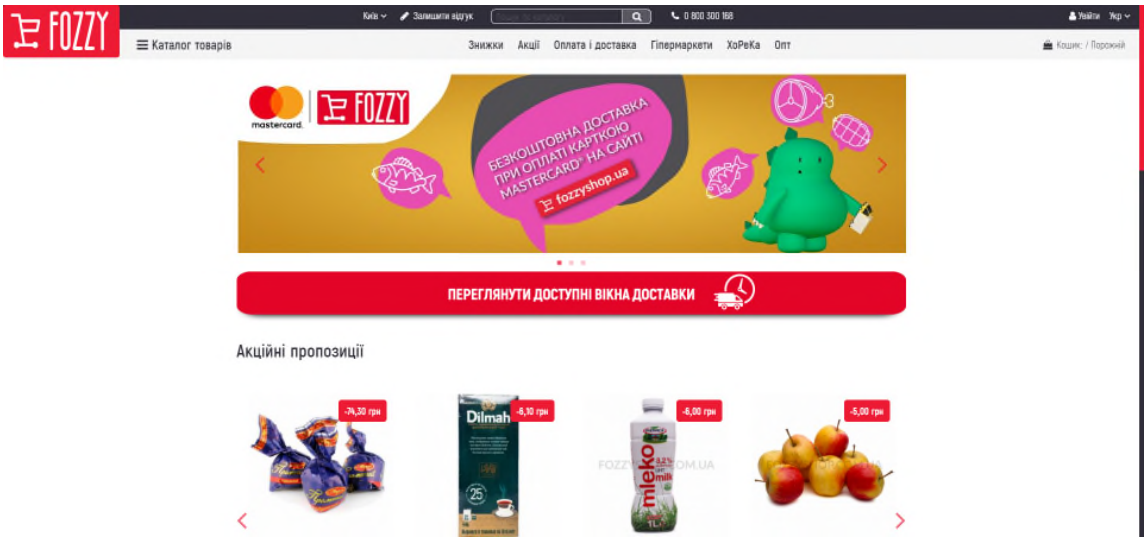

Рисунок 1.6 – Головна сторінка вебсайту Fozzyshop

Але доставку можна замовити тільки в перших трьох. На сайті можна зробити покупки, є послуги з експрес доставки та безкоштовної від певної суми покупок, також наявний варіант самовивозу.

Під час пандемії сервіс набрав обертів що збільшило затримки в доставці і кількість нових замовлень на сервісі зростає з кожним днем що не покращує ситуацію.

Мережа Glovo – найбільш направлений сервіс саме на доставку страв з ресторанів та кафе про який в період пандемії дізналась велика кількість Українців.

Наразі Glovo(рисунок 1.7) працює у 17 українських містах. Glovo зручний у використанні, є можливість робити замовлення з ПК на сайті та за допомогою мобільних додатків.

Компанія також запустила послугу безконтактної доставки, що вкрай важливо в умовах карантину.

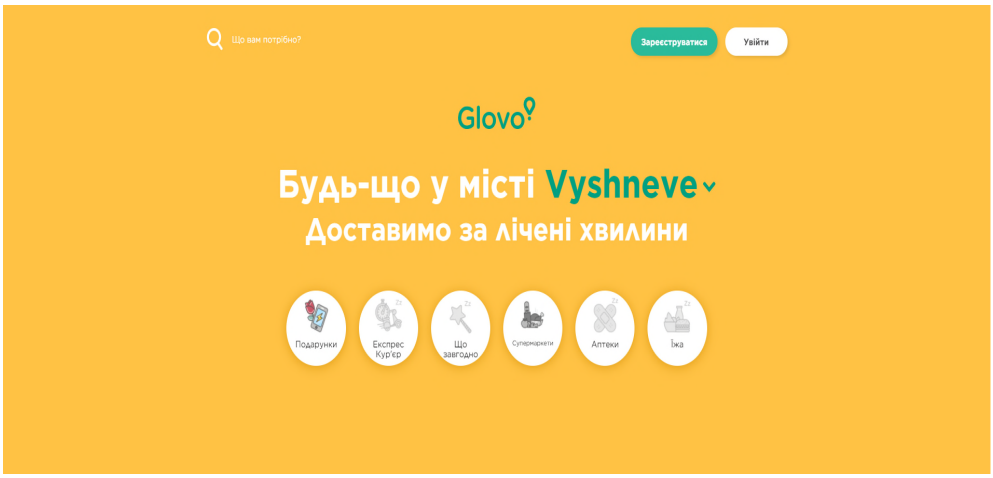

Рисунок 1.7 – Головна сторінка вебсайту Glovo

Modesto – сервіс позиціонує себе як доставка своїх страв. На даний момент Modesto(рисунок 1.8) невеликий сервіс, що надає послуги доставки, проте порівняно невеликий потік покупців робить замовлення «час в час» реальнішим.

Рисунок 1.8 – Головна сторінка вебсайту Modesto

1.4 Аналіз методів оптимізації доставки замовлень

Аналізуючи способи оптимізації доставки, не можливо оминути одну з найвідоміших задач комбінаторної оптимізації, а саме «задачу комівояжера». Дана задача полягає в знаходженні найбільш вигідного маршруту, що проходить через всі точки, та повертається на стартову позицію. Завдання комівояжера як лінійного програмування є найбільш вивченої на сьогоднішній день та має велику актуальність для компаній, які постачають свої товари в різні міста[20].

Існує безліч різних методів вирішення задачі комівояжера[21]:

1. Точні методи
   - метод повного перебору;
   - метод гілок і меж.
2. Евристичні методи
   - метод найближчого сусіда;
   - метод найдешевшого включення;
   - метод включення найближчого міста;
   - метод мінімального кістяка дерева;
   - метод генетичних алгоритмів
   - метод мурашиних колоній.

Відомі методи розв'язання поділяють на дві групи, що можна комбінувати. Точні методи знаходять, маючи достатньо часу, гарантовано оптимальний шлях. Евристичні методи знаходять, часто за коротший час, гарні розв'язки, що, в загальному випадку, можуть бути гіршими за оптимальні [21].

1.5 Постановка задачі

Метою магістерської роботи є розробка інформаційної системи оптимізації маршрутів доставки онлайн замовлень в мережі закладів швидкого

харчування, яка б дала можливість скоротити час доставки замовлень під час підвищеного попиту в зв’язку з пандемією.

Для досягнення поставленої мети визначені наступні задачі:

– дослідити методи оптимізації доставки замовлень;

– дослідити способи отримання відстаней між точками доставки та навігацію по ним;

– вдосконалити метод оптимізації доставки з врахування динамічної зміни точок доставки;

– спроектувати та розробити інформаційну систему оптимізації маршрутів доставки.

Програмна реалізація інформаційної системи також повинна забезпечити функціональні вимоги для здійснення та обробки онлайн-замовлень.

Висновки до розділу 1.

В першому розділі розглянута проблема оптимізації доставки замовлень в мережах закладів швидкого харчування в період пандемії.

Проведений аналіз видів вебдодатків та розглянуті їх види. В результаті аналізу було проведено дослідження існуючих систем доставки для аналізу програмних рішень та виявлення недоліків існуючих систем.

Розглянуті методи оптимізації маршрутів для аналізу рішення подібних проблем.

Отже, вирішено спроектувати та розробити інформаційну систему оптимізації маршрутів доставки.

Розділ 2

Розробка інформаційної системи для оптимізації доставки замовлень в мережі закладів швидкого харчування.

2.1 Загальний опис інформаційної системи оптимізації доставки замовлень.

Аналізуючи існуючі сервіси з доставки онлайн замовлень в більшості з зростанням навантаження на сервіс з'являється проблема перевищення граничного часу доставки замовлень. Час доставки доставки замовлень важливий в закладах швидкого харчування, так як температурний режим замовлень та товарний вигляд з часом псується, що може спричинити відмову від них, що в свою чергу тягне за собою економічні збитки для закладу. Існуючі сервіси з доставки замовлень розглянуті в розділі 1.3, доставляють страви швидкого харчування протягом години з моменту їх готовності. Також слід зазначити що для передачі замовлення кур'єром також знадобиться час, приблизно до 5 хвилин, виходячи з цього маємо наступні обмеження для ІС у вигляді часу до однієї години та у вигляді кількості замовлень 4-5 за годину: близько 10 хвилин між пунктами та до 5 хвилин на передачу, що для п'яти замовлень становитиме 1-1.15 години.

Тому потрібно розробити інформаційну систему з оптимізацією маршруту доставки для закладів швидкого харчування, для вирішення цього недоліку. Запропонована інформаційна система, з використанням сучасних зовнішніх систем визначення відстані, дозволить реалізувати в собі оптимізацію маршрутів доставки замовлень використовуючи методи рішення задачі комівояжера.

Інформаційна система складається з наступних кроків (рисунок 2.1):

1. Вхідна інформація: список поточних замовлень закладу.
2. Підготовка замовлень для обробки сервісом визначення відстані.

3. Визначення часу проїзду до точки доставки за допомогою сервісу визначення відстані та включення її в маршрут.

4. Оптимізація маршруту методом розв’язку задачі комівояжера.

5. Вихідна інформація: оптимізований маршрут доставки.

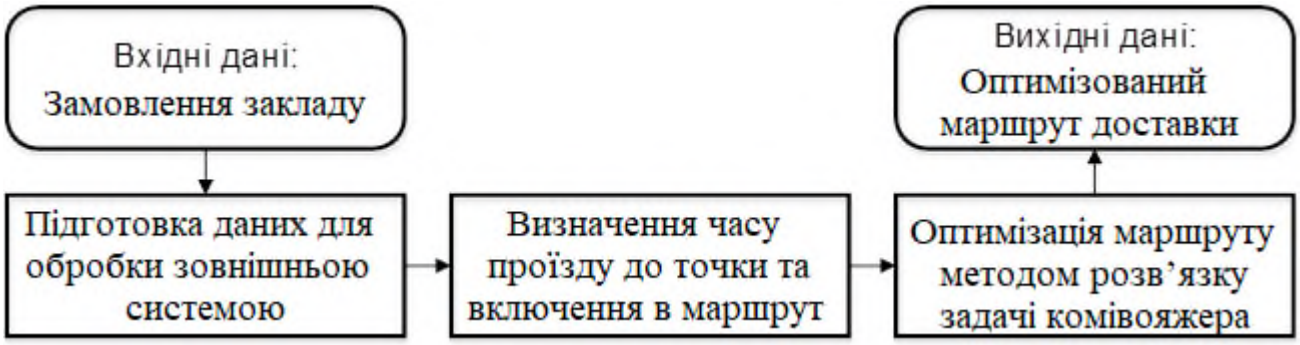

Рисунок 2.1 – Схема інформаційної системи оптимізації доставки замовлень

На рисунку 2.1 наведені основні кроки, що передбачають оптимізацію маршруту шляхом методу розв’язку задачі комівояжера. Задача комівояжера має достатню кількість методів для розв’язку тому потребує дослідження для вибору оптимального методу.

2.2 Постановка задачі комівояжера

Задача комівояжера полягає у знаходженні найоптимальнішого маршруту доставки, де комівояжер (бродячий торговець, а в нашому випадку кур’єр), повинен відправитись з стартової точки, відвідати по разу в невідомому порядку пункти доставки і повернутися в стартову точку.

Відстані між усіма точками доставки – відомі. У якому порядку слід доставити замовлення, щоб замкнутий шлях кур’єра був найкоротшим? Завдання комівояжера є так званої NP-повною задачею, тобто завданням, точне рішення якої в загальному випадку може бути отримано тільки за методом повного перебору. Але вирішувати її методом повного перебору не ефективно при великій кількості пунктів доставки.

Одним з підходів до її вирішення є скорочення перебору методом гілок і меж. Метод гілок і меж дозволяє визначити не вигідні маршрути по точкам з початку, в результаті значно зменшується кількість точок що прискорює побудову маршруту, але відкинутий на початку невигідний маршрут в кінці може бути найкращим, що дає змогу отримати хороший маршрут, але не оптимальний. Вперше метод гілок і меж був запропонований Лендом і Дойг в 1960 для вирішення загальної задачі цілочислового лінійного програмування.

В основі методу гілок і меж лежить ідея послідовного розбиття множини допустимих рішень на підмножини (стратегія «розділяй і володарюй»). На кожному кроці методу елементи розбиття піддаються перевірці для з'ясування, містить дане підмножина оптимальне рішення чи ні. Як приклад конкретного застосування методу може бути запропонована прикладна задача, пов’язана з проблемою розміщення устаткування, в якій потрібно визначити оптимальну траєкторію руху транспортера по траєкторії цеху.

Також метод з категорії так званих жадібних алгоритмів - найближчого сусіда, ідея його проста, яка полягає в відвідуванні кожної точки маршруту по одному разу та виборі на кожному кроці найближчої точки, що в результаті має дати гарний результат зважаючи на те що на кожному кроці вибір був оптимальним.

2.2 Сервіси побудови маршрутів

У сучасному світі з розвитком міст та значному збільшенню кількості вулиць не можливо орієнтуватися в мегаполісі без карт, особливо коли потрібно відвідати незнайомі міста.

Також при необхідності відвідування деякої кількості точок важко продумати маршрут без додаткових сервісів, не кажучи про їх оптимізацію. Тут на допомогу приходять сервіси які надають картографічні послуги.

Для дослідження та аналізу було вибрано такі платформи: Google Maps та Bing Maps, розглянемо кожен картографічний детально.

2.2.1 Google Maps

Це безкоштовний картографічний сервіс від компанії Google, а також набір застосунків, побудованих на основі цього сервісу й інших технологій Google.

Сервіс являє собою карту та супутникові знімки всього і надає користувачам можливості панорамного перегляду вулиць, аналізу трафіку у реальному часі, прокладання маршруту: автомобілем, пішки, велосипедом або громадським транспортом. З сервісом інтегрований бізнес-довідник і карта автомобільних доріг, з пошуком маршрутів[15]. Сервіс представлений в вигляді вебсайту та мобільних додатків для ОС Android та IOS. Сервіс (рисунок 2.2) реалізований з адаптивним інтерфейсом що захоплює велику кількість різноманітних пристроїв.

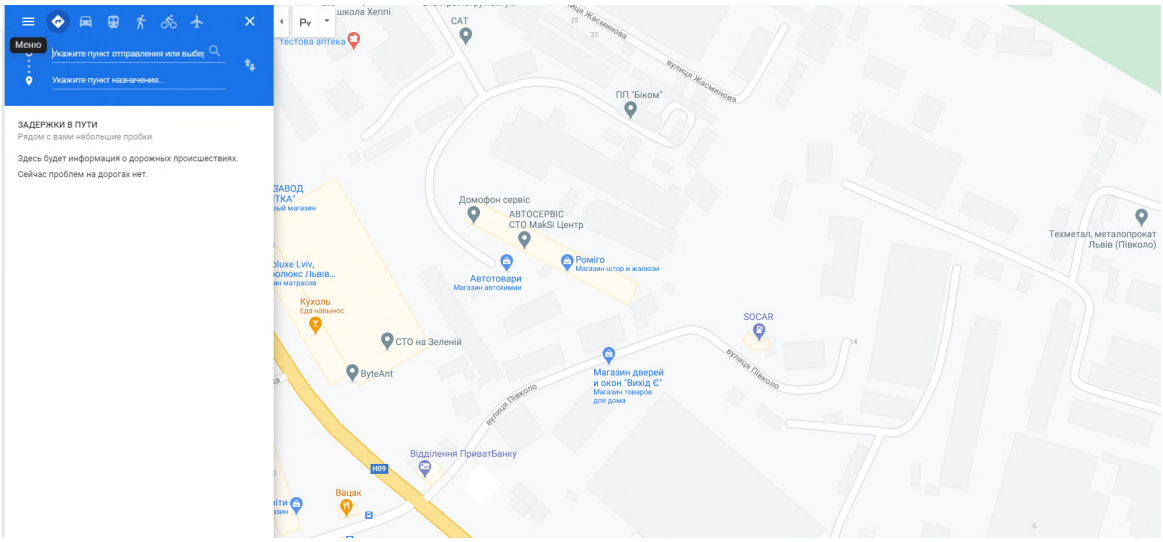

Рисунок 2.2 – Головна сторінка Google Maps

Також Google реалізувало своє API для роботи з їх картами для розробників різних вебдодатків. API Google Maps [14] – це абсолютно

безкоштовний картографічний сервіс, який надається компанією Google. Для прикладу карту можна легко розмістити на своєму сайті і періодично відзначати своє місце положення, щоб клієнти могли легко визначити ваше місце розташування.

Основні переваги сервісу API Google Maps:

1. Простота. Сервіс досить простий і зрозумілий у використанні, користувачеві потрібно всього лише вказати своє місце розташування і прописати дані, які потрібно відобразити на карті. На вказаному місці з'явиться маркер, при натисканні на який відобразиться вся актуальна інформація об'єкта.

2. Доступний функціонал. Сервіс можна оформити під загальну тематику сайту, додатково розширивши його можливості під потреби відвідувачів.

API для визначення відстані приймає наступну форму запиту(рисунок 2.3), де outputFormat – це тип відповіді, а в parameters – маємо передати адреси або координати двох точко між якими шукаєм відстань наприклад: origins=41.43206,-81.38992|-33.86748,151.20699 [18].

```
https://maps.googleapis.com/maps/api/distancematrix/outputFormat?parameters
```

Рисунок 2.3 – Приклад запиту до Google Maps Api

А у відповідь віддає json массив з даними по точкам (рисунок 2.4).


```
{
  "status": "OK",
  "origin_addresses": [
    "Vancouver, BC, Canada",
    "Seattle, État de Washington, États-Unis"
  ],
  "destination_addresses": [
    "San Francisco, Californie, États-Unis",
    "Victoria, BC, Canada"
  ],
  "rows": [
    {
      "elements": [
        {
          "status": "OK",
          "duration": {
            "value": 340110,
            "text": "3 jours 22 heures"
          },
          "distance": {
            "value": 1734542,
            "text": "1 735 km"
          }
        }
```


Рисунок 2.4 – Приклад відповіді від Google Maps Api

2.2.2 Bing Maps

Bing Maps – картографічний сервіс від Microsoft [16], що є частиною порталу Bing. Компанія Microsoft намагалась зайняти нішу найкращого картографічного сервісу, але конкурувати з Google Maps не вдалось, тому що карти від Google набули своєї популярності через доступне API для інтеграції їх карт на різноманітні сайти. Сервіс карт від Microsoft представлений у вигляді сервісу (рисунок 2.5) без адаптивного дизайну для різних пристроїв та без наявності мобільних додатків для ОС Android та IOS.

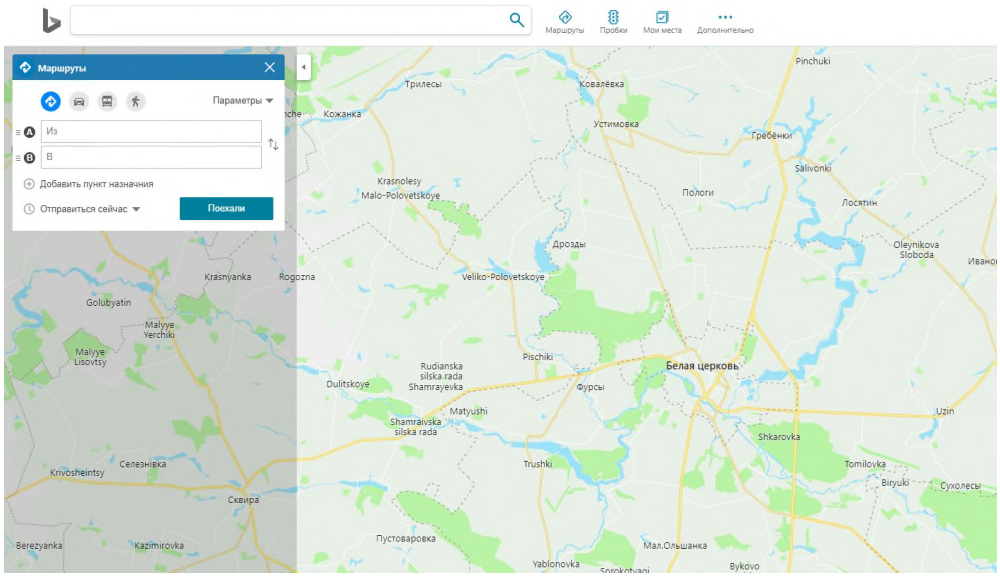

Рисунок 2.5 – Головна сторінка Bing Maps

У Bing Maps також є власне API для роботи з їх картами для розробників різних вебдодатків. Для прикладу карту також можна легко розмістити на своєму сайті. Bing Maps API для початку роботи потрібно підключити скрипт на сторінку де буде виводитись карта та сформувати javascript класс (рисунок 2.6).

