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 оптимізації маршрутів торгових агентів*

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* _____ *А.Ю. Буров*
Підпис Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КНІТ* _____ *С.С.Петровський*
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*» *9* *2020* року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ МАГІСТРА

1. Тема дипломної роботи магістра: *«Інформаційна система оптимізації маршрутів торгових агентів»*

2. Завдання видано студенту Бурову Андрію Юрійовичу

(прізвище, ім'я, по батькові)

3. Керівник роботи *к.т.н., доцент Петровський Сергій Степанович*

(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «*9*» *9* 202 р. № *22*

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

Мета роботи – розробка системи для контролю торгових агентів, системи, що дозволить працювати з декількома торговими агентами, розрізняючи їх маршрути та акумулюючи зібрану інформацію. Відображаючи маршрути агентів на онлайн карті інтерактивно змінюючи фільтри, щоб динамічно відображати потрібну для користувача інформацію в програмі. На кортах також зображена актуальна інформація про торгові точки. В якості даних для аналізу необхідно виводити користувачеві інформацію по минулому маршруті з можливістю відобразити або приховувати стоянки торгових агентів. Система має бути універсальною, щоб її можна було інтегрувати в поточні процеси для компаній з торговими агентами та представниками.

Реферат

Дипломна робота магістра присвячена розробці системи призначеної для ефективного управління торговими агентами та представниками, розробці технологій , що дозволять спростити роботу супервайзера та збільшити кількість активних клієнтів.

Актуальність теми. На даний момент, в умовах запеклої конкуренції велике значення в успішній діяльності підприємства набуває управління просуванням виробів на ринок, що є однією з функцій маркетингу. Це потрібно для того, щоб створити систему управління просуванням виробів підприємства на ринок.

Ця проблема ставить завдання з просування виробів фірми на ринок, тобто завдання логістичні. Для їх успішного вирішення підприємству необхідно досягти деякого мінімального обсягу продажів. У іншому випадку всі зусилля з організації служби доставки товару до споживача виявляються малорентабельними.

До особливостей масової торгівлі слід в першу чергу віднести широкий асортимент продукції, що випускається, велику кількість покупців і мінливість у потребах. Успіх підприємства на ринку великою мірою забезпечує доступність товару для покупця, яка досягається різними формами торгівлі. Останнім часом в промисловості широко використовуються методи дрібнооптової торгівлі із залученням торгових агентів (ТА), кожен з яких пов'язаний з групою клієнтів, що постійно оновлюється. В організації роботи з торговими агентами є особливості, які можна назвати як перевагами, так недоліками: значні обсяги клієнтської бази; велика швидкість ротації клієнтів; висока нестабільність характеристик бізнесу клієнтів - починаючи з обсягів і структури продажів і закінчуючи присутністю на ринку; невеликі масштаби ринку більшості торгових посередників; територіальна віддаленість їх один від одного і від центрального складу виробника або оптовика; обмеженість агентів в асортименті (що, з іншого боку, дає можливість економити на складських приміщеннях); відсутність гарантійних запасів, тощо[1].

Робота з товарними агентами передбачає формування невеликих за обсягом цільових замовлень. Це передбачає створення на складах фірм-оптовиків великих запасів товарів, необхідних для безперебійного постачання ТА нових партій товару. Вони формуються залежно від характеру замовлень покупців, а тому часто дрібні партії товарів можуть об'єднуватись у великі, або навпаки – великі обсяги можуть ділитись на декілька замовлень. Виконання замовлення здійснюється на основі товарного чеку, який друкується у кількох примірниках. Як правило, один з них призначається для формування замовлення на складі, другий для покупця і третій має бути підписаний покупцем в якості підтвердження отримання товару та повернений на підприємство. Така система формування замовлення полегшує контроль видачі, перевезення та отримання товару сторонами. Часто підприємства застосовують контрольно-ревізійні перевірки для звірення даних в товарних чеках [1; 2].

Мета і задачі роботи. Мета роботи полягає у розробці інформаційної системи, яка дозволить спланувати супервайзеру роботу торгових агентів шляхом подубови планових маршрутів на поточний день з можливістю спланувати час прибуття торгового агента на торгову точку. Для перевірки роботи агента потрібно проаналізувати його минулі маршрути, що дозволить виявити затримки і відхилення, а в подальшому їх запобігти. Для досягнення поставленої мети визначені наступні задачі дослідження:

– дослідження організації роботи торгових агентів для визначення можливих варіантів побудови для них маршрутів та визначення їх видів;

– розробка інформаційної технології відображення маршрутів торгових агентів у вигляді списку та графіку, оцінкою відстаней між торговими точками і відображеня цих точок у звучному вигляді;

– створення автоматизованої системи визначення оптимальних маршрутів з можливістю порівняти результати, дисперсійної оцінки та різниці відстаней;

– аналіз ефективності після перегляду попередніх маршрутів і висновки по побудові наступних, за допомогою системи відображення та планування маршрутів торгових агентів.

Об’єкт дослідження – маршрути торгових агентів як результат планування та аналізу попередніх маршрутів.

Предмет дослідження – методи побудови та відображення маршрутів.

Методи дослідження, застосовані для вирішення поставлених завдань: для визначення маршрутів між точками; для визначення критеріїв зупинок і стоянок торгових агентів; для реалізації інформаційної системи - методології проектування систем та об’єктно-орієнтований підхід для порівняння минулих маршрутів торгових агентів з їх візуальним відображеням

Наукова новизна одержаних результатів. У результаті роботи були отримані наступні результати:

– Вперше розроблено метод визначення ключових точок в пройденому торговим агентом маршрутах, а саме було визначено критерії побудови точок стоянок і зупинок, згідно даних GPS з пристрою торгового агента..

– Набули подальшого розвитку системи по управлінню торговими агентами. Існуючі системи можуть використати у своїх продуктах розроблені методи аналізу маршрутів.

– Досліджено практичну ефективність якісного планування маршрутів торгових агентів для збільшення кількості торгових точок, яких може відвідати торговий агент, а також більш ефективне планування його робочого дня.

Практичне значення одержаних результатів. На основі розроблений методів аналізу маршрутів торгових агентів було створено автоматизовану систему для планування та відображення маршрутів з можливістю виводу результату роботи самих торгових агентів.

При розробці системи планування маршрутів торгових агентів було зібрано пересування торгових представників з декількох компаній для більш достовірної вибірки

За результатами дослідження, проведеного з використанням розробленої системи аналізу та побудови маршрутів, одержано наступні результати:

– встановлено причини некоректної побудови маршрутів при слабкому сигналі GPS з отриманням даних від мобільних мереж;

– визначено показники точності і достовірності зібраних даних;

– встановлено причини некоректної побудови маршрутів користувачами програми.

Для підвищення ефективності планування маршрутів визначено ключові точки, вказуючи які можна побудувати більш точніший плановий маршрут.

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 40 найменувань та 2 додатків. Загальний обсяг дипломної роботи магістра становить 97 сторінок, з них 72 сторінка основного тексту та 25 сторінок додатків. У роботі наведено 16 рисунків.

Ключові слова: торгові агенти, побудова маршрутів, інформаційна система, планування маршрутів, мобільні продажі.

Зміст

Перелік скорочень 4

Вступ 5

Розділ 1

Аналіз сучасного стану проблеми контролю над торговими агентами і представниками 9

1.1 Аналіз предметної області 9

1.2 Аналіз сучасних наукових публікацій по плануванню роботи торгових агентів 15

1.3 Дослідження інформаційного забезпечення для планування роботи торгових агентів 17

1.4 Постановка задачі 25

Висновки до розділу 1 26

Розділ 2

Розробка інформаційної технології аналізу маршрутів торгових агентів 27

2.1 Загальний опис технології аналізу маршрутів торгових агентів 27

2.2 Метод формування проаналізованих маршрутів 30

Висновки до розділу 2 32

Розділ 3

Розробка методів та компонентів системи для планування та відображення маршрутів торгових агентів 33

3.1 Розробка методу взаємодії системи із допоміжними додатками для збору координат пересування торгових агентів 33

3.2 Проектування програмного забезпечення для аналізу та побудови маршрутів 35

3.2.1 Проектування архітектури майбутньої програми 35

3.2.2 Проектування інтерфейсу користувача програми для побудови маршрутів 45

3.2.3 Аналіз та автоматизація обробки потоків даних та проектування структури бази даних 48

3.2.4 Аналіз та вибір технологій і методів реалізації програми для побудови маршрутів 55

Висновки до розділу 3 59

Розділ 4

Дослідження ефективності системи для аналізу та планування маршруту торгового агента 60

4.1 Робота системи відображення маршрутів торгових агентів 60

4.2 Тестування програмної системи аналізу та побудови маршрутів 63

Висновки до розділу 4 68

Загальні висновки 69

Перелік посилань 70

Додатки

Перелік скорочень

Скорочення, термін, позначення	Пояснення
АІС	Автоматизована інформаційна система
БД	База даних
ДРМ	Дипломна робота магістра
ТА	Торговий агент
ТТ	Торгова точка
ПЗ	Програмне забезпечення

Вступ

Актуальність теми. На даний момент, в умовах запеклої конкуренції велике значення в успішній діяльності підприємства набуває управління просуванням виробів на ринок, що є однією з функцій маркетингу. Це потрібно для того, щоб створити систему управління просуванням виробів підприємства на ринок.

Ця проблема ставить завдання з просування виробів фірми на ринок, тобто завдання логістичні. Для їх успішного вирішення підприємству необхідно досягти деякого мінімального обсягу продажів. У іншому випадку всі зусилля з організації служби доставки товару до споживача виявляються малорентабельними.

До особливостей масової торгівлі слід в першу чергу віднести широкий асортимент продукції, що випускається, велику кількість покупців і мінливість у потребах. Успіх підприємства на ринку великою мірою забезпечує доступність товару для покупця, яка досягається різними формами торгівлі.

Останнім часом в промисловості широко використовуються методи дрібнооптової торгівлі із залученням торгових агентів (ТА), кожен з яких пов'язаний з групою клієнтів, що постійно оновлюється. В організації роботи з торговими агентами є особливості, які можна назвати як перевагами, так недоліками: значні обсяги клієнтської бази; велика швидкість ротації клієнтів; висока нестабільність характеристик бізнесу клієнтів - починаючи з обсягів і структури продажів і закінчуючи присутністю на ринку; невеликі масштаби ринку більшості торгових посередників; територіальна віддаленість їх один від одного і від центрального складу виробника або оптовика; обмеженість агентів в асортименті (що, з іншого боку, дає можливість економити на складських приміщеннях); відсутність гарантійних запасів, тощо[1].

Робота з товарними агентами передбачає формування невеликих за обсягом цільових замовлень. Це передбачає створення на складах фірм-

оптовиків великих запасів товарів, необхідних для безперебійного постачання ТА нових партій товару. Вони формуються залежно від характеру замовлень покупців, а тому часто дрібні партії товарів можуть об’єднуватись у великі, або навпаки – великі обсяги можуть ділитись на декілька замовлень.

Виконання замовлення здійснюється на основі товарного чеку, який друкується у кількох примірниках. Як правило, один з них призначається для формування замовлення на складі, другий для покупця і третій має бути підписаний покупцем в якості підтвердження отримання товару та повернений на підприємство. Така система формування замовлення полегшує контроль видачі, перевезення та отримання товару сторонами. Часто підприємства застосовують контрольно-ревізійні перевірки для звірення даних в товарних чеках[1; 2].

Мета і задачі роботи. Мета роботи полягає у розробці інформаційної системи, яка дозволить спланувати супервайзеру роботу торгових агентів шляхом подубови планових маршрутів на поточний день з можливістю спланувати час прибуття торгового агента на торгову точку. Для перевірки роботи агента потрібно проаналізувати його минулі маршрути, що дозволить виявити затримки і відхилення, а в подальшому їх запобігти. Для досягнення поставленої мети визначені наступні задачі дослідження:

– дослідження організації роботи торгових агентів для визначення можливих варіантів побудови для них маршрутів та визначення їх видів;

– розробка інформаційної технології відображення маршрутів торгових агентів у вигляді списку та графіку, оцінкою відстаней між торговими точками і відображеня цих точок у звучному вигляді;

– створення автоматизованої системи визначення оптимальних маршрутів з можливістю порівняти результати, дисперсійної оцінки та різниці відстаней;

– аналіз ефективності після перегляду попередніх маршрутів і висновки по побудові наступних, за допомогою системи відображення та планування маршрутів торгових агентів.

Об’єкт дослідження – маршрути торгових агентів як результат планування та аналізу попередніх маршрутів.

Предмет дослідження – методи побудови та відображення маршрутів.

Методи дослідження, застосовані для вирішення поставлених завдань: для визначення маршрутів між точками; для визначення критеріїв зупинок і стоянок торгових агентів; для реалізації інформаційної системи - методології проектування систем та об’єктно-орієнтований підхід для порівняння минулих маршрутів торгових агентів з їх візуальним відображеням

Наукова новизна одержаних результатів. У результаті роботи були отримані наступні результати:

– Вперше розроблено метод визначення ключових точок в пройденому торговим агентом маршрутах, а саме було визначено критерії побудови точок стоянок і зупинок, згідно даних GPS з пристрою торгового агента..

– Набули подальшого розвитку системи по управлінню торговими агентами. Існуючі системи можуть використати у своїх продуктах розроблені методи аналізу маршрутів.

– Досліджено практичну ефективність якісного планування маршрутів торгових агентів для збільшення кількості торгових точок, яких може відвідати торговий агент, а також більш ефективне планування його робочого дня.

Практичне значення одержаних результатів. На основі розроблений методів аналізу маршрутів торгових агентів було створено автоматизовану систему для планування та відображення маршрутів з можливістю виводу результату роботи самих торгових агентів.

При розробці системи планування маршрутів торгових агентів було зібрано пересування торгових представників з декількох компаній для більш достовірної вибірки

За результатами дослідження, проведеного з використанням розробленої системи аналізу та побудови маршрутів, одержано наступні результати:

– встановлено причини некоректної побудови маршрутів при слабкому сигналі GPS з отриманням даних від мобільних мереж;

– визначено показники точності і достовірності зібраних даних;

– встановлено причини некоректної побудови маршрутів користувачами програми.

Для підвищення ефективності планування маршрутів визначено ключові точки, вказуючи які можна побудувати більш точніший плановий маршрут.

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 40 найменувань та 2 додатків. Загальний обсяг дипломної роботи магістра становить 97 сторінок, з них 72 сторінка основного тексту та 25 сторінок додатків. У роботі наведено 16 рисунків.

Розділ 1

Аналіз сучасного стану проблеми контролю над торговими агентами і представниками

1.1 Аналіз предметної області

Торговий агент відвідує магазини, пропонуючи товар для поставки, збирає заявки. Це представник підприємства-виробника, оптової або дистриб'юторської фірми, що відповідає за просування товару в роздрібних торгових точок.

Використання торгових агентів в сучасних умовах ведення бізнесу - це практично невід'ємна частина для великої кількості компаній.

Торговий агент оформляє договори купівлі-продажу, контролює їх виконання. Іноді він же забезпечує доставку і відвантаження товарів, мерчандайзинг (стежить за викладкою товару і його наявністю на прилавку).

Наявність сильної конкуренції вимагає використання ефективних засобів, наприклад, мобільних продажів. Мобільні продажі - термін, що позначає доставку товару виробником в торгівлю або покупцеві за рахунок широких можливостей вибудовування логістики руху товару і грошей, документообігу будь-якого рівня складності. Це дозволяє реалізовувати ефективні бізнес-процеси і організувати торгівлю з урахуванням всіх нюансів логістики і здійснити якісний стрибок в організації бізнесу.

Одна з найбільш популярних форм подібної організації роботи - van selling (вен-селлінг), дрібнооптова торгівля «з коліс», за якої вибір товару і оформлення угоди здійснюється прямо в офісі клієнта. Вен-селлінг переслідує чотири такі цілі, як: реклама, доставка, розширення асортименту, маркетинг.

Вен-селлінг створює певні переваги для постачальників і продавців, перш за все вони пов’язані з оптимізацією процесу продажу та використання складського приміщення через закупівлю товару без залишків, спрощенням

процесу доставки та продажу для покупця та полегшення контролю наявності товару, оперативністю доставки, тощо.

Особливо відчутний виграш від застосування вен-селлінга для компаній, що виробляють або продають дрібні споживчі товари повсякденного попиту. Найчастіше це продукти харчування, миючі засоби, недорога парфумерія і т. п. Такий товар швидко розходиться, і поповнювати його запаси в роздрібній торговельній мережі треба дуже оперативно. При цьому асортимент продукції може бути будь-яким - і дуже широким, і порівняно невеликим.

Практична реалізація вен-селлінга виглядає наступним чином. З ранку «вен» завантажується товаром і потім разом з торговим представником відправляється за маршрутом точками роздрібної торгівлі. У кожній точці проводяться переговори, відразу ж оформляються всі документи і відбувається відвантаження товару[2].

Ефективним також є впровадження системи pre-selling (пре-селлінгу). Це організація попереднього збору замовлень для реалізації товару підприємствами. Мобільна торгівля за цим принципом передбачає використання електронного пристрою (комунікатора), який має повну інформацію про наявність товару на складах.

Таким чином торгові агенти можуть виконувати завдання як моделі пре-селлінгу, так і вен-селлінгу. Перший – пре-селлінг – зводиться до формування портфеля замовлень в процесі спілкування з актуальними та потенційними клієнтами. Цей режим передбачає максимальну свободу в ухваленні рішення. В першу чергу мова йде про цінову політику. Мобільний модуль може раціонально зв'язати три найважливіші компоненти процесу формування замовлень − прайс-листи, торгові операції та клієнтів. Таким чином, ціна може бути обрана виходячи з умов угоди, на які погодиться той чи інший конкретний клієнт. Інформація про клієнтів та історія відносин з ними зберігається в стаціонарній частині комплексу. Агенту немає необхідності самостійно аналізувати їх, варіанти рішень можуть бути завантажені в його кишеньковий комп'ютер до

початку роботи. Крім того, завчасно може бути сформована політика компанії щодо нових, незнайомих клієнтів, для них можуть бути визначені дозволені операції і допустимі варіанти цін. Інформація про актуальних клієнтів, після завершення певного періоду «скидається» в стаціонарну частину комплексу.

Вен-селлінг зводиться до дрібнооптового продажу в різних точках роздрібної торгівлі. У більшості випадків це простіші завдання, які можуть бути зведені до функції доставки зроблених замовлень. Тут, як правило, велике значення має якість формування маршрутів і висока надійність блоку друку. Природньо, обидва види робіт можуть бути суміщені в різних варіантах.

Широкі можливості вибудовування логістики руху товару і документів, документообігу будь-якого рівня складності, дозволяє реалізувати бізнес-процеси, які неможливо впровадити без автоматизованої системи, або застосовувати методи податкової оптимізації, що допомагають економити до 80% податків, при цьому не порушуючи меж чинного законодавства. Реалізувати такий підхід вручну, без автоматизованої до рівня кожного робочого місця системи, практично нереально. Фактично відбувається якісний стрибок в організації бізнесу, який в принципі не можна зробити без автоматизації [3].

Будь-яке підприємство роздрібної або оптової торгівлі прагне до адекватної автоматизації своїх бізнес-процесів. Два основних мотивуючих фактора в даній сфері, в загальному, давно відомі - це глобальне зниження витрат і підвищення ефективності компанії. Безумовно, до самого процесу поліпшення систем відносяться по-різному. Можна діяти з розмахом: всюди, де потрібно, поставити комп'ютери, об'єднати їх в дротову мережу, закупити необхідну кількість промислових терміналів для складу, встановити спеціальне програмне забезпечення. Це дорогувато, але, як правило, ефективно - особливо якщо на підприємстві точно знають, роботу якої ділянки потрібно поліпшити. Інший варіант - повна протилежність: автоматизуватися по мінімуму, поставивши лише потрібну техніку в бухгалтерії і в службі збору

замовлень, зробити інтернет-сайт з формою замовлення, а доставкою керувати по телефону.

У цьому сенсі система, де використовуються мобільні пристрої зі спеціальним ПЗ, виявляється як не можна до речі: вона поєднує в собі гнучкі можливості масштабування (стаціонарна техніка встановлюється тільки в офісі компанії, а всі мобільні співробітники оснащуються портативними пристроями), її не складно експлуатувати (первинне навчання персоналу найчастіше мінімально - інтерфейс у подібних систем досить простий), а завдяки невеликій вартості кінцевих пристроїв інсталяція такої системи не вимагає великих грошових вкладень. До того ж кишенькові комп'ютери дозволяють автоматизувати облік в таких областях, де раніше це було просто неможливо - наприклад, в кур'єрських службах, при обслуговуванні дрібних торгових точок, які не мають виходу в Інтернет, і т.д.

Автоматизована система управління мобільною торгівлею - програмний комплекс, що дозволяє ефективно вирішувати завдання автоматизації торгових представників з використанням компактних комп'ютерів. Система дозволяє істотно підвищити ефективність продажів, помітно знизити витрати на підтримку структури торгових агентів, принципово посилити контроль над їх діяльністю, підвищити точність і швидкість виконання замовлень, значно прискорити виведення на ринок нової продукції і кардинально зменшити терміни навчання нових торгових представників. При цьому система забезпечує оперативне отримання інформації для прийняття необхідних управлінських рішень, доступ керівництва до будь-якого рівня інформації про продажі, дозволяє досить просто реалізовувати побудову складних, територіально розподілених систем для підтримки мобільних продажів. Система автоматизації мобільної торгівлі дозволяє підняти на принципово новий рівень інформаційну підтримку системи дистрибуції товарів і прийняття управлінських рішень, поліпшити імідж компанії і підвищити лояльність клієнтів.

Цього досягають шляхом: зменшення часу на прийняття рішень торговим агентом в точці продажів, яке забезпечується оперативним доступом до інформації на маршруті; автоматичного виконання основних розрахункових операцій; зменшення помилок при створенні документів за рахунок автоматизації контролю обмежень для конкретної торгової точки; оптимізації маршрутів ТА і посилення контролю над діяльністю торгових агентів; прискорення введення інформації в облікову систему підприємства про створені торговим агентом документи та його роботу; можливості не відвідувати офіс, а використовувати віддалену синхронізацію; підвищення точності і швидкості виконання; скорочення операторів з введення інформації від торгових агентів; оптимізації залишків продукції на складі за рахунок своєчасного надходження замовлень і оперативного відвантаження товару; підвищення якості обслуговування клієнтів.

У базі даних компанії зберігається інформація, необхідна для роботи ТА і для обміну з корпоративною обліковою системою (товари ,склади, ціни, знижки, замовники, торгові агенти), дані по маршрутах і управління правами торгових агентів, плани та цілі, сценарії роботи мобільних працівників, об'єкти обліку та багато іншого.

Система управління командою торгових агентів, яка дозволяє керівництву компанії та менеджерам підрозділу ТА оперативно отримувати актуальну інформацію про роботу працівників, формувати маршрути, дистанційно керувати роботою мобільних працівників і ефективно контролювати їх діяльність.

Особливо актуальним є використання мобільних продажів та автоматизації системи управління мобільною торгівлею в період пандемії COVID через можливість зменшити кількість контактів між людьми, перевести працівників на віддалену роботу чи роботу «в полі» (торговим агентом або водієм), скоротити кількість працівників, тощо. Крім того особливо зростає цінність цього методу

через можливість скоротити витрати на податки, що компенсує негативний вплив епідемії на рівень продажів.

Підбиваючи підсумки варто зазначити, що організація збуту за системою мобільних продажів дозволяє спростити значну частину процесів, що стосуються оптової чи роздрібної торгівлі. Ряд переваг для покупців та продавців, які надає система дозволяє оптимізувати роботу підприємствах на різних рівнях. Вен-селлінг, окрім того, дає можливість підприємствам економити місце на складських приміщеннях або ефективніше використовувати наявне, а покупцям нівелювати дефіцит певних товарів одразу на місці. Пре-селлінг дозволяє здійснювати точкові продажі задовольняючи короткострокові потреби клієнтів в перспективі, уникаючи або доповнюючи довгострокові угоди[4].

У більшості випадків торгові компанії автоматизують діяльність торгових представників, мерчендайзерів, супервайзерів, тобто польових співробітників, які за родом своєї діяльності більшу частину робочого часу повинні перебувати поза офісом (в торгових точках, на маршруті), але в той же час залишатися на зв'язку з офісом - скидати замовлення, отримувати нові завдання і т.д. Збільшенню кількості «мобілізованих» компаній сприяє кілька чинників. Найважливіший показник в даному випадку - потенційне зростання продажів за рахунок більш швидкої та ефективної роботи торгового представника. Озброєний автоматизованим інструментом, він на місці робить угоди, може автоматично розрахувати знижки, націнки і т.д. Бездротовий зв'язок офісом забезпечує оперативну передачу замовлень безпосередньо в електронному вигляді, що знімає навантаження з операторів або взагалі дозволяє позбутися від них. Одночасно торговий представник отримує оперативний доступ до всієї необхідної йому інформації з офісної системи по кожній конкретній торговій точці (історія продажів, прайс-листи, баланси з клієнтами, що проходять рекламні акції і т.п.), що значно прискорює його роботу в порівнянні з тим , коли він використовував паперові носії. Таким чином, якщо раніше на формування замовлення для середнього за розміром магазину йшло до

1 год, то тепер це займає 5-10 хв. що значно прискорює його роботу в порівнянні з тим, коли він використовував паперові носії.

До того ж, торгові представники зі спеціальними мобільними пристроями в руках і виглядають солідніше (а значить, підвищується лояльність клієнтів), вони і працюють швидше, і замовлення виконують точніше. Автоматизована система економить їх час - їм немає необхідності щодня відвідувати офіс, відстоювати чергу до оператора, вбивається замовлення, не потрібно возити з собою стоси документів з маршрутними і прайс-листами і т.д.

Автоматизована система управління мобільною торгівлею надає можливість спростити та автоматизувати значну частину процесів та уникнути ряду помилок, які спричиняє людський фактор. Особливо ефективно ця система працює в мобільній торгівлі створюючи додатковий зв’язок між різними рівнями працівників підприємства, полегшуючи доступ до інформації та її використання. Додаткової актуальності модель набула після початку епідемії COVID та загострення необхідності оптимізації роботи торгових підприємств, створення можливості здійснення безконтактного адміністрування та економії коштів.

1.2 Аналіз сучасних наукових публікацій по плануванню роботи торгових агентів

При підготовці до написанні дипломної роботи було проаналізовано різні пов'язані наукові статті, які розглядають роботу торгових агентів та побудову для них маршрутів.

Наприклад в статті “Автоматизация построение маршрутов перевозок мелкопартионных грузов”[7] обґрунтована необхідність в автоматизації побудови маршрутів в умовах щоденно змінюючого попиту. Описанні умови, які необхідно виконувати при побудові маршрутів, визначенні

критерії завантаження та розвантаження, розміщення постачальників та покупців(торгових точок).

Також приведена математична постановка задачі об'єднання пунктів доставки з нульовими потребами з врахуванням собівартості перевезення. Побудовано алгоритм, який враховує обмеження вантажомісткості автомобіля, собівартість перевезень, заявки на поставку та зайнятість автомобіля.

Після її аналізу було досліджено варіанти автоматизації формування та побудови маршрутів для торгових агентів та представників супервайзером. До цього ж було розглянуто варіації типів маршрутів і методи для обрахунку вартості перевезення товарів у випадку продажу товарів самим торговим агентом.

Наступною статтею було розглянуто статтю “Сферы применения распределительной логистики”[8] в якій було розглянуто деякі аспекти застосування розподільчої логістики в діяльності різноманітних підприємств в умовах сучасного господарювання.

Стаття допомогла більше зрозуміти сфери дії роботи торгових агентів, а також проаналізувати необхідність зміни логіки побудови маршрутів за допомогою автоматизованого програмного рішення.

У розглянутих статтях не було знайдено методи та алгоритми для аналізу маршруту і визначення контрольних точок маршруту, і тому ці алгоритми будуть розроблені в рамках цієї роботи.

Наразі торгові агенти успішних компаній укомплектовані мобільними пристроями (телефон або планшет) зазвичай на базі Android зі спеціальною програмою.

Специфікою роботи торгового представника є те, що він працює в "полях" без будь-якого серйозного контролю. Не секрет, що працюючи в полях, ці співробітники надані самі собі. Найчастіше відсутність контролю веде до зниження дисципліни. Без належного контролю можливо нецільове

використання робочого часу, збір заявок по телефону замість того, , щоб презентувати нову продукцію або прийняти замовлення, помилкові (вигадані) дані мерчандайзингу. У цей момент конкуренти можуть відвести навіть самого лояльного клієнта, а обсяг ваших замовлень стрімко піде вниз.

Щоб клієнти завжди обслуговувалися якісно, а продажі компанії не залежали від окремих неурядових торгових представників, потрібно добре контролювати їх роботу "в полях".

Вигода досягається за рахунок застосування мобільних додатків з функцією GPS-контролю. Різні сервіси дозволяють ефективно планувати роботу торгових представників, контролювати їх розташування протягом дня, а також точно вимірювати транспортні витрати і продуктивність співробітників.

Як правило, торгові представники заповнюють звіти про виконану роботу самостійно, від руки. Така звітність не достовірна, її легко сфабрикувати - звідси і виникають підозри в нечесності співробітника.

GPS-контроль торгових представників виключає можливість махінації. А значить вам не доведеться вірити співробітникам на слово, підозрювати їх і змушувати писати звіти, щоб оцінити продуктивність.

Тому було вирішено розробити програму для контролю переміщення торгових агентів та представників.

1.3 Дослідження інформаційного забезпечення для планування роботи торгових агентів

На сьогоднішній день існують певні сервіси, які надають послуги відслідковування пересування торгових представників.

Також є програми для самих торгових агентів. Ці програми дозволяють агентам бачити оперативну інформацію, таку як залишки товарів на складах, детальну інформацію про товар чи його фото, а також записати замовлення у клієнта.

Роботою торгового агента управляє супервайзер. Він повинен спланувати маршрути по території і створити для торгових представників план візитів на кожен день; запланувати планові цільові показники для кожного маршруту по запланованим візитам; організувати дії своєї команди шляхом щоденної постановки задач по візитах на маршрути.

Також повинен контролювати результати виконання завдань і давати конструктивний зворотний зв'язок; давати об'єктивну оцінку внеску кожного торгового представника і приймати рішення, хто залишається на роботі, а кому запропонувати іншу посаду або звільнити.

Планування маршруту дозволяє створити оптимальну схему відвідування клієнтів і зменшити витрати компанії. Також дозволяє охопити більшу територію, збільшити кількість клієнтів, цим самим збільшити товарообіг й загальні прибутки.

Супервайзер компанії повинен мати змогу контролювати виконання торговими агентами поставленого їм маршрута, та відслідковувати їх пересування.

Нижче розглянемо декілька сервісів для супервайзера, які допомогають йому аналізувати роботу торгових агентів.

Одним із них є сервіс Yaware.Mobile, рисунок 1.1.

Цей сервіс він дозволяє переглядати пройдені маршрути торгових агентів. Також він дозволяє вивести на карту маршрути декількох агентів одночасно.

Недоліком цього сервісу є те, що він не може будувати планові маршрути, а також не відображається причина зупинки (відвідування точки маршруту, або звичайна зупинка).

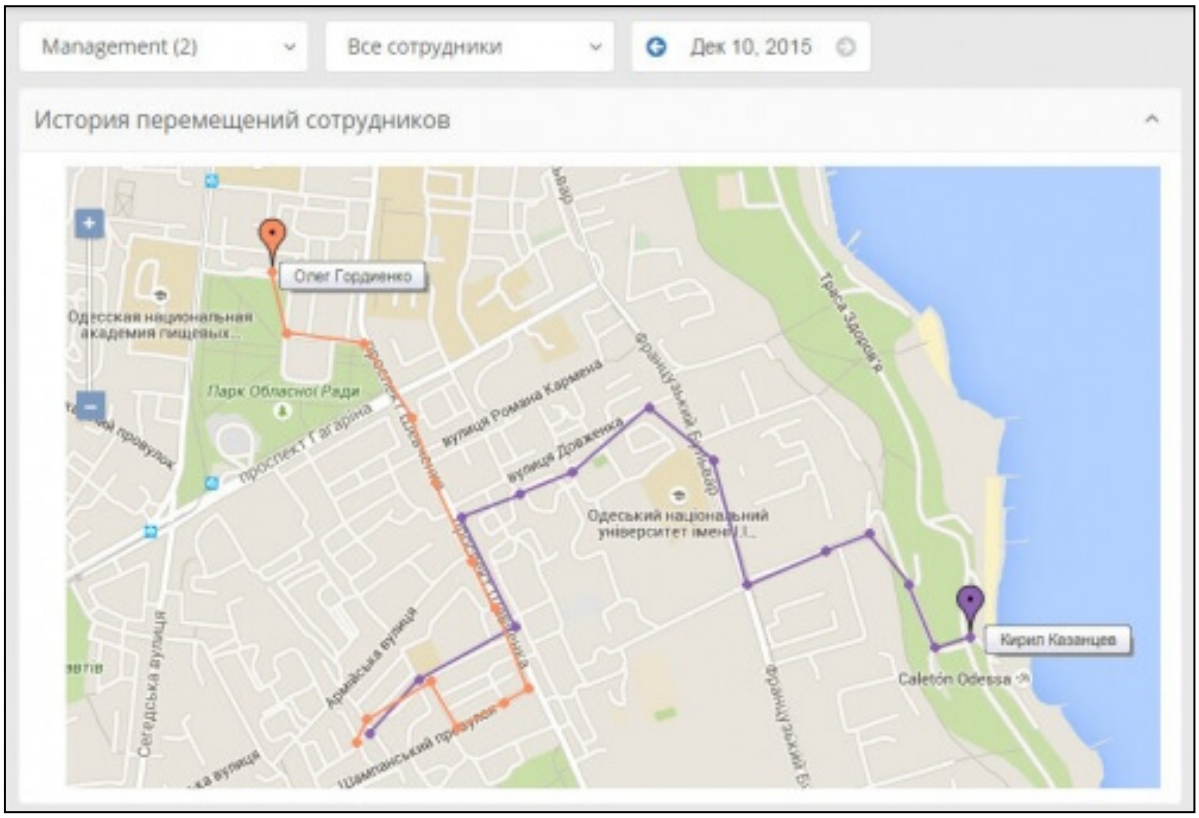


Рисунок 1.1 – Сервіс Yaware.Mobile

Також є система “Контроль Плюс”, що зображена на рисунку 1.2.

Вона також дозволяє переглянути відвідувані точки агента, а також відхилення від заданого маршута у вигляді таблиці. Система ще дозволяє проглянути простої агента, а також вирахувати пробіг та витрату пального.

Недоліком цієї системи, є те, що вона відображає точки списком, який на відміну від зображеного на самій карті є менш інформативний.

Тому буде більш логічним рішення зображати комбіновано маршрут як списком, так і на карті паралельно.

Цей сервіс застосовує власні пристрої для відстеження пересування торгових агентів

Рисунок 1.2 – Система “Контроль Плюс”

Також перевагою є те, що в списку відображається тривалість перебування торгового агента на торговій точці, і на основі цієї інформації можна зробити висновок скільки торговий агент провів часу з клієнтом.

На рисунку 1.3 зображено вікно роботи сервісу ANTOR TeamMaster, який виводить маршрут торгового агента на карті..

Сервіс має застарілий дизайн та не має змоги відобразити плановий маршрут агента з усіма точками простою, а тільки фактичний з позначеними місцями заявок.

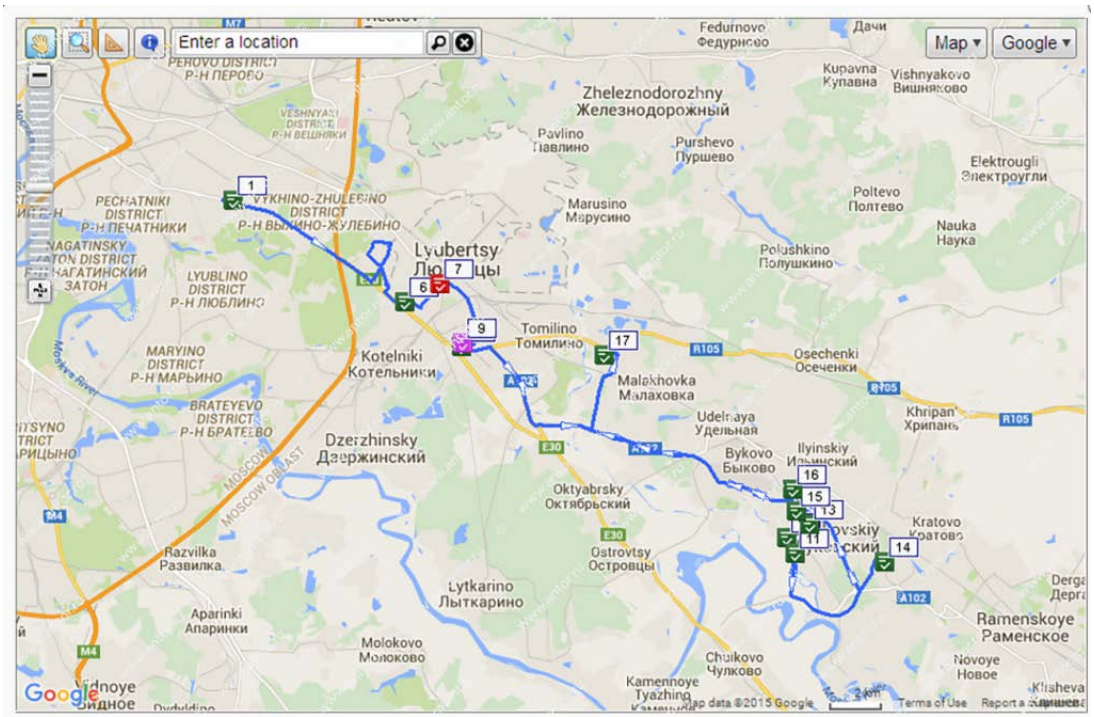


Рисунок 1.3 – Сервис ANTOR TeamMaster

На рисунку 1.4 зображено сервіс “ProfGPS”, який також дозволяє моніторити пересування агентів, а також відображати їх зупинки та швидкість.

Він призначений тільки для відслідковування автомобіля і не може відображати плановий маршрут і точки відвідувань по маршруту.

Також цей сервіс дозволяє відобразити різнити кольорами різні види контрольних точок, що дозволяє швидко оцінити ситуацію на маршруті та втрутитися за необхідністю.

Зручною функцією є також відображення інформації про зупинку на самій карті, тобто можна переглянути інформацію в один клік.

Різними кольорами на карті позначається швидкість торгового агента, і цим самим супервайзер може оцінити проблемні ділянки, щоб в подальшому побудувати маршрут з урахуванням цих даних і пришвидшити роботу торгового агента

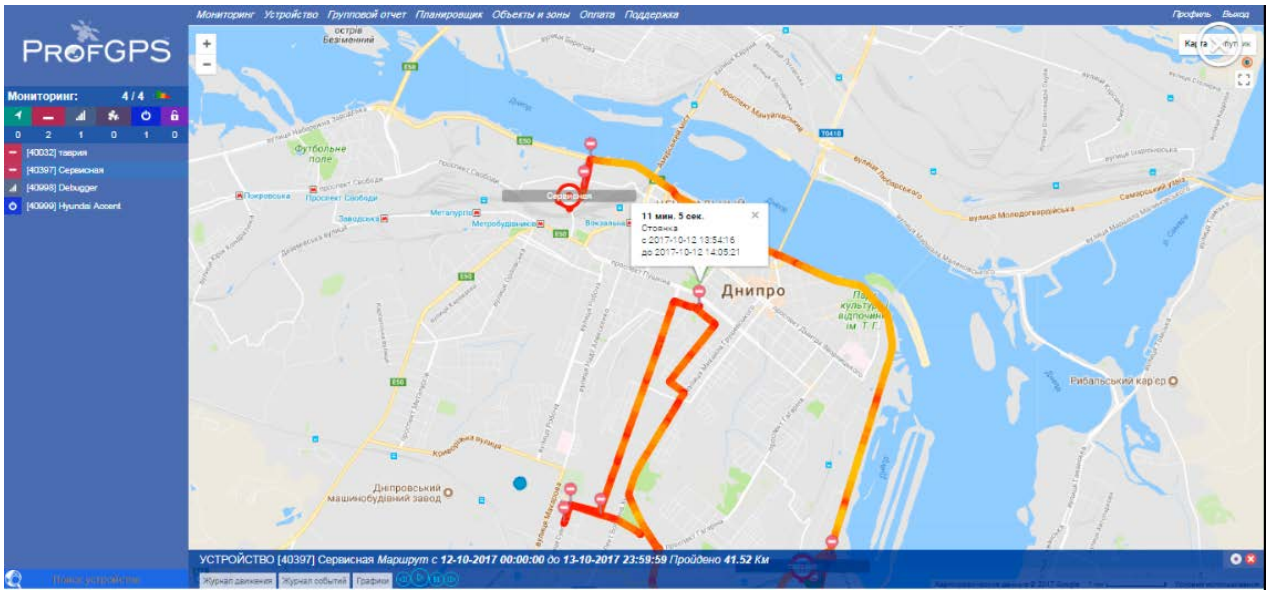


Рисунок 1.4 – Сервіс “ProfGPS”

На рисунку 1.5 зображено програма “MapXPlus+Trade”. Ця програма також зображує торгові точки на карті, але вона може відобразити точки для різних агентів на одній карті, щоб їх можна було перекомпонувати, для більш ефективного розподілу всіх точок по торговим агентам.

Також ця програма дозволя оцінити, як близько точки відносяться одна до одної, щоб їх можна було згрупувати при об їзді маршруту.

Але ця програма надає менше можливостей для порівняння попереднього маршруту з поточним. І також немає відображених точок, які агент не мав відвідати, або відвідував по свої справах.

На рисунку 1.6 зображена система управління агентами Агент Плюс.

Додаток «Агент Плюс: Мобільна торгівля» підтримує роботу з маршрутами для торгових агентів на поточний день, а також дає можливість переглянути або змінити маршрут на майбутні дні або переглянути маршрути за попередні дні.

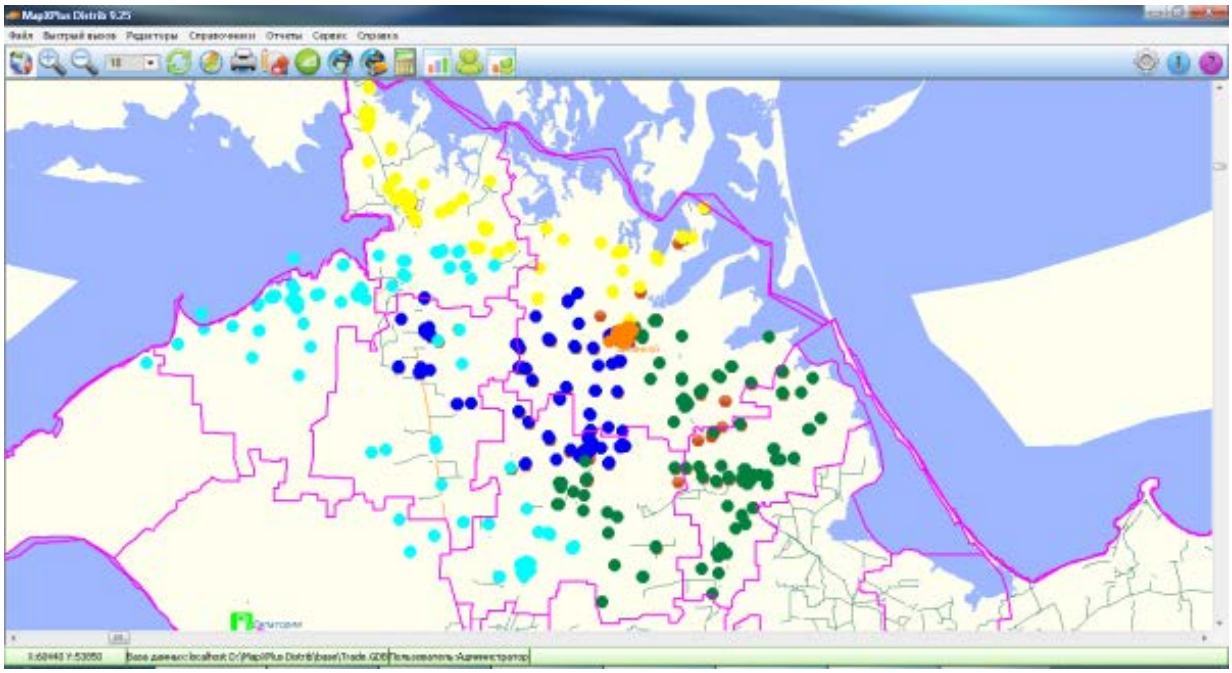


Рисунок 1.5 – Сервіс “MapXPlus+Trade”

Маршрут може бути створений з урахуванням заданого порядку відвідування торгових точок, а також (при необхідності) із зазначенням бажаного часу відвідування торгових точок.

Мобільний додаток розроблений, в основному, для системи 1С. І сам сервіс реалізований, як модуль для 1С. Налаштування маршрутів проводиться за допомогою документа «План відвідувань агента». Після вивантаження «Плану відвідувань» на МУ торгового агента, він буде представлений у вигляді набору маршрутів на кожен день, заданих в плані відвідувань.

В тому, що це модуль програми 1С, цей сервіс не може повноцінно використовуватися в поточних реаліях.

Більшість рішень призначені для відслідковування автомобілей торгових агентів або самих агентів через програми на пристроях, але ці рішення не мають змоги відображати планові маршрути або порівнювати здійснювані відвідування торгового агента з плановими. Також важливим є відображення цих маршрутів на карті з позначками торгових точок та позапланових зупинок торгового агента.