JavaScript HTML TypeScript

```
1 var map = new Microsoft.Maps.Map(document.getElementById('myMap'), {});
2 map.setOptions({
3     maxZoom: 12,
4     minZoom: 5
5 });
6
```

Рисунок 2.6 – Приклад створення класу для показу карти.

Після чого на бажаній сторінці в елементі з класом у прикладі це «myMap» з'явиться карта(рисунок 2.7).

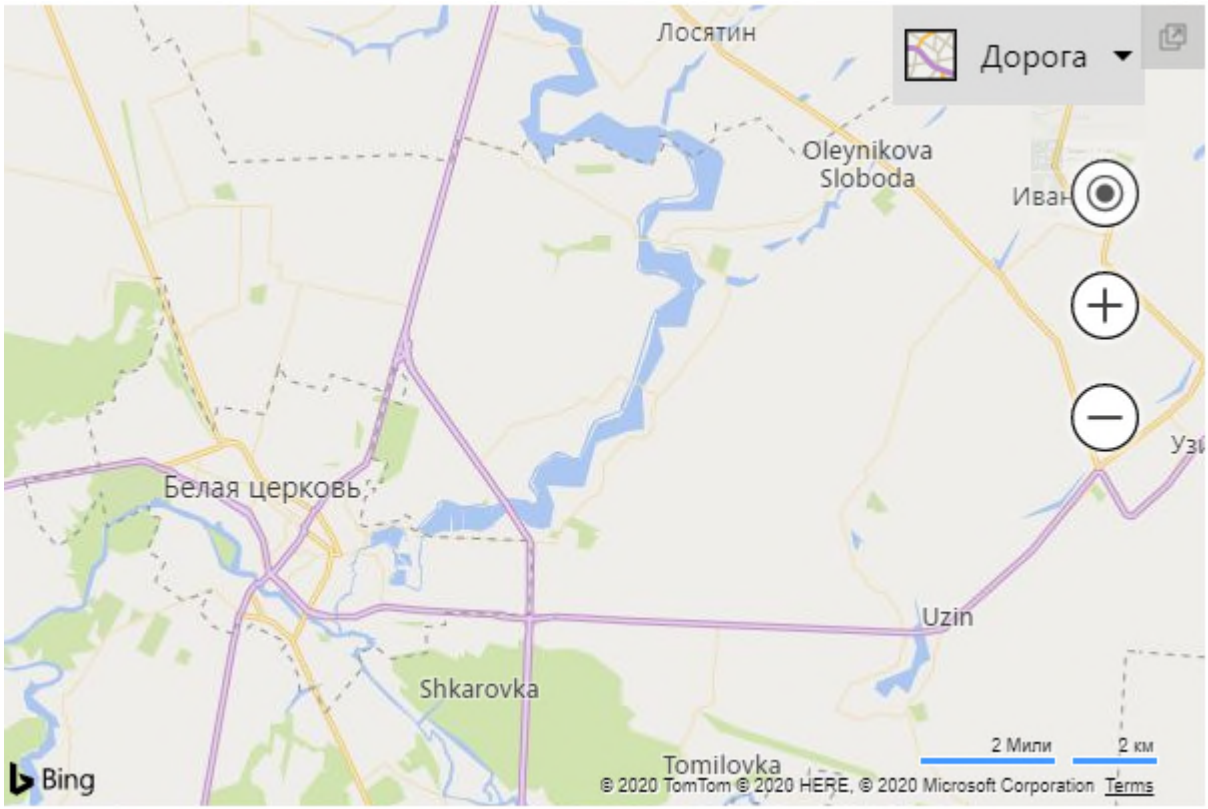

Рисунок 2.7 – Приклад карти.

Але взявши до уваги значно більшу розповсюдженість Google Maps, та зручність використання для розробників і їх легку інтеграцію при потребі з сторонніми сервісами, використання Bing Maps стає не таким привабливим.

2.3 Порівняння методів розв’язку задачі комівояжера

Для визначення методу оптимізації маршрутів доставки проаналізуймо три методи рішення задачі комівояжера: повного перебору, найближчого сусіда та метода гілок і меж. Після аналізу яких буде виявлено оптимальний метод який буде задіяний в інформаційній системі.

Для прикладу використаємо вхідні дані, що наведені в Таблиця 2.1 та відображенні на рисунку 2.8.

Таблиця 2.1. Вхідні данні.

Назва	№	0	1	2	3	4
ЖД Вокзал	0	M	10	11	12	9
ТЦ «Либідь Плаза»	1	8	M	6	6	4
Речовий Ринок	2	7	7	M	4	9
ХНУ	3	10	10	3	M	12
ТЦ «Оазис»	4	8	4	9	9	M

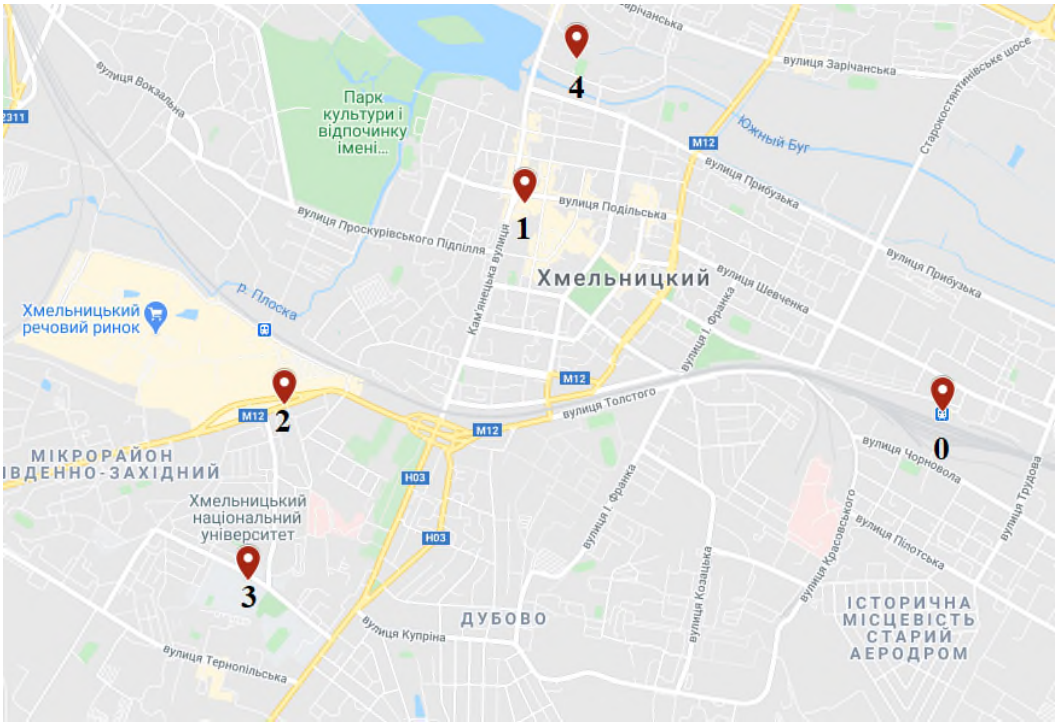

Рисунок 2.8 – Карта точок вхідних даних

Для проведення порівняльного аналізу методів розв'язку задачі комівояжера використаємо координати відомих місць в місті Хмельницькому, час дороги між ними, отриманий за допомогою сервісу GoogleMaps [15], зазначених в хвилинах. В ролі початкової точки було обрано «ЖД Вокзал».

Розрахуємо оптимальний час доставки за вхідними даними за допомогою кожного з методів.

Метод повного перебору може застосовуватися тільки для задач малої розмірності при розгляді до двох десятків відвідуваних пунктів.

Для реалізації методу повного перебору необхідно навчитися виробляти генерацію всіх перестановок заданого числа елементів. Зробити це можна кількома способами, але найпоширеніший (самий використовуваний в інших переборних алгоритмах) - це перебір в лексикографічному порядку [17].

Розглянемо як приклад перебір перестановок з п'яти елементів, позначених цифрами 1 ... 5. Лексикографічно першої перестановкою буде 1-2-3-4-5, другий - 1-2-3-5-4 останньою - 5-4-3-2-1. Для безпосередньої реалізації генерації перестановок потрібно визначити загальний алгоритм перетворення будь-якої перестановки в наступну.

Принцип роботи алгоритму:

Нехай дана перестановка 1-3-5-4-2. Потрібно рухатися по перестановці справа наліво, поки вперше не буде виявлено число, менше, ніж попереднє. Припустимо, що знайдене число розташовується на позиції P_i-1.

Міняємо знайдене число місцями з найменшим з великих чисел, які розташовані правіше позиції P_i-1. Після чого, числа правіше позиції P_i-1 необхідно впорядкувати по зростанню. В результаті виходить безпосередньо наступна перестановка: у прикладі це 1-4-2-3-5 , за нею йде 1-4-2-5-3 і так далі.

Після генерації перестановки рахується сума проїзду між пунктами і отримується результат. Після того, як згенеровані всі перестановки (Рисунок 2.9) і пораховані суми проїзду для них, вибирається маршрут, відповідний мінімальному часу об'їзду пунктів.

Назва	№	0	1	2	3	4
ЖД Вокзал	0	M	10	11	12	9
ТЦ «Либідь Плаза»	1	8	M	6	6	4
Речовий Ринок	2	7	7	M	4	9
ХНУ	3	10	10	3	M	12
ТЦ «Оазис»	4	8	4	9	9	M

Метод повного перебору генерує 24 маршрути

012340 031240
012430 031420
013240 032140
013420 032310
014230 034120
014320 034210
021340 041230
021430 041320
023140 042130
023410 042310
024130 043120
024310 043210

Рисунок 2.9 – Можливі варіанти маршрутів згенеровані методом повного перебору.

В результаті кур'єр пройшов за оптимальним маршрутом 0-4-1-3-2-0 (Рисунок 2.10), витративши на об'їзд 9 + 4 + 6 + 3 + 10 = 32 хвилини.

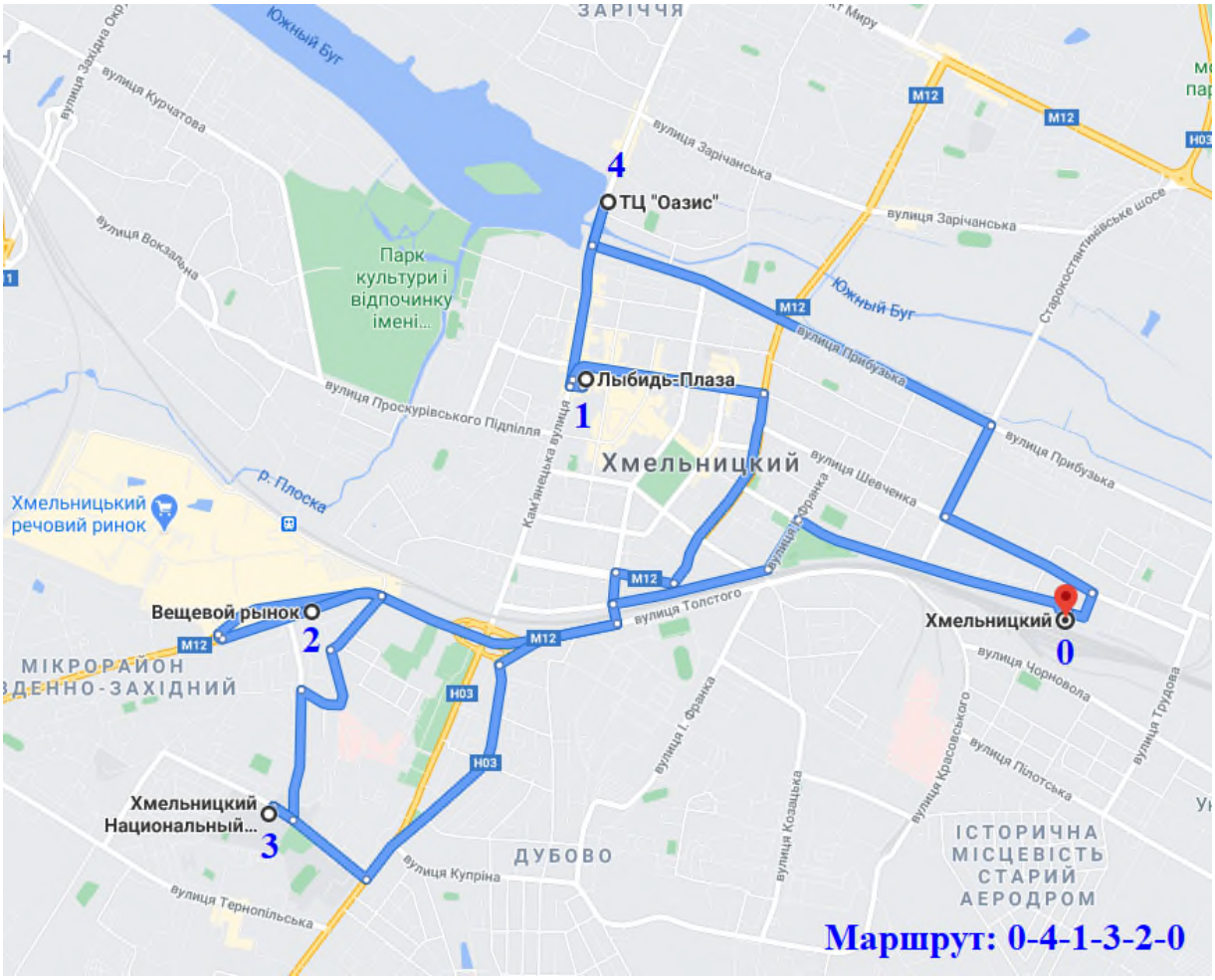

Рисунок 2.10 – Маршрут побудований методом повного перебору.

Метод найближчого сусіда відноситься до так званих жадібних алгоритмів. Жадібний алгоритм має на увазі під собою прийняття локально оптимальних рішень, допускаючи що кінцеве рішення також виявиться оптимальним.

Існує досить велика кількість завдань, для яких жадібні алгоритми дають оптимальні рішення. Однак для ряду завдань, жадібні алгоритми не дають оптимального рішення. В таких завданнях, в тому числі і в завданні комівояжера, вони дають досить непогане наближене рішення.

Ідея алгоритму найближчого сусіда заснована на простому правилі: якщо відвідувати найближчий пункт на кожному кроці, то маршрут вийде досить хороший в цілому. Перед комівояжером ставиться завдання відвідувати найближчий з ще не відвіданих пунктів. В алгоритмі існують два важливих обмеження:

1) Недопущення повторного заїзду в пункт. Воно пов'язане з необхідністю (за умовою завдання) знаходження Гамільтона циклу, тобто циклу, в якому всі пункти відвідуються раз.

2) Недопущення передчасного повернення в початковий пункт. Ця заборона вводиться для запобігання передчасного зациклення маршруту, яке потягне за собою неправильну роботу алгоритму.

Схема алгоритму зображена на рисунку 2.11.

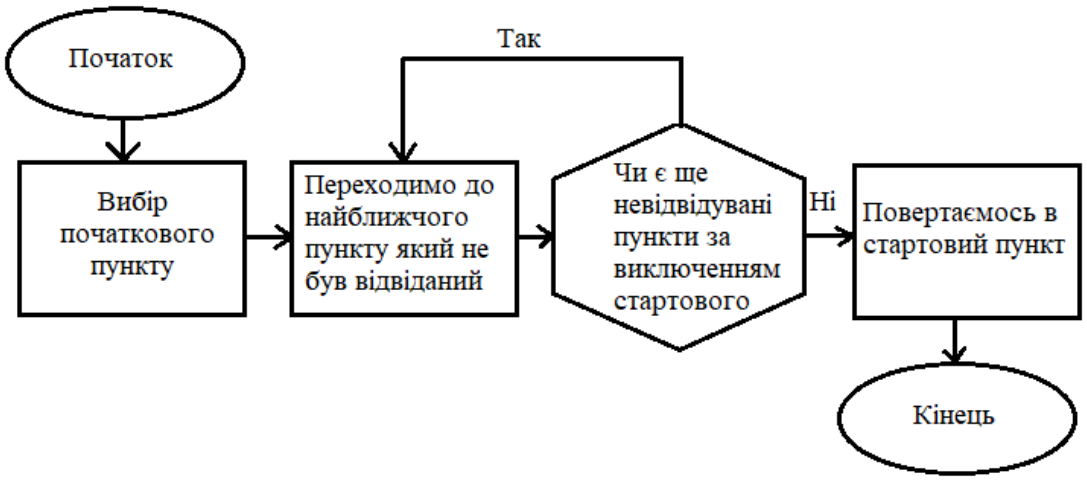

Рисунок 2.11 – Схема алгоритму найближчого сусіда

Для аналізу роботи методу взяті дані з матриця часів проїзду між популярними транспортними вузлами м. Хмельницького (Таблицю 2.1).

На першому і наступних етапах кур'єру необхідно вибирати найближчий з ще не відвіданих пунктів. Також необхідно пам'ятати, що повернення в початковий пункт завчасно неможливий.

На першому кроці вибір проводиться без обмежень, так як повторне відвідування поки не можливо, і повернення в стартовий пункт навіть теоретично не представляється реальним через знаходження в ньому. Тобто перебуваючи в базовому пункті «ЖД Вокзал» кур'єр вибере для відвідування пункт 4 «ТЦ Оазис», час проїзду до якого 9 хвилин мінімально дивись таблицю 2.2.

Таблиця 2.2. Крок перший.

Назва	№	0	1	2	3	4
ЖД Вокзал	0	M	10	11	12	9
ТЦ «Либідь Плаза»	1	8	M	6	6	4
Речовий Ринок	2	7	7	M	4	9
ХНУ	3	10	10	3	M	12
ТЦ «Оазис»	4	8	4	9	9	M

Опинившись в 4-му пункті для подальшого відвідування кур'єр вибере 1-ий пункт, тому що дистанція до нього найменша та він не суперечить умовам задачі дивись таблицю 2.3.

Таблиця 2.3. Крок другий.

Назва	№	0	1	2	3	4
ЖД Вокзал	0	M	10	11	12	9
ТЦ «Либідь Плаза»	1	8	M	6	6	4
Речовий Ринок	2	7	7	M	4	9
ХНУ	3	10	10	3	M	12
ТЦ «Оазис»	4	8	4	9	9	M

З 1-го пункту кур'єр піде в 2-й, так як час до 2-го і 3-го однаковий, але 2-ий перший по порядку та ігноруючи повторне відвідування 4-го пункту дивись таблицю 2.4.

Таблиця 2.4. Крок третій.

Назва	№	0	1	2	3	4
ЖД Вокзал	0	M	10	11	12	9
ТЦ «Либідь Плаза»	1	8	M	6	6	4
Речовий Ринок	2	7	7	M	4	9
ХНУ	3	10	10	3	M	12
ТЦ «Оазис»	4	8	4	9	9	M

На цьому етапі так само, як і раніше вибераємо найблищий пункт який не суперечить умовам – це «Хмельницький Національний Університет» дивись таблицю 2.5.

Таблиця 2.5. Крок четвертий.

Назва	№	0	1	2	3	4
ЖД Вокзал	0	M	10	11	12	9
ТЦ «Либідь Плаза»	1	8	M	6	6	4
Речовий Ринок	2	7	7	M	4	9
ХНУ	3	10	10	3	M	12
ТЦ «Оазис»	4	8	4	9	9	M

Як тільки всі точки відвідані, кур'єр може повертатися в базовий пункт. Це і відбувається на останньому етапі дивись таблицю 2.6.

Таблиця 2.6. Крок п’ятий.

Назва	№	0	1	2	3	4
ЖД Вокзал	0	M	10	11	12	9
ТЦ «Либідь Плаза»	1	8	M	6	6	4
Речовий Ринок	2	7	7	M	4	9
ХНУ	3	10	10	3	M	12
ТЦ «Оазис»	4	8	4	9	9	M

В результаті кур'єр пройшов за маршрутом 0-4-1-2-3-0 (Рисунок 2.12), витративши на об'їзд 9 + 4 + 6 + 4 + 10 = 33 хвилини.

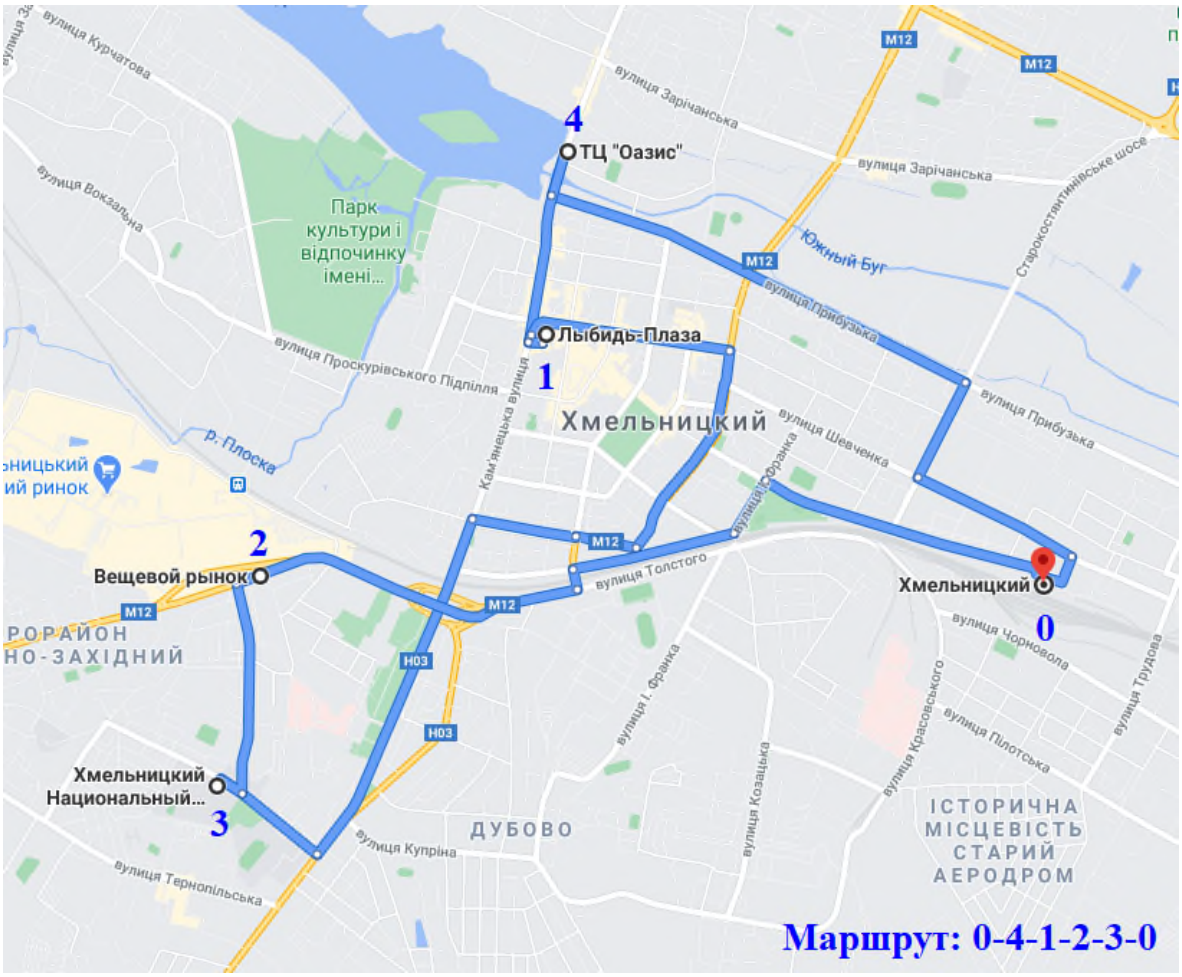

Рисунок 2.12 – Маршрут побудований методом найближчого сусіда

Метод гілок і меж - загальний алгоритмічний метод, який широко застосовується для вирішення завдань комбінаторної оптимізації.

По суті, метод є вдосконаленим методом повного перебору з послідовним відсівом рішень, що здаються невигідними.

Внаслідок того, що в процесі роботи методу деякі рішення не розглядаються, метод гілок і меж не може гарантувати знаходження точного рішення задачі. Відкинуте на початковому етапі «невигідне» рішення може виявитися в кінці кінців кращим.

У 1963 році групою авторів була запропонована модифікація методу гілок і меж, розроблена спеціально для вирішення завдання комівояжера. Згодом цей алгоритм отримав назву «Алгоритм Літтла».

В основі методу гілок і меж лежить ідея послідовного розбиття множини рішень шляхом розгалуження і знаходження оцінок границь схема методу приведена на рисунку 2.13.

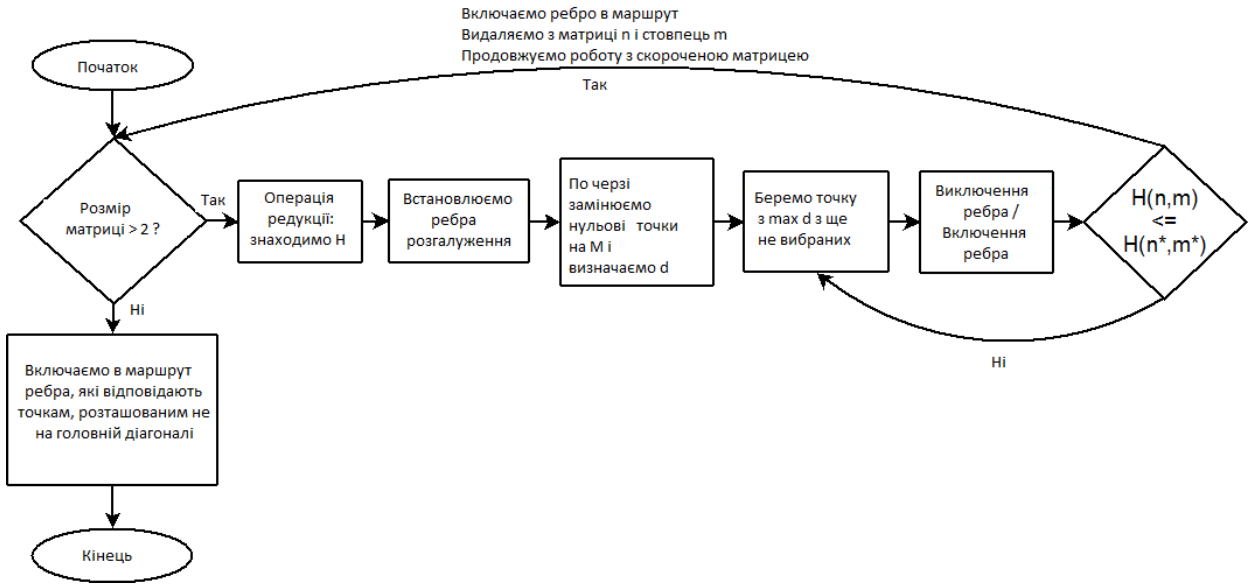

Рисунок 2.13 – Схема метода гілок і меж.

Метод гілок і меж вважається універсальним методом, так як він добре підходить для вирішення антисиметричною завдання комівояжера.

Алгоритм складається з двох етапів:

Перший етап : операція приведення матриці і обчислення нижньої оцінки вартості маршруту H.

Знаходження мінімальних елементів в кожному рядку константи приведення *di* таблиця 2.8.

Таблиця 2.8. Крок перший.

№	0	1	2	3	4	di
0	M	10	11	12	9	9
1	8	M	6	6	4	4
2	7	7	M	4	9	4
3	10	10	3	M	12	3
4	8	4	9	9	M	4

З елементів рядків матриці віднімаються константи приведення di таблиця 2.9.

Таблиця 2.9. Крок два.

№	0	1	2	3	4
0	M	1	2	3	0
1	4	M	2	2	0
2	3	3	M	0	5
3	7	7	0	M	9
4	4	0	5	5	M

Знаходження мінімального елемента в кожному стовпці та константи приведення dj таблиця 2.10

Таблиця 2.10. Крок третій

№	0	1	2	3	4
0	M	1	2	3	0
1	4	M	2	2	0
2	3	3	M	0	5
3	7	7	0	M	9
4	4	0	5	5	M
dj	3	0	0	0	0

З елементів стовпців матриці віднімаються константи приведення таблиця 2.11.

Таблиця 2.11. Крок четвертий.

№	0	1	2	3	4
0	M	1	2	3	0
1	1	M	2	2	0
2	0	3	M	0	5
3	4	7	0	M	9
4	1	0	5	5	M

В результаті отримуємо наведену матрицю, в якій в кожному рядку і в кожному стовпці є хоча б один нульовий елемент. Обчислення H (нижньої межі) як суму констант приведення d_i и d_j.

$$H = \sum d_i + \sum d_j$$

$$H = 9 + 4 + 4 + 3 + 4 + 3 + 0 + 0 + 0 + 0 = 27$$

Другий етап :

Обчислення штрафу за невикористання для кожного нульового елемента наведеної матриці витрат.

Штраф за невикористання елемента з індексом (*i*, *j*) означає, що відповідне Ребров не буде включено в маршрут, а значить мінімальна вартість невикористання цього ребра дорівнює сумі мінімальних елементів в рядку *i* і стовпці *j*.

а) Шукаються всі нульові елементи в наведеній матриці таблиця 2.12

Таблиця 2.12. Крок п’ятий.

№	0	1		2	3	4
0	M	1		2	3	0
1	1	M		2	2	0
2	0	3		M	0	5
3	4	7		0	M	9
4	1	0		5	5	M

б) Нульові елементи по черзі замінюються на M (нескінченність).

Кожному нульового елементу зіставляється штраф сума констант приведення за його невикористання таблиця 2.13.

Таблиця 2.13. Крок шість.

№	0	1	2	3	4	di
0	M	1	2	3	0 (9)	9
1	1	M	2	2	0 (4)	4
2	0 (7)	3	M	0 (4)	5	4
3	4	7	0 (3)	M	9	3
4	1	0 (4)	5	5	M	4
dj	3	0	0	0	0	

в) Вибирається елемент з ще не обраних, якому відповідає

максимальний штраф.

Вибір елемента з максимальним штрафом обумовлений тим, що виключення з маршруту цього ребра призведе до максимального збільшення вартості оптимального маршруту, знаходження якого є нашою метою. Найбільша сума констант приведення дорівнює (9 + 0) = 9 для ребра (0,4), отже, вибираємо цей елемент.

Тепер вихідна безліч розбивається на дві підмножини - (i *, j *) (містить ребро (i, j)) і (i, j) (що не містить ребро (i, j)). Проводиться обчислення нижніх меж для підмножин. У нашому випадку безліч розбивається на підмножини (0,4) і (0*,4*).

а) Виключення ребра ($i*$, $j*$)

Нижня межа для підмножини (i *, j *) дорівнює сумі нижньої межі H вихідного безлічі і штрафу за невикористання ребра (i, j).

Виняток ребра (0,4) проводиться шляхом заміни елемента $d(0,4) = 0$ на M, після чого здійснюється чергове приведення матриці відстаней для утворився підмножини (4 *, 0 *), в результаті отримаємо скороченої матрицю таблиця 2.14.

Таблиця 2.14. Крок сім.

№	0	1	2	3	4	di
0	M	1	2	3	M	9
1	1	M	2	2	0	0
2	0	3	M	0	5	0
3	4	7	0	M	9	0
4	1	0	5	5	M	0
dj	0	0	0	0	0	

Нижня межа гамільтонових циклів цього підмножини:

$$H(0*,4*) = 27 + 9 = 36$$

б) Включення ребра (i, j)

При обчисленні нижньої межі для безлічі (i, j) необхідно взяти до уваги, що раз ребро (i, j) входить в маршрут, то симетричне ребро (j, i) в маршрут входити не може, тому в матриці цей елемент замінюється на M .

А в зв'язку з тим, що з пункту i – «вже пішли», а в пункт j – «вже прийшли», то жодне ребро, що виходить з i, і жодне ребро, що приходить в j, вже використовуватися не можуть, тому викреслюємо з матриці рядок i і стовпець j.

Проводимо операцію приведення для вийшла матриці. Нижня межа для (i, j) буде дорівнює нижній межі H вихідного безлічі плюс сума констант приведення для вийшла скороченої матриці.

Включення ребра (0,4) проводиться шляхом виключення всіх елементів 0-го рядка і 4-го стовпця, в якій елемент $d(4,0)$ замінюється на M, для запобігання утворенню негамільтонова циклу. В результаті виходить інша скорочена матриця (4 × 4), яка підлягає операції приведення.

Після операції приведення скорочена матриця буде мати вигляд (таблиця 2.15):

Таблиця 2.15. Крок вісім.

№	1	2	3	4	di
0	1	M	2	2	1
1	0	3	M	0	0
2	4	7	0	M	0
3	M	0	5	5	0
dj	0	0	0	0	

Сума констант приведення скороченою матриці:

$$\sum d_i + \sum d_j = 1$$

Якщо $(i, j) \leq (i^*, j^*)$, то ребро включається в маршрут, і подальша робота методу відбувається з матрицею меншої розмірності.

В іншому випадку відбувається повернення до пункту 1 і розглядається елемент з наступним по спадаючій штрафом за невикористання.

Нижня межа підмножини (0,4) дорівнює:

$$H\,(0,4) = 27 + 1 = 28 < 36$$

Оскільки нижня межа цієї підмножини (0,4) менше, ніж підмножини (0 *, 4 *), то ребро (0,4) включається в маршрут.

Процес розгалуження шляхом включення і виключення ребер триває до тих пір, поки в розгляді не опиниться матриця розмірності 2. З неї додавання ребер в маршрут відбувається тривіальним чином. При вирішенні завдання комівояжера методом гілок і меж необхідно перед кожною ітерацією алгоритму забороняти для відвідування ребра, які можуть привести до передчасного зациклення алгоритму. Робиться це шляхом присвоєння відповідного елементу матриці значення M.

В даному випадку алгоритмом в кінці роботи був запропонований наступний маршрут: 0-4-3-1-2-0 (Рисунок 2.14). Час об'їзду маршруту, запропонованого методом гілок і меж, займе 36 хвилини

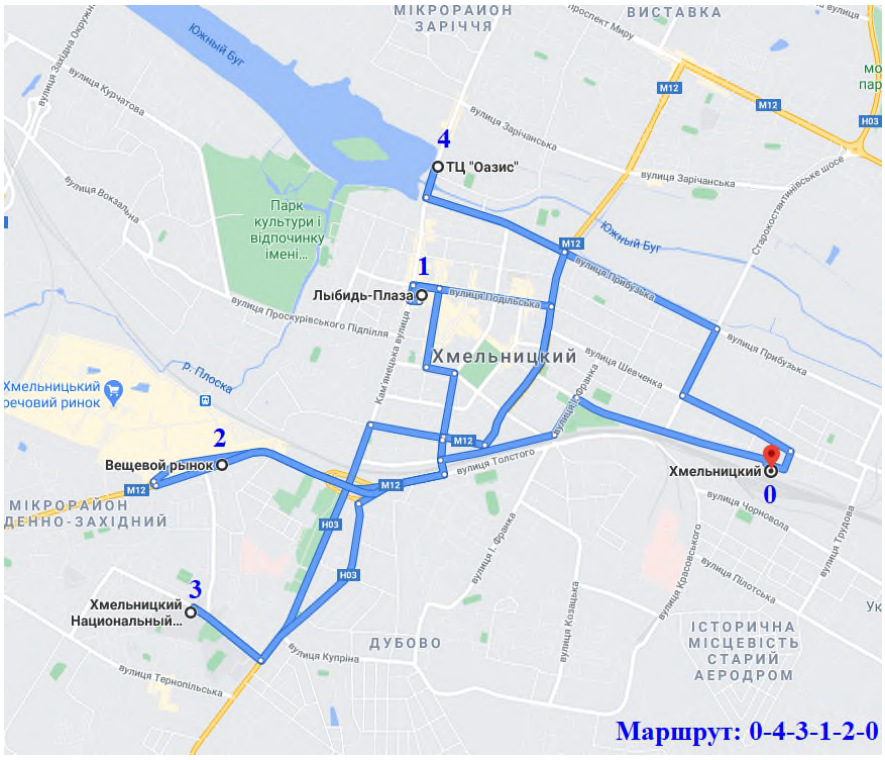

Рисунок 2.14 – Маршрут побудований методом меж і границь.

Виходячи з результатів (Таблиця 2.16) методів рішення задачі комівояжера можна зробити висновок, що кращий результат дав метод повного перебору, але при великій кількості точок доставки обрахунки маршруту системи дуже зростають [17].

Тому методом оптимізації маршрутів інформаційної системи доцільніше буде обрати метод найближчого сусіда який дав наступний кращий результат, на основі якого буде побудована інформаційна система оптимізації доставки замовлень.

Таблиця 2.16. Результати оптимізованих маршрутів.

Метод	Маршрут по точкам	Час маршруту
Повного перебору	0-4-1-3-2-0	32 хвилини
Найближчого сусіда	0-4-1-2-3-0	33 хвилини
Гілок і меж	0-4-3-1-2-0	36 хвилин

Висновки до розділу 2

Запропоновано інформаційну систему для оптимізації маршрутів доставки замовлень в мережі закладів швидкого харчування. Для визначення відстані між точками було запропоноване використання API Google Maps.

Досліджена задача комівояжера для рішення поставленої завдання. Розглянуто три методи рішення задачі комівояжера, що можуть бути використанні для оптимізації доставки замовлень: повного перебору, найближчого сусіда та метода гілок і меж.

Проаналізувавши методи рішення задачі комівояжера, в рамках інформаційної технології, запропоновано використати метод найближчого сусіда.

Розділ 3
Проектування інформаційної системи доставки онлайн замовлень

3.1 Основні вимоги для проектування інформаційної технології

Проектування інформаційних системи завжди починається з визначення мети проекту та у формуванні основних функціональних вимог.

Метаю роботи є розробка інформаційної системи оптимізації маршрутів доставки онлайн замовлень в мережі закладів швидкого харчування, яка б дала можливість скоротити час доставки:

– функціональну модель, що відповідає узагальненим вимогам;

– модель інтерфейсу взаємодії користувача;

– компонент взаємодії системи з Google Maps API для отримання відстаней між точками доставки та навігацію по ним;

– компонент формування оптимізованого маршруту з використанням методу рішення задачі комівояжера.

Крім того, необхідно описати основні функціональні блоки моделі, що складають систему, їх взаємодію та залежності.

Також програмна реалізація інформаційної системи повинна забезпечити функціональні вимоги для здійснення та обробки онлайн-замовлень.

3.2 Проектування загальної схеми інформаційної системи

Для забезпечення функціональних вимог в рамках інформаційної системи необхідно запропонувати функціональну модель, яка складається з декількох основних функціональних блоків (рисунок 3.1).

Рисунок 3.1 – Функціональна модель інформаційної системи

Блок оформлення замовлення формує замовлення зробленого користувачем системи та спрямовує сигнал на обробку адреси вказаної в замовлені на сервіс Google Maps API.

Блок модуль взаємодії з Google Maps API відповідає за відправку та обробку відповідей з сервісу Google Maps. Таким чином, даний блок виконує функцію визначення часу дороги до точок, на виході з цього функціонального блоку отримується матриця часу.

Блок оптимізації маршруту отримавши матрицю часу, займається оптимізацією маршруту доставки по точкам за допомогою метода розв’язку задачі комівояжера, а саме методу найближчого сусіда.

Запропонована функціональна модель описує процес обробки вхідної інформації, вхідні та вихідні параметри кожного блоку, що дозволить реалізувати в цілому інформаційну систему оптимізації маршрутів доставок з використанням методу розв’язку задачі комівояжера.

3.3 Проектування інтерфейсної частини інформаційної системи

Веб-додаток має різні інтерфейси для користувацької частини і для адміністративної частини, при цьому обидві частини мають використовувати однакову структуру, але з різними наборами модулів інтерфейсу.

Виходячи з розгляду існуючих сервісів доставки з розділу 1.3, можна зробити висновок, що всі представлені сервіси реалізують адаптивний дизайн для мобільних пристроїв, а також мають зручне навігаційне меню на основі існуючих сервісів розробимо шаблон нашої інформаційної системи.

Також в адміністративну панель потрібно включити виведення бокового меню, для забезпечення зручної навігації. На рисунку 3.2 показаний загальний шаблон інформаційної системи який буде використовуватись як основа для забезпечення функціональних вимог для здійснення та обробки онлайн-замовлень.