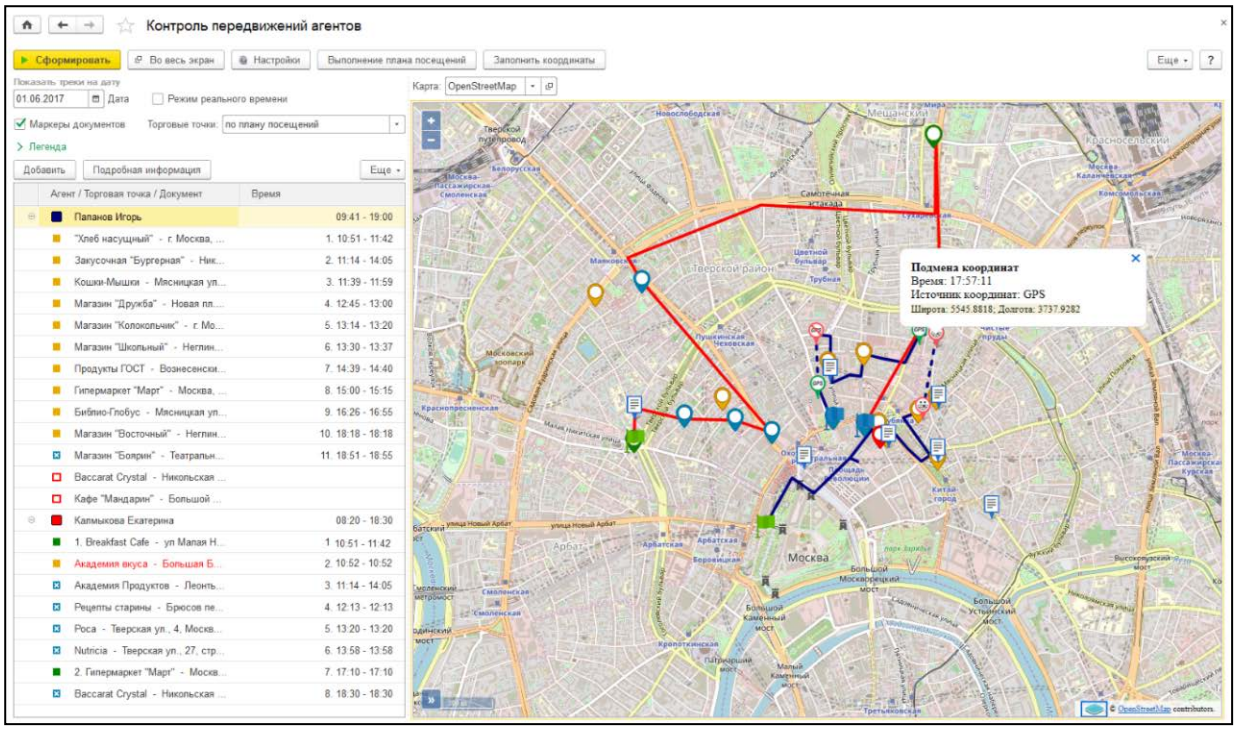


Рисунок 1.6 – Сервіс “Агент Плюс”

Також більшість сервісів працюють з конкретним GPS-пристроєм або зі своєю програмою, що не дає змогу використати продукт для інших в чомусь більш вдалих реалізаціях програм для самих торгових агентів.

Програма також повинна бути більш універсальною, щоб мати змогу її підключати у вже існуючі рішення для конкретних задач.

Після аналізу аналогів було розглянуто всі позитивні і негативні функції існуючих рішень і сформовані вимоги до майбутньої системи.

Тому в програмі потрібно розробити можливість відображати список агентів, з можливістю змінювати маршрути одного агента на другого. Також повинні бути розроблені фільтри по відображенню, такі як: показати/сховати планований маршрут, показати/сховати фактичний маршрут і відобразити точки заявок та точки зупинок торгового агента.

1.4 Постановка задачі

Потрібно розробити систему для аналізу і побудови маршрутів торгових агентів. Також необхідно розробити алгоритми визначення контрольних точок маршрутів для подальшого їх аналізу.

Також метою дослідження є виявлення існуючих методів аналізу маршрутів і на основі них побудови нового алгоритму, який дозволить більш точно відобразити на карті контрольні точки маршрутів торгових агентів.

Програма має бути проста у використання та легко інтегрована в існуючі системи, з мінімальним зміненням основної системи обліку та автоматизації.

Розроблена програма має дозволяти працювати з декількома торговими агентами, розрізняючи їх маршрути та акумулювати зібрану інформацію.

Маршрути агентів потрібно відображати на карті та інтерактивно змінювати фільтри, щоб динамічно відображати потрібну для користувача інформацію в програмі.

В програмі мають бути фільтри, що дозволять переглядати маршрут для одного торгового агента за заданий день, а також за частину дня. Також має бути можливість користувачем відображати або приховувати стоянки торгових агентів. В налаштуваннях програми має бути можливість встановлювати критерій, що буде визначати тривалість зупинки торгового агента, щоб відобразити мітку стоянки.

Програма має вміти працювати з онлайн картами для побудови планового маршруту по заданим координатам торгових точок, а також необхідно виводити на карту фактичний маршрут по заданим точкам зібраних торговим агентом під час своєї роботи на маршруті.

Також потрібно виводити спеціальні маркери, що будуть відображати стоянки та торгові точки маршрута, з можливістю перегляду детальної інформації при натисненні на маркер.

Для зручності користувачів додати можливість зберігати карту на пристрої для можливості перегляду попередніх маршрутів без підключення до мережі.

Список торгових агентів з якими працює програма має вводитися та динамічно змінюватися в програмі.

Програма повинна мати простий та зручний інтерфейс, а також швидко працювати та легко інтегруватися в існуючі системи.

Розроблена програма має допомогти супервайзерам контролювати торгових агентів, переглядаючи їхню активність на маршруті та порівнюючи плановані маршрути з фактичними маршрутами.

На карті мають бути відображено точки маршрути які торговий агент повинен відвідати, а також інформація про ці точки, така, як назва торгової точки, адреса, час відвідування, коментар для агента, а також можливо графік роботи та інша потрібна інформація.

Висновки до розділу 1

За результатами першого розділу було проаналізовано існуючі наукові статті по темам пов'язаними з побудовою маршрутів перевезень та маршрутів для самих торгових агентів.

Також було проаналізовано перелік існуючих програмних продуктів зі схожим функціоналом для виявлення сильних та слабких сторін в кожного з них.

Після цього було сформовано уточнення до майбутньої системи аналізу та планування маршрутів та визначенні ключові аспекти функціювання системи планування.

Розділ 2

Розробка інформаційної технології аналізу маршрутів торгових агентів

2.1 Загальний опис технології аналізу маршрутів торгових агентів

Розглянемо такий критерії побудови маршруту торгового представника, як найкоротша відстань.

При плануванні маршруту, в програму відразу будуть занесені торгові точки, які необхідно відвідати. Кожен раз мапа буде показувати найкоротший маршрут до наступної точки. Навіть, якщо в маршрутний лист в режимі реального часу були внесені зміни, програма це врахує.

Відвідування клієнтів відбувається в порядку їх пріоритетності, з урахуванням необхідної частоти візитів в точку і в залежності від поставлених конкретних завдань. Враховуються місця скупчення торговельних точок, де торговий представник може максимально охопити точки продажів. Програма складання маршруту – це відмінне рішення для дистриб'юторів по збільшенню кількості продажів, підвищенню ефективності роботи торгових агентів при пересуванні[8].

Однією з переваги програм з планування маршруту є маршрутизація торгових представників, яка має максимальну ефективність при використанні програм планування маршруту в комплексі. Перевагами є об'єктивна оцінка потенціалу території здійснюється на підставі аналізу щільності населення, присутності цільових груп та ін.

Балансування території дозволяє найбільш ефективно розпланувати відвідування торгових точок по днях і по торговим агентам. Для реалізації цього завдання використовуються власні розробки і вимоги бізнесу. Оптимізація маршрутів відрізняється точністю і актуальністю даних, оскільки процес проходить на розробленому картографічному сервісі. Його особливістю є можливість обліку мінливої інфраструктури території.

Контроль і коригування маршруту відрізняються наочністю і простотою експлуатації. Використання автоматизованих сервісів планування маршрутів вигідно для всіх. Керівники і менеджери за допомогою цього інструменту: Повністю контролюють роботу польових співробітників, а також підвищують ефективність їх роботи, здійснюють якісне планування.

Контроль торгових представників, можливо, створює певний дискомфорт у польових співробітників, але найбільшим плюсом є грамотне прокладання маршруту, що дозволяє торговому представнику отримати максимальний прибуток і виконати всі поставлені завдання. Керування роботою мерчендайзерів, торгових представників за допомогою поточної системи здатне підвищити ефективність їх роботи приблизно на 20%. З урахуванням того, що маршрути стають більш компактними, збалансованими, економиться робочий час співробітників. Відповідно, менеджери можуть розробляти інші території. Спостерігається зниження витрат на ПММ приблизно на 30%[9; 10].

За допомогою автоматизованого планування пересування польових співробітників, можна контролювати роботу виїзної торговельної команди, відстежуючи активність співробітників безпосередньо на карті. Наприклад, можна легко оцінити покриття території або відобразити співвідношення часу в дорозі до часу, витраченого на роботу в торговій точці по кожному окремому співробітнику або команді в цілому. Формування маршруту за допомогою програмного забезпечення суттєво економить час, зменшує транспортні витрати, підвищує продажі. При правильному плануванні, такий маршрут торгового представника є найбільш ефективним.

Функціонал для програми управління:

- побудова оптимального маршруту;
- розрахунок планової відстані;
- визначення часу в дорозі;
- планування пересування польових співробітників; к
- контроль мобільних співробітників;

- графічне відображення треку переміщення агента на карті;
- побудова оптимального маршруту по декількох точках;
- рішення аналітичних задач.

В даний момент існують декілька технологій, призначених для визначення місцезнаходження. Системи, що базуються на цих технологіях, можуть працювати як назовні, так і в будівлях.

Однак місце розташування пристрою у вигляді координат необхідно обробити і привести до зручного виду. Для ефективної обробки таких даних про місцезнаходження і побудови маршрутів призначені різні навігаційні сервіси. Подібні сервіси прив'язують дані про розташування до спеціально обробленими картами місцевості або будівель, на яких потім будуються маршрути. Для підвищення зручності такі сервіси часто випускаються так само і у вигляді додатків для смартфонів, в яких користувач може на свій розсуд редагувати необхідні йому карти. Ключовою властивістю подібних сервісів є коригування даних про карти, яка дозволяє швидко і своєчасно вносити зміни, що в свою чергу забезпечує постійну актуальність інформації.

Одним із способів пошуку оптимальних маршрутів називають завданням комівояжера. Алгоритми для вирішення завдання комівояжера можна розділити на точні (exact algorithm) і неточні (non-exact algorithm). Точні алгоритми включають в себе перебір всіх можливих варіантів, в окремих випадках рішення можуть бути швидко знайдені, але в цілому здійснюється перебір. Другі в загальних випадках застосовуються для задач, які неможливо вирішити точно (обчислення певних інтегралів, рішення нелінійних рівнянь, витяг квадратного кореня та інші), якщо існуючі точні рішення вимагають значних і невиправданих витрат часу при високій складності завдання, і як частина більш складного алгоритму, з допомогою якого завдання вирішується точно.

2.2 Метод формування проаналізованих маршрутів

Польові співробітники поза офісом працюють близько 90% робочого часу, відвідуючи торгові точки. Кількість відвідувань в день може бути десять і більше. Але маршрут повинен бути побудований грамотно, інакше ефективність роботи значно падає, а відповідно скорочується і прибуток. Тому планування пересування польових співробітників є одним з ключових аспектів ефективної організації їх діяльності. Врахувати всі точки призначення самостійно може бути досить складно, особливо, якщо в процесі будуть корегування.

Сучасні навігаційно-картографічні системи мають багато різних форм і уявлень, починаючи з онлайнових сервісів, доступних на будь-якому комп'ютері, і закінчуючи мобільними пристроями, які завжди можна взяти з собою.

Використання спеціальних програм дозволяє:

- Збільшити покриття території;
- ефективно використовувати робочий час;
- скоротити логістичні витрати;
- оптимізувати склад польової команди;
- контролювати мобільних працівників;
- побудувати маршрут з проміжними точками.

Використання статистичних даних, таких як: щільність населення, цільові групи споживачів, купівельна спроможність, для оцінки та розвитку території з метою ефективного використання ресурсів торгової команди допомагає оцінити потенціал території.

Розуміючи існуючі можливості території, менеджер знаходить оптимальний баланс для поділяння всієї території за днями та за торговими представниками, тобто зробити балансування території

Для оптимального використання часу менеджер розпланувує розбивку території і створюємо маршрутний лист торгового представника для кожного польового працівника в рамках його денного маршруту. Він використовує

програму з картографічний сервісом, яка гарантує актуальність даних, враховує мінливу інфраструктуру території.

Використовуючи моніторинг співробітників по GPS можна відстежувати місцезнаходження мобільних співробітників по GPS в режимі реального часу. Будь-якої хвилини ви можна побачити де знаходиться агент, які точки він вже відвідав і які йому ще належить відвідати[14; 15].

Перевагам від використання контролю і моніторингу з використанням GPS є:

- контроль часу і маршруту переміщення торгових агентів;
- збільшення кількості відвідуваних агентами торговельних точок в результаті раціонального планування;
- надання інформації про місцезнаходження торгових агентів в реальному часі;

аналіз результатів роботи торгових агентів.

GPS трек необхідний для візуалізації маршруту переміщення торгового агента за день і для побудови звіту по маршруту.

Для перегляду маршруту переміщення торгового агента на базі GPS треку агента можна побудувати візуальне відображення його переміщення.

Моніторинг агентів відбувається в реальному часі, супутникова система GPS моніторингу торгових агентів дозволяє в реальному часі визначити і показати поточне місцезнаходження всіх торгових агентів. На карті динамічно відображається поточний стан і місце розташування торгових агентів.

Контроль агентів за допомогою GPS дуже ефективний при зборі мерчендайзинговой інформації. Мерчендайзинг вимагає обов'язкової присутності агента в торговій точці.

Моделі візуалізації маршрутів лежать в основі методики аналізу даних, що застосовуються в запропонованому підході, підтримують описану схему аналітичного процесу. Ключовими елементами методики є нейронні мережі SOM, що використовуються для формування груп співробітників, що мають

схожі шаблони переміщень, і теплові карти, які застосовуються для виявлення періодів аномального поведінки. Вони доповнюються двома моделями візуалізації - графом контрольованих зон і вкладеної моделлю візуалізацією BandView, що відбиває послідовність відвіданих зон і тривалість перебування в них [15].

Транспортна логістика вирішує завдання, пов'язані з організацією доставки різного роду речей з одних місць в інші по оптимально обраному маршруту. На перший погляд може здатися, що рішення таких задач пов'язано суто з організаційними проблемами. На практиці таке судження призводить до зайвих витрат і обертається зниженням прибутковості логістичних підприємств. Саме вказівка на використання оптимального маршруту доставки змушує підключати до вирішення цієї проблеми теорію. Теоретичною основою логістики служать такі дисципліни як теорія соціально-економічних систем, теорія організацій і системотехніка, а методологічну основу для вирішення різних логістичних проблем забезпечують дисципліни: математика, дослідження операцій, системний аналіз, економетрія, менеджмент.

Висновки до розділу 2

В поточному розділі було розглянуто методи відображення маршрутів (списком та на карті), було переглянуто необхідність обробляти дані GPS для більш точного аналізу маршруту і побудови контрольних точок торгового агента.

Також було розглянуто, як за допомогою планування маршруту та відслідковування торгового агента можна збільшити ефективність продаж.

Розділ 3

Розробка методів та компонентів системи для планування та відображення маршрутів торгових агентів

3.1 Розробка методу взаємодії системи із допоміжними додатками для збору координат пересування торгових агентів

На пристрої торгового агента має бути встановлений додаток, який буде збирати точки пересування торгового агента.

Дані про пересування торгових агентів будуть зберігатися або у файлі на FTP або на сайті з Api модулем, який буде віддавати точки по запиту.

Також програма повинна зберігати дані про пересування в локальній базі даних для оперативного відображення даних по маршрутам.

Розроблена програма повинна мати змогу зчитувати дані з файлу у форматі JSON, XML або CSV, який може бути архівований. Також програма повинна мати можливість отримувати дані із зовнішнього сайту за допомогою API методів.

Для заповнення планових має бути розроблена бібліотека, що дозволить обмінюватися даними з обліковою системою. За допомогою неї має бути можливість заповнити всю інформацію по плановому маршруті, а також за необхідністю заповнити фактичний маршрут, це потрібно для полегшення інтеграції в системи, де дані про пересування агента передаються напряму в облікову систему.

Для зручності перегляду маршрути потрібно відображати візуально на карті. Цей тип відображення буде зрозумілим для користувача, на відміну від списків або таблиць карта дозволяє візуально оцінити відхилення від маршрута, або загальну інформацію, що дозволить на основі цих даних скоригувати наступні маршрути.

На карті має бути можливість відображати фактичні та планові маршрути окремо, а також відразу на одній карті, для того, щоб можна було візуально

порівняти плановий маршрут з фактичним. Планові маршрути та фактичні маршрути мають візуально відрізнятися.

Інтерфейс програми повинен бути простим та зрозумілим для користувача.

В програмі має бути можливість відображати список агентів, з можливістю перемикати маршрути з одного агента на другого. Також повинні бути фільтри по відображенню, такі як: показати/сховати планований маршрут, показати/сховати фактичний маршрут, відобразити точки заявок та точки зупинок торгового агента.

Розроблена програма має допомогти супервайзерам контролювати торгових агентів, переглядаючи їхню активність на маршруті та порівнюючи плановані маршрути з фактичними маршрутами[16].

На карті мають бути відображено точки маршрути які торговий агент повинен відвідати, а також інформація про ці точки, така, як назва торгової точки, адреса, час відвідування, коментар для агента, а також можливо графік роботи та інша потрібна інформація.

Зупинки торгового агента також повинні відображати на карті. Період зупинки, який буде відображатися повинен задаватися в налаштуваннях, наприклад якщо торговий агент зупинився на 15 хв, то ми будемо вважати, що це стоянка і її потрібно відобразити на карті.

Також на карті мають відзначатися точки оформлення замовлення, якщо ці данні будуть передаватися з мобільного пристрою. Це дозволить запобігти тому, що торговий агент може взяти замовлення дистанційно.

В програмі має бути можливість переглядати маршрути торгових агентів за попередні періоди, за допомогою фільтра по даті. Для зручності відображення маршрут повинен виводитися за один день, який вибере користувач.

Також повинен бути фільтр по часу, щоб можна було вивести маршрут за певний проміжок часу.

Отже, після перегляду існуючих рішень було визначено програмні засоби та технології для реалізації проекту.

3.2 Проектування програмного забезпечення для аналізу та побудови маршрутів

3.2.1 Проектування архітектури майбутньої програми

Завдання проектування включають в себе дві складові: логічне і фізичне проектування програмних продуктів. Логічне проектування полягає в розробці класів для реалізації їх примірників - об'єктів. Для цього потрібно докладний опис полів і методів класів, а також зв'язків між ними. Для цього використовуються статичні діаграми класів і об'єктів, динамічні - послідовностей станів і кооперації. Фізичне проектування передбачає побудову програмних компонентів з раніше визначених класів і об'єктів та розміщення їх на конкретних обчислювальних пристроях. Розробляються на цьому етапі діаграми - компонентів і розгортання.

Розробка структури програмного забезпечення при об'єктному підході, на етапі проектування уточнюються поля і методи класів, а також відносини між класами. Все це знаходить відображення на діаграмі класів.

Інженерія програмного забезпечення для досягнення позитивних результатів вимагає застосування спеціалізованих методик управління, як самим процесом, так і всіма стадіями аналізу, проектування, реалізації та подальшого використання розробленого програмного продукту. В якості основи даного процесу розглянуто уніфікована мова моделювання UML (Unified Modeling Language) і способи його застосування для управління процесами розробки додатків в цілому.

Приступаючи до розробки кожної програми, слід мати на увазі, що вона, як правило, є великою системою, тому ми повинні вжити заходів для її

спрощення. Для цього таку програму потрібно розробити на частини, які називаються програмними модулями.

Пропонується для оцінки прийнятності програмного модуля використовувати більш конструктивні його характеристики: розмір модуля, міцність модуля, зчеплення з іншими модулями, рутинність модуля (незалежність від передісторії звертань до нього).

Розмір модуля вимірюється числом, скільки містяться в ньому операторів або рядків. Модуль не повинен бути занадто маленьким або занадто великим. Маленькі модулі призводять до громіздкої модульної структурі програми і можуть не окупати накладних витрат, пов'язаних з їх оформленням.

Великі модулі незручні для вивчення і змін, вони можуть істотно збільшити сумарний час повторних трансляцій програми при налагодженні програми. Зазвичай рекомендуються програмні модулі розміром від кількох десятків до кількох сотень операторів.

Міцність модуля - це міра його внутрішніх зв'язків. Чим вище міцність модуля, тим більше зв'язків він може сховати від зовнішньої по відношенню до нього частини програми і, отже, тим більший внесок в спрощення програми він може внести.

Функціонально міцний модуль - це модуль, що виконує (реалізує) одну яку-небудь певну функцію. При реалізації цієї функції такий модуль може використовувати і інші модулі. Такий клас програмних модулів рекомендується для використання.

Інформаційно міцний модуль - це модуль, що реалізує кілька операцій над однією і тією ж структурою даних, яка вважається невідомою поза цим модуля. Для кожної з цих операцій в такому модулі є свій вхід зі своєю формою звернення до нього.

Такий клас слід розглядати як клас програмних модулів з вищим ступенем міцності. Інформаційно міцний модуль може реалізовувати, наприклад, абстрактний тип даних.

Архітектура являє собою структуру програми, що визначає її роботу на найвищому концептуальному рівні, включаючи апаратні і програмні компоненти, видимі зовні властивості цих компонентів, відносини між ними, а також документування системи. Документування архітектури спрощує процес взаємодії між учасниками проекту, що дозволяє зафіксувати прийняті на ранніх етапах проектування рішення про високорівневий дизайн системи і використовувати елементи цього дизайну і шаблони повторно в інших проектах.

Розробка архітектури є дуже важливим етапом, бо якщо розроблена архітектура не буде реалізовувати поставлені замовником мети, то це може, наприклад, збільшити терміни виконання проекту (за рахунок того, що необхідно буде робити доопрацювання щодо виправлення недоліків архітектури), а отже, знизити прибуток за розробку.

На рівні розробки архітектури програми повинні вирішуватися основні завдання, важливі для замовника.

Одним із завдань розробки архітектури є поліпшення і підвищення продуктивності процесів, що включає в себе зменшення часу, що витрачається на виконання різних дій; прискорення виконання операцій; автоматизація процесів; різні поліпшення, одержувані за рахунок масштабованості, яка допоможе справлятися зі збільшенням робочого навантаження (збільшувати свою продуктивність) при додаванні ресурсів, зазвичай апаратних.

Також спроектована архітектура дозволить зменшити витрати. Однією з цілей розробки може стати зменшення витрат, необхідних при здійсненні будь-яких дій. Це може здійснюватися як за рахунок підвищення продуктивності процесів, так і за рахунок прискорення виконання операцій.

Розроблена архітектура дозволяє зменшити ризики при розробці програмного забезпечення. Також архітектурне рішення може мати на меті підвищення ефективності управління[17; 18].

Для уявлення архітектури (точніше до різних структур, що входять до неї) зручно використовувати графічні мови. На даний момент найбільш

опрацьованим і найбільш широко використовуваним з них є уніфікована мова моделювання (Unified Modeling Language, UML), хоча на високому рівні абстракції архітектуру системи зазвичай описують просто набором іменованих прямокутників, з'єднаних лініями і стрілками, що представляють можливі зв'язки.