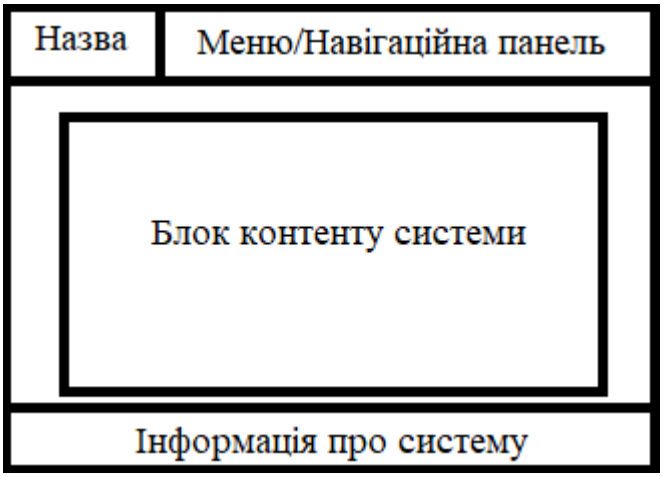

Рисунок 3.2 – Типовий шаблон інтерфейсу інформаційної системи.

Для реалізації користувацької частини для здійснення замовлень інтерфейс має мати системи має такі інтерфейсні елементи: списки з блоками інформації про товар, кошик для замовлення, інформаційні блоки про систему. На рисунку 3.3 зображена схема яка реалізує вказані вище елементи.

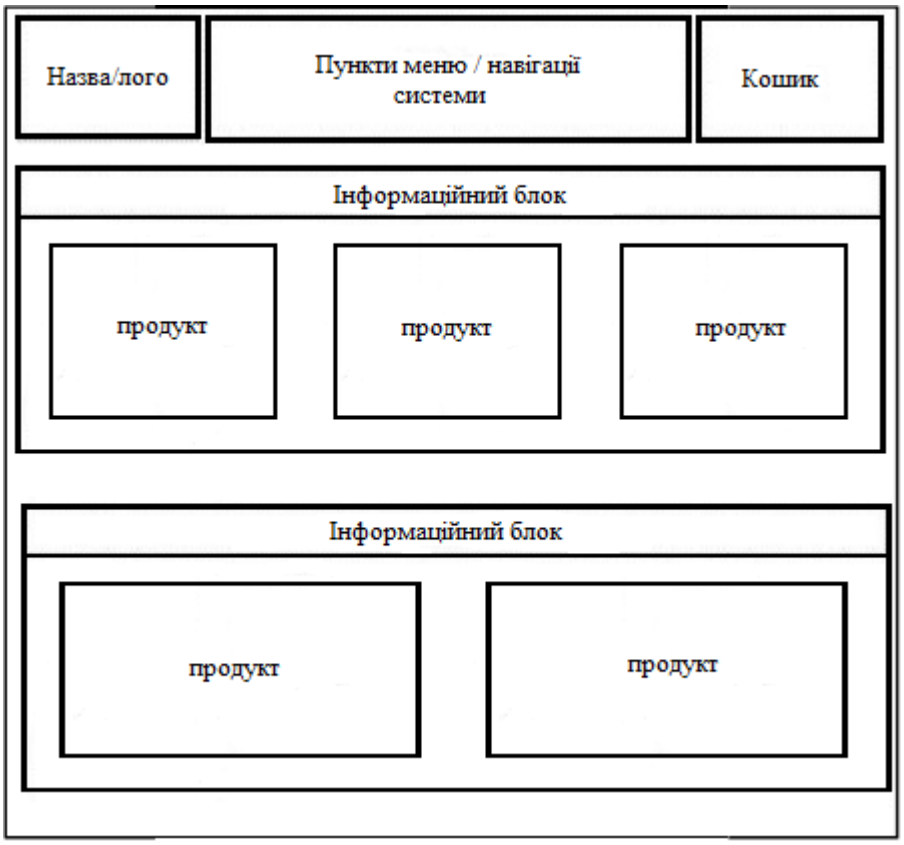

Рисунок 3.3 – Схема користувацької частини

Також потрібно спроектувати схему (Рисунок 3.4) користувацького інтерфейсу для мобільних пристроїв які будуть користуватись системою, по якій буде реалізований адаптивний дизайн.

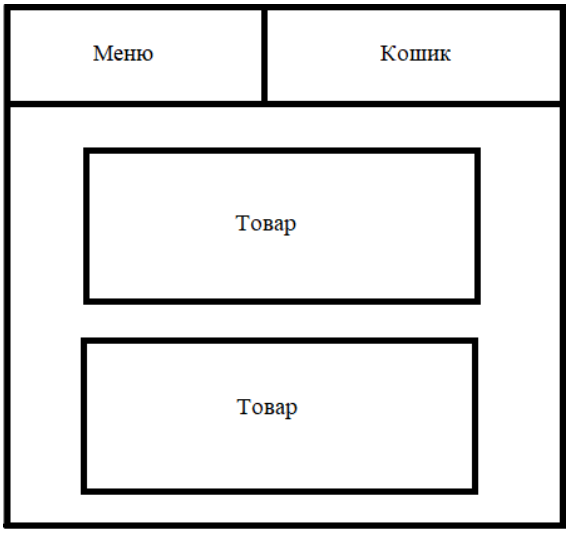

Рисунок 3.4 – Смеха адаптивного інтерфейсу для мобільних пристроїв

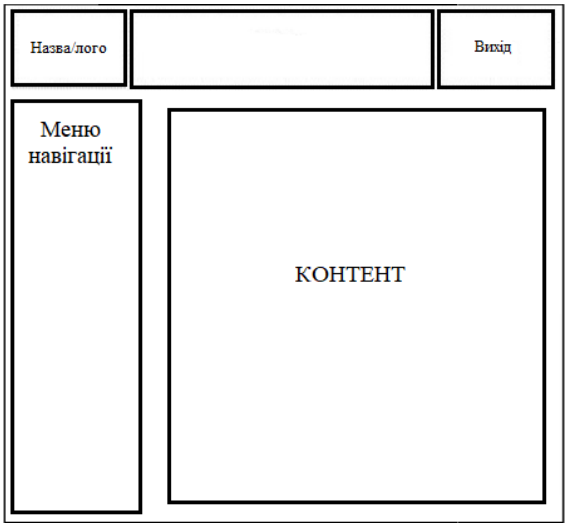

Рисунок 3.5 – Схема адміністративної частини

Для розробки адміністративної частини системи за основу також був взяти шаблон (рисунок 3.2) та спроектована схема (Рисунок 3.5).

Для перегляду маршруту доставки кур’єром буде використовуватись адміністративна частина системи, тому окремого проектування вона не потребує.

Розроблені блок схеми будуть використанні в подальшій програмній розробці у вигляді макетних інтерфейсних заготовок.

3.4 Проектування компонентів взаємодії з Google Map API

Для взаємодії з сервісом Google Maps API буде розроблений окремий сервіс в інформаційній системі. Для визначення відстані між пунктами буде використовуватись компонент сервісу під назвою Distance Matrix API.

Distance Matrix API - це сервіс, який надає інформацію про відстань і час в дорозі для матриці пунктів відправлення і призначення. API повертає інформацію на основі рекомендованого маршруту між початковим і кінцевим пунктом, розрахованого за допомогою Google Maps API, і складається з рядків, що містять значення тривалості і відстані для кожної пари[18].

Також для відображення карти маршруту кур’єру буде використаний компонент Google Maps API - Maps JavaScript API.

Maps JavaScript API дозволяє налаштовувати карти з вашим власним контентом і зображеннями для відображення на веб-сторінках і мобільних пристроях. Maps JavaScript API пропонує чотири основних типи карт (дорожня карта, супутникова, гібридна і рельєфна), які ви можете змінювати, використовуючи шари і стилі, елементи управління і події, а також різні служби і бібліотеки[19].

Використання даних сервісів зображене на блок схемі(рисунок 3.6).

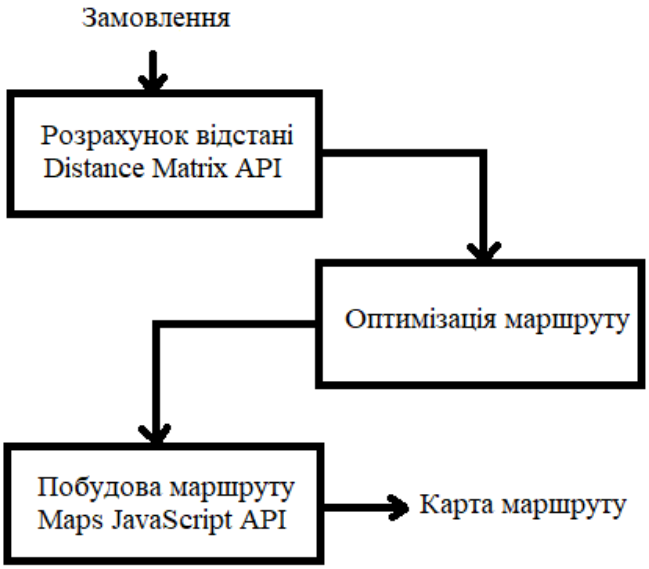

Рисунок 3.6 – Блок схема оптимізації маршруту з використанням Google Maps API

За даною блок схемою необхідно розробити модуль для побудови оптимізованого маршруту відображеного на карті. Приклад такого побудованого маршруту показано на рисунку 3.7.

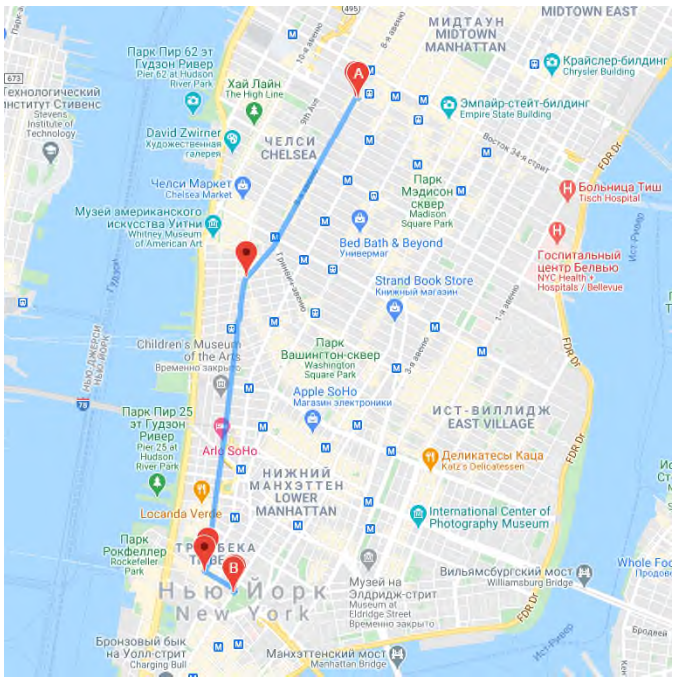

Рисунок 3.7 – Приклад результату роботи сервісу Maps JavaScript API

3.5 Проектування структури сховища даних

Для забезпечення функціональних можливостей системи потрібне проектування сховища даних для збереження інформації яка буде використовуватись системой. На рисунок 3.8 спроектована структура БД, яка буде мати відповідні таблиці “order”, “order_item”,“products”, “categories”, “photos”, “restaurants”, “couriers”, “user”.

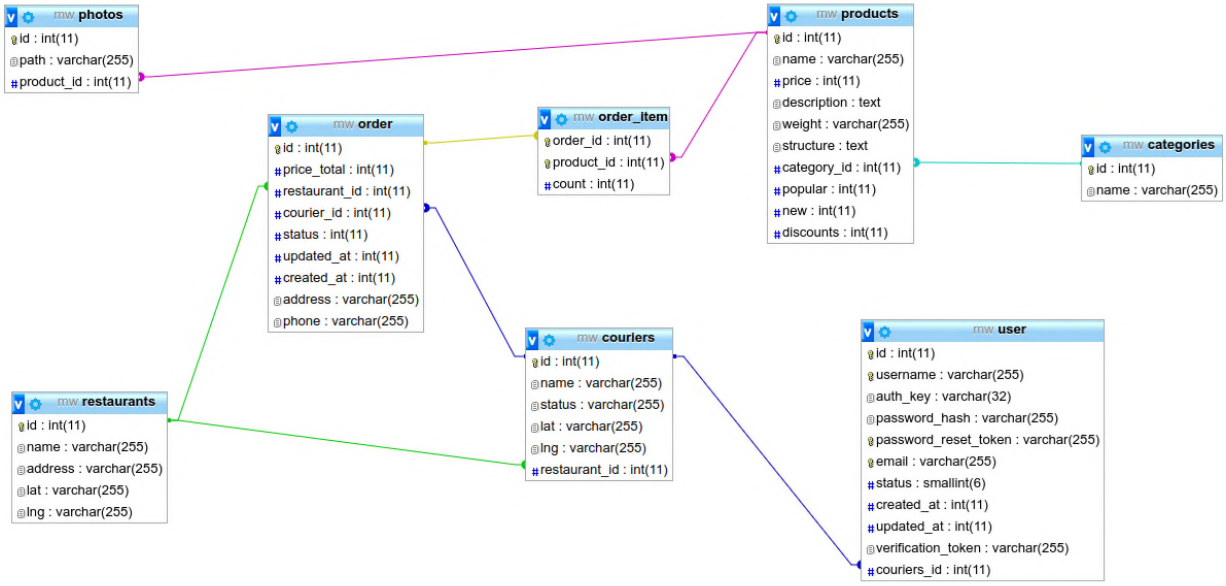

Рисунок 3.8 – Структура БД

Таблиця “order” буде зберігати в собі інформацію по замовленням користувачів. Таблиця “user” буде зберігати в собі інформацію про адміністраторів сервісу. Таблиця “courier” буде зберігати в собі інформацію про кур’єрів. Таблиця “restaurants” буде зберігати в собі інформацію про ресторани системи. Таблиця “photos” для збереження фотографій товарів. Таблиця “products” буде зберігати в собі інформацію про товари системи. Таблиця “categories” міститиме інформацію про категорії товарів в системі. Таблиця

“order_item” буде використовуватись в системі для переліку товарів в замовленні.

Висновки до розділу 3

В результаті виконання даного розділу спроектовано інформаційну систему оптимізації маршрутів доставок з використанням методу розв’язку задачі комівояжера.

Сформульовано вимоги, які необхідно врахувати при проектуванні інформаційної системи. Запропоновано функціональну модель для забезпечення цих вимог в рамках інформаційної системи.

Спроектовані інтерфейси для користувацької та адміністративної частини. Розглянуті та спроектована робота з сервісами Google Maps API. Спроектоване сховище даних для інформаційної системи.

Розділ 4
Програмна реалізація

4.1 Обґрунтування вибору мови програмування для розробки інформаційної системи

Проаналізувавши готові рішення розглянути в розділі 1.3, можна зробити висновок: що використання однієї з мобільних платформ IOS або Android буде не доцільним, бо для повного охвату аудиторії потрібно буде реалізовувати два мобільних додатка.

Тому оптимальним рішенням для реалізації інформаційної системи обрано вебсайт, який добре працюватиме як на мобільних пристроях через свою обробку скриптів на вебсервері та адаптивний дизайн який буде підлаштовуватись під розширення екрану. Так і на стаціонарних комп'ютерах, що значно зменшить витрачений час на її розробку та підтримку через використання однієї системи.

Для аналізу мов програмування було обране дослідження порталу "Wappalyzer" з кращими мовами програмування для веб-розробки за 2020 рік(рисунок 4.1).

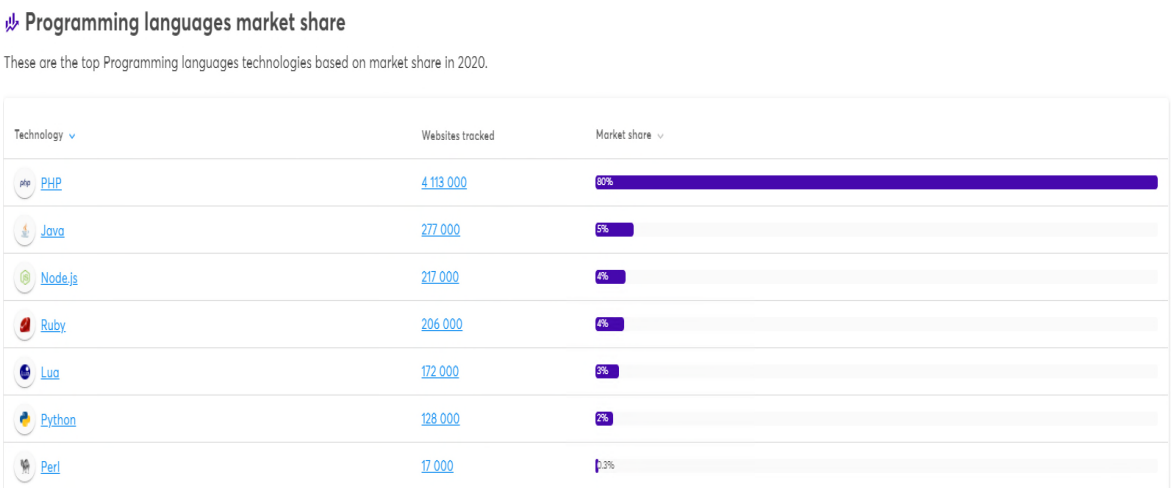

Рисунок 4.1 – Діаграма дослідження мов програмування [9]

Розглянемо трійку лідерів цього списку:

PHP – це поширена мова програмування загального призначення з відкритим вихідним кодом. PHP спеціально сконструйований для веб-розробок та його код може впроваджуватися безпосередньо в HTML [10].

Основна відмінність PHP від JavaScript це те, що PHP-скрипти виконуються на сервері і генерують HTML, який надсилається клієнту. Вся обробка коду лежить на сервері що зменшує використання ресурсів пристроями які відвідують вебсторінку, по факту клієнт отримає тільки результат обробки, та не може знати який саме код його обробив що забезпечує ще і безпечність.

PHP вкрай простий для освоєння, але разом з тим здатний задовольнити запити професійних програмістів. PHP це мова яка не стоїть на місці та постійно розвивається, нещодавно анонсували нову версію для тестування, а саме PHP 8.0.

Java – це технологія, яка використовується для розробки додатків, які роблять роботу в мережі Інтернет більш захоплюючою і зручною. Java відрізняється від мови програмування javascript, який представляє собою просту технологію для створення веб-сторінок і виконується тільки в браузері[12].

За допомогою Java можна реалізувати будь які забаганки, це складний інструмент в якому можна реалізувати такі функціональні можливості як: завантажувати фотографії, спілкуватися в режимі онлайн, здійснювати віртуальні екскурсії і користуватися такими послугами як дистанційне навчання, дистанційне банківське обслуговування та перегляд інтерактивних карт та навіть ігри[13] . Java потребує встановлення свого середовища, для запоруки коректної роботи додатків і веб-сайтів без якого вони просто не будуть працювати.

Java це скадна і потужна система яка використовується для розробки різного роду віртуальних і звичних додатків.

Node.Js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Якщо

раніше Javascript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників[13].

Node.Js це свого роду оболонка яка робить з мови програмування JavaScript яка в основному застосовується для взаємодії з користувачем, керування браузером, асинхронного обміну даними з сервером, змінювання структури та зовнішнього вигляду вебсторінки в повноцінний інструмент для розробки вебсайтів. Так само як і php цілеспрямовано використовується в основному для розробки вебсайтів за що і отримала свою популярність у розробників.

Спираючись на розгляд з описами мов програмування робимо висновок що розробка вебсайту можлива на всіх цих мовах, але Java більш різностороння мова яка важка в освоєнні та використовується для більш важких задач.

Node.Js це мова яка використовується більше для користувацької частини додатку та за часту працює в парі з PHP, де NodeJs займається фронтовой частиною, а PHP бек частину додатку. В свою чергу PHP був спеціально розроблений для веб-розробки – що і підтверджує його популярність. Також популярність PHP дала старт різним CMS та фрейморк системам, що значно спрощують розробку вебзастосунків на PHP, що робить його оптимальним для розробки.

4.2 Обґрунтування вибору програмної технології для розробки користувальницького інтерфейсу

Керуючись висновками розділу 4.1 було вирішено розробляти дану інформаційну систему на мові програмування PHP, для прискорення та полегшення цього процесу буде використаний програмний фреймворк Yii2.

Програмний фреймворк – це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Відповідно – програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту [22].