UML пропонує використовувати для опису архітектури 8 видів діаграм. Не всяка діаграма на UML описує архітектуру - 9-й вид діаграм, діаграми варіантів використання, не належать до архітектурних уявленням, крім того, і інші види діаграм можна використовувати для опису внутрішньої структури компонентів або сценаріїв дій користувачів і інших елементів, до архітектури не відносяться.

Визначення типу додатки є дуже важливим при створенні архітектури. При проектуванні приділяють окрему увагу питанням надання інформації користувачу, а також взаємодії користувача з системою, що найбільшою мірою визначається типом програми і видається замовником у вигляді вимоги. В даний час існує велика кількість різних програмних рішень, які дозволяють розробнику вибрати найбільш оптимальне для конкретної проблеми.

Дана програма має розроблятися, як класичні desktop-додатки. Такі додатки також називають «віконними». У них використовується графічний інтерфейс і введення інформації здійснюється за допомогою клавіатури і миші. Дані програми дозволяють створювати практично будь-які за складністю рішення. Основним мінусом є складність оновлення програм у кінцевих користувачів.

Інколи важко визначити різницю між веб-і настільними додатками, тому що веб-програми можуть запускати настільні програми, а програми для робочого столу можуть надсилати та отримувати дані через Інтернет.

Найчастіше програмні продукти мають певну конструкцію (архітектуру) побудови - склад і взаємозв'язок програмних модулів, тобто володіють

внутрішньою організацією або внутрішньою структурою, утвореної взаємопов'язаними програмними модулями.

Модуль - це самостійна частина програми, що має певне призначення і забезпечує задані функції обробки автономно від інших програмних модулів. Модуль складається з логічно взаємопов'язаної сукупності функціональних елементів.

Структуризація програм виконується в першу чергу для зручності розробки, програмування, налагодження та внесення змін до програмний продукт.

Структуризація програмних продуктів переслідує такі цілі: розподілити роботи по виконавцям, забезпечивши прийнятну їх завантаження і необхідні терміни розробки програмних продуктів; побудувати календарні графіки проектних робіт і здійснювати їх координацію в процесі створення програмних виробів; контролювати трудовитрати і вартість проектних робіт та ін.

Структурне розбиття програм на окремі складові є основою і для вибору інструментальних засобів їх створення, хоча має місце і зворотний вплив - вибір інструментальних засобів розробника ПЗ визначає типи програмних модулів.

При створенні програмних продуктів виділяються багаторазово використовувані модулі, проводиться їх типізація і уніфікація, за рахунок чого скорочуються терміни і трудовитрати на розробку програмного продукту в цілому.

Деякі програмні продукти використовують модулі з готових бібліотек стандартних підпрограм, процедур, функцій, об'єктів, методів обробки даних.

В роботі програмного продукту активізуються необхідні програмні модулі. Керувальні модулі задають послідовність виклику на виконання чергового модуля.

Інформаційний зв'язок модулів забезпечується за рахунок використання загальної бази даних або міжмодульної передачі даних через змінні обміну.

Кожен модуль може оформлятися як самостійно зберігається файл; для функціонування програмного продукту необхідна наявність програмних модулів в повному складі.

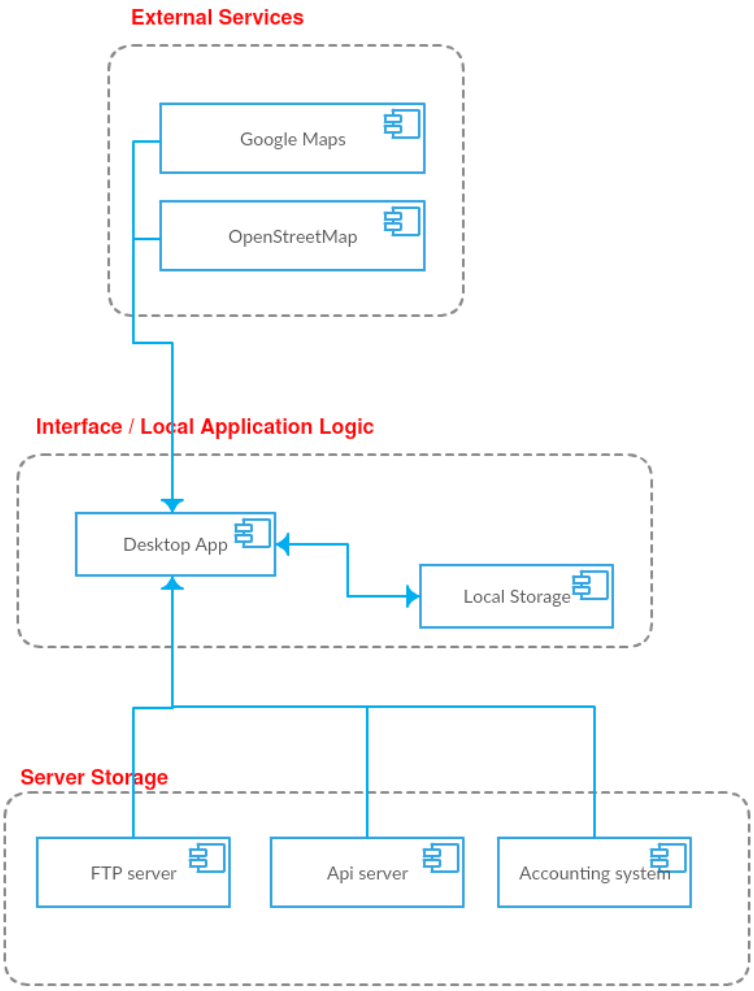


Рисунок 3.1 – Діаграма компонентів

Архітектуру програмного продукту можна схематично зобразити UML діаграмами, наприклад на рисунку 3.1 зображено діаграму компонентів, яка використовується для моделювання фізичних аспектів системи.

Діаграма компонентів не описує функціональність системи, але описує компоненти, які використовуються для створення цих функцій.

З даної діаграми можна дізнатися з яких частин складається програмний продукт. Програма буде використовувати сервіси Google maps та OpenStreetMap

для відображення карти. Дані про переміщення торгових агентів можуть передаватися через файл на FTP, Api-сервер або з облікової системи.

Програма матиме свою локальну базу даних для збереження всіх даних, які потребуватиме програма для коректної роботи.

Після проектування програми було сформовано список компонентів та основні функціональні особливості програми та архітектуру.

Детальне проектування - це повний обсяг робіт з проектування, виключаючи архітектуру і реалізацію. Воно містить визначення класів предметної області і класів архітектури. Детальне проектування - це технічна діяльність, яка слідує за етапом вибору архітектури. Основною метою цієї діяльності є якомога більш повна підготовка проекту до його реалізації, тобто до створення програмного коду.

Існує певний взаємозв'язок варіантів використання, архітектури та детального проектування . Варіанти використання є частиною вимог, відштовхуючись від яких розробники вибирають архітектуру, після чого вони розробляють детальний проект для реалізації необхідних варіантів використання з урахуванням обраної архітектури.

Завдання етапу детального проектування включають аналіз відповідності обраного для детальної розробки варіанту проекту загальним вимогам з урахуванням повного і детального аналізу підсистем і виявити при цьому відхилення в зв'язку з прийняттям компромісних інженерних рішень. На цьому ж етапі аналізується реалізованість проекту[20].

На першому кроці проектування ПО варіанти використання фіксуються як частина вимог. На другому кроці проектування варіанти використання використовуються для визначення класів предметної області. На третьому кроці - розробляється програмна архітектура, і додаються відповідні класи проекту. Четвертий крок полягає в перевірці того факту, що архітектура і детальний проект задовольняють необхідним варіантів використання.

Перш ніж починати детальне проектування, програмних засобів, необхідно визначити, які функції краще виконуються за допомогою програмного забезпечення мікрокомп'ютера, а які - за допомогою апаратних засобів. Всі функції попередньо повинні бути розподілені між програмними і апаратними засобами. Бажано, щоб з цього моменту в структуру проекту не вносилися ніякі зміни, хоча на практиці це рідко вдається.

Після завершення етапу детального проектування зазвичай створюється дослідний зразок або макет, який піддається всебічному критичному обговоренню.

Структурно-складні програмні продукти розробляються як пакети програм, і найчастіше вони мають прикладний характер - пакети прикладних програм.

Більшість програмних продуктів, особливо прикладного характеру, орієнтованих на кінцевого користувача, працюють в діалоговому режимі взаємодії з користувачем таким чином, що ведеться обмін повідомленнями, що впливають на обробку даних.

Графічний інтерфейс користувача є обов'язковим компонентом більшості сучасних програмних продуктів, орієнтованих на роботу з кінцевим користувачем. До графічного інтерфейсу користувача пред'являються високі вимоги як з чисто інженерної, так і з мистецького боку розробки, при його розробці орієнтуються на можливості людини.

Найбільш часто графічний інтерфейс реалізується в інтерактивному режимі роботи користувача для програмних продуктів, що функціонують в середовищі Windows, і будується в вигляді системи розкривних меню з використанням в якості засобу маніпуляції миші і клавіатури.

Робота користувача здійснюється з екранними формами, що містять об'єкти управління, панелі інструментів з піктограмами режимів і команд обробки.

До основних принципів структурної методології відноситься принцип абстракції. Абстракція дозволяє програмісту уявити необхідну рішення проблеми без одномоментного обліку безлічі деталей. Використовуючи принцип абстракції, розробник може розглядати програму по рівням.

Верхній рівень показує велику абстракцію, спрощує погляд на проект, в той час як нижній рівень показує дрібні деталі реалізації.

Перший крок розробки програмного модуля в значній мірі є суміжний контроль структури програми знизу: вивчаючи специфікацію модуля, розробник повинен переконатися, що вона йому зрозуміла і достатня для розробки цього модуля.

На завершення цього кроку вибирається мова програмування: хоча мова програмування може бути вже визначений для всього ПЗ, все ж в ряді випадків (якщо система програмування це допускає) може бути вибрана інша мова, яка більше підходить для реалізації даного модуля.

На другому кроці розробки програмного модуля необхідно з'ясувати, чи не відомі чи є вже якісь алгоритми для вирішення поставленого і чи близькою до неї завдання. І якщо знайдеться відповідний алгоритм, то доцільно їм скористатися. Вибір відповідних структур даних, які будуть використовуватися при виконанні модулем своїх функцій, в значній мірі визначає логіку і якісні показники розроблюваного модуля, тому його слід розглядати як вельми відповідальне рішення.

На третьому кроці здійснюється побудова тексту модуля на обраною мовою програмування. Велика кількість всіляких деталей, які повинні бути враховані при реалізації функцій, зазначених у специфікації модуля, легко можуть привести до створення досить заплутаного тексту, що містить масу помилок і неточностей.

Шукати помилки в такому модулі і вносити в нього необхідні зміни може виявитися досить трудомістким завданням. Тому дуже важливо для побудови

тексту модуля користуватися технологічно обґрунтованої і практично перевіреної дисципліною програмування.

Наступний крок розробки модуля пов'язаний з приведенням тексту модуля в завершеному вигляді відповідно до специфікації якості ПЗ. При програмуванні модуля розробник основну увагу приділяє правильності реалізації функцій модуля, залишаючи недопрацьованими коментарі і допускаючи деякі порушення вимог до стилю програми.

При шліфуванні тексту модуля він повинен відредагувати наявні в тексті коментарі і, можливо, включити в нього додаткові коментарі з метою забезпечити необхідні примітиви якості. З цією ж метою проводиться редагування тексту програми для виконання стилістичних вимог.

Крок перевірки модуля являє собою ручну перевірку внутрішньої логіки модуля до початку його налагодження (використовує виконання його на комп'ютері), реалізує загальний принцип, сформульований для обговорюваної технології програмування, про необхідність контролю прийнятих рішень на кожному етапі розробки ПЗ.

І, нарешті, останній крок розробки модуля означає завершення перевірки модуля (за допомогою компілятора) і перехід до процесу налагодження модуля.

Застосування профілю захисту інформації на стадії детального проектування полягає в тому, щоб структурувати розподіл функцій захисту і реалізують їх механізмів між компонентами системи, які визначаються при деталізації її структури. Кожній групі функцій профілю захисту інформації повинні відповідати конкретні компоненти системи, відповідальні за виконання цих функцій. Відмінності в уразливості різних компонентів по відношенню до зовнішніх і внутрішніх негативних впливів, що впливає на інформаційну безпеку, визначають різні вимоги до цих компонентів.

3.2.2 Проектування інтерфейсу користувача програми для побудови маршрутів

Призначений для користувача інтерфейс (UI) - це спосіб, яким виконується якась задача за допомогою будь-якого продукту, а саме здійснюванні дії і те, що отримується у відповідь.

Програмний інтерфейс не тільки вирішує нашу проблему взаємодії з додатком, а й робить це взаємодія максимально комфортним. Важлива наявність інтерфейсу, що дозволяє при меншій кількості зусиль ознайомитися з можливостями програми та зрозуміти принципи роботи в ньому.

Інтерфейс має величезне значення для довільної програмної системи і є обов'язковою її складовою, спрямованої, перш за все, на кінцевого користувача. Якраз по інтерфейсу користувач судить про програму в цілому; до того ж, часто висновок про використання програмного забезпечення користувач сприймає по тому, в якій мірі йому комфортний і зрозумілий інтерфейс. Разом з тим, трудомісткість проектування та розроблення інтерфейсу досить чимала.

На основі наданого опису роботи інтерфейсу створюється список завдань (призначених для користувача сценаріїв), які може виконувати користувач в рамках інтерфейсу.

Як і розробка програми в цілому, так і генерування призначеного для користувача інтерфейсу для нього - процес ітераційний. Малоймовірно, що подібні етапи, як прототипування, конструювання та випробовування інтерфейсу можуть бути закінчені за один прохід. Тому, якщо внаслідок випробування виявлені недоробки, то вони, якщо таке можливо, усуваються шляхом вторинного конструювання або розробляється інший прототип інтерфейсу.

Залежно від потреб і ступеня готовності проекту можна проектувати: логіку, функціонал та графічне представлення.

Логіку - яким чином система вирішує проблеми користувачів, базовий рівень, з якого починається робота проектувальника.

Функціонал - яким чином людина взаємодіє з призначеним для користувача інтерфейсом сайту, що саме, в якому порядку і з використанням яких технічних засобів робить, як різні частини системи взаємодіють між собою.

Графічне представлення - візуалізація дизайну: розташування блоків, колірні і інші оформлювальні рішення, використання графіки для управління увагою.

На етапі подання формується скелет інтерфейсу і визначаються принципи і правила внутрішньої роботи системи і її взаємодії з користувачами. Фактично це найважливіший етап проектування, так як саме тут закладається базова логіка проекту.

Помилки і недоліки, допущені на цьому етапі, множаться в геометричній прогресії в міру подальшої роботи над проектом - і їх усунення в готовому продукті часто або неможливо в принципі, або невиправдано дорого.

Зазвичай для одного інтерфейсу розробляють кілька рівноцінних версій. В ході A/B-тестувань, за результатами інтерв'ю і онлайн опитувань вибираються рішення, які в більшій мірі відповідають бізнес-завдань і потреб аудиторії.

За результатами юзабіліті-тестів і на основі затвердженого з клієнтом прототипу створюється графічне оформлення інтерфейсу. На цьому ж етапі розробляється функціонал і бекенд проекту.

Інтерфейс повинен бути інтуїтивно зрозумілим. Таким, щоб користувачеві не потрібно пояснювати як ним користуватися.

Розроблена програма повинна складатися з основного вікна, з вікна налаштувань програми, а також з додаткових вікон та діалогів, які знадобляться.

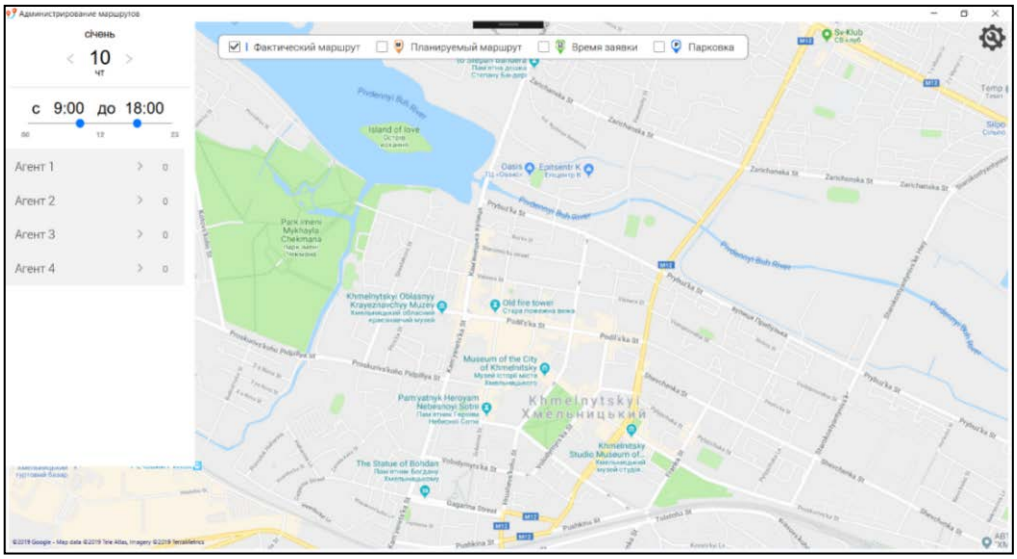


Рисунок 3.2 – Головне меню програми

На головному екрані додатка, що показаний на рисунку 3.2, зліва знаходиться список торгових агентів, а також фільтр по даті і часу.

Найбільш часто графічний інтерфейс реалізується в інтерактивному режимі роботи користувача для програмних продуктів, що функціюють в середовищі Windows. Робота користувача здійснюється з екранними формами, що містять об'єкти управління, панелі інструментів з піктограмами режимів і команд обробки.

Стандартний графічний інтерфейс користувача повинен відповідати ряду вимог: підтримувати інформаційну технологію роботи користувача з програмним продуктом - містити звичні і зрозумілі користувачеві пункти меню, що відповідають повноваженням обробки, розташовані в природній послідовності використання.

Орієнтуватися на кінцевого користувача, який спілкується з програмою на зовнішньому рівні взаємодії.

А також керуватися правилом "шести" - в одну лінійку меню включати не більше 6 понять, кожне з яких містить не більше 6 опцій.

І слідкувати, щоб графічні об'єкти зберігали своє стандартизоване призначення і по можливості розташування на екрані.

3.2.3 Аналіз та автоматизація обробки потоків даних та проектування структури бази даних

Для вирішення завдання функціонального моделювання на базі структурного аналізу застосовуються два типи моделей: IDEF0-діаграми і діаграми потоків даних.

Методологія розробки діаграми процесів зазвичай застосовується при проведенні обстежень підприємств в рамках проектів управлінського консалтингу, а також в проектах автоматизації великих об'єктів при експрес-обстеження (зазвичай для складання розгорнутого плану робіт).

Нотація діаграм потоків даних дозволяє відображати на діаграмі як кроки бізнес-процесу, так і потік документів і управління (в основному, управління, оскільки на верхньому рівні опису процесних областей значення має передача управління). Також на діаграмі можна відображати засоби автоматизації кроків бізнес-процесів. Зазвичай використовується для відображення третього і нижче рівня декомпозиції бізнес-процесів (першим рівнем вважається ідентифікований перелік бізнес-процесів, а другим - функції, що виконуються в рамках бізнес-процесів).

Діаграми потоків даних можуть доповнити те, що вже відображено в моделі IDEF0, оскільки вони описують потоки даних , дозволяючи простежити, яким чином відбувається обмін інформацією як усередині системи між бізнес-функціями, так і системи в цілому із зовнішнім інформаційним середовищем

У разі наявності в моделюється системі програмної/програмованої частини (практично завжди) перевага, як правило, віддається DFD з таких міркувань.

Якщо при моделюванні за методологією IDEF0 система розглядається як мережа взаємопов'язаних функцій, то при створенні DFD-діаграми система

розглядається як мережа пов'язаних між собою функцій, тобто як сукупність сутностей (предметів).

Структурний аналіз - це системний покроковий підхід до аналізу вимог і проектування специфікацій системи незалежно від того, чи є вона існуючої або створюється знову.

Оскільки проектування фактично закладає основи успішного або неуспішної реалізації завдання розробки продукту, необхідно жорстко контролювати процес проектування і вчасно виправляти невірні проектні рішення. З цією метою весь етап проектування ділять на дві великі стадії: розробка ескізного проекту, розробка технічного проекту.

На стадії ескізного проектування здійснюється основна робота з остаточного теоретичному і експериментальному на рівні моделей обгрунтування і опису пристрою і роботи програм. Основним видом роботи на стадії ескізного проектування є аналіз потоків даних, що дозволяє виділити основні структурні одиниці даних і методів, які здійснюють обробку інформації і перетворюють дані з однієї форми в іншу.

Поряд з розумінням того, які дані обробляються і в якій послідовності відбувається ця обробка, описуючи модель потоків даних, визначається безліч інформаційних потреб користувачів в даних, де під користувачем розглядаються не тільки люди, які використовують інформаційну систему для роботи з даними, а й сама інформаційна система, а точніше, її модулі та підсистеми, які звертаються до бази даних для отримання додаткових допоміжних відомостей.

Модель потоків даних може представлятися в двох варіантах: діаграма потоків даних (ДПД/ОРО), де вказуються завдання учасника процесу по перетворенню вихідних відомостей в результуючі, і діаграма робіт з даними, коли описуються операції обробки даних, які проводять в автоматизованій системі перетворення відомостей, включаючи їх збереження в базі даних або видалення.

У першому випадку, при формуванні діаграми потоків даних в рамках розгляду бізнес-процесу або допоміжного процесу, поява елементів "Сховище" та відображення завдань обробки даних призводить до необхідності їх відображення в дереві функцій, визначаючи необхідні структури даних для виконання відповідного процесу.

У другому випадку, при формуванні діаграми робіт, дерево функцій не поповнюється новими завданнями, але формується дерево зв'язків функцій, що відбиває послідовність використання операцій вибірки, додавання, модифікації і видалення в рамках застосування основних і допоміжних функцій, відображених в дереві функцій. Ця модель (дерево зв'язків функцій) дозволяє виділити укрупнений алгоритм перетворення даних і визначити функції, реалізація які дозволяють виконати відповідні операції.

У проектування структури бази даних під базою даних розуміється об'єктивна форма представлення та організації сукупності даних, систематизована таким способом, щоб ці дані могли бути знайдені і оброблені за допомогою найменшої кількості запитів.

Дійсно, процеси обробки інформації мають загальну природу і спираються на опис фрагментів реальності, виражене у вигляді сукупності взаємопов'язаних даних. Бази даних є ефективним засобом представлення структур даних і маніпулювання ними. Концепція баз даних припускає використання інтегрованих засобів зберігання інформації, що дозволяють забезпечити централізоване управління даними і обслуговування ними багатьох користувачів.

Інформаційні системи, що використовують бази даних, дозволили подолати обмеження файлових систем, такі як:

- надмірність даних;
- слабкий контроль;
- недостатні можливості управління даними;
- великі затрати праці програміста.

Всі питання, пов'язані з ефективним веденням і використанням бази даних, вирішує група фахівців, які називаються адміністраторами бази даних. Нові методи доступу до даних сильно спростили процес зв'язування елементів даних, що в свою чергу призвело до розширення можливостей роботи з даними. Всі ці характеристики систем управління базами даних спрощують процес програмування і зменшують необхідність програмної підтримки.