Yii2 – це високопродуктивний компонентний PHP-фреймворк, призначений для швидкої розробки сучасних веб-додатків. Слово Yii на китайській мові означає «простий і еволюціонує». Також Yii2 може розшифровуватись акронім «Так, це так!». Для яких завдань найбільше підходить Yii? Yii – це універсальний фреймворк і може бути задіяний у всіх типах веб-додатків. Завдяки його компоненту і відмінною підтримки кешування, фреймворк особливо підходить для розробки таких великих проектів, як портали, форуми, CMS, магазини або RESTful-додатки.

Для організації коду Yii використовує архітектурний паттерн MVC (Model-View-Controller). Yii дотримується філософії простого і елегантного коду, не намагаючись ускладнювати дизайн тільки заради проходження будь-яким шаблонами проектування. Yii є full-stack фреймворком і включає в себе перевірені і добре зарекомендували себе можливості, такі як ActiveRecord для реляційних і NoSQL баз даних, підтримку REST API, багаторівневе кешування і інші. Yii відмінно розширюємо. Ви можете налаштувати або замінити практично будь-яку частину основного коду. Використовуючи архітектуру розширень, легко ділитися кодом або використовувати код спільноти. Одна з головних цілей Yii – продуктивність. Він підтримується і розвивається сильною командою і великим співтовариством розробників, які їй допомагають. Автори фреймворка стежать за тенденціями веб-розробки і розвитком інших проектів. Найбільш підходящі можливості і кращі практики регулярно впроваджуються в фреймворк у вигляді простих і елегантних інтерфейсів [23].

Можливості:

- висока продуктивність;
- паттерн Модель-вид-контролер;
- інтерфейси DAO та Active Record для роботи з базами даних (PDO);
- підтримка інтернаціоналізації;
- кешування сторінок та окремих фрагментів;
- перехоплення та обробка помилок;
- введення та валідація веб-форм;
- автентифікація та авторизація;
- використання AJAX та інтеграція з jQuery;
- генерація базового PHP-коду для CRUD-операцій (скаффолдінг);
- підтримка тем оформлення для їх легкої зміни;
- можливість підключення сторонніх бібліотек;
- міграції бази даних;
- автоматизоване тестування;
- підтримка REST.

Yii це потужний інструмент для розробки веб-застосунків які базуються на мові програмування php, який через свою популярність обріс великою кількістю програмних застосунків які значно полегшують виконання поставленого завдання, тому і був обраний для виконання як інструмент в розробці інформаційної системи.

Також для роботи реалізації інформаційної системи потрібен веб сервер та систему керування базами даних (СКБД).

В ролі веб сервера був вибраний безкоштовний, популярний, перевірений часом та найпоширеніший у світі веб-сервер Apache.

Веб сервер – це сервер, що приймає HTTP-запити від клієнтів, зазвичай веб браузерів, видає їм HTTP-відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. Веб сервер – одна із

основ Всесвітньої павутини. Веб-сервером називають як програмне забезпечення, що виконує функції веб-сервера, так і комп'ютер, на якому це програмне забезпечення працює.

Apache HTTP-сервер – відкритий веб-сервер Інтернет для UNIX-подібних, Microsoft Windows, Novell NetWare та інших операційних систем. Apache розроблюється та підтримується спільнотою розробників відкритого програмного забезпечення під керівництвом Apache Software Foundation.

Web-сервер Apache є самостійним, некомерційним, вільно розповсюджуваним продуктом. Продукт підтримує безліч можливостей, багато з яких реалізовані як скомпільовані модулі, які розширюють основні функціональні можливості. Вони різняться від серверної підтримки мов програмування до схем автентифікації. Існують інтерфейси для підтримки мов програмування Perl, Python, Tcl і PHP. Популярні методи стискування на Apache включають зовнішній модуль mod_gzip, створений для зменшення розміру веб-сторінок, переданих по HTTP.

Функції віртуального хостингу дозволяють одній інсталяції Apache обслуговувати різні веб-сайти. Apache передусім використовується для передачі через HTTP статичних та динамічних веб-сторінок у всесвітній павутині. Багато веб-застосунків спроектовано, зважаючи на середовище і можливості, які надає цей веб-сервер [24].

Для ролі СКБД було обрано MySQL яка добре взаємодіє в тандемі з Apache сервером.

MySQL – вільна система керування реляційними базами даних. MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується,

в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL – компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому [25].

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Також для адміністрування СКБД було обрано phpMyAdmin.

phpMyAdmin – веб-додаток з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування бази даних MySQL або MariaDB. phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних.

Ця програма користується великою популярністю у веб-розробників, оскільки дозволяє керувати базу даних MySQL без вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера під'єднаного до інтернету без необхідності встановлення додаткового програмного забезпечення.

На сьогоднішній день phpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, з огляду на всі нововведення СКБД MySQL [26].

4.3 Розробка модуля інформаційної системи для обробки онлайн-замовлень

Розроблений модуль реалізує наступні функціональні можливості: надає користувацький інтерфейс для замовлень страв та надає можливості адміністрування товарів та замовлень в системі.

Отже, можна виділити декілька основних компоненти розробленої модуля:

– компонент функціоналу для замовлення страв;

– компонент адміністрування замовлень та контенту сервісу;

Продемонструємо життєвий цикл інформаційної системи – запиту на відображення головної сторінки інформаційної системи для функціоналу замовлення страв... При переході на головну сторінку сайту отримуємо запит в екшен index на обробку:

```
public function actionIndex() {
$new = Products::find()->where(['new' => 1])->all();
$popular = Products::find()->where(['popular' => 1])->all();
$hot = Products::find()->where(['id' => [32,31]])->all();
return $this->render('index',['new'=> $new, 'popular' => $popular, 'hot' => $hot]);
}
```

Контролер робить запит в модель Products, яка в свою чергу робить запит до БД для пошуку товарів по заданим умовам. Основною перевагою реалізацією моделі в фрейморку Yii є задіяння паттерну системи ORM Active Record.

Він дозволяє значно спростити роботу з базою даних. Код моделі наведений нижче містить опис прив’язки моделі до таблиці та реалізацію зв’язків між таблицями , інші моделі є аналогічними:

```
<?php
namespace common\models;
class Products extends \yii\db\ActiveRecord{
    public static function tableName()
    {
        return '{{%products}}';
```

```
    }
   public function getOrderItems()
    {
        return $this->hasMany(OrderItem::className(), ['product_id' => 'id']);
    }
public static function getProduct($id)
    {
        return self::findOne(['id' => $id]);
    }

    /**
     * Gets query for [[Orders]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getOrders()
    {
        return $this->hasMany(Order::className(), ['id' => 'order_id'])-
>viaTable('order_item', ['product_id' => 'id']);
}
}
```

Після чого модель отриманні данні з БЦ повертає в контролер, і контролер передає данні на представлення програмний код якого виглядає наступним чином:

```
            <div class="feature_product_inner">
                <div class="main_title">
                    <h2>Нові</h2>
                </div>
                <div class="feature_p_slider owl-carousel">
                    <?php foreach ($new as $item) : ?>
                    <div class="item">
                        <div class="f_p_item">
                            <div class="f_p_img">
                                <img class="img-fluid" src="<?= $item->photos->path ?>"
alt="">
                                <div class="p_icon cart-cart" data-id="<?= $item->id
?>">
                                    <a href="javascript:void(0);"><i class="lnr lnr-cart
" data-id="<?= $item->id ?>"></i></a>
                                </div>
                            </div>
                            <a href="<?= Url::to(['/site/product','id' => $item-
>id]);?>"><h4><?= $item->name ?></h4></a>
                            <h5><?= $item->price ?> грн</h5>
                        </div>
                    </div>
                    <?php endforeach; ?>
```

Результат виконання запиту на відображення головної сторінки (рисунку 4.2).

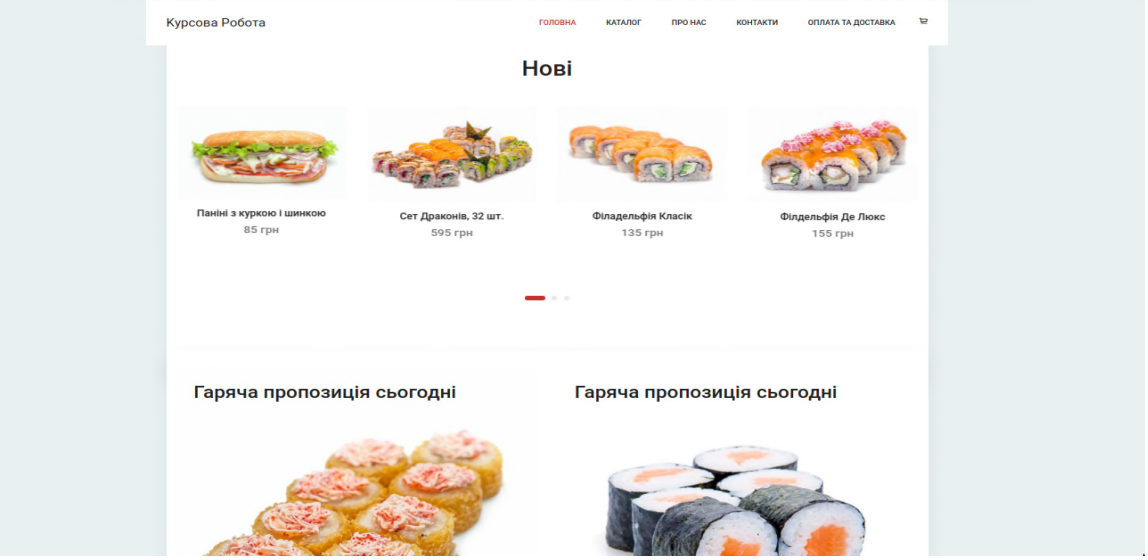

Рисунок 4.2 – Головна сторінка веб сайту

Розглянувши функціонал для замовлення страв, проглянемо ще сторінку адміністрування контенту сервісу життєвий цикл якого працює по аналогії з функціоналом замовлення страв. Розглянемо програмну реалізацію основного розділу для наповнення сервісу – Продуктів.

```
<?php
use yii\helpers\Html;
use yii\grid\GridView;
use yii\widgets\Pjax;
/* @var $this yii\web\View */
/* @var $searchModel backend\models\ProductsSearch */
/* @var $dataProvider yii\data\ActiveDataProvider */

$this->title = 'Продукти';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="products-index">
    <p>
        <?= Html::a('Створити продукт', ['create'], ['class' => 'btn btn-success']) ?>
    </p>

    <?php Pjax::begin(); ?>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'filterModel' => $searchModel,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],
```

```
            'name',
            [
                'attribute' => 'category_id',
                'value' => function ($model) {
                    return $model->category ? $model->category->name : 'Нема';
                },
                'filter'                                                    =>
\yii\helpers\ArrayHelper::map(\common\models\Categories::find()->all(), 'id','name')
            ],
            'description:ntext',
        [
                'attribute' => 'price',
                'value' => function ($model) {
                    return $model->price . ' грн';
                }
            ],
            ['class' => 'yii\grid\ActionColumn'],
        ],
]); ?>
    <?php Pjax::end(); ?>
</div>
```

Yii реалізовує в собі зручні віджети для будування таблиць що значно спрощує реалізцію таких компонетнів. Відображення коду представленного вище зображено на рисунку 4.3.

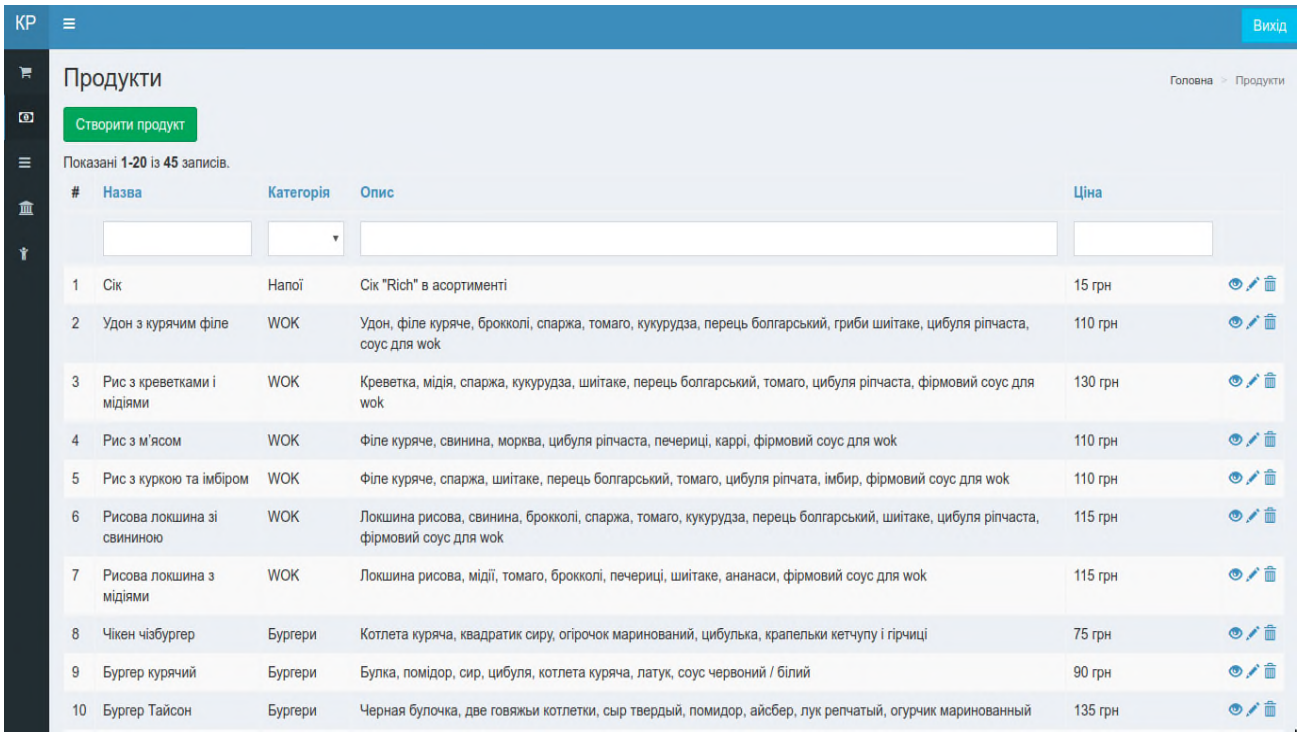

Рисунок 4.3 – Головна сторінка продуктів.

Також на рисунку 4.4 представлена форма створення нового продукту.

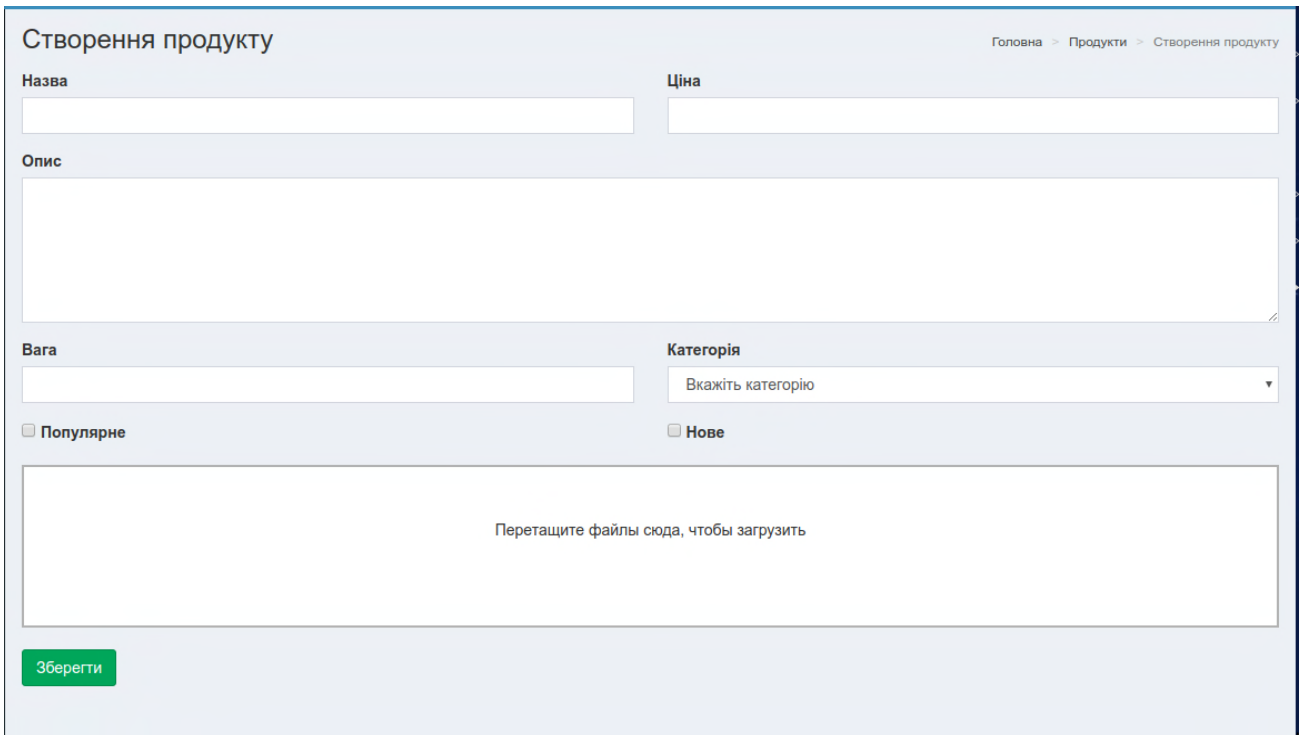

Рисунок 4.4 – Сторінка створення продукту.

4.4 Розробка програмних компонентів та модулів інформаційної системи для оптимізації маршрутів доставки в мережі закладів швидкого харчування

Програма оптимізації маршруту інформаційної системи складається з модулів за якими закріпленні певні функції системи. Модуль роботи з Google API та модуль оптимізації маршруту є основними в інформаційній системі.

Для виконання завдання обрано Yii2 Framework, який повністю відповідає вимогам для проектування структури та реалізації взаємодії з Google API. На рисунку 4.5 зображена діаграма взаємодії модулів інформаційної системи.

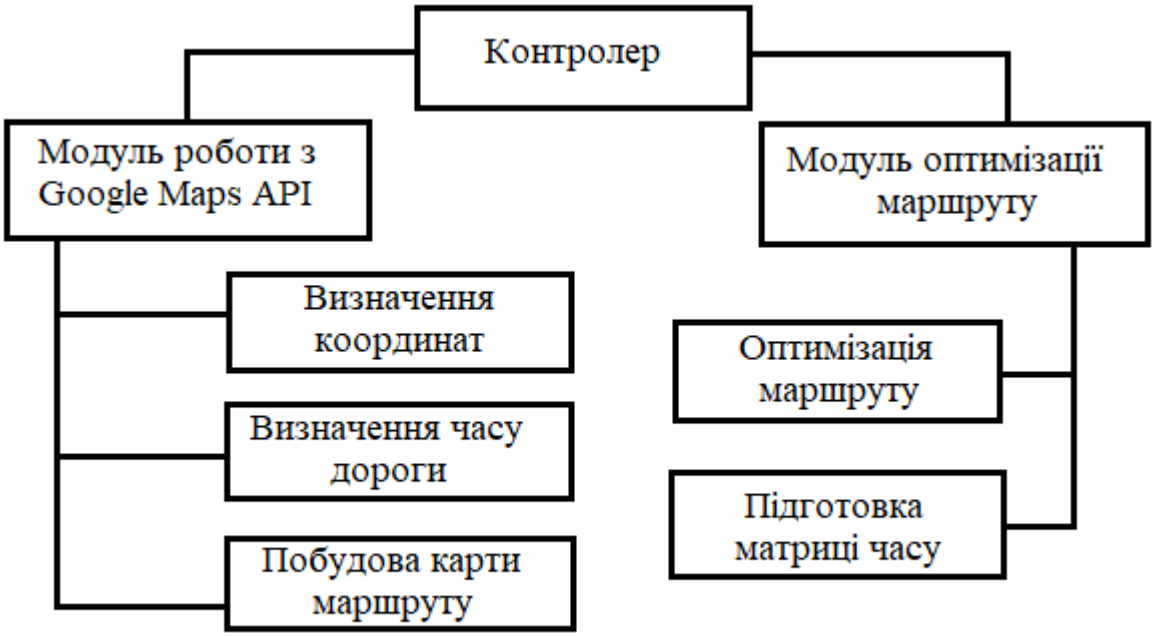

Рисунок 4.5 – Діаграма модулів для оптимізації маршрутів.

Для кур’єра інформаційної системи розроблений відповідний функціонал де можливо подивитись перелік замовлень та переглянути карту маршруту (рисунок 4.6)

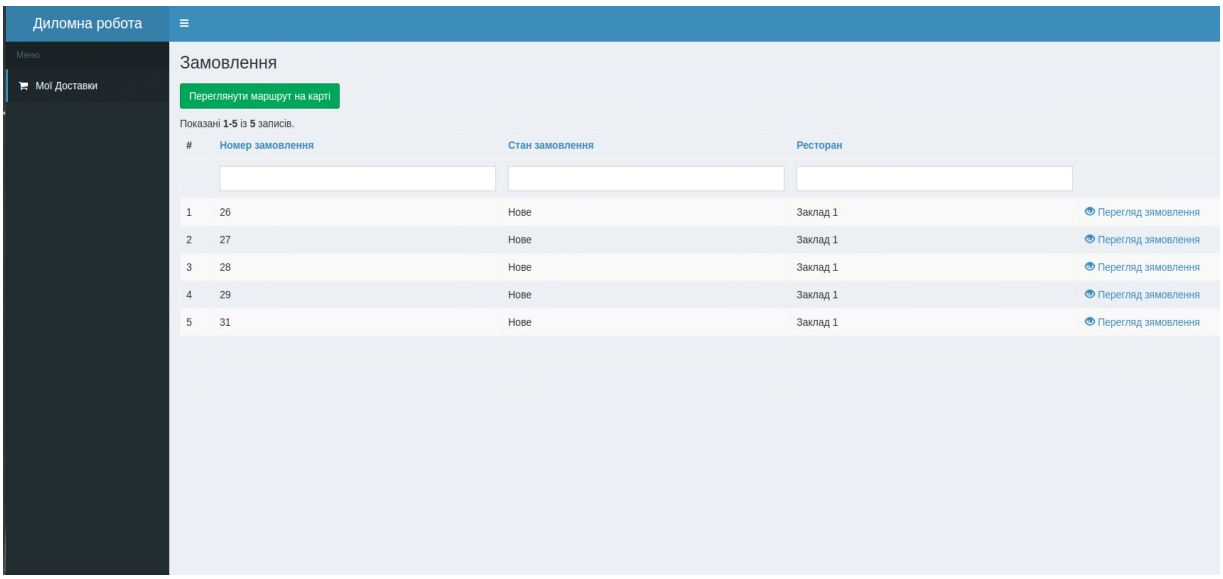

Рисунок 4.6 – Панель перегляду замовлень кур’єра.

Для перегляду оптимізованого маршруту доставки, кур’єру потрібно натиснути на відповідну кнопку в панелі. Після чого буде запущений алгоритм оптимізації маршруту по відповідним замовленням в інформаційній системі.

А саме контролер відправить список адрес замовлень в модуль оптимізації маршруту для побудови матриці часу.

```
        $orders = Order::find()->where(['status' => Order::NEWS])->andWhere(['courier_id'
=> Yii::$app->user->identity->couriers_id])->all();
                    foreach($orders as $key => $order){
                        for ($i = 0 ; $i < count($orders); $i++) {
                            if ($key == $i ) {
                                $array[$key][$i] = "M";
                            } else {
                                $result = $this->getDistance($order->address,
$orders[$i]->address);
                                $array[$key][$i] = $result[0];
                                $coordinates[$key] = $result[1];
                                $address[$key] = $order->address;
                            }

                        }
                    }
```

Метод побудови матриці часу використовує модуль для роботи з Google Maps API для визначення координат точок доставки частина коду якого представлена нижче:

```
        $formattedAddrFrom    = str_replace(' ', '+', $addressFrom);
        $formattedAddrTo     = str_replace(' ', '+', $addressTo);

                $geocodeFrom =
file_get_contents('https://maps.googleapis.com/maps/api/geocode/json?address='.$formatte
dAddrFrom.'&sensor=false&key='.$api);
                $outputFrom = json_decode($geocodeFrom);
                if(!empty($outputFrom->error_message)){
                    return $outputFrom->error_message;
```

```
                }

                $geocodeTo =
file_get_contents('https://maps.googleapis.com/maps/api/geocode/json?address='.$formatte
dAddrTo.'&sensor=false&key='.$api);
                $outputTo = json_decode($geocodeTo);
                if(!empty($outputTo->error_message)){
                    return $outputTo->error_message;
                }
```

Код, що представлений вище форматує адресу з замовлення до вимог Google Maps API та після отримання результату передає отриманні результати для визначення часу проїзду між точками, приклад коду запиту представлений нижче:

```
                $latitudeFrom    = $outputFrom->results[0]->geometry->location->lat;
                $longitudeFrom    = $outputFrom->results[0]->geometry->location->lng;
                $latitudeTo        = $outputTo->results[0]->geometry->location->lat;
                $longitudeTo    = $outputTo->results[0]->geometry->location->lng;

                $responce =
file_get_contents("https://maps.googleapis.com/maps/api/distancematrix/json?origins=$lat
itudeFrom,$longitudeFrom&destinations=$latitudeTo,$longitudeTo&key=$api");
        $result = json_decode($responce);
                return [trim($result ->rows[0]->elements[0]->duration->text),
[$latitudeFrom,$longitudeFrom]];
```

Код представлений вище в результаті своєї роботи відає час проїзду між двома переданими точками. Після чого отримуємо масив з відстанями між точками у вигляді масиву:

```
$array[0] = ["M", 11, 14, 13, 10];
$array[1] = [9, "M", 7, 5, 12];
$array[2] = [11, 10, "M", 7, 13];
$array[3] = [12, 10, 9, "M", 13];
```

```
$array[4] = [8, 15, 10, 10, "M"];
```

Даний масив передається в метод оптимізації маршруту який використовує метод рішення задачі комівояжера, а саме метод найближчого сусіда. Програмна реалізація цього методу представлена нижче:

```
private function optimize($array)
    {
        $key = 0;
        $stopPoint = [];
        $pointValue = [];
        $len = count($array);
        for ($i = 0 ; $i <= count($array); $i++) {
            $stopPoint[] = $key;
            $arrayNow = $array[$key];
            if ($i == $len - 1) {
                $stopPoint[] = 0;
                $pointValue[] = $arrayNow[0];
                return [$stopPoint, $pointValue];
            } else {
                foreach ($stopPoint as $item) {
                    $arrayNow[$item] = 9999;
                }
            }
            $way = array_keys($arrayNow, min($arrayNow));
            $pointValue[] = min($arrayNow);
            $key = $way[0];
        }
    }
```

Після обробки алгоритмом матриці часу, отримуємо результат роботи у вигляді маршруту 0-4-1-2-3-0, де цифри це ключі масиву адрес точок доставки.

Далі контролер спрямовує отриманий маршрут в метод побудови карти. Спочатку створюється карта і проставляються маркери на точки доставки:

```
$map = new Map([
```

```
            'center' => $coord,
            'zoom' => 12,
        ]);
        $map->width = 750;
        $map->height = 750;
        $len = count($ssCoordinates);
        foreach ($ssCoordinates as $key => $item) {
            $waypoints[] = new DirectionsWayPoint(['location' => new LatLng(['lat' =>
$item[0], 'lng' => $item[1]])]);
            if($key != $len) {
                $marker = new Marker([
                    'position' => new LatLng(['lat' => $item[0], 'lng' => $item[1]]),
                ]);
            }
        }
```

Після чого будується маршрут доставки у відповідному порядку по точкам:

```
$directionsRequest = new DirectionsRequest([
    'origin' => $coord,
    'destination' => $coord,
    'waypoints' => $waypoints,
    'travelMode' => TravelMode::DRIVING
]);
$polylineOptions = new PolylineOptions([
    'strokeColor' => '#FFAA00',
]);
$directionsRenderer = new DirectionsRenderer([
    'map' => $map->getName(),
    'polylineOptions' => $polylineOptions
]);
$directionsService = new DirectionsService([
    'directionsRenderer' => $directionsRenderer,
    'directionsRequest' => $directionsRequest
]);
$map->appendScript($directionsService->getJs());
```

Та потім всі маркери та маршрут заносить на карту маршруту, результат роботи – оптимізований маршрут представлений на рисунку 4.7.

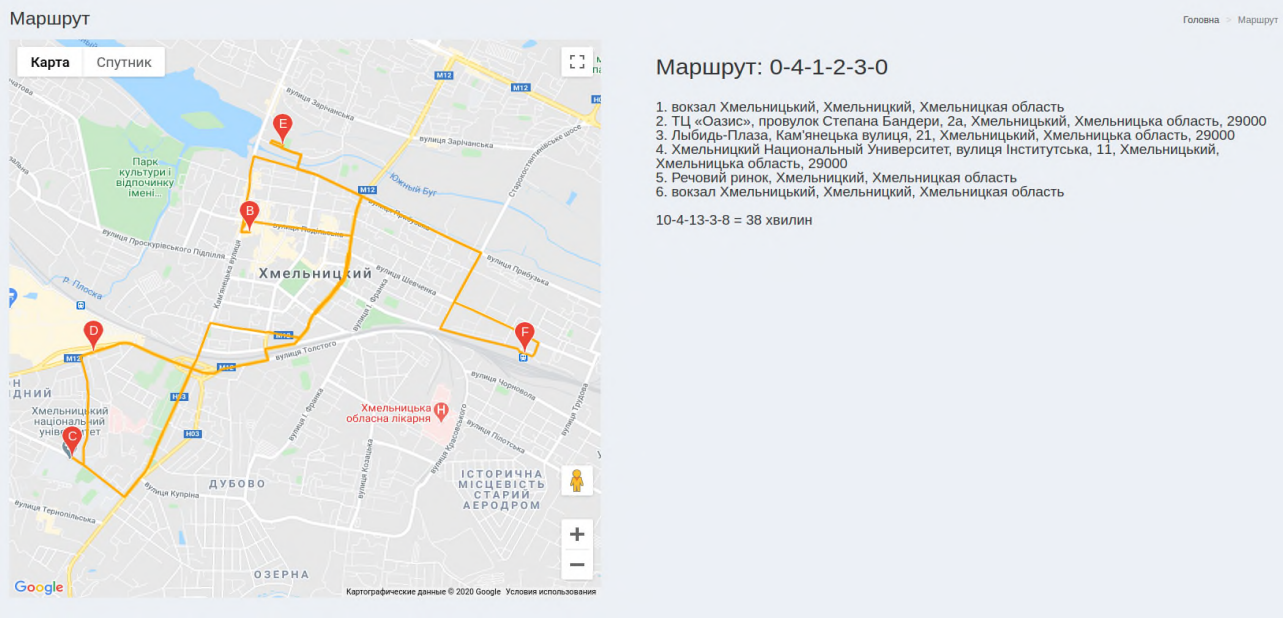

Рисунок 4.7 – Оптимізований маршрут.

Вимоги до програмного забезпечення:

– PHP 7.0 та більш нові;
– Операційна система Windows 7 та більш нові версії або Ubuntu 16.04 та більш нові версії;
– MySQL 5.4;
– APATCH;
– Yii2.

Висновки до розділу 4

Проаналізовано та обґрунтовано вибір мови програмування для реалізації інформаційної системи. Обґрунтовано вибір програмної технології для розробки інформаційної системи.

Розроблений та реалізований користувацький інтерфейс для користувачів та адміністраторів інформаційної системи. Також розроблені і реалізовані модулі для роботи з Google Maps API та оптимізації маршрутів.

В результаті виконання даного розділу було зроблено програмну реалізацію інформаційної системи для оптимізації маршрутів доставки в мережі закладів швидкого харчування.

Висновки

Запропонована інформаційна система призначена для оптимізації маршрутів доставки в мережах закладів швидкого харчування. Інтеграція з API Google Maps дає можливість визначати відстані між точками та візуалізувати отримані маршрути доставки.

Досліджено існуючі сервіси з доставки замовлень та проаналізовано методи оптимізації маршрутів доставки. Досліджена задача комівояжера та методи її розв’язку. Детально розглянуто та проведене порівняння таких методів розв’язку задачі комівояжера: повного перебору, найближчого сусіда та методу гілок і меж.

Розглянуто та проведено порівняння картографічних сервісів. Запропоновано та описано інформаційну систему для оптимізації маршрутів доставки в мережі закладів швидкого харчування.

Спроектовано структуру інформаційної системи, її користувацькі та адміністративні частини, а також модулі для роботи з Google Maps API та оптимізації маршрутів.

Розроблено програмний продукт, який реалізував всі функціоналі вимоги інформаційної системи для оптимізації маршрутів доставки замовлень в мережі закладів швидкого харчування.

Перелік посилань

1. COVID-19 [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Коронавірусна_хвороба_2019
2. Вебзастосунок [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Вебзастосунок
3. Вебсайт [Електронний ресурс]. – Режим доступу: http://www.webtec.com.ua/uk/articles/index/view/2011-05-05/web-site.
4. Що таке «Адаптивний веб-дизайн»? [Електронний ресурс]. – Режим доступу: http://webmark.com.ua/ua/web-site-development/responsive-web-design.html
5. Мобільний застосунок [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Мобільний_застосунок
6. Операційна система [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Операційна_система
7. Android [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Android
8. IOS [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/IOS
9. Wappalyzer – [Електронний ресурс]. – Режим доступу: https://www.wappalyzer.com/technologies/programming-languages;
10. PHP ABOUT – [Електронний ресурс]. – Режим доступу: https://www.php.net/manual/ru/intro-whatis.php
11. PHP – [Електронний ресурс]. – Режим доступу: https://www.php.net
12. JAVA – [Електронний ресурс]. – Режим доступу: https://www.java.com/ru/about/whatis_java.jsp
13. Node.Js – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Node.js

14. API Google Maps – [Електронний ресурс]. – Режим доступу: https://cloud.google.com/maps-platform/
15. Google Maps – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Карти_Google
16. Bing Maps – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Bing_Maps
17. Полный перебор – [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Полный_перебор
18. Distance Matrix API – [Електронний ресурс]. – Режим доступу: https://developers.google.com/maps/documentation/distance-matrix/overview
19. Google Maps JavaScript API– [Електронний ресурс]. – Режим доступу: https://developers.google.com/maps/documentation/javascript/overview?hl=ru
20. Задача комівояжера – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Задача_комівояжера
21. Методи розв’язку задачі комівояжера – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Задача_комівояжера#Методи_розв'язання
22. Програмний фреймворк – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/ wiki/Програмний_каркас;
23. Yii2 – [Електронний ресурс]. – Режим доступу: https://www.yiiframework.com/doc/guide/2.0/ru/intro-yii;
24. APACHE – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Apache_HTTP_Server;
25. APACHE – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Apache_HTTP_Server;
26. PhpMyAdmin – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/PhpMyAdmin;
27. НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ «АПКН-2020» – [Електронний ресурс]. – Режим доступу: https://kn.khnu.km.ua/page.aspx?r=3&p=7;