В даний час процес створення максимально потужних систем управління базами даних йде повним ходом. За кілька десятиліть послідовно з'являлися системи, засновані на трьох базових моделях даних, або концептуальних методах структурування даних: ієрархічної, мережевої і реляційної.

Бази даних (БД) складають в даний час основу комп'ютерного забезпечення інформаційних процесів, що входять практично в усі сфери людської діяльності.

Дійсно, процеси обробки інформації мають загальну природу і спираються на опис фрагментів реальності, виражене у вигляді сукупності взаємопов'язаних даних. Бази даних є ефективним засобом представлення структур даних і маніпулювання ними. Концепція баз даних припускає використання інтегрованих засобів зберігання інформації, що дозволяють забезпечити централізоване управління даними і обслуговування ними багатьох користувачів. При цьому БД повинна підтримуватися в середовищі ЕОМ єдиним програмним забезпеченням, що називається системою управління базами даних (СКБД). СУБД разом із прикладними програмами називають банком даних.

Проектування БД являє собою складний трудомісткий процес відображення предметної області у внутрішню модель даних. В процесі проектування розробляється моделі різних рівнів архітектури БД, перевіряється можливість відображення об'єктів однієї моделі об'єктами іншої моделі[21; 22].

Перший крок на шляху реалізації структури реляційної бази даних це нормалізація таблиць бази даних.

Процес приведення бази даних до виду, в якому вона буде відповідати правилам нормальних форм, називається нормалізацією бази даних.

Головна мета нормалізації бази даних — усунення надмірності та дублювання інформації. В ідеалі при нормалізації треба домогтися, щоб будь-яке значення зберігалося в базі в одному примірнику, причому значення це не повинно бути отримано розрахунковим шляхом з інших даних, що зберігаються в базі.

В таблицях поля мають формати, які дозволяють вказати формати виводу тексту, чисел, дат і значень часу на екран і на друк. Формат поля для елементів управління задається у вікні властивостей, а для поля в таблиці або запиті в режимі конструктора таблиці (в розділі властивостей поля) або у вікні запиту (у вікні властивостей поля). Формати можна вибирати зі списку вбудованих форматів для полів, що мають числовий, грошовий, логічний типи даних, а також типи даних лічильника і дати/часу. Також для будь-яких типів полів, відмінних від об'єктів OLE є можливість створення власних спеціальних форматів. Крім того, значення даного властивості можна задати в макросі або в програмі Visual Basic .

Зв'язок між таблицями встановлює зав'язки між співпадаючими значеннями в ключових полях, зазвичай між полями різних таблиць, що мають однакові імена. У більшості випадків з ключовим полем однієї таблиці, що є унікальним ідентифікатором кожного запису, зв'язується зовнішній ключ іншої таблиці.

Рисунок 3.3 – Структура таблиці «RoutePoint»

Для прикладу візьмемо таблицю «RoutePoint», яка зображена на рисунку 3.3, в якій знаходяться точки маршрутів торгового агента.

Для оптимізації було створено таблицю «Route», що зображена на рисунку 3.4, в котру було перенесено поля «TypeRoute», що відповідає за тип маршруту (плановий чи фактичний) і «Agent» в якому зберігається торговій агент, якому належить дана точка, так як вони є спільні для усього маршруту і їх не має потреби дублювати.

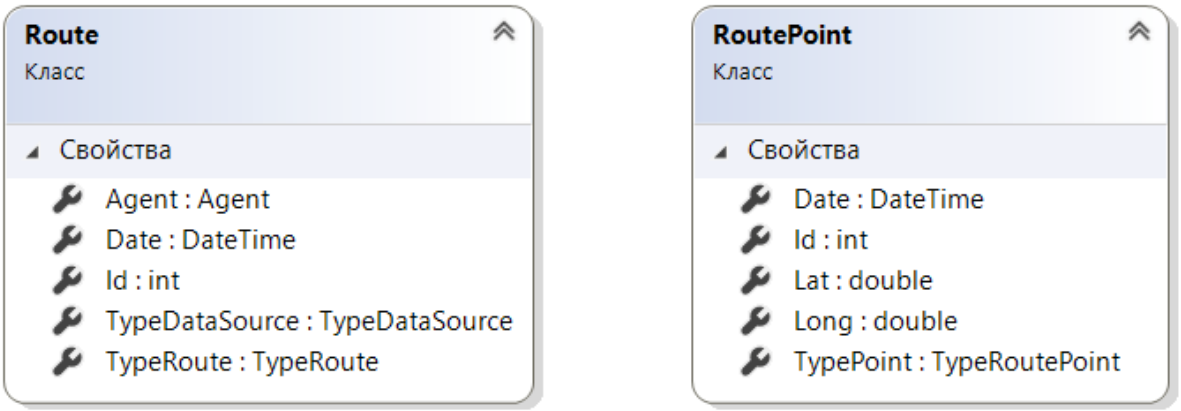


Рисунок 3.4 – Структура таблиці «RoutePoint»

В таблицю з точками додано поле «Route», що вказує в до якого маршруту відноситься дана точка, зображено на рисунку 3.5.

Цей зв'язок можна побудувати завдяки тому, що в нас поле «Id» в таблиці «Route» є ключовим полем.

Модель «сутність-зв'язок» передбачає кілька типів зв'язку: «один-до-одного», «один-до-багатьох», «багато-до-багатьох». Зв'язок «один-до-одного» означає, що в кожен момент часу кожному екземпляру сутності А відповідає 1 і тільки 1 екземпляр сутності В і навпаки. Зв'язок «один-до-багатьох» позначає, що одному представнику сутності А відповідають 0, 1 або кілька представників сутності В, але кожному екземпляру сутності В відповідає тільки 1 екземпляр сутності А. Зв'язок «багато-до-багатьох» показує, що одному представнику суті А відповідають 0, 1 або кілька представників сутності В і навпаки.

Для підтримки взаємозв'язку об'єктів в інформаційній моделі, між сутностями встановимо зв'язки між таблицями RoutePoint і Route.

Рисунок 3.5 – Створення зв’язку між таблицями

Для зв’язку між сервером баз даних і самим додатком буде використовуватись Entity Framework 6.0 це є ORM модель бази даних, і в ньому використовують три кардинально різні підходи для роботи з базою даних. Принцип роботи Entity Framework полягає в тому, що інформація надходить в

моделі, після того виконуються функції, вони повертають моделі в запити для запам’ятовування інформації, а потім виконуються запити, і навпаки при виведені інформації.

СУБД зазвичай працюють з БД значного розміру. Цей розмір істотно перевищує доступний об'єм оперативної пам'яті. При зверненні до будь-якого елементу даних проводиться обмін із зовнішньою пам'яттю, і система працює зі швидкістю пристрою зовнішньої пам'яті. Єдиним способом збільшення цієї швидкості є буферизація даних в оперативній пам'яті. Тому в СУБД підтримується набір буферів оперативної пам'яті з дисциплінами заміни буферів.

СУБД повинна забезпечувати надійне зберігання даних у зовнішній пам'яті, тобто СУБД повинна мати можливість відновити останній узгоджений стан БД після будь-якого апаратного або програмного збою.

Model First представляє один підхід до роботи з Entity Framework. Суть даного підходу полягає в тому, що спочатку робиться модель, а потім по ній створюється база даних.

У Entity Framework файли моделі і зіставлення слугують для виконання операцій створення, читання, оновлення та видалення, виконуваних над сутностями і зв'язками концептуальної моделі, в еквівалентні операції в джерелі даних. Entity Framework підтримує навіть зіставлення сутностей в концептуальної моделі з збереженими процедурами в джерелі даних.

Entity Framework дозволяє спростити розробку бази даних, дозволяє підключати різні реляційні бази даних без зміни бізнес-логіки програми, для цього достатньо налаштувати підключення до бази.

3.2.4 Аналіз та вибір технологій і методів реалізації програми для побудови маршрутів

На сьогоднішній день для вирішення поточної задачі є багато технологій і мов програмування, що дозволяють реалізувати потрібний функціонал.

Є три варіанта створення додатка – це стандартна desktop-програма, мобільний додаток або сайт.

Для поточної цілі найкраще підійде реалізація стандартної desktop-програми.

Вона має перевагу над мобільним додатком у тому, що комп’ютерну програму буде простіше інтегрувати у вже існуючі облікові системи, а також дозволить працювати з програмою на комп'ютері, одночасно з обліковою програмою, тобто не потрібно буде докупляти додаткове обладнання, наприклад планшети.

Веб-додаток також має право на існування, він дозволяє працювати на комп’ютері, а також можна відносно легко інтегрувати у існуючу облікову систему (методом Api запитів). Але desktop-програма на відміну від веб додатка, дозволяє працювати в offline-режимі, тобто завантажити або закешувати карту для перегляду без доступу до інтернету. Також розмістивши програму локально, на відміну від веб-додатка, можна добитися кращої продуктивності, та не залежити від хостинга і інтернету.

Для написання desktop-програми можна скористанися декількома мовами програмуваннями, наприклад Java, C#, C++/QT. Дані мови являються об'єктно-орієнтованими.

Java. На сьогоднішній день мова Java є одна з найбільш поширених і популярних мов програмування. Вона поширена в основному в мобільній розробці під Android, а також для написання великих Enterprise серверів. На цій також можна створювати програми різної складності під різні платформи.

C#. На сьогоднішній момент мова програмування C# є одною з найпотужніших, що швидко розвиваються і затребуваних мов в IT-галузі. На даний момент на ній пишуться найрізноманітніші програми: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

Мова програмування C++ високорівнева компільована мова програмування загального призначення зі статичної типізацією, яка підходить для створення самих різних додатків для різних платформ.

Для написання відносно невеликої програми найкраще підійде мова програмування C#, так як вона має велику кількість написаних бібліотек, для неї на мою думку найзручніший редактор, та вона дозволяє швидко створювати готові рішення різної складності. C# найкраще підходить для розробки під ОС Windows, але за необхідністю можна скористатися проектом Mono, який дозволить доробити додаток, щоб він запускався під ОС сімейства Linux.

Для написання програми буде використано технологію WPF (Windows Presentation Foundation), яка є частиною платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

Якщо при створенні традиційних додатків на основі WinForms за вивід елементів управління і графіки відповідали такі частини ОС Windows, як User32 і GDI +, то додатки WPF засновані на DirectX.

В порівнянні з додатками на Windows Forms обсяг програм на WPF і споживання ними пам'яті в процесі роботи в середньому трохи вище. Але це компенсується більш широкими графічними можливостями і підвищеною продуктивністю при відображенні графіки. В основі WPF лежить векторна система візуалізації, яка не залежить від розширення і створена з розрахунком можливостей сучасного графічного обладнання.

Для зберігання даних буде використана база даних MySql, так як вона є безкоштовною і підходить для наших потреб. Також буде можливість підключати інші бази даних.

Зв’язок з базою даних буде забезпечувати Entity Framework - це структура ORM (об'єктно-реляційне відображення), яку Microsoft надає як частину розробки .NET. Його мета - абстрагувати зв'язки з реляційною базою даних, таким чином, щоб розробник міг відноситись до об'єкта бази даних до набору об'єктів, а потім до класів на додатково до їх властивостей.

В основу суті даної технології покладено принцип поділу програмування і написання інтерфейсу. Введення мови XAML робить останнім максимально схожим на принцип Web-програмування.

Крім того, WPF, на відміну від класичних додатків на WinAPI і додатків, написаних з використанням WindowsForms, активно використовує безпосередньо ресурси відеокарти. З використанням технології WPF можна писати програми, що відображають тривимірну графіку, підключивши необхідні посилання.

Є можливість використання градієнтів в проектуванні користувацького інтерфейсу, що може зробити його барвистим і живим.

Після побудови програми Windows Presentation Foundation (WPF), вони повинні бути розгорнуті. Windows і .NET Framework підтримують кілька технологій розгортання. Технологія розгортання, яка використовується для розгортання програми WPF, залежить від типу програми.

Інсталятор Windows дозволяє упаковувати додатки як самодостатні виконувані файли, які можна легко поширювати на клієнтах і запускати. Крім того, інсталятор Windows встановлюється разом з Windows і підтримує інтеграцію з робочим столом, меню "Пуск" і компонентом панелі управління "Програми".

Автономні додатки розгортаються за допомогою ClickOnce або інсталятор Windows . У будь-якому випадку для запуску автономних додатків потрібно повна довіра. Повна довіра автоматично надається автономним додатків, які розгортаються за допомогою інсталятор Windows . Автономні програми, які розгортаються за допомогою ClickOnce , не отримують повної довіри автоматично. Замість цього ClickOnce виводить діалогове вікно з попередженням системи безпеки, яке користувач повинен підтвердити перед установкою автономного додатки.

Висновки до розділу 3

В поточому розділі була спроектована програма для аналізу та побудови маршрутів, було розділено програму на модулі. І була спроектована база даних для швидкого і отимального отримання потрібних результатів

Були побудовані діаграми UML для більш наглядної взаємодії модулів програми.

Також були визначені функціональні вимоги до майбутньої програми, а саме формати та способи обміну зі сторонніми системами(обліковими програмами).

Розділ 4

Дослідження ефективності системи для аналізу та планування маршруту торгового агента

4.1 Робота системи відображення маршрутів торгових агентів

В результаті було розроблено програму, що дозволяє переглянути маршрути торгових агентів з відображенням контрольних точок, а також переглянути плановий маршрут з інформацією по торговій точці.

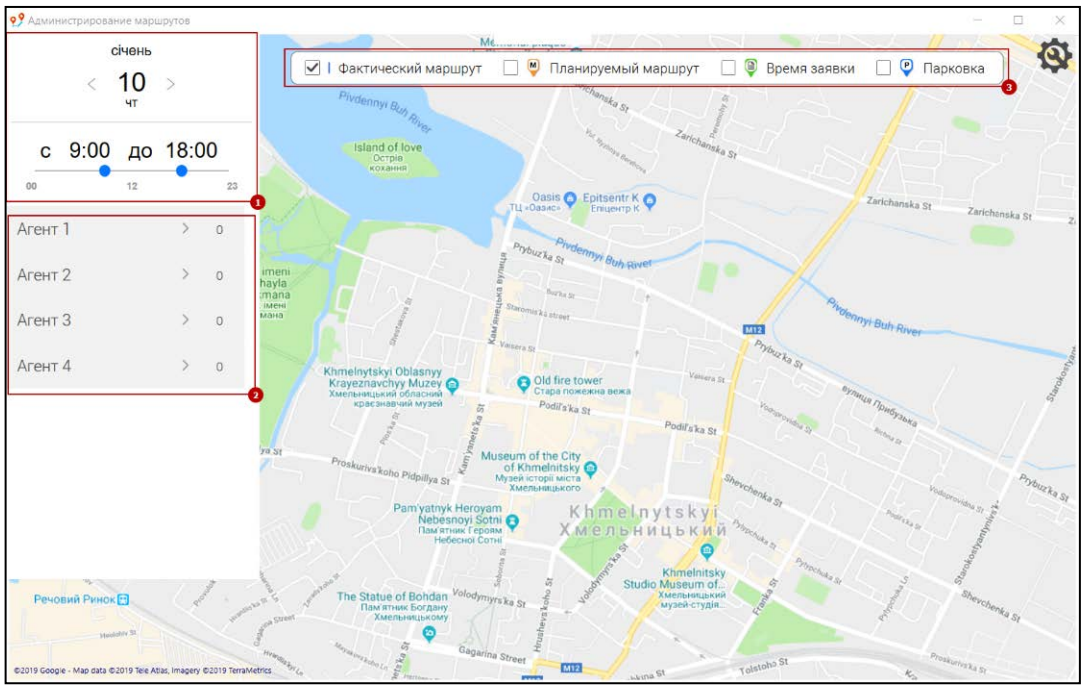


Рисунок 4.1 – Головне вікно програми

Програма складається з головного вікна, що зображено на рисунку 4.1, з вікна налаштувань та допоміжних діалогових вікон.

На рисунку 4.1 показано фільтри по даті і часі, які допомагають вибрати точки агента за певний період.

Під фільтром по даті знаходиться список торгових агентів при виборі яких відображаються їх точки.

Також у верхній частині додатка знаходиться фільтр по типу видимих точок, можна відобразити наступні типи точок:

– фактичний маршрут;

– плановий маршрут;

– точки заявок;

– точки парковок.

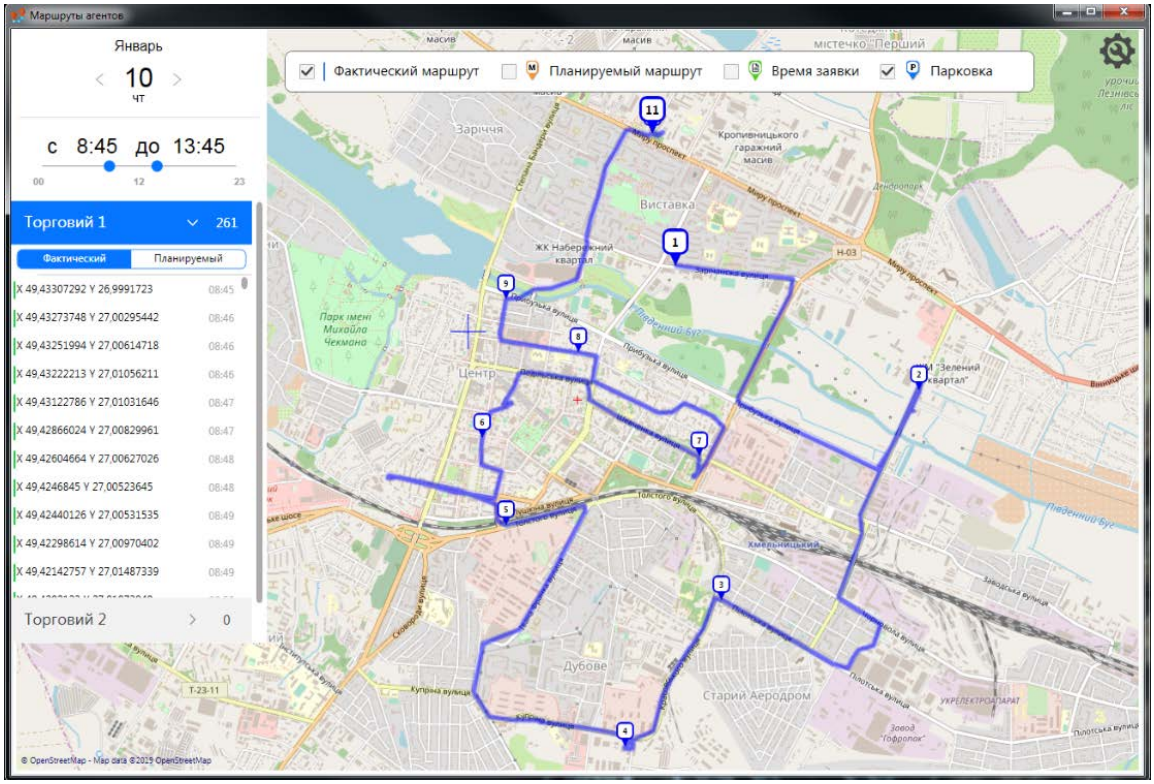


Рисунок 4.2 – Візуалізація маршруту

При виборі торгового агента відображається маршрут поточного агента, що зображено на рисунку 4.2. Також на маршруті можна відобразити точки парковок і проставити точки по маршруті кожні декілька хвилин.

Для налаштування додатку потрібно натиснути на кнопку «налаштування» у верхньому правому куті.

В додатку можна налаштувати імпорт торгових агентів і їхніх планових і фактичних маршрутів, для цього потрібно вказати «Ключ доступа» у формі, що показана на рисунку 4.3.

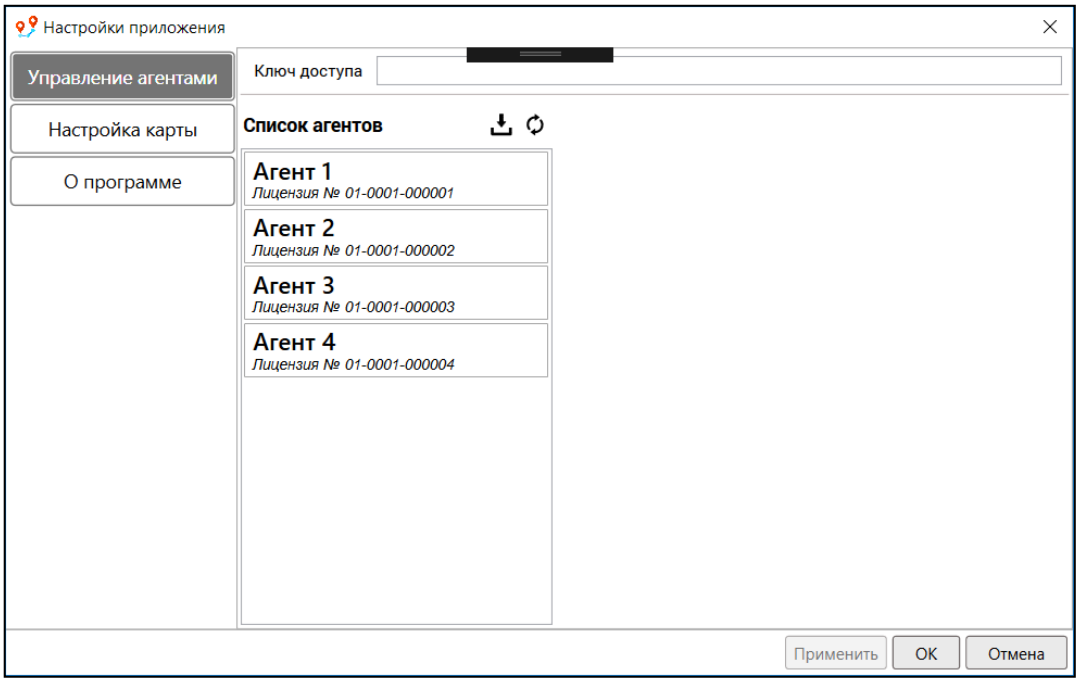


Рисунок 4.3 – Налаштування списку агентів

Також можна переназивати торгових агентів вибравши необхідного агента, а також можна приховувати неактивних агентів.

Для налаштування параметрів карти потрібно перейти на вкладку «Налаштування карти», налаштування зображені на рисунку 4.4.

При налаштуваннях карти можна змінити наступні параметри для зручнішої роботи з картою:

– тип карти(GoogleMap або OpenStreetMap);

– клавіша миші для переміщення карти;

– параметри для зупинки (при яких налаштуваннях буде ставитися маркер із зупинкою);

– маркування маршрута (встановлюється періодичність маркерів на маршруті);

– групування точок, для зручнішого перегляду.