ДОДАТКИ

Додаток А

Програмні коди

```
<?php

namespace backend\controllers;

use Yii;
use common\models\Order;
use backend\models\OrderSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * OrderController implements the CRUD actions for Order
model.
 */
class OrderController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all Order models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new OrderSearch();
        $dataProvider = $searchModel->search(Yii::$app-
>request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single Order model.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionView($id)
    {
        Yii::$app->formatter->locale = 'ru';
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new Order model.
     * If creation is successful, the browser will be
redirected to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new Order();

        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            return $this->redirect(['view', 'id' => $model-
>id]);
        }

        return $this->render('create', [
            'model' => $model,
        ]);
    }

    /**
     * Updates an existing Order model.
     * If update is successful, the browser will be
redirected to the 'view' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            return $this->redirect(['view', 'id' => $model-
>id]);
        }
```

```
        return $this->render('update', [
            'model' => $model,
        ]);
    }

    /**
     * Updates an existing Order model.
     * If update is successful, the browser will be
redirected to the 'view' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionClose($id)
    {
        $model = $this->findModel($id);
        $model->status = Order::DONE;
        $model->save();
        Yii::$app->session->setFlash('success', 'Статус
оновленно');
        return $this->redirect(['view', 'id' => $model->id]);
    }

    /**
     * Deletes an existing Order model.
     * If deletion is successful, the browser will be
redirected to the 'index' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    /**
     * Finds the Order model based on its primary key value.
     * If the model is not found, a 404 HTTP exception will
be thrown.
     * @param integer $id
     * @return Order the loaded model
     * @throws NotFoundHttpException if the model cannot be
found
     */
    protected function findModel($id)
    {
        if (($model = Order::findOne($id)) !== null) {
            return $model;
        }
        throw new NotFoundHttpException('The requested page
does not exist.');
    }
}
<?php

namespace backend\controllers;

use backend\models\SignupForm;
use common\models\User;
use Yii;
use common\models\Couriers;
use backend\models\CouriersSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * CouriersController implements the CRUD actions for
Couriers model.
 */
class CouriersController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all Couriers models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new CouriersSearch();
        $dataProvider = $searchModel->search(Yii::$app-
>request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single Couriers model.
     * @param integer $id
```

```
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new Couriers model.
     * If creation is successful, the browser will be
redirected to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new Couriers();
        $userForm = new SignupForm();

        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            $userForm->load(Yii::$app->request->post());
            $userForm->signup($model->id);
            return $this->redirect(['view', 'id' => $model-
>id]);
        }

        return $this->render('create', [
            'model' => $model,
            'user' => $userForm,
        ]);
    }

    /**
     * Updates an existing Couriers model.
     * If update is successful, the browser will be
redirected to the 'view' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
        $user = User::findOne(['couriers_id' => $id]);
        $userForm = new SignupForm();
        if(empty($user) === false) {
            $userForm->username = $user->username;
        }

        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            $userForm->load(Yii::$app->request->post());
            $userForm->signup($model->id);
            return $this->redirect(['view', 'id' => $model-
>id]);
        }

        return $this->render('update', [
            'model' => $model,
            'user' => $userForm,
        ]);
    }

    /**
     * Deletes an existing Couriers model.
     * If deletion is successful, the browser will be
redirected to the 'index' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    /**
     * Finds the Couriers model based on its primary key
value.
     * If the model is not found, a 404 HTTP exception will
be thrown.
     * @param integer $id
     * @return Couriers the loaded model
     * @throws NotFoundHttpException if the model cannot be
found
     */
    protected function findModel($id)
    {
        if (($model = Couriers::findOne($id)) !== null) {
            return $model;
        }

        throw new NotFoundHttpException('The requested page
does not exist.');
    }
}
<?php

namespace backend\controllers;

use Yii;
use common\models\Categories;
use backend\models\CategoriesSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
```

```
/**
 * CategoriesController implements the CRUD actions for
Categories model.
 */
class CategoriesController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all Categories models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new CategoriesSearch();
        $dataProvider = $searchModel->search(Yii::$app-
>request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single Categories model.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new Categories model.
     * If creation is successful, the browser will be
redirected to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new Categories();

        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            return $this->redirect(['view', 'id' => $model-
>id]);
        }

        return $this->render('create', [
            'model' => $model,
        ]);
    }

    /**
     * Updates an existing Categories model.
     * If update is successful, the browser will be
redirected to the 'view' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            return $this->redirect(['view', 'id' => $model-
>id]);
        }

        return $this->render('update', [
            'model' => $model,
        ]);
    }

    /**
     * Deletes an existing Categories model.
     * If deletion is successful, the browser will be
redirected to the 'index' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    /**
```

```
     * Finds the Categories model based on its primary key
value.
     * If the model is not found, a 404 HTTP exception will
be thrown.
     * @param integer $id
     * @return Categories the loaded model
     * @throws NotFoundHttpException if the model cannot be
found
     */
    protected function findModel($id)
    {
        if (($model = Categories::findOne($id)) !== null) {
            return $model;
        }

        throw new NotFoundHttpException('The requested page
does not exist.');
    }
}
<?php

namespace backend\controllers;

use common\models\Photos;
use Yii;
use common\models\Products;
use backend\models\ProductsSearch;
use yii\base\ErrorException;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\web\UploadedFile;

/**
 * ProductsController implements the CRUD actions for
Products model.
 */
class ProductsController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all Products models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new ProductsSearch();
        $dataProvider = $searchModel->search(Yii::$app-
>request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single Products model.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new Products model.
     * If creation is successful, the browser will be
redirected to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new Products();

        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            $model->linkPhoto();
            return $this->redirect(['view', 'id' => $model-
>id]);
        }

        return $this->render('create', [
            'model' => $model,
        ]);
    }

    /**
     * Updates an existing Products model.
     * If update is successful, the browser will be
redirected to the 'view' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
```

```
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
        if ($model->load(Yii::$app->request->post()) &&
$model->save()) {
            $model->linkPhoto();
            return $this->redirect(['view', 'id' => $model-
>id]);
        }

        return $this->render('update', [
            'model' => $model,
        ]);
    }

    /**
     * Deletes an existing Products model.
     * If deletion is successful, the browser will be
redirected to the 'index' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be
found
     */
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    /**
     * Finds the Products model based on its primary key
value.
     * If the model is not found, a 404 HTTP exception will
be thrown.
     * @param integer $id
     * @return Products the loaded model
     * @throws NotFoundHttpException if the model cannot be
found
     */
    protected function findModel($id)
    {
        if (($model = Products::findOne($id)) !== null) {
            return $model;
        }

        throw new NotFoundHttpException('The requested page
does not exist.');
    }


    /**
     * Remove file from request.
     **
     * @return mixed
     *
     * @throws \Throwable If file can't delete.
     * @throws \yii\db\StaleObjectException If file can't
delete.
     */
    public function actionFileRemove($id)
    {
        $file = Photos::findOne($id);
        if (false === empty($file)) {
            $this->asJson(['status' => $file->delete()]);
        }
        return true;
    }

    /**
     * Save file and return his id.
     *
     * @return \yii\web\Response
     * @throws ErrorException If file don't save.
     */
    public function actionFileSave()
    {
        $file = UploadedFile::getInstanceByName('file');
        if (false === empty($file)) {
            return $this->asJson(['id' =>
Photos::saveFile($file)]);
        } else {
            throw new ErrorException('File is not upload!');
        }
    }
}
<?php

namespace backend\controllers;

use Yii;
use common\models\Restaurants;
use backend\models\RestaurantsSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * RestaurantsController implements the CRUD actions for
Restaurants model.
 */
class RestaurantsController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
```

```
                    'delete' => ['POST'],
                ],
            ],
        ];
    }

    /**
     * Lists all Restaurants models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new RestaurantsSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single Restaurants model.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be found
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

    /**
     * Creates a new Restaurants model.
     * If creation is successful, the browser will be redirected to the 'view' page.
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new Restaurants();

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('create', [
            'model' => $model,
        ]);
    }

    /**
     * Updates an existing Restaurants model.
     * If update is successful, the browser will be redirected to the 'view' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be found
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('update', [
            'model' => $model,
        ]);
    }

    /**
     * Deletes an existing Restaurants model.
     * If deletion is successful, the browser will be redirected to the 'index' page.
     * @param integer $id
     * @return mixed
     * @throws NotFoundHttpException if the model cannot be found
     */
    public function actionDelete($id)
    {
        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    /**
     * Finds the Restaurants model based on its primary key value.
     * If the model is not found, a 404 HTTP exception will be thrown.
     * @param integer $id
     * @return Restaurants the loaded model
     * @throws NotFoundHttpException if the model cannot be found
     */
    protected function findModel($id)
    {
        if (($model = Restaurants::findOne($id)) !== null) {
            return $model;
        }
```

```
        throw new NotFoundHttpException('The requested page
does not exist.');
    }
}
<?php
namespace frontend\controllers;

use common\models\Categories;
use common\models\Couriers;
use common\models\Order;
use common\models\OrderItem;
use common\models\Products;
use common\models\Restaurants;
use common\models\User;
use Yii;
use yii\data\Pagination;
use yii\web\Controller;

/**
 * Site controller
 */
class SiteController extends Controller
{

    /**
     * {@inheritdoc}
     */
    public function actions()
    {
        return [
            'error' => [
                'class' => 'yii\web\ErrorAction',
            ],
            'captcha' => [
                'class' => 'yii\captcha\CaptchaAction',
                'fixedVerifyCode' => YII_ENV_TEST ? 'testme'
: null,
            ],
        ];
    }

    /**
     * Displays homepage.
     *
     * @return mixed
     */
    public function actionIndex()
    {
        $new = Products::find()->where(['new' => 1])->all();
        $popular = Products::find()->where(['popular' => 1])-
>all();
        $hot = Products::find()->where(['id' => [32,31]])-
>all();
        return $this->render('index',['new'=> $new, 'popular'
=> $popular, 'hot' => $hot]);
    }

    /**
     * Displays contact page.
     *
     * @return mixed
     */
    public function actionContact()
    {
        return $this->render('contact');

    }

    /**
     * Displays about page.
     *
     * @return mixed
     */
    public function actionAbout()
    {
        return $this->render('about');
    }

    /**
     * Displays about page.
     *
     * @return mixed
     */
    public function actionDelivery()
    {
        return $this->render('delivery');
    }

    /**
     * Displays about page.
     *
     * @param integer $category
     * @param mixed $sort
     *
     * @return mixed
     */
    public function actionCatalog($category = null, $sort =
null)
    {
        $query = Products::find();
        if (empty($category) === false) {
            $query->where(['category_id' => $category]);
        }
        if (empty($sort) === false) {
            $query->andWhere([$sort => true]);
        }
        $countQuery = clone $query;
        $pages = new Pagination(['totalCount' => $countQuery-
>count(),'pageSize' => 9]);
        $products = $query->offset($pages->offset)
            ->limit($pages->limit)
            ->all();
```

```
        $categories = Categories::find()->all();
        return $this->render('catalog',['products' =>
$products, 'categories' => $categories, 'pages' => $pages,]);
    }

    /**
     * Displays about page.
     *
     * @param integer $id
     *
     * @return mixed
     */
    public function actionProduct($id)
    {
        $product = Products::find()->where(['id' => $id])-
>one();

        return $this->render('product',['product' =>
$product]);
    }

    /**
     * Add on cart.
     *
     *
     * @return mixed
     */
    public function actionAdd()
    {
        $data = Yii::$app->request->post();
        $session = Yii::$app->session;
        $id = $data['id'];
        $count = isset($data['count']) ? $data['count'] : 1;

        $cart = $session->get('cart');
        if(isset($cart) === true) {
            $products = json_decode($cart, true);
            if(isset($products[$id]) === true) {
                $products[$id] = $products[$id] += $count;
//                dd($products);
                $session->set('cart',json_encode($products));
            } else {
                $products[$id] = $count;
                $session->set('cart',json_encode($products));
            }

        } else {
            $session->set('cart',json_encode([$id =>
$count]));
        }
    }

    /**
     * Change cart.
     *
     *
     * @return mixed
     */
    public function actionChange()
    {
        $data = Yii::$app->request->post();
        $session = Yii::$app->session;
        $id = $data['id'];
        $count = $data['count'];
        $cart = $session->get('cart');
        if(isset($cart) === true) {
            $products = json_decode($cart, true);

            if($count < 1) {
                unset($products[$id]);
                $session->set('cart',json_encode($products));
            } else {
                $products[$id] = $count;
                $session->set('cart',json_encode($products));
            }

        }
    }
    /**
     * Requests password reset.
     *
     * @return mixed
     */
    public function actionCart()
    {
        $session = Yii::$app->session;
        $cart = $session->get('cart');
        $model = new Order();

        if ($model->load(Yii::$app->request->post())) {
            $model->courier_id = rand(Couriers::find()-
>min('id'),Couriers::find()->max('id'));
            $model->restaurant_id = rand(Restaurants::find()-
>min('id'),Restaurants::find()->max('id'));
            $model->save();
            $products = json_decode($cart, true);
            if (empty($products) === false) {
                foreach ($products as $product => $count) {
                    $item = new OrderItem();
                    $item->product_id = $product;
                    $item->count = $count;
                    $item->order_id = $model->id;
                    $item->save();
                }
                $session->destroy();
                return $this->render('order', ['id' =>
$model->id]);
            }
            return $this->redirect('/');

        }
        return $this->render('cart',
            [
```

```
                'cart' => json_decode($cart, true),
                'model' => $model,
            ]
        );
    }

}
<?php

namespace common\models;

use Yii;

/**
 * This is the model class for table "categories".
 *
 * @property int $id
 * @property string|null $name
 *
 * @property Products[] $products
 */
class Categories extends \yii\db\ActiveRecord
{
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return '{{%categories}}';
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['name'], 'string', 'max' => 255],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'name' => 'Назва',
        ];
    }

    /**
     * Gets query for [[Products]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getProducts()
    {
        return $this->hasMany(Products::className(),
['category_id' => 'id']);
    }


    public static function getName($id)
    {
        $model = self::findOne(['id' => $id]);
        return $model->name;
    }
}
<?php

namespace common\models;

use Yii;
use yii\behaviors\TimestampBehavior;

/**
 * This is the model class for table "order".
 *
 * @property int $id
 * @property int|null $price_total
 * @property int|null $restaurant_id
 * @property int|null $courier_id
 * @property int|null $status
 * @property string $address
 * @property string $phone
 * @property int $created_at
 * @property int $updated_at
 *
 * @property Couriers $courier
 * @property Restaurants $restaurant
 * @property OrderItem[] $orderItems
 * @property Products[] $products
 */
class Order extends \yii\db\ActiveRecord
{


    const NEWS = 1;
    const DONE = 2;

    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'order';
    }

    /**
     * @return array
     */
    public function behaviors()
```

```
    {
        return [
            TimestampBehavior::class,
        ];
    }

    /**
     * {@inheritdoc}
     */
    public static function statuses()
    {
        return [
            self::NEWS => 'Нове',
            self::DONE => 'Виконане'
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['address','phone'],'required'],
            [['price_total', 'restaurant_id',
'courier_id','status','created_at','updated_at'], 'integer'],
            [['status'], 'default', 'value' => self::NEWS ],
            [['courier_id'], 'exist', 'skipOnError' => true,
'targetClass' => Couriers::className(), 'targetAttribute' =>
['courier_id' => 'id']],
            [['restaurant_id'], 'exist', 'skipOnError' =>
true, 'targetClass' => Restaurants::className(),
'targetAttribute' => ['restaurant_id' => 'id']],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'id' => 'Номер замовлення',
            'price_total' => 'Всього',
            'restaurant_id' => 'Ресторан',
            'courier_id' => 'Кур\'єр',
            'status' => 'Стан замовлення',
            'address'=> 'Адреса доставки',
            'phone'=> 'Телефон',
        ];
    }

    /**
     * Gets query for [[Courier]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getCourier()
    {
        return $this->hasOne(Couriers::className(), ['id' =>
'courier_id']);
    }

    /**
     * Gets query for [[Restaurant]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getRestaurant()
    {
        return $this->hasOne(Restaurants::className(), ['id'
=> 'restaurant_id']);
    }

    /**
     * Gets query for [[OrderItems]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getOrderItems()
    {
        return $this->hasMany(OrderItem::className(),
['order_id' => 'id']);
    }

    /**
     * Gets query for [[Products]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getProducts()
    {
        return $this->hasMany(Products::className(), ['id' =>
'product_id'])->viaTable('order_item', ['order_id' => 'id']);
    }
}
<?php

namespace common\models;

use Yii;

/**
 * This is the model class for table "order_item".
 *
 * @property int $order_id
 * @property int $product_id
 * @property int|null $count
 *
 * @property Order $order
 * @property Products $product
 */
class OrderItem extends \yii\db\ActiveRecord
{
```

```
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'order_item';
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['order_id', 'product_id'], 'required'],
            [['order_id', 'product_id', 'count'], 'integer'],
            [['order_id', 'product_id'], 'unique',
'targetAttribute' => ['order_id', 'product_id']],
            [['order_id'], 'exist', 'skipOnError' => true,
'targetClass' => Order::className(), 'targetAttribute' =>
['order_id' => 'id']],
            [['product_id'], 'exist', 'skipOnError' => true,
'targetClass' => Products::className(), 'targetAttribute' =>
['product_id' => 'id']],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'order_id' => 'Order ID',
            'product_id' => 'Product ID',
            'count' => 'Count',
        ];
    }

    /**
     * Gets query for [[Order]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getOrder()
    {
        return $this->hasOne(Order::className(), ['id' =>
'order_id']);
    }

    /**
     * Gets query for [[Product]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getProduct()
    {
        return $this->hasOne(Products::className(), ['id' =>
'product_id']);
    }
}
<?php

namespace common\models;

use Yii;

/**
 * This is the model class for table "products".
 *
 * @property int $id
 * @property string|null $name
 * @property int|null $price
 * @property string|null $description
 * @property string|null $weight
 * @property string|null $structure
 * @property int|null $category_id
 * @property int|null $popular
 * @property int|null $new
 * @property int|null $discounts
 *
 * @property OrderItem[] $orderItems
 * @property Order[] $orders
 * @property Photos[] $photos
 * @property Categories $category
 */
class Products extends \yii\db\ActiveRecord
{
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return '{{%products}}';
    }

    /**
     * Images array.
     *
     * @var $photoArray
     */
    public $photoArray;

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['price', 'category_id', 'popular', 'new',
'discounts'], 'integer'],
            [['description', 'structure'], 'string'],
            [
                'photoArray',
```

```
                'safe',
            ],
            [['name', 'weight'], 'string', 'max' => 255],
            [['category_id'], 'exist', 'skipOnError' => true,
'targetClass' => Categories::className(), 'targetAttribute'
=> ['category_id' => 'id']],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'name' => 'Назва',
            'price' => 'Ціна',
            'description' => 'Опис',
            'weight' => 'Вага',
            'structure' => 'Склад',
            'category_id' => 'Категорія',
            'popular' => 'Популярне',
            'new' => 'Нове',
            'discounts' => 'Знижка',
        ];
    }

    /**
     * Gets query for [[OrderItems]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getOrderItems()
    {
        return $this->hasMany(OrderItem::className(),
['product_id' => 'id']);
    }

    /**
     * Get product.
     *
     * @param int $id
     *
     * @return self
     */
    public static function getProduct($id)
    {
        return self::findOne(['id' => $id]);
    }

    /**
     * Gets query for [[Orders]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getOrders()
    {
        return $this->hasMany(Order::className(), ['id' =>
'order_id'])->viaTable('order_item', ['product_id' => 'id']);
    }

    /**
     * Gets query for [[Photos]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getPhotos()
    {
        return $this->hasOne(Photos::className(),
['product_id' => 'id']);
    }

    /**
     * Gets query for [[Category]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getCategory()
    {
        return $this->hasOne(Categories::className(), ['id'
=> 'category_id']);
    }

    /**
     * Get array with data of file for dropzone.
     *
     * @return array
     */
    public function getFileForDropZone()
    {
        $data = [];
        if (true === empty($this->photos)) {
            return $data;
        }

            $data[] = [
                'id' => $this->photos->id,
                'thumbnail' => $this->photos->path,
                'size' =>
@filesize(\Yii::getAlias('@webroot') . $this->photos->path),
                'name' => $this->photos->path,
            ];

        return $data;
    }

    /**
     * Get array with data of file for dropzone.
     *
     * @return array
     */
    public function linkPhoto()
    {
```

```
        $photo = json_decode($this->photoArray);
        if(!empty($photo)) {
            foreach ($photo as $item) {
                $file_gallery = Photos::findOne(['id' =>
$item]);
                $file_gallery->product_id = $this->id;
                $file_gallery->save();
            }
        }

    }
}
<?php
use yii\helpers\Url;
$this->title = 'Про нас';
?>
<section class="banner_area">
    <div class="banner_inner d-flex align-items-center">
        <div class="container">
            <div class="banner_content text-center">
                <h2>Про нас</h2>
                <div class="page_link">
                    <a href="<?=
Url::to(['/']);?>">Головна</a>
                    <a href="javascript:void(0);">Про нас</a>
                </div>
            </div>
        </div>
    </div>
</section>
<section class="sample-text-area main_box">
    <div class="container">
        <h3 class="text-heading title_color">Інформаційна
система доставки онлайн замовлень в мережі закладів швидкого
харчування</h3>
        <p class="sample-text">
            <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Morbi eget ex sem.
            Nullam a ultrices ligula, sit amet fermentum
risus. Proin erat nisl,
            tempor in orci quis, aliquam hendrerit risus.
Maecenas pulvinar est mi,
            ut elementum sapien dapibus eu. Class aptent
taciti sociosqu ad litora torquent per conubia nostra,
            per inceptos himenaeos. Morbi aliquam tristique
elit, posuere dictum ex consequat sit amet.
            Quisque commodo lectus sed vulputate tincidunt.
Proin pharetra ex nec sapien convallis mattis.
            Mauris interdum est eget turpis pharetra, et
mollis dolor efficitur. Maecenas nec placerat enim,
            eget faucibus arcu. Sed tincidunt condimentum
dignissim. Morbi varius lacus dui, ut rutrum ante cursus
eget.</p>

        <p>Integer accumsan convallis dui. Sed vitae est ex.
Proin consequat maximus dolor,
            nec auctor lectus molestie quis. Praesent sem
nisl, mollis eu accumsan a, tempor at risus.
            Integer rutrum sem eu diam condimentum, id
lobortis justo hendrerit. Orci varius natoque penatibus
            et magnis dis parturient montes, nascetur
ridiculus mus. Mauris accumsan erat eu vestibulum luctus.
            Etiam rhoncus est purus, quis suscipit arcu
volutpat vel. In ac dolor quam. Vestibulum sit
            amet augue pretium, pellentesque metus eget,
facilisis dui. Sed ut lacus faucibus, viverra nulla nec,
            tempor est. Morbi faucibus enim cursus lectus
pellentesque sagittis. Duis accumsan dui eget dolor
efficitur,
            et porta urna eleifend. Aenean fermentum justo
non nunc blandit volutpat. Fusce varius quam sit
            amet arcu eleifend, ac blandit enim facilisis.
Pellentesque mauris risus, dignissim at convallis eget,
            posuere eu lectus.</p>

        <p>Morbi ligula nisl, feugiat vitae massa facilisis,
pretium semper metus. Mauris venenatis felis est.
            Nullam mollis lacus metus, id consequat diam
eleifend non. Nunc varius scelerisque mi nec ornare.
            Morbi eu consectetur tellus, a egestas lacus.
Aliquam ultricies velit non est finibus,
            sit amet malesuada erat tempor. Aenean vel cursus
velit. In in lorem sit amet tellus faucibus ultricies.</p>
        </p>
    </div>
</section>
<?php
use yii\helpers\Url;
?>
<div class="banner_inner d-flex align-items-center">

</div>

<!--================Latest Product Area =================-->
<section class="feature_product_area latest_product_area">
    <div class="main_box">
        <div class="container">
            <div class="feature_product_inner">
                <div class="main_title">
                    <h2>Нові</h2>
                </div>
                <div class="feature_p_slider owl-carousel">
                    <?php foreach ($new as $item) : ?>
                    <div class="item">
                        <div class="f_p_item">
                            <div class="f_p_img">
                                <img class="img-fluid"
src="<?= $item->photos->path ?>" alt="">
                                <div class="p_icon cart-cart"
data-id="<?= $item->id ?>">
                                    <a
href="javascript:void(0);"><i class="lnr lnr-cart " data-
id="<?= $item->id ?>"></i></a>
```

```
                            </div>
                        </div>
                        <a href="<?=
Url::to(['/site/product','id' => $item->id]);?>"><h4><?=
$item->name ?></h4></a>
                        <h5><?= $item->price ?> грн</h5>
                    </div>
                </div>
                <?php endforeach; ?>
            </div>
        </div>
    </div>
</section>
<!--================End Latest Product Area
=================-->
<section class="feature_product_area">
    <div class="main_box">
        <div class="container">
            <div class="row hot_product_inner">
                <?php foreach ($hot as $item) : ?>
                <div class="col-lg-6">
                    <div class="hot_p_item">
                        <img class="img-fluid" src="<?=
$item->photos->path ?>" alt="">
                        <div class="product_text">
                            <h4>Гаряча пропозиція
сьогодні</h4>
                            <a  class="cart-
cart"href="javascript:void(0);" data-id="<?= $item->id
?>">Купити зараз</a>
                        </div>
                    </div>
                </div>
                <?php endforeach; ?>
            </div>
            <div class="feature_product_inner">
                <div class="main_title">
                    <h2>Популярне</h2>
                </div>
                <div class="feature_p_slider owl-carousel">
                    <?php foreach ($popular as $item) : ?>
                        <div class="item">
                            <div class="f_p_item">
                                <div class="f_p_img">
                                    <img class="img-fluid"
src="<?= $item->photos->path ?>" alt="">
                                    <div class="p_icon cart-
cart" data-id="<?= $item->id ?>">
                                        <a
href="javascript:void(0);"><i class="lnr lnr-cart"></i></a>
                                    </div>
                                </div>
                                <a href="<?=
Url::to(['/site/product','id' => $item->id]);?>"><h4><?=
$item->name ?></h4></a>
                                <h5><?= $item->price ?>
грн</h5>
                            </div>
                        </div>
                    <?php endforeach; ?>
                </div>
            </div>
        </div>
    </div>
</section>
<?php

use yii\helpers\Url;
$css = <<< CSS
.product_image_area {
     padding-top: 0px;
}
CSS;
$this->registerCss($css);
?>
<!--================Home Banner Area =================-->
<section class="banner_area">
    <div class="banner_inner d-flex align-items-center">
        <div class="container">
            <div class="banner_content text-center">
                <h2><?= $product->name ?></h2>
                <div class="page_link">
                    <a href="<?=
Url::to(['/']);?>">Головна</a>
                    <a href="<?=
Url::to(['/site/catalog','category' => $product->category_id
]);?>"><?= $product->category->name ?></a>
                    <a href="javascript:void(0);"><?=
$product->name ?></a>
                </div>
            </div>
        </div>
    </div>
</section>
<div class="product_image_area main_box">
    <div class="container">
        <div class="row s_product_inner">
            <div class="col-lg-6">
                <div class="s_product_img">
                    <div id="carouselExampleIndicators"
class="carousel slide" data-ride="carousel">
                        <div class="carousel-inner">
                            <div class="carousel-item
active">
                                <img class="d-block w-100"
src="<?= $product->photos->path ?>" alt="First slide">
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-lg-5 offset-lg-1">
```

```
                <div class="s_product_text">
                    <h3><?= $product->name ?></h3>
                    <h2><?= $product->price ?> грн</h2>
                    <ul class="list">
                        <li><a class="active" href="<?=
Url::to(['/site/catalog','category' => $product->category_id
]);?>"><span>Категорія</span> : <?= $product->category->name
?></a></li>
                        <li><a href="#"><span>Вага</span> :
<?= $product->price ?> грам</a></li>
                    </ul>
                    <p><?= $product->description ?></p>
                    <div class="product_count">
                        <label for="qty">Кількість:</label>
                        <input type="text" name="qty"
id="sst" maxlength="12" value="1" title="Quantity:"
class="input-text qty">
                        <button onclick="var result =
document.getElementById('sst'); var sst = result.value; if(
!isNaN( sst )) result.value++;return false;" class="increase
items-count" type="button"><i class="lnr lnr-chevron-
up"></i></button>
                        <button onclick="var result =
document.getElementById('sst'); var sst = result.value; if(
!isNaN( sst ) &amp;&amp; sst > 0 ) result.value--;return
false;" class="reduced items-count" type="button"><i
class="lnr lnr-chevron-down"></i></button>
                    </div>
                    <div class="card_area">
                        <a class="main_btn cart-cart"
href="javascript:void(0);" data-id="<?= $product->id ?>">У
кошик</a>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<?php
use yii\helpers\Url;
?>
<div class="banner_inner d-flex align-items-center">

</div>

<!--================Latest Product Area =================-->
<section class="feature_product_area latest_product_area">
    <div class="main_box">
        <div class="container">
            <div class="feature_product_inner">
                <div class="main_title">
                    <h2>Нові</h2>
                </div>
                <div class="feature_p_slider owl-carousel">
                    <?php foreach ($new as $item) : ?>
                    <div class="item">
                        <div class="f_p_item">
                            <div class="f_p_img">
                                <img class="img-fluid"
src="<?= $item->photos->path ?>" alt="">
                                <div class="p_icon cart-cart"
data-id="<?= $item->id ?>">
                                    <a
href="javascript:void(0);"><i class="lnr lnr-cart " data-
id="<?= $item->id ?>"></i></a>
                                </div>
                            </div>
                            <a href="<?=
Url::to(['/site/product','id' => $item->id]);?>"><h4><?=
$item->name ?></h4></a>
                            <h5><?= $item->price ?> грн</h5>
                        </div>
                    </div>
                    <?php endforeach; ?>
                </div>
            </div>
        </div>
    </div>
</section>
<!--================End Latest Product Area
=================-->
<section class="feature_product_area">
    <div class="main_box">
        <div class="container">
            <div class="row hot_product_inner">
                <?php foreach ($hot as $item) : ?>
                <div class="col-lg-6">
                    <div class="hot_p_item">
                        <img class="img-fluid" src="<?=
$item->photos->path ?>" alt="">
                        <div class="product_text">
                            <h4>Гаряча пропозиція
сьогодні</h4>
                            <a  class="cart-
cart"href="javascript:void(0);" data-id="<?= $item->id
?>">Купити зараз</a>
                        </div>
                    </div>
                </div>
                <?php endforeach; ?>
            </div>
            <div class="feature_product_inner">
                <div class="main_title">
                    <h2>Популярне</h2>
                </div>
                <div class="feature_p_slider owl-carousel">
                    <?php foreach ($popular as $item) : ?>
                    <div class="item">
                        <div class="f_p_item">
                            <div class="f_p_img">
                                <img class="img-fluid"
src="<?= $item->photos->path ?>" alt="">
                                <div class="p_icon cart-
cart" data-id="<?= $item->id ?>">
```

```
                                    <a
href="javascript:void(0);"><i class="lnr lnr-cart"></i></a>
                                </div>
                            </div>
                            <a href="<?=
Url::to(['/site/product','id' => $item->id]);?>"><h4><?=
$item->name ?></h4></a>
                            <h5><?= $item->price ?>
грн</h5>
                        </div>
                    </div>
                <?php endforeach; ?>
            </div>
        </div>
    </div>
</section>
```

Додаток Б

Ксерокопії наукових публікацій, виконаних при роботі над дипломною роботою магістра

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XII всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2020»

9-10 листопада 2020

Хмельницький 2020

УДК 004:37:001:62

Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020». Хмельницький – 2020. – 365с.

У збірнику наукових праць подані перспективні практичні розробки аспірантів, студентів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, комп'ютерної інженерії, прикладної математики й інженерії програмного забезпечення, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки сучасних систем пошуку, обробки й захисту інформації, медійних та комунікаційних системи.

УДК 004:37:001:62

Матеріали конференції відтворені з авторських оригіналів. Оргкомітет конференції висловлює подяку учасникам конференції та сподівається на подальшу співпрацю.

З питань проведення конференції та подальшого обміну інформацією звертатись на e-mail конференції: *apkt.khnu@gmail.com*

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК - 2020

XII Всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

СЕКЦІЇ КОНФЕРЕНЦІЇ:

1. Комп'ютерні науки та прикладні інформаційні технології.
2. Комп'ютерна інженерія та системи захисту інформації.
3. Математичне моделювання та інженерія програмного забезпечення
4. Телерадіокомунікації, медійні та комунікаційні системи.
5. Проблеми впровадження інформаційних технологій у виробництво та управління.

Робочі мови конференції: українська, англійська

ОРГКОМІТЕТ:

СИНЮК О. М. голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор

СОРОКАТИЙ Р. В. заступник голови оргкомітету, завідувач кафедри Комп'ютерних наук та інформаційних технологій ХНУ, доктор технічних наук, професор

БАРМАК О. В. заступник голови оргкомітету, доктор технічних наук, професор

САВЕНКО О. С. декан Факультету програмування та комп'ютерних і телекомунікаційних систем ХНУ, доктор технічних наук, професор

ВИСОЦЬКА О. В. доктор технічних наук, завідувач кафедри радіоелектронних та біомедичних комп'ютеризованих засобів і технологій Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», професор

ЛАВРОВ Є. А. доктор технічних наук, професор (Сумський державний університет)

ТІМОФЄЄВА Л. В. відповідальна за студентську науково-дослідну роботу ХНУ

МАЗУРЕЦЬ О. В. секретар конференції, старший викладач кафедри комп'ютерних наук та інформаційних технологій ХНУ

КОНТАКТНА ІНФОРМАЦІЯ:

e-mail для листування: *apkt.khnu@gmail.com*

УДК 004.02

Ткачук Є. А., Багрій Р. О., Скрипник Т. К.

Хмельницький національний університет

МЕТОДИ ОПТИМІЗАЦІЇ ДОСТАВКИ ЗАМОВЛЕНЬ

Розглянуто основні аспекти розробки інформаційної системи для оптимізації маршрутів доставки замовлень, які сприятимуть зменшенню кількості витрачених ресурсів та часу на доставку. Проведений огляд та порівняння методів розв'язку задачі комівояжера: повного перебору, найближчого сусіда та метода гілок і меж. Запропонована інформаційна система для оптимізації доставки замовлень, яка генерує оптимальний маршрут доставки.

The main aspects of information system development for optimization of order delivery routes are considered, which will help to reduce the amount of resources spent and delivery time. A review and comparison of methods for solving the problem of the salesman: a complete search, the nearest neighbor and the method of branches and boundaries. An information system for optimizing the delivery of orders that generates the optimal delivery route is proposed.

У житті сучасних підприємств різного роду діяльності значне місце займають транспортні задачі, які під час пандемії COVID-19 набули не аби якої актуальності. Для кожної компанії є важливим питання про своєчасну доставку товару споживачам в найкоротші терміни. Для здійснення цього керівництво компанії, по суті, займається вирішенням так званої задачі комівояжера. Оптимальні маршрути дозволяють компаніям не тільки пришвидшити виконання доставки, але й заощаджують значну кількість ресурсів, які компанії можуть задіяти в покращенні інших напрямках сервісу.

Метою дослідження є розробка інформаційної системи оптимізації маршрутів доставки на основі методів розв'язання задачі комівояжера.

У зв'язку з актуальністю проблеми під час пандемії, значно зросла кількість запитів на доставку того чи іншого товару, такі запити можна об'єднати в загальний клас «транспортних задач». У цей розділ входить величезне різноманіття різних завдань, які пов'язані тим, що цільова функція в них носить той чи інший економічний сенс [1]. Так, наприклад, класична транспортна задача [1] полягає в доставці товару декількома транспортними засобами до певної кількості споживачів. Особливе місце в класі транспортних завдань займає задача комівояжера [2], яка полягає в знаходженні самого вигідного маршруту, що проходить через задані пункти.

Принцип роботи інформаційної системи (Рисунок 1) складається в формуванні списку замовлень закладу, отримання матриці часу доставки за допомогою API Google Maps [7] та побудові оптимального маршруту доставки за допомогою методів оптимізації.

Рисунок 1 – Інформаційна системи

Для визначення оптимального маршруту доставки розглянемо три методи рішення задачі комівояжера: повного перебору, найближчого сусіда та метода гілок і меж. Після аналізу яких буде виявлено оптимальний метод який буде задіяний в інформаційній системі.

Вхідні дані (Таблиця 1) для проведення аналізу це відомі точки міста Хмельницького, час дороги від точки до точки отриманий за допомогою сервісу GoogleMaps [3] та зазначений в хвилинах. В ролі початкової точки було вибрано «ЖД Вокзал».

Таблиця 1 – Вхідні данні

Назва	**№**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ЖД Вокзал	1	M	10	11	12	9	8	5	14	5	10
ТЦ «Либідь Плаза»	2	8	M	6	6	4	13	6	13	10	9
Речовий Ринок	3	7	7	M	4	9	14	6	15	11	7
ХНУ	4	10	10	3	M	12	18	9	19	14	6
ТЦ «Оазис»	5	8	4	9	9	M	11	7	12	8	12
Центральний автовокзал	6	8	11	15	16	10	M	7	7	3	13
Готель «Поділля»	7	3	6	7	9	6	9	M	11	5	7
ТЦ «Агора»	8	10	11	16	17	10	9	10	M	8	17
ТЦ «WoodMall»	9	5	9	13	15	8	4	5	9	M	10
Ринок "Дубово"	10	9	12	8	6	13	13	7	19	10	M

Розглянемо більш детально кожен з методів розв'язку задачі комівояжера та розрахуємо оптимальний час доставки за вхідними даними (Таблиця 1).

Метод повного перебору, по-іншому званий методом грубої сили (англ. Brute force), є простим, логічним і широко використовуваним математичним методом. Він може застосовуватись у багатьох, якщо не у всіх, областях математики: завдання комівояжера також не є винятком.

Ідея brute force проста: перебираються всілякі рішення і з них вибирається рішення (або безліч рішень) відповідає умові завдання. У задачі комівояжера, відповідно, потрібно з різних варіантів об'їзду пунктів вибрати маршрут, який займає найкоротший.

Величезною перевагою методу повного перебору перед іншими методами вирішення завдання комівояжера є гарантованість знаходження найкращого

маршруту. Інші методи радять лише «Хороший» маршрут, який зовсім не обов'язково є кращим. Крім того, до переваг методу відноситься простота його програмної реалізації.

Метод найближчого сусіда відноситься до так званих жадібних алгоритмів. Жадібний алгоритм (англ. Greedy algorithm) має на увазі під собою прийняття локально оптимальних рішень, допускаючи що кінцеве рішення також виявиться оптимальним. Існує досить велика кількість завдань, для яких жадібні алгоритми дають оптимальні рішення. Однак в таких завданнях, в тому числі і в завданні комівояжера, вони дають досить непогане наближене рішення. Ідея алгоритму найближчого сусіда заснована на простому правилі: якщо ми будемо відвідувати найближчий пункт на кожному кроці, то маршрут вийде досить хороший в цілому. Перед комівояжером ставиться завдання відвідувати найближчий з ще не відвіданих пунктів. В алгоритмі існують два важливих обмеження:

1) Недопущення повторного заїзду в пункт. Воно пов'язане з необхідністю (за умовою завдання) знаходження Гамільтона циклу, тобто циклу, в якому всі пункти відвідуються раз.
2) Недопущення передчасного повернення в початковий пункт. Ця заборона вводиться для запобігання передчасного зациклення маршруту, яке потягне за собою неправильну роботу алгоритму.

Метод гілок і меж (англ. Branch and bound) - загальний алгоритмічний метод, який широко застосовується для вирішення завдань комбінаторної оптимізації.

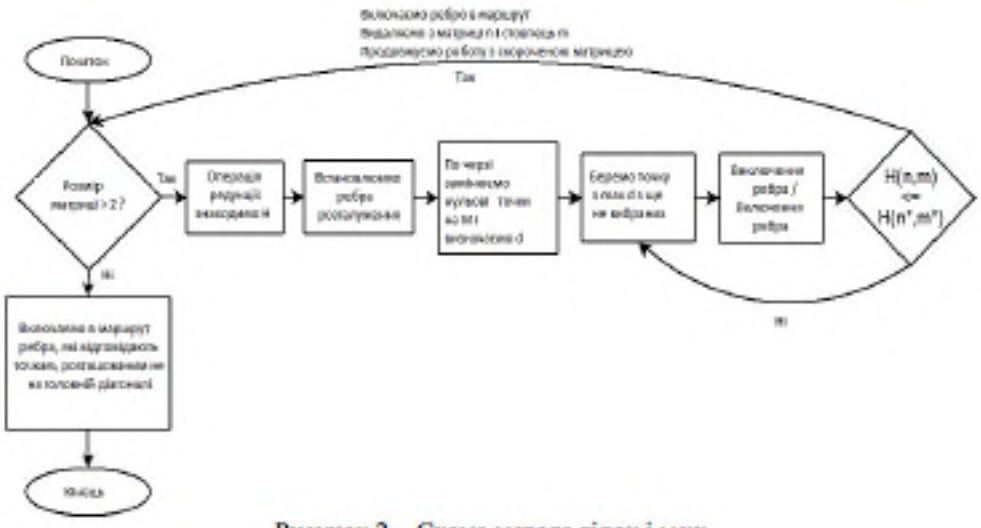

Рисунок 2 – Схема метода гілок і меж

По суті, метод є вдосконаленим методом повного перебору з послідовним відсівом рішень, що здаються невигідними.

Внаслідок того, що в процесі роботи методу деякі рішення не розглядаються, метод гілок і меж не може гарантувати знаходження точного рішення задачі. Відкинуте на початковому етапі «невигідне» рішення може виявитися в кінці кінців кращим. В основі методу гілок і меж лежить ідея послідовного розбиття множини рішень шляхом розгалуження і знаходження оцінок границь. Схема методу приведена на рисунку 2.

Метод гілок і меж вважається універсальним методом, так як він добре підходить для вирішення завдання комівояжера.

Виходячи з результатів (Таблиця 2) методів рішення задачі комівояжера можна зробити висновок, що кращий результат дав метод повного перебору на основі якого буде побудована інформаційна система оптимізації доставки замовлень.

Таблиця 2 – Результати оптимізації маршруту

Метод	**Час оптимізованого маршруту**
Повного перебору	51 хвилина
Найближчого сусіда	60 хвилин
Гілок і меж	56 хвилин

Отже, інформаційна система оптимізації маршрутів доставки дозволить отримувати оптимальні маршрути для доставки замовлень. Ця система може бути задіяна в закладах де надаються послуги з доставки замовлень. Основними перевагами системи є:

- зменшення часу на доставку замовлень;
- економія ресурсів транспортних засобів.

Інформаційній системі доступне покращення функціоналу, за рахунок використання інших оптимальніших засобів побудови маршруту.

Перелік посилань

1. Бронштейн Е. М., Заико Т. А. Детерменірованние оптимізаційні завдання транспортної логістики // Автоматика і телемеханіка, 2010. №10. С. 133-147.
2. Мудров В.І. Завдання про комівояжера. 1969. 62 с.
3. Задача комівояжера [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Задача_комівояжера
4. Пошукова система Google [Електронний ресурс]. – Режим доступу: https://www.google.com/
5. Модульне середовище для навчання [Електронний ресурс]. – Режим доступу: https://msn.khnu.km.ua/
6. Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/
7. Google Maps API [Електронний ресурс]. – Режим доступу: https://developers.google.com/maps/documentation

Wed Dec 02 22:21:37 EET 2020, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

	Документ		Сумарний збіг по Базі Даних	
ID: 82191 Назва: Інформаційна система доставки онлайн замовлень в мережі закладів швидкого харчування Додано в БД: 2020-12-02 Автора: Ткачук Євгеній Андрійович Керівники: Багрій Р.О. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	63865	560	7826 (12%)	83 (15%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ КАФЕДРИ **КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ** ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

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

Назва: **Інформаційна система доставки онлайн замовлень в мережі закладів швидкого харчування**
Автор: **Ткачук Є.А.**
Спеціальність: **122 Комп'ютерні науки**
Науковий керівник: **к.т.н., доц. Багрій Р.О.**

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	-
3	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	-
4	Інше:	-

Підтвердження: Виявленні запозичення не є плагіатом т.я. розміщенні в розділах, які не описують безпосередньо авторське дослідження, складають 18,5% та мають посилання на приведений список літературних джерел

01.11.2020
Дата

Підпис керівника

Підпис завідувача кафедри

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

ВІДГУК ОПОНЕНТА

на дипломну роботу магістра

Магістра *гр. КНм-19-1 Ткачука Євгенія Андрійовича*

На тему: *Інформаційна система доставки онлайн замовлень в мережі закладів швидкого харчування.*

1. Актуальність і значення теми

Інформаційна система доставки онлайн замовлень в мережі закладів швидкого харчування є особливо актуальною в період пандемії COVID-19. Через впровадження карантину кількість замовлень з доставкою в закладах швидкого харчування значно зросла. Розроблена інформаційна система дозволяє оптимізувати маршрути доставки замовлень в мережах закладів швидкого харчування, що забезпечує економію ресурсів та часу на доставку замовлень та позитивно впливає на економічне становище цих закладів.

2.Оцінка якості та достовірності проведених досліджень.

Отримані результати добре співвідносяться з результатами, наведеними в наукових роботах. Проведені в роботі дослідження підтвердили достовірність одержаних результатів.

3. Оцінка запропонованих заходів та пропозицій, практичної цінності та ефективності.

Проведені дослідження представляють науково-технічну цінність, є ефективними методами оптимізації, їх можна використати з метою оптимізації маршрутів доставки замовлень.

4. Загальний висновок та оцінка

Робота виконана в повному обсязі. Пояснювальна записка оформлена в відповідності з нормами. Відмічені недоліки не знижують цінності дипломної роботи. За своєю структурою, практичними цінностями, поставленій меті та вирішеними задачами робота відповідає вимогам вищої школи і вимогам, що пред'являються до освітньо-кваліфікаційного рівня «магістр», а її автор Є. А. Ткачук заслуговує присвоєння кваліфікації магістра з комп'ютерних наук.

Робота заслуговує на оцінку « добре ».

Опонент к.т.н., доцент кафедри галузевого машинобудування та архітектури Ладвещук Н.Г.