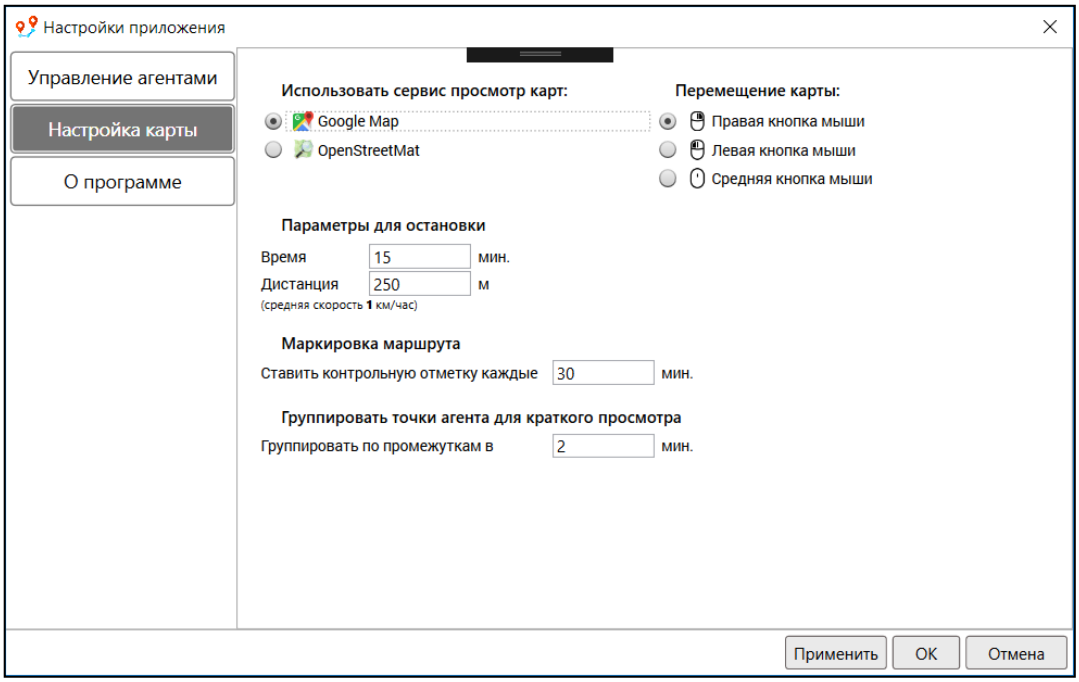


Рисунок 4.4 – Налаштування карти

4.2 Тестування програмної системи аналізу та побудови маршрутів

Існує безліч підходів до вирішення завдання тестування і верифікації ПЗ, але ефективне тестування складних програмних продуктів - це процес надзвичайно творчий, що не зводиться до проходження суворим і чітким процедурам або створення таких.

Тестування ПЗ - спроба визначити, чи виконує програма то, що від неї очікують. Як правило, ніяке тестування не може дати абсолютної гарантії працездатності програми в майбутньому.

Для наочності: майже всі виробники комерційного ПЗ виправляють помилки в своїх продуктах.

Тестування поділяється на наступні рівні:

Модульне тестування (юніт-тестування) - тестується мінімально можливий для тестування компонент, наприклад, окремий клас або функція. Часто модульне тестування здійснюється розробниками ПЗ.

Інтеграційне тестування - тестуються інтерфейси між компонентами, підсистемами. При наявності резерву часу на даній стадії тестування ведеться ітеграційно, з поступовим підключенням наступних підсистем.

Інтеграційне тестування (integration testing) направлено на перевірку взаємодії між декількома частинами додатка (кожна з яких, в свою чергу, перевірена окремо на стадії модульного тестування). Тестування навантаження (load testing, capacity testing) - дослідження здатності додатки зберігати задані показники якості при навантаженні в допустимих межах і деяке перевищення цих меж (визначення «запасу міцності»). Опис розроблених модулів програми, які відповідають вимогам. Функціональне тестування даних модулів показало їх повну працездатність і відсутність помилок, тобто всі функціональні вимоги були успішно реалізовані.

Системне тестування - тестується інтегрована система на її відповідність вимогам.

Функціональне тестування – це сама трудомістка частина випробування ресурсу. На цьому етапі тестуються всі функціональні вимоги програмного продукту, робота посилань, пошук неробочих гіперпосилань. Йде перевірка підгрузки файлів на сервер, роботи лічильників на сторінках порталу, призначених для користувача форм (наприклад, контакти, зворотний зв'язок, підписка, додавання повідомлень і т.д.). Перевіряється, чи відповідає вміст сторінок сайту вихідному коду.

Usability Testing - іншими словами, ступінь зручності роботи користувача з програмним продуктом. Usability тестування ґрунтується на залученні в якості тестувальників користувачів, аналізуються всі результати і думки[24].

Тестування безпеки - на цьому етапі тестувальник займається перевіркою захисту програми.

Для тестування даного програмного забезпечення було використано модульне тестування (юніт-тестування) для перевірки окремих модулів, Usability Testing для тестування зручності користування програмою.

Тестування навантаження проводилося для дослідження швидкодії таких процесів, як, наприклад, запис точки маршруту (оскільки цей процес відбувається велику кількість разів, то передбачається, що його виконання не повинно займати багато часу) і пошук точок маршруту (без індексування таблиці).

Юніт-тестуванні програмного забезпечення це тестування в якому перевіряються окремі одиниці/компоненти програмного забезпечення. Мета полягає в тому, щоб перевірити, що кожна одиниця програмного забезпечення виконується так, як розроблено. Пристрій є найменшою частиною будь-якого програмного забезпечення, що перевіряється. Зазвичай він має один або декілька входів і зазвичай єдиний вихід. У процедурному програмуванні одиниця може бути окремою програмою, функцією, процедурою тощо. У об'єктно-орієнтованому програмуванні найменша одиниця - це метод, який може належати базовому/супер-класу, абстрактному класу або похідному/дочірньому класу. Деякі з них розглядають модуль програми як одиницю. Це не рекомендується, оскільки в цьому модулі, ймовірно, буде багато окремих одиниць. Для полегшення тестування використовуються модульні тестові рамки, драйвери, заглушки та макетні/підроблені об'єкти.

Однією з переваг юніт-тестування є підвищена впевненість у зміні і підтримці коду. Якщо записуються хороші тестові одиниці, і якщо вони виконуються кожного разу, коли змінюється будь-який код, ми зможемо негайно зловити всі дефекти, введені внаслідок зміни. Крім того, якщо коди вже зроблені менш взаємозалежними, щоб зробити тестування блоку можливим, непередбачений вплив змін на будь-який код є меншим[25].

Тестування програмного забезпечення і процес аналізу програмного продукту і супутньої документації з метою виявлення недоліків в роботі і підвищення якості програмного продукту є дуже важливим.

Для того, щоб модульне тестування було можливим, код повинен бути модульними. Це означає, що код легше використовувати.

Вартість встановлення дефекту, виявленого під час модульного тестування, менша в порівнянні з дефектами, виявленими на більш високих рівнях. Порівняйте вартість (час, зусилля, руйнування, приниження) дефекту, виявленого під час приймального тестування або коли програмне забезпечення є живим[26].

При автоматизованому тестуванні для виконання тестів і перевірки результатів використовуються програмні засоби. При використанні методології гнучкого програмування TDD (Test Driven Development, Розробка через тестування) тести пишуться до написання коду програми, і програма вважається закінченою, коли проходить всі тести. У цьому підході автоматизація тестування закладена в самій методології. Автоматизація тестування при правильному застосуванні дозволяє значно знизити час, необхідний на тестування. Воно не може повністю замінити ручне тестування, але зате високо ефективно при модульному, регрессионном, нагрузочном, інсталяційному і деяких інших видах тестування.

Полегшене налагодження. Коли тест не вдався, потрібно лише налагодити останні зміни. При тестуванні на більш високих рівнях необхідно провести сканування змін, що відбуваються протягом декількох днів/тижнів/місяців.

Найбільш поширеною формою автоматизації є тестування додатків через графічний користувацький інтерфейс. Популярність такого виду тестування пояснюється двома факторами: по-перше, додаток тестується тим же способом, яким його буде використовувати людина, по-друге, можна тестувати додаток, не маючи при цьому доступу до вихідного коду.

Для перевірки програми на працездатність було використано декілька типів тестування.

Під час користувацького тестування всі знайдені помилки були виправлені.

Також було здійснено Unit тестування, написані тести для автоматичного тестування програми.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using AgentRoutes;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AgentRoutes.Tests
{
   [TestClass()]
   public class DateTimeExtentionTests
   {
      [TestMethod()]
      public void TimeStampToDateTimeTest()
      {
         int timeStamp = 1560124994;
         DateTime ex = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc).AddSeconds(timeStamp);
         Assert.AreEqual(new DateTime().TimeStampToDateTime(timeStamp), ex);
      }
   }
}
```

Тести до класу RouteInterval:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using AgentRoutes;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using GMap.NET;

namespace AgentRoutes.Tests
{
   [TestClass()]
   public class RouteIntervalTests
   {
      [TestMethod()]
      public void GetMidPointTest()
      {
         List<RouteInterval> intervals = new List<RouteInterval>();
         PointLatLng point;
         if (intervals.Count() == 0)
         {
            point = PointLatLng.Empty;
         }
         if (intervals.Count() % 2 == 1)
         {
            point = intervals.OrderBy(x => x.DateFrom).ElementAt(intervals.Count() / 2).GetMidPoint();
         }
```

```
            else
            {
                point = intervals.OrderBy(x => x.DateFrom).ElementAt(intervals.Count() / 2).PointFrom;
            }

            Assert.IsNotNull(point);
        }

        [TestMethod()]
        public void GetMidPointTest1()
        {
            List<RouteInterval> intervals = new List<RouteInterval>();
            PointLatLng point;
            PointLatLng PointFrom = PointLatLng.Empty;
            PointLatLng PointTo = PointLatLng.Empty;
            point = new PointLatLng((PointFrom.Lat + PointTo.Lat) / 2, (PointFrom.Lng + PointTo.Lng) / 2);

            Assert.IsNotNull(point);
        }
    }
}
```

Після запуску тестів було виправлено всі помилки, які виявили тести.

Висновки до розділу 4

В процесі роботи були проаналізовані всі вимоги та розроблена програма для аналізу та побудови маршрутів.

Програма була протестована для виявлення помилок, які могли б виникнути в майбутньому в користувачів. Після завершення тестування програма відповідала усім поставленим завданням.

Було сформовано інструкцію по використанню та описаний функціонал програмного продукту.

Загальні висновки

В даній дипломній роботі було проаналізовано роботу торговий агентів і представників, було розглянуто роботу супервайзера, який планує маршрут для торгових агентів і аналізує їх попередні маршрути, щоб знайти місця, де їх можна оптимізувати.

За допомогою проведеного аналізу спроектовано та розроблено архітектуру і створено базу даних перевірки і роботи програми. Було розроблено програмне забезпечення, що дозволяє аналізувати маршрути торгових агентів і представників, а також планувати для них маршрути.

Під час роботи над даною роботою було проведені дослідження для визначення різних варіантів роботи торгових агентів і представників, щоб система могла інтегруватися у різні існуючі робочі процеси.

Так, як бізнес постійно потребує розширення і збільшення продажів, то торгові агенти є невід’ємною складовою розвитку компаній, що займаються транспортуванням та доставкою товарів, а також ті, що налагоджують зв'язок з новими покупцями пропонуючи нову продукцію. І як наслідок із оптимізацією процесів роботи торгових агентів збільшується прибуток компанії.

Наступним кроком у розвитку системи має бути створення нових модулів аналізу маршрутів з визначенням нових контрольних точок, а також створення нових алгоритмів побудови планового маршруту.

Перелік посилань

1. Куцик П. О. Діяльність торговельних підприємств у конкурентному середовищі: контрольно-аналітичне забезпечення системи управління. Чернівці. Технодрук –. 2015 – 370 с.

2. Левченко М. Є. (2016). Формування омнікальної збутової політики підприємства. Київ. 42 с. [Електронний ресурс]. – Режим доступу: http://marketing.kpi.ua/files/studentam/AR_mag_2016/AR_Levchenko.pdf

3. Абрамов та ін. Завдання з програмування. М.: Наука, Гл.ред. фіз.-мат. літ.1988-2241.

4. Куцик П. О. Діяльність торговельних підприємств у конкурентному середовищі: контрольно-аналітичне забезпечення системи управління. Чернівці. Технодрук. – 2015 – 370 с.

5. Арсеновски Даниэль. Рефакторинг в C# и .NET для профессионалов. — М.: «Диалектика», 2009. — С. 528.

6. Г. Буч, Дж. Рамбо, А. Джекобсон. Язык UML. Руководство пользователя. М., ДМК, 2000.

7. Семенов Ю. Н. Семенова О. С. Автоматизация построение маршрутов перевозок мелкопартионных грузов / Ю. Н. Семенов О. С Семенова // Вестник Кубансского государственного технического университета. Кемерово – 2016. – С.192-197.

8. Абрамян Михаил Технология LINQ на примерах. Практикум с использованием электронного задачника Programming Taskbook for LINQ; ДМК Пресс - М., 2014. - 161 с.

9. Гарнаев А. Самоучитель Visual Studio .NET 2003; БХВ-Петербург - М., 2003. - 354 с.

10. Гриффитс Иэн Программирование на C# 5.0; Эксмо - М., 2012. - 826 с.

11. Зиборов В. MS Visual C++ 2010 в среде .NET; Питер - М., 2012. - 320 с.

12. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Питер-ДМК, 2001.

13. Э. Дж. Брауде. Технология разработки программного обеспечения. Питер, 2004.

14. И. Соммервилл. Инженерия программного обеспечения. Вильямс, 2002.

15. Камерон Роб, Михалк Дэйл.NET 3.5, компоненты AJAX и серверные элементы управления для профессионалов; Вильямс - М., 2009. - 608 с.

16. Кристофер Б. Джонс 140 технологий раскрутки сайтов; Рид Групп - М., 2011. - 352 с.

17. Лотка Р. C# и CSLA .NET Framework. Разработка бизнес-объектов; Диалектика / Вильямс - М., 2010. - 842 с.

18. Орлов С. – Технологии розроботки програмного обеспечение, Москва, 2011;

19. Кузьменко Н.И. Сферы применения распределительной логистики / Н.И. Кузьменко // Территория науки. 2017. № 2 – 2017 – С.179-183

20. Токарев Андрій - Про C# 5.0 і NET 4.5 Framework, Київ, 2012;

21. М. Фаулер и др. Архитектура корпоративных программных приложений. Вильямс, 2004.

22. М. Фаулер, К. Скотт. UML в кратком изложении. М., Мир, 1999.

23. Торстейнсон П., .Ганеш Г. А Криптография и безопасность в технологии .NET; Бином. Лаборатория знаний - М., 2007. - 480 с.

24. Цвалина Кржиштоф , Абрамс Брэд Инфраструктура программных проектов. Соглашения, идиомы и шаблоны для многократно используемых библиотек .NET Вильямс - М., 2011. - 416 с.

25. Чакраборти Ангшуман , Кранти Юдай , Роопендра Джит Сандху, Кранти Юдай Microsoft .NET Framework. Разработка профессиональных проектов; БХВ-Петербург - М., 2003. - 884 с.

26. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. Pattern-Oriented Software Architecture. Wiley, 2002.

27. Matthew MacDonald, Mario Szpuszta. .Net in C # 2008. Second Edition.

28. Форум програмістів [Електронний ресурс]. – Режим доступу: http://cyberforum.ru/

29. Робота з базами даних на мові C# [Електронний ресурс]. – Режим доступу: http://sernam.ru/book_cbd.php

30. Сайт по C# і платформі .NET Framework [Електронний ресурс]. – Режим доступу: https://professorweb.ru/

31. Сайт по програмуванні [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/general.php

32. Измайлова М.А. Деловое общение. 4-е изд. М.: Дашков и Ко , – 2011. – С. 8-29.

33. Лабунская В.А., Менджерицкая Ю.А., Бреус Е.Д. Психология затруднённого общения. М.: Академия, – 2001 – 288 с.

34. Муздыбаев А. К. Психология ответственности. Л.: Изд-во ЛГУ, – 1983 – 168 с.

35. Наследов А.Д. SPSS: Компьютерный анализ данных в психологии и социальных науках. СПб.: Питер – 2007 – 416с.

36. Поздняков В.П., Филинкова Е.Б. Психология успешного предпринимательства: опыт исследования и практической работы. // Прикладная психология.1998. №5 – 1998 – С.32-43

37. Рач В.А., Вереіна Л.В., Могільний Г.А. Візуалізація інформації: психологічні та організаційні аспекти – Луганськ: Вид-во Східноукр. нац. ун-ту, – 2000. – 160 с.

38. Романчиков В. І. Основи наукових досліджень: Навчальний посібник. – К.: ІЗМН, – 1997. – 244 с.

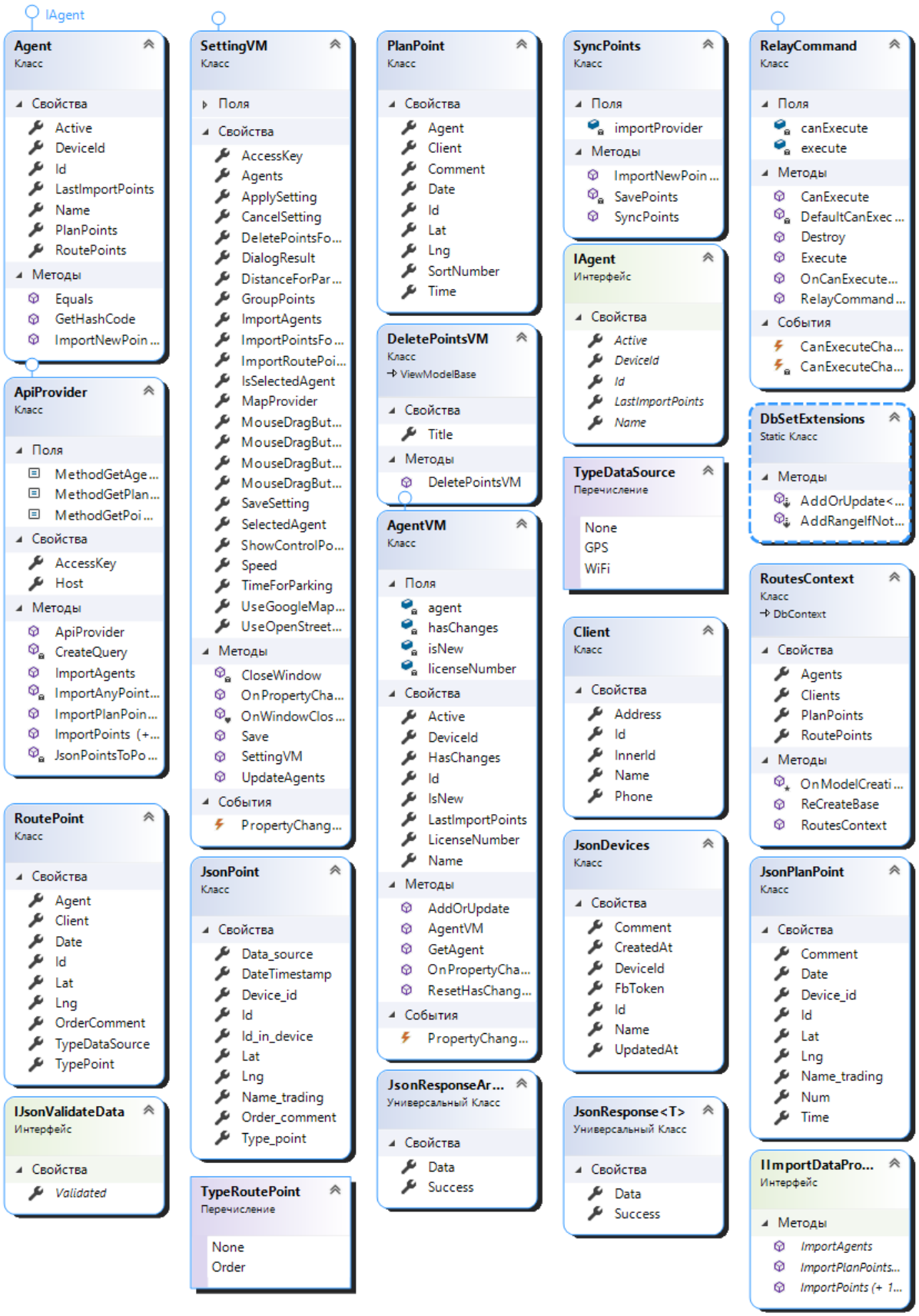
39. Всемирнова Ю.В. Мотивационные особенности менеджеров по продажам с разной профессиональной успешностью // Вестн. ЮУрГУ. Сер. Психология. 2012. № 6. – 2012 – С. 90–95.

40. Диагностика оптимизма как атрибутивного стиля (опросник СТОУН) / Т.О. Гордеева [и др.]. М. – 2008. – С. 84

ДОДАТКИ

Додаток А

Розгорнута структура класів програми для відображення маршрутів



Додаток Б

Програмні коди

Клас MainWindow.cs

```
using AgentRoutes.Controls;
using GMap.NET;
using GMap.NET.MapProviders;
using GMap.NET.WindowsPresentation;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Windows;
using System.Windows.Media;
using System.Windows.Shapes;

namespace AgentRoutes
{
   public partial class MainWindow : Window
   {
      public DateTime Date
      {
         get { return dateTime; }
         set
         {
            dateTime = value;
            tbMounth.Text = dateTime.ToString("MMMM");
            tbDay.Text = dateTime.Day.ToString();
            tbDayOfWeek.Text = dateTime.ToString("ddd").ToLower();
         }
      }
      private DateTime dateTime;
      private DateTime LoverDatetime
      {
         get
         {
            return dateTime.AddHours((int)timeSlider.LowerValue).AddMinutes(timeSlider.LowerValue % 1 * 60);
         }
      }
      private DateTime UpperDateTime
      {
         get
         {
            return dateTime.AddHours((int)timeSlider.UpperValue).AddMinutes(timeSlider.UpperValue % 1 * 60).AddSeconds(59);
         }
      }
      private readonly bool isSyncPoints = false;

      public MainWindow()
      {
         InitializeComponent();


         mainMap.Position = new PointLatLng(49.4281378, 26.9822486);
         mainMap.Zoom = 15;
      }

      public void ImportAgents()
      {
         var agents = new List<Agent>(); //new ApiProvider().GetAgents();
         using (RoutesContext context = new RoutesContext())
         {
            foreach (var agent in agents)
            {
               //if (agent.LastImportPoints == null || agent.LastImportPoints == DateTime.MinValue)
               // agent.LastImportPoints = DateTime.Now.AddMonths(-1);
               var id = agent.Id;

context.Agents.AddOrUpdate(context, agent, x => x.Id == id);
               context.SaveChanges();
            }
         }
      }

      public void RefreshAgents()
      {
```

```
        using (RoutesContext context = new RoutesContext())
        {
            List<Agent> agents = context.Agents.Where(x => x.Active).OrderBy(x => x.Name).ToList();
            Agent currentAgent = null;
            foreach (CustomExpander item in pnAgents.Children)
            {
                if (item.IsExpanded && item.Tag is Agent agent)
                {
                    currentAgent = agent;
                    break;
                }
            }
            pnAgents.Children.Clear();
            foreach (var agent in agents)
            {
                CustomExpander expander = new CustomExpander
                {
                    Text = agent.Name,
                    Tag = agent
                };
                expander.TotalText = context.RoutePoints.Where(x => x.TypeDataSource == TypeDataSource.GPS && x.Agent.Id == agent.Id && x.Date >= LoverDatetime && x.Date <= UpperDateTime).Count().ToString();
                if (currentAgent?.Id == agent.Id)
                {
                    expander.IsExpanded = true;
                }
                expander.ToggleChecked += Expander_ToggleChecked;
                expander.Expanded += Expander_Expanded;
                pnAgents.Children.Add(expander);
            }
        }
    }

    public void ImportPoints(Agent agent)
    {
        using (RoutesContext context = new RoutesContext())
        {
            //agent.ImportNewPoints();
            context.SaveChanges();
        }
    }

    public List<RoutePoint> GetRoutePoints(DateTime dateFrom, DateTime dateTo, Agent agent)
    {
        if (isSyncPoints)
        {
            ImportPoints(agent);
        }
        List<RoutePoint> resultList;
        using (RoutesContext context = new RoutesContext())
        {
            resultList = context.RoutePoints.Where(x => x.TypeDataSource == TypeDataSource.GPS && x.Date >= dateFrom && x.Date <= dateTo && x.Agent.Id == agent.Id).OrderBy(x => x.Date).ToList();
        }
        return resultList;
    }

    public List<PlanPoint> GetPlanPoints(DateTime dateFrom, DateTime dateTo, Agent agent)
    {
        if (isSyncPoints)
        {
            ImportPoints(agent);
        }
        List<PlanPoint> resultList;
        dateTo = dateTo.AddDays(1);
        using (RoutesContext context = new RoutesContext())
        {
            resultList = context.PlanPoints.Where(x => x.Date >= dateFrom.Date && x.Date <= dateTo.Date && x.Agent.Id == agent.Id).OrderBy(x => x.SortNumber).ToList();
        }
        return resultList;
    }

    public void ShowRoute(List<PointLatLng> points, Brush Stroke)
```

```
    {
        if (points.Count == 0)
        {
            return;
        }

        GMapRoute gmRoute = new GMapRoute(points);
        gmRoute.RegenerateShape(mainMap);
        ((Path)gmRoute.Shape).Stroke = Stroke;

        mainMap.Markers.Add(gmRoute);
    }
    public void GenericRoute(List<PointLatLng> points, Brush Stroke)
    {
        if (points.Count == 0)
        {
            return;
        }

        List<PointLatLng> genericPoints = new List<PointLatLng>();

        RoutingProvider routingProvider = mainMap.MapProvider as RoutingProvider ?? GMapProviders.OpenStreetMap;

        for (int i = 0; i < points.Count - 1; i++)
        {
            if (points[i].Equals(points[i + 1]))
            {
                continue;
            }

            MapRoute route = routingProvider.GetRoute(
                points[i], //start
                points[i + 1], //end
                false, //avoid highways
                false, //walking mode
                (int)mainMap.Zoom);


genericPoints.AddRange(route.Points);
        }

        ShowRoute(genericPoints, Stroke);
    }

    public void RefreshRoute()
    {
        foreach (CustomExpander expander in pnAgents.Children)
        {
            if (expander.IsExpanded)
            {
                RefreshRoute(expander.Tag as Agent);
                return;
            }
        }

        RefreshAgents();
    }

    public void ShowListPoints()
    {
        CustomExpander expander = null;
        foreach (CustomExpander OneExpander in pnAgents.Children)
        {
            if (OneExpander.IsExpanded)
            {
                expander = OneExpander;
                break;
            }
        }
        if (expander == null)
        {
            return;
        }

        expander.ClearItems();
        if (expander.CheckedToggleButton == ExpanderToggleButtons.RightButton)
        {
            List<PlanPoint> points = GetPlanPoints(LoverDatetime, UpperDateTime, (Agent)expander.Tag);
            foreach (PlanPoint point in points)
            {
                expander.AddItem(new LinePoint()
                {
                    TextValue = point.Client,
                    TimeValue = point.Time,
                    ViewIcon = Visibility
                });
            }
        }
        else
```

```
        {
            List<RoutePoint> points = GetRoutePoints(LoverDatetime, UpperDateTime, (Agent)expander.Tag);

            PointLatLng lastPoint = PointLatLng.Empty;
            DateTime lastDate = DateTime.MinValue;

            string lastTimeStr = null;
            double allDistance = 0;
            int i = 1;
            foreach (RoutePoint point in points)
            {
                string timeStr = point.Date.ToString("HH:mm");
                if (lastTimeStr == null)
                {
                    lastTimeStr = timeStr;
                }
                //else
                //{
                //    if (lastTimeStr == timeStr)
                //    {
                //        continue;
                //    }
                //}

                double distance = 0;
                double speed = 0;
                if (!lastPoint.IsEmpty)//
                {
                    distance = lastPoint.GetDistanceTo(new PointLatLng(point.Lat, point.Lng));
                    if (point.Date - lastDate != TimeSpan.Zero)
                    {
                        speed = distance / (point.Date - lastDate).TotalHours;
                    }
                }
                allDistance += distance;

                expander.AddItem(new LinePoint()
                {
                    TextValue = $"{i++.ToString()}|t={point.Date.ToString("HH:mm:ss")},d={distance.ToString("0.###")}/{allDistance.ToString("0.###")},s={speed.ToString("0.#")}",
                    //TextValue = (point.Client != null && point.Client != "") ? point.Client : $"X {point.Lat} Y {point.Lng} {additionalText}",
                    TimeValue = timeStr,
                    ViewIcon = (point.TypePoint == TypeRoutePoint.Order) ? Visibility.Visible : Visibility.Hidden
                });
                lastTimeStr = timeStr;


                lastPoint = new PointLatLng(point.Lat, point.Lng);//
                lastDate = point.Date;
            }
        }
    }

    public void RefreshRoute(Agent agent)
    {
        mainMap.Markers.Clear();

        if (cbShowPlanRoute.IsChecked == true)
        {
            List<PlanPoint> routePlanPoints = GetPlanPoints(LoverDatetime, UpperDateTime, agent);
            List<PointLatLng> points = new List<PointLatLng>();
            foreach (var point in routePlanPoints)
            {
                points.Add(new PointLatLng(point.Lat, point.Lng));//+0.00005
            }
            GenericRoute(points, new SolidColorBrush(Colors.Red));
            foreach (var point in routePlanPoints)
            {
                mainMap.AddMarker(new PointLatLng(point.Lat, point.Lng), TypeMarker.PlanPoint, point.SortNumber.ToString(), point.Client, 3);
            }
        }
        if (cbShowFactRoute.IsChecked == true)
        {
```

```
            List<RoutePoint> routePoints =
GetRoutePoints(LoverDatetime,
UpperDateTime, agent).Where(x =>
x.TypeDataSource ==
TypeDataSource.GPS).ToList();
            List<PointLatLng> points = new
List<PointLatLng>();
            foreach (var point in routePoints)
            {
               points.Add(new
PointLatLng(point.Lat, point.Lng));
            }
            ShowRoute(points, new
SolidColorBrush(Colors.Blue));

            if (points.Count > 0)
            {
               var point = routePoints.First();
               mainMap.AddMarker(new
PointLatLng(point.Lat, point.Lng),
TypeMarker.ControlPoint, "1",
point.Date.ToString("HH:mm"), 3);
            }

            TimeSpan periodControlPoints =
Properties.Settings.Default.ShowControlPointsP
erMin;
            int num = 2;
            if (periodControlPoints >
TimeSpan.Zero &&
cbShowAllPoints.IsChecked == true)
            {
               DateTime timeFrom =
DateTime.MinValue;
               foreach (var point in routePoints)
               {
                  if (timeFrom ==
DateTime.MinValue)
                  {
                     timeFrom = point.Date;
                     continue;
                  }
                  if (point.Date - timeFrom >=
periodControlPoints)
                  {
                     mainMap.AddMarker(new
PointLatLng(point.Lat, point.Lng),
TypeMarker.ControlPoint, num.ToString(),
point.Date.ToString("HH:mm"), 2);
                     num++;
                     timeFrom = point.Date;
                  }
               }
            }

            if (points.Count > 0)
            {
               var point = routePoints.Last();
               mainMap.AddMarker(new
PointLatLng(point.Lat, point.Lng),
TypeMarker.ControlPoint, num.ToString(),
point.Date.ToString("HH:mm"), 3);
            }

            if (cbShowParkingPoints.IsChecked
== true && routePoints.Count > 0)
            {
               List<ParkingOnRoute> parkings =
ParkingOnRoute.GetParkings(routePoints);
               foreach (var parkingPoint in
parkings)
               {

mainMap.AddMarker(parkingPoint.Point,
TypeMarker.Stop,
DoubleToStringTime((parkingPoint.DateTo -
parkingPoint.DateFrom).TotalHours),

parkingPoint.DateFrom.ToString("HH:mm") +
"-" + parkingPoint.DateTo.ToString("HH:mm"),
3);
               }
            }
            if (cbShowTimeOrder.IsChecked ==
true)
            {
               List<RoutePoint> orderPoints =
routePoints.Where(x => x.TypePoint ==
TypeRoutePoint.Order).ToList();
               foreach (var point in orderPoints)
               {
                  mainMap.AddMarker(new
PointLatLng(point.Lat, point.Lng),
TypeMarker.Order,
point.Date.ToString("HH:mm"));
               }
            }
         }
         RefreshAgents();
         ShowListPoints();
      }
```

```
    public string DoubleToStringTime(double countHours)
    {
        return ((int)countHours).ToString() + ":" + (countHours % 1 * 60).ToString("00");
    }

    private void timeSlider_LowerValueChanged(double value)
    {
        tbLowerTime.Text = DoubleToStringTime(value);

        Debug.WriteLine("timeSlider_LowerValueChanged");
    }
    private void timeSlider_UpperValueChanged(double value)
    {
        tbUpperTime.Text = DoubleToStringTime(value);

        Debug.WriteLine("timeSlider_UpperValueChanged");
    }
    private void btPriorDay_Click(object sender, RoutedEventArgs e)
    {
        Date = Date.AddDays(-1);
        RefreshRoute();
    }
    private void btNextDay_Click(object sender, RoutedEventArgs e)
    {
        Date = Date.AddDays(1);
        RefreshRoute();
    }
    private void mainMap_OnMapZoomChanged()
    {
        foreach (var item in mainMap.Markers)
        {
            if (item.Shape is CustomMarker image)
            {

                ((CustomMarker)item.Shape).Zoom = mainMap.Zoom;
            }
        }
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        GMapMarker marker = new GMapMarker(new PointLatLng(49.4281378, 26.9822486));
        marker.Shape = new CustomMarker()
        {
            RealHeigth = 3,
            Text = "1",
            Zoom = mainMap.Zoom,
            Fill = new SolidColorBrush(Colors.Orange)
        };

        mainMap.Markers.Add(marker);
    }
    private void Button_Click_2(object sender, RoutedEventArgs e)
    {
        RoutingProvider routingProvider = mainMap.MapProvider as RoutingProvider ?? GMapProviders.OpenStreetMap;

        MapRoute route = routingProvider.GetRoute(
            new PointLatLng(49.4381378, 26.9822486), //start
            new PointLatLng(49.4281378, 26.9622486), //end
            false, //avoid highways
            false, //walking mode
            (int)mainMap.Zoom);

        GMapRoute gmRoute = new GMapRoute(route.Points);
        gmRoute.RegenerateShape(mainMap);
        ((Path)gmRoute.Shape).Stroke = new SolidColorBrush(Colors.Red);
        ((Path)gmRoute.Shape).StrokeThickness = 20;

        mainMap.AddMarker(new PointLatLng(49.4381378, 26.9822486), TypeMarker.PlanPoint, "start");
        mainMap.Markers.Add(gmRoute);
        mainMap.AddMarker(new PointLatLng(49.4281378, 26.9622486), TypeMarker.Stop, "end");
    }
```

```
        private void Button_Click_3(object sender, RoutedEventArgs e)
        {
            if (mainMap.MapProvider == GMapProviders.OpenStreetMap)
            {
                mainMap.MapProvider = GMapProviders.GoogleMap;
            }
            else
            {
                mainMap.MapProvider = GMapProviders.OpenStreetMap;
            }
        }
        private void Button_Click_4(object sender, RoutedEventArgs e)
        {
            CustomExpander expander = new CustomExpander();
            pnAgents.Children.Add(expander);
        }
        private void Button_Click_5(object sender, RoutedEventArgs e)
        {
            ImportAgents();
        }
        private void Expander_Expanded(object sender, RoutedEventArgs e)
        {
            foreach (CustomExpander expander in pnAgents.Children)
            {
                if (!expander.Equals(sender))
                {
                    expander.IsExpanded = false;
                }
                else
                {
                    expander.IsExpanded = true;

                }
            }

            RefreshRoute(((CustomExpander)sender).Tag as Agent);
        }
        private void Expander_ToggleChecked(object sender)
        {
            var a = ((CustomExpander)sender).CheckedToggleButton;

            Debug.WriteLine("Expander_ToggleChecked = " + a);

            ShowListPoints();
        }
        private void BtSettinds_Click(object sender, RoutedEventArgs e)
        {
            SettingForm form = new SettingForm()
            {
                Owner = this
            };
            bool? result = form.ShowDialog();
            Debug.WriteLine("XXXXX "+result);
            RefreshAgents();
            RefreshRoute();
            mainMap.DragButton = Properties.Settings.Default.MouseDragButton;
            mainMap.Manager.UseMemoryCache = Properties.Settings.Default.UseCache;
        }
        private void RefreshRoute_Click(object sender, RoutedEventArgs e)
        {
            RefreshRoute();
        }
        private void TimeSlider_PreviewMouseUp(object sender, System.Windows.Input.MouseButtonEventArgs e)
        {
            RefreshRoute();

            Debug.WriteLine("TimeSlider_PreviewMouseUp");
        }
        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {

            Properties.Settings.Default.LoverTimeValue = timeSlider.LowerValue;

            Properties.Settings.Default.UpperTimeValue = timeSlider.UpperValue;
```

```
Properties.Settings.Default.IsCheckedShowFactRoute = cbShowFactRoute.IsChecked == true;

Properties.Settings.Default.IsCheckedShowPlanRoute = cbShowPlanRoute.IsChecked == true;

Properties.Settings.Default.IsCheckedShowTimeOrder = cbShowTimeOrder.IsChecked == true;

Properties.Settings.Default.IsCheckedShowParkings = cbShowParkingPoints.IsChecked == true;
            Properties.Settings.Default.Save();
            //var b = mainMap.Manager.ExportToGMDB(((GMap.NET.CacheProviders.SQLitePureImageCache)mainMap.Manager.PrimaryCache).CacheLocation+"1.gmdb");
        }
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            timeSlider.LowerValue = Properties.Settings.Default.LoverTimeValue;
            timeSlider.UpperValue = Properties.Settings.Default.UpperTimeValue;
            cbShowFactRoute.IsChecked = Properties.Settings.Default.IsCheckedShowFactRoute;
            cbShowPlanRoute.IsChecked = Properties.Settings.Default.IsCheckedShowPlanRoute;
            cbShowTimeOrder.IsChecked = Properties.Settings.Default.IsCheckedShowTimeOrder;
            cbShowParkingPoints.IsChecked = Properties.Settings.Default.IsCheckedShowParkings;
            mainMap.DragButton = Properties.Settings.Default.MouseDragButton;

            //Date = DateTime.Now.Date;
            Date = DateTime.Parse("10.01.2019");
            RefreshAgents();
            //var b = mainMap.Manager.ImportFromGMDB(((GMap.NET.CacheProviders.SQLitePureImageCache)mainMap.Manager.PrimaryCache).CacheLocation + "1.gmdb");
        }
        private void DtpDate_CalendarClosed(object sender, RoutedEventArgs e)
        {
            Date = dtpDate.SelectedDate ?? Date;
            RefreshRoute();
        }
        private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            dtpDate.IsDropDownOpen = true;
        }

        private void Button_Click_6(object sender, RoutedEventArgs e)
        {
            switch (((System.Windows.Controls.Button)sender).Content)
            {
                case "1":
                    mainMap.MouseWheelZoomType = MouseWheelZoomType.MousePositionAndCenter;
                    break;
                case "2":
                    mainMap.MouseWheelZoomType = MouseWheelZoomType.MousePositionWithoutCenter;
                    break;
                case "3":
                    mainMap.MouseWheelZoomType = MouseWheelZoomType.ViewCenter;
                    break;
                default:
                    break;
            }
        }

        private void Button_Click_7(object sender, RoutedEventArgs e)
        {
            switch (((System.Windows.Controls.Button)sender).Content)
            {
                case "1":
                    mainMap.SelectionUseCircle = !mainMap.SelectionUseCircle;
```

```
            break;
        case "2":
            mainMap.ShowTileGridLines = !mainMap.ShowTileGridLines;
            break;
        case "3":
            mainMap.ShowCenter = !mainMap.ShowCenter;
            break;
        case "4":

Debug.WriteLine(mainMap.Manager.MemoryCache.Size);
            break;
        default:
            break;
      }
    }

    private void Button_Click_8(object sender, RoutedEventArgs e)
    {
      switch (((System.Windows.Controls.Button)sender).Content)
      {
        case "1":
            mainMap.HelperLineOption = HelperLineOptions.DontShow;
            break;
        case "2":
            mainMap.HelperLineOption = HelperLineOptions.ShowAlways;
            break;
        case "3":
            mainMap.HelperLineOption = HelperLineOptions.ShowOnModifierKey;
            break;
        default:
            break;
      }
    }

    private void Button_Click_9(object sender, RoutedEventArgs e)
    {
      RectLatLng area = mainMap.SelectedArea;

      if (!area.IsEmpty)
      {
        for (int i = (int)mainMap.Zoom; i <= mainMap.MaxZoom; i++)
        {
          Stopwatch timer = new Stopwatch();
          timer.Start();
          TilePrefetcher obj = new TilePrefetcher();
          obj.Title = "Prefetching Tiles";
          obj.Icon = Icon;
          obj.Owner = this;
          obj.ShowCompleteMessage = false;
          obj.Start(area, i, mainMap.MapProvider, 100);
          timer.Stop();
          rtbMemo.AppendText(i + " - " + timer.ElapsedMilliseconds / 1000 + Environment.NewLine);
        }
      }
      else
      {
        MessageBox.Show("No Area Chosen", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
      }
    }

    private void Button_Click_10(object sender, RoutedEventArgs e)
    {
      if (mainMap.MapProvider == GMapProviders.GoogleMap)
        mainMap.MapProvider = GMapProviders.OpenStreetMap;
      else
        mainMap.MapProvider = GMapProviders.GoogleMap;
    }
  }
}
```

Клас ParkingOnRoute

```
using GMap.NET;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace AgentRoutes
{
   public class RouteInterval
   {
      public PointLatLng PointFrom { get; }
      public PointLatLng PointTo { get; }
      public DateTime DateFrom { get; }
      public DateTime DateTo { get; }
      public double Distance { get; }
      public TimeSpan TimeSpan { get; }
      public double Speed { get; }

      public RouteInterval(PointLatLng pointFrom, PointLatLng pointTo, DateTime dateFrom, DateTime dateTo)
      {
         PointFrom = pointFrom;
         PointTo = pointTo;
         DateFrom = dateFrom;
         DateTo = dateTo;

         Distance = PointFrom.GetDistanceTo(PointTo);
         TimeSpan = DateTo - DateFrom;
         Speed = Distance / TimeSpan.TotalHours;
      }

      public static PointLatLng GetMidPoint(IEnumerable<RouteInterval> intervals)
      {
         if (intervals.Count() == 0)
         {
            return PointLatLng.Empty;
         }
         if (intervals.Count() % 2 == 1)
         {
            return intervals.OrderBy(x => x.DateFrom).ElementAt(intervals.Count() / 2).GetMidPoint();
         }
         else
         {
            return intervals.OrderBy(x => x.DateFrom).ElementAt(intervals.Count() / 2).PointFrom;
         }
      }
      public PointLatLng GetMidPoint()
      {
         return new PointLatLng((PointFrom.Lat + PointTo.Lat) / 2, (PointFrom.Lng + PointTo.Lng) / 2);
      }
   }
   public class ParkingOnRoute
   {
      public PointLatLng Point { get; set; }
      public DateTime DateFrom { get; set; }
      public DateTime DateTo { get; set; }
      public double Distance { get; set; }

      public static ParkingOnRoute IntervalToParkingPoint(RouteInterval interval)
      {
         return new ParkingOnRoute()
         {
            DateFrom = interval.DateFrom,
            DateTo = interval.DateTo,
            Distance = interval.Distance,
            Point = GetMidPoint(interval.PointFrom, interval.PointTo)
         };
      }

      public static ParkingOnRoute IntervalToParkingPoint(List<RouteInterval> intervals, int fromId, int toId)
      {
         return new ParkingOnRoute()
         {
            DateFrom = intervals[fromId].DateFrom,
            DateTo = intervals[toId].DateTo,
            Distance = intervals.GetRange(fromId, toId - fromId + 1).Sum(x => x.Distance),
            Point = RouteInterval.GetMidPoint(intervals.GetRange(fromId, toId - fromId + 1))
         };
      }
      public static List<ParkingOnRoute> GetParkings(List<RoutePoint> routePoints)
      {
         List<ParkingOnRoute> returnPoints = new List<ParkingOnRoute>();
```

```
        double distanceForParking = Properties.Settings.Default.DistanceForParking / 1000;
        TimeSpan timeForParking = Properties.Settings.Default.TimeForParking;
        double speedForParking;
        if (timeForParking == TimeSpan.Zero || distanceForParking == 0 || routePoints.Count < 2)
        {
            return returnPoints;
        }
        else
        {
            speedForParking = (distanceForParking) / timeForParking.TotalHours;
        }

        List<RouteInterval> intervals = new List<RouteInterval>();

        for (int i = 1; i < routePoints.Count; i++)
        {
            intervals.Add(new RouteInterval(new PointLatLng(routePoints[i - 1].Lat, routePoints[i - 1].Lng),
                new PointLatLng(routePoints[i].Lat, routePoints[i].Lng), routePoints[i - 1].Date, routePoints[i].Date));
        }
        intervals = intervals.OrderBy(x => x.Distance).ToList();
        for (int i = intervals.Count - 1; i >= 0; i--)
        {
            RouteInterval currentInterval = intervals[i];

            if (currentInterval.Distance >= distanceForParking)
            {
                if (currentInterval.Speed <= speedForParking)
                {

returnPoints.Add(IntervalToParkingPoint(currentInterval));
                }
                intervals.RemoveAt(i);
            }
            else
            {
                break;
            }
        }
        intervals = intervals.OrderBy(x => x.DateFrom).ToList();
        for (int i = 0; i < intervals.Count; i++)
        {
            double distance = intervals[i].Distance;
            TimeSpan time = intervals[i].TimeSpan;
            int j;
            for (j = i + 1; j < intervals.Count; j++)
            {
                if (intervals[j - 1].PointTo != intervals[j].PointFrom)
                {
                    j--;
                    break;
                }
                distance += intervals[j].Distance;
                time += intervals[j].TimeSpan;
                if (time>timeForParking && distance<distanceForParking)
                {
                    for (int k = j+1; k < intervals.Count; k++)
                    {
                        if (intervals[k].Distance<0.025)
                        {
                            j++;
                        }
                        else
                        {
                            break;
                        }
                    }
returnPoints.Add(IntervalToParkingPoint(intervals, i, j));
                    i = j + 1;
                    break;
                }
            }
        }
        return returnPoints;
    }
```

```
    public static PointLatLng GetMidPoint(PointLatLng point1, PointLatLng point2)
    {
        return new PointLatLng((point1.Lat + point2.Lat) / 2, (point1.Lng + point2.Lng) / 2);
    }
  }
}
```

Додаток В

Ксерокопії наукових публікацій, виконаних при роботі над дипломною роботою магістра

УДК 004.4

Буров А. Ю.

Хмельницький національний університет

ОРГАНІЗАЦІЯ ЗБУТУ ЗА СИСТЕМОЮ МОБІЛЬНИХ ПРОДАЖІВ

Досліджено значення логістики в організації збуту, а саме особливості функціонування системи мобільних продажів. Досліджено віддалені та альтернативні моделі продажів, такі як вен-селлінг та пре-селлінг та дано оцінку перевагам та недолікам, пов'язаним з ними. Окрему увагу зосереджено на актуальності використання автоматизованих систем управління мобільною торгівлею. Проаналізовано особливості впровадження та поєднання моделей і автоматизованої системи, а також адаптації в умовах епідемії COVID.

The article examines the importance of logistics in the organization of sales, namely, the features of the functioning of the mobile sales system. Remote and alternative sales models, such as Ven-selling and pre-selling, are investigated and the advantages and disadvantages associated with them are evaluated. Special attention is paid to the relevance of using automated mobile trading management systems. The features of implementation and combination of models and an automated system, as well as adaptation in the context of the COVID epidemic, are analyzed.

Постановка проблеми. Розвиток та поглиблення економічних зв'язків спричинив ускладнення торгової системи на різних рівнях. Це призвело до необхідності модифікації логістики та методів ведення торгівлі шляхом зміни, диференціації та доповнення класичних методів. Мобільна торгівля як явище з'явилась порівняно недавно і постійно видозмінювалась відповідно до потреб та умов галузі. Необхідність дослідження цієї моделі торгівлі і зумовила актуальність написання цієї статті.

Мета роботи полягає у дослідженні організації збуту за системою мобільних продажів та автоматизованої системи управління мобільною торгівлею, особливостей їх функціонування та впровадження.

Виклад основного матеріалу. Сьогодні в умовах запеклої конкуренції велике значення в успішній діяльності підприємства набуває управління просуванням виробів на ринок, що є однією з функцій маркетингу. Це потрібно для того, щоб створити систему управління просуванням виробів підприємства на ринок.

Ця проблема ставить завдання з просування виробів фірми на ринок, тобто завдання логістичні. Для їх успішного вирішення підприємству необхідно досягти деякого мінімального обсягу продажів. У іншому випадку всі зусилля з організації служби доставки товару до споживача виявляються малорентабельними.

До особливостей масової торгівлі слід в першу чергу віднести широкий асортимент продукції, що випускається, велику кількість покупців і мінливість у потребах. Успіх підприємства на ринку великою мірою забезпечує доступність товару для покупця, яка досягається різними формами торгівлі. Останнім часом в промисловості широко використовуються методи дрібнооптової торгівлі із залученням торгових агентів (ТА), кожен з яких пов'язаний з групою клієнтів, що постійно оновлюється. В організації роботи з торговими агентами є особливості, які можна назвати як перевагами, так недоліками: значні обсяги клієнтської бази; велика швидкість ротації клієнтів; висока нестабільність характеристик бізнесу клієнтів - починаючи з обсягів і структури продажів і закінчуючи присутністю на ринку; невеликі масштаби ринку більшості торгових посередників; територіальна віддаленість їх один від одного і від центрального складу виробника або оптовика; обмеженість агентів в асортименті (що, з іншого боку, дає можливість економити на складських приміщеннях); відсутність гарантійних запасів, тощо [1; 4; 7].

Робота з товарними агентами передбачає формування невеликих за обсягом цільових замовлень. Це передбачає створення на складах фірм-оптовиків великих запасів товарів, необхідних для безперебійного постачання ТА нових партій товару. Вони формуються залежно від характеру замовлень покупців, а тому часто дрібні партії товарів можуть об'єднуватись у великі, або навпаки – великі обсяги можуть ділитись на декілька замовлень. Виконання замовлення здійснюється на основі товарного чеку, який друкується у кількох примірниках. Як правило, один з них призначається для формування замовлення на складі, другий для покупця і третій має бути підписаний покупцем в якості підтвердження отримання товару та повернений на підприємство. Така система формування замовлення полегшує контроль видачі, перевезення та отримання товару сторонами. Часто підприємства застосовують контрольно-ревізійні перевірки для звірення даних в товарних чеках.

Наявність сильної конкуренції вимагає використання ефективних засобів, наприклад, мобільних продажів. Мобільні продажі - термін, що позначає доставку товару виробником в торгівлю або покупцеві за рахунок широких можливостей вибудовування логістики руху товару і грошей, документообігу будь-якого рівня складності. Це дозволяє реалізовувати ефективні бізнес-процеси і організувати торгівлю з урахуванням всіх нюансів логістики і здійснити якісний стрибок в організації бізнесу [3; 6].

Одна з найбільш популярних форм подібної організації роботи - van selling (вен-селлінг), дрібнооптова торгівля «з коліс», за якої вибір товару і оформлення угоди здійснюється прямо в офісі клієнта. Вен-селлінг переслідує чотири такі цілі, як: реклама, доставка, розширення асортименту, маркетинг.

Вен-селлінг створює певні переваги для постачальників і продавців, перш за все вони пов'язані з оптимізацією процесу продажу та використання складського приміщення через закупівлю товару без залишків, спрощенням процесу доставки та продажу для покупця та полегшення контролю наявності товару, оперативністю доставки, тощо.

Особливо відчутний виграш від застосування вен-селлінга для компаній, що виробляють або продають дрібні споживчі товари повсякденного попиту. Найчастіше це продукти харчування, миючі засоби, недорога парфумерія і т. п. Такий товар швидко розходиться, і поповнювати його запаси в роздрібній торговельній мережі треба дуже оперативно. При цьому асортимент продукції може бути будь-яким - і дуже широким, і порівняно невеликим [1].

Практична реалізація вен-селлінга виглядає наступним чином. З ранку «вен» завантажується товаром і потім разом з торговим представником відправляється за маршрутом точками роздрібної торгівлі. У кожній точці проводяться переговори, відразу ж оформляються всі документи і відбувається відвантаження товару.

Ефективним також є впровадження системи pre-selling (пре-селлінгу). Це організація попереднього збору замовлень для реалізації товару підприємствами. Мобільна торгівля за цим принципом передбачає використання електронного пристрою (комунікатора), який має повну інформацію про наявність товару на складах [3; 8; 9].

Таким чином торгові агенти можуть виконувати завдання як моделі пре-селлінгу, так і вен-селлінгу. Перший – пре-селлінг – зводиться до формування портфеля замовлень в процесі спілкування з актуальними та потенційними клієнтами. Цей режим передбачає максимальну свободу в ухваленні рішення. В першу чергу мова йде про цінову політику. Мобільний модуль може раціонально зв'язати три найважливіші компоненти процесу формування замовлень − прайс-листи, торгові операції та клієнтів. Таким чином, ціна може бути обрана виходячи з умов угоди, на які погодиться той чи інший конкретний клієнт. Інформація про клієнтів та історія відносин з ними зберігається в стаціонарній частині комплексу. Агенту немає необхідності самостійно аналізувати їх, варіанти рішень можуть бути завантажені в його кишеньковий комп'ютер до початку роботи. Крім того, завчасно може бути сформована політика компанії щодо нових, незнайомих клієнтів, для них можуть бути визначені дозволені операції і допустимі варіанти цін. Інформація про актуальних клієнтів, після завершення певного періоду «скидається» в стаціонарну частину комплексу.

Вен-селлінг зводиться до дрібнооптового продажу в різних точках роздрібної торгівлі. У більшості випадків це простіші завдання, які можуть бути зведені до функції доставки зроблених замовлень. Тут, як правило, велике значення має якість формування маршрутів і висока надійність блоку друку. Природньо, обидва види робіт можуть бути суміщені в різних варіантах.

Ще однією цікавою особливістю комплексу є можливість роботи в режимі on-line із залученням засобів мобільного зв'язку. Це особливо актуально в двох випадках. По-перше, при організації запитів на товар може виникнути необхідність негайно отримати інформацію про наявність товару на складі та можливість зарезервувати деяку його частину. По-друге, іноді виникає необхідність фіксування

переміщення агента протягом дня для того, щоб відкоригувати його маршрут і направити на новий найближчий до нього об'єкт.

Широкі можливості вибудовування логістики руху товару і документів, документообігу будь-якого рівня складності, дозволяє реалізувати бізнес-процеси, які неможливо впровадити без автоматизованої системи, або застосовувати методи податкової оптимізації, що допомагають економити до 80% податків, при цьому не порушуючи меж чинного законодавства. Реалізувати такий підхід вручну, без автоматизованої до рівня кожного робочого місця системи, практично нереально. Фактично відбувається якісний стрибок в організації бізнесу, який в принципі не можна зробити без автоматизації [3].

Автоматизована система управління мобільною торгівлею - програмний комплекс, що дозволяє ефективно вирішувати завдання автоматизації торгових представників з використанням компактних комп'ютерів. Система дозволяє істотно підвищити ефективність продажів, помітно знизити витрати на підтримку структури торгових агентів, принципово посилити контроль над їх діяльністю, підвищити точність і швидкість виконання замовлень, значно прискорити виведення на ринок нової продукції і кардинально зменшити терміни навчання нових торгових представників. При цьому система забезпечує оперативне отримання інформації для прийняття необхідних управлінських рішень, доступ керівництва до будь-якого рівня інформації про продажі, дозволяє досить просто реалізовувати побудову складних, територіально розподілених систем для підтримки мобільних продажів. Система автоматизації мобільної торгівлі дозволяє підняти на принципово новий рівень інформаційну підтримку системи дистрибуції товарів і прийняття управлінських рішень, поліпшити імідж компанії і підвищити лояльність клієнтів.

Цього досягають шляхом: зменшення часу на прийняття рішень торговим агентом в точці продажів, яке забезпечується оперативним доступом до інформації на маршруті; автоматичного виконання основних розрахункових операцій; зменшення помилок при створенні документів за рахунок автоматизації контролю обмежень для конкретної торгової точки; оптимізації маршрутів ТА і посилення контролю над діяльністю торгових агентів; прискорення введення інформації в облікову систему підприємства про створені торговим агентом документи та його роботу; можливості не відвідувати офіс, а використовувати віддалену синхронізацію; підвищення точності і швидкості виконання; скорочення операторів з введення інформації від торгових агентів; оптимізації залишків продукції на складі за рахунок своєчасного надходження замовлень і оперативного відвантаження товару; підвищення якості обслуговування клієнтів.

У базі даних компанії зберігається інформація, необхідна для роботи ТА і для обміну з корпоративною обліковою системою (товари ,склади, ціни, знижки, замовники, торгові агенти), дані по маршрутах і управління правами торгових агентів, плани та цілі, сценарії роботи мобільних працівників, об'єкти обліку та багато іншого.

Система управління командою торгових агентів, яка дозволяє керівництву компанії та менеджерам підрозділу ТА оперативно отримувати актуальну інформацію про роботу працівників, формувати маршрути, дистанційно керувати роботою мобільних працівників і ефективно контролювати їх діяльність [2; 3; 5].

Для впровадження моделі мобільної торгівлі підприємство слід вирішити п'ять наступних завдань:

1. Розвиток регіонального продажу. Йдеться про розширення географії продажів. Підприємство-розповсюджувач розбиває свою територію на ділянки. За кожною з таких ділянок закріплюється «вен», на якому працюють водій і торговий агент. Спочатку команда «вена» діє наосліп, але з часом з'являються постійні партнери, напрацьовуються оптимальні маршрути.
2. Розробка маршруту та періодичність візитів. Маршрут не повинен бути ні занадто коротким, ні занадто довгим. У першому випадку «вен» простоює, у другому - багато часу витрачається на дорогу. Ключовим моментом є дотримання періодичності візитів до постійних партнерів: до наступного приїзду «Вена» у реалізатора не повинно бути як надлишків, так і дефіциту тих чи інших видів продукції. Крім того, при низькій циклічності торгові агенти компаній, що конкурують, можуть перехопити клієнта.
3. Встановлення розміру партії. Важливо також точно визначити співвідношення місткості автомобіля і розмірів партії товару з урахуванням тривалості маршруту, числа торгових партнерів і звичайних обсягів їх замовлень.
4. Створення менеджменту. При впровадженні вен-селлінгу необхідна дуже чітка організація роботи команд ТА: поділ території на ділянки, організація пошуку та відсіву клієнтів, відпрацювання маршрутів руху машин, організація обліку та контролю товарів і документації та багато іншого, що вимагає управлінської кваліфікації.
5. Створення комунікатора. Клієнтська частина системи мобільної торгівлі базується на сучасних смартфонах і комунікаторах, в яких функціонал персонального комп'ютера і стільникового телефону об'єднаний в одному корпусі [3].

Особливо актуальним є використання мобільних продажів та автоматизації системи управління мобільною торгівлею в період пандемії COVID через можливість зменшити кількість контактів між людьми, перевести працівників на віддалену роботу чи роботу «в полі» (торговим агентом або водієм), скоротити кількість працівників, тощо. Крім того особливо зростає цінність цього методу через можливість скоротити витрати на податки, що компенсує негативний вплив епідемії на рівень продажів.

Підбиваючи підсумки варто зазначити, що організація збуту за системою мобільних продажів дозволяє спростити значну частину процесів, що стосуються оптової чи роздрібної торгівлі. Ряд переваг для покупців та продавців, які надає система дозволяє оптимізувати роботу підприємствах на різних рівнях. Вен-селлінг, окрім того, дає можливість підприємствам економити місце на складських

приміщеннях або ефективніше використовувати наявне, а покупцям нівелювати дефіцит певних товарів одразу на місці. Пре-селлінг дозволяє здійснювати точкові продажі задовольняючи короткострокові потреби клієнтів в перспективі, уникаючи або доповнюючи довгострокові угоди.

Автоматизована система управління мобільною торгівлею надає можливість спростити та автоматизувати значну частину процесів та уникнути ряду помилок, які спричиняє людський фактор. Особливо ефективно ця система працює в мобільній торгівлі створюючи додатковий зв'язок між різними рівнями працівників підприємства, полегшуючи доступ до інформації та її використання. Додаткової актуальності модель набула після початку епідемії COVID та загострення необхідності оптимізації роботи торгових підприємств, створення можливості здійснення безконтактного адміністрування та економії коштів.

Перелік посилань

1. Володько В. Ф. (2020). Инновационные модели маркетинговой деятельности предприятия. Наука и техника, (2), 130-138. https://cyberleninka.ru/article/n/innovatsionnye-modeli-marketingovoy-deyatelnosti-predpriyatiya.
2. Куцик П. О. (2015). Діяльність торговельних підприємств у конкурентному середовищі: контрольно-аналітичне забезпечення системи управління. Чернівці. Технодрук. 370 с.
3. Лавриненко Д.В.(2008). Методы увеличеня продаж товаров массового спроса за счет использования современных мобильных систем. Научные труды Вольного экономического общества России, 98 , 131-137. https://cyberleninka.ru/article/n/metody-uvelicheniya-prodazh-tovarov-massovogo-sprosa-za-schet-ispolzovaniya-sovremennyh-mobilnyh-sistem.
4. Мазаракі А. А. (2016). Внутрішня торгівля України. Київ : Київ. нац. торг.-екон. ун-т, 2016. – 864 с. https://knute.edu.ua/file/MjExMzA=/508cf1120fbd640bbc5337b81b163898.pdf.
5. Цвєтов Ю. М. (2013). Особливості автоматизованого обліку товарообігу в сучасних умовах. Інші сфери економіки. С.307-311.
6. Шереметинська О. В. (2016). Стимулювання збуту: заходи та засоби, які допомагають при формуванні маркетингової діяльності підприємства при здійсненні ЗЕД. Нац. унів. харч. тех.
7. Левченко М. Є. (2016). Формування омнікальної збутової політики підприємства. Київ. 42 с. http://marketing.kpi.ua/files/studentam/AR_mag_2016/AR_Levchenko.pdf
8. Muthiani J. (2009). The impact of van selling on product distribution. School of business. 57 p. http://erepository.uonbi.ac.ke/bitstream/handle/11295/13263/Muthiani_The%20impact%20of%20van%20selling%20on%20product%20distribution.pdf?sequence=3.
9. Oler D. (2008). Rumors and Pre-Announcement Trading: Why Sell Target Stocks Before Acquisition Announcements? 39 p. https://www.researchgate.net/publication/228260198_Rumors_and_Pre-Announcement_Trading_Why_Sell_Target_Stocks_Before_Acquisition_Announcements.

Перелік наукових публікацій:

1. Буров А. Ю. Організація збуту за системою мобільних продажів / Буров А. Ю. // Збірник наукових праць «Актуальні проблеми комп'ютерних наук АПКН-2020» Хмельницький, 2020. – С.47-52.

Додаток Г

Презентація

Презентація до дипломного проекту на тему

Інформаційна система оптимізації маршрутів торгових агентів

Розробив Буров А. Ю.

Керівник Петровський С. С.

Торгові представники та агенти

Одним із інструментів впливу на покупця є персональний продаж товарів, який передбачає особисте пред'явлення товарів покупцю або групі покупців під час бесіди з метою продажу цих товарів.

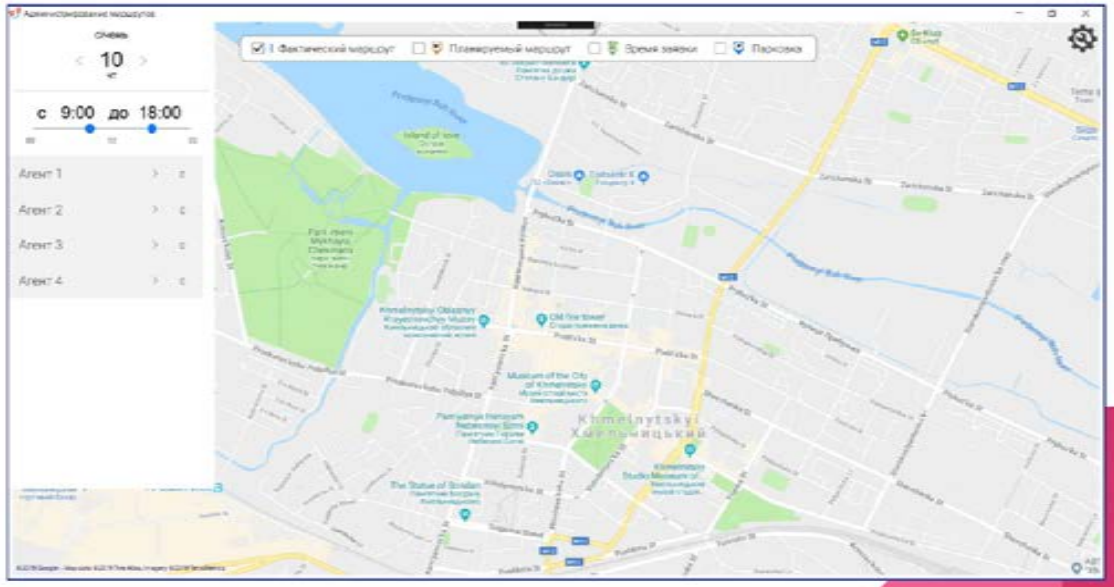
До категорії персонального продажу товарів і послуг належать усі форми продажу за участю представників виробника товару або дистриб'ютора

Контроль торгових агентів

Використання програми для контролю діяльності торгових агентів та інших мобільних співробітників досить швидко призводить до підвищення якості обслуговування клієнтів, кількість торгових точок, точність і обсяг заявок, зібраних даних і відмінно дисциплінує.

Також це дозволяє візуально відслідковувати переміщення торгових представників і коректність скоєних ними візитів до торгових точок.

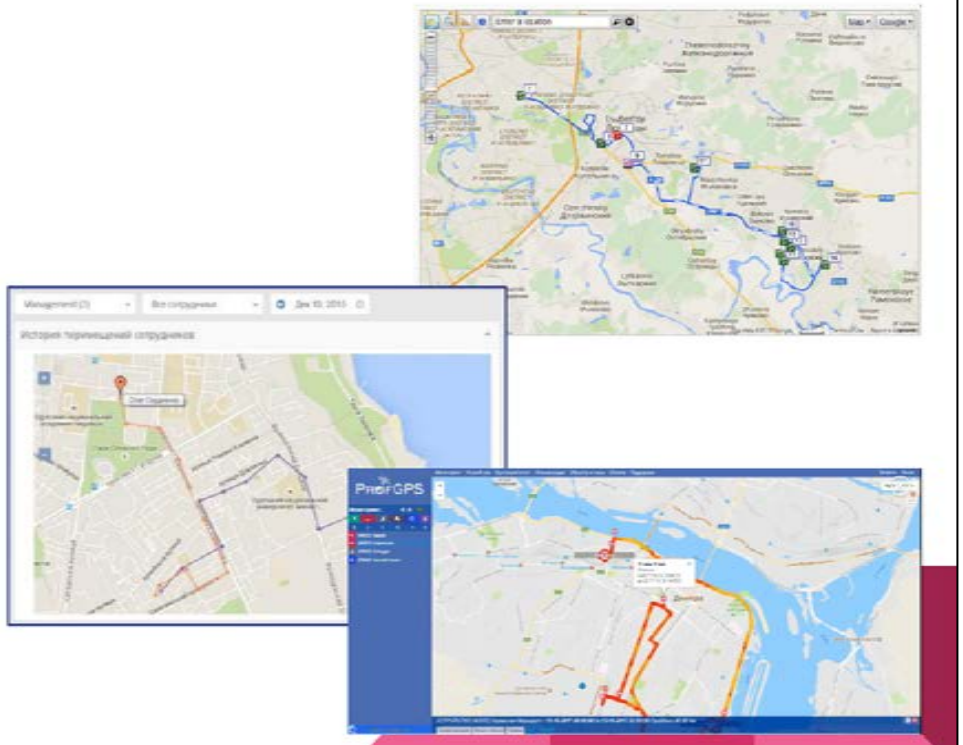
При виконанні дипломного проекту було розроблено програму для контролю торгових агентів



Було розглянуто існуючі рішення

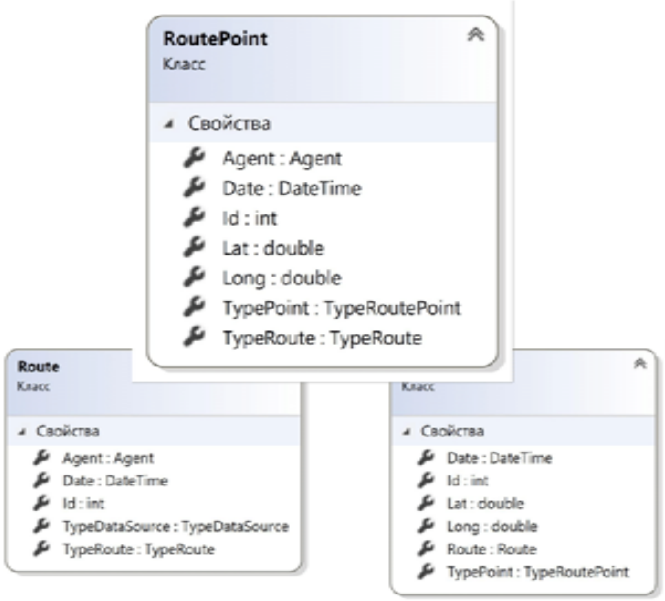
Було проаналізовано ряд існуючих програм, щоб визначити позитивні і негативні сторони кожного програмного засобу.

Це допомогло не допустити помилок або незручностей у створеному програмному засобі.



Була спроектована база даних для зберігання маршрутів агентів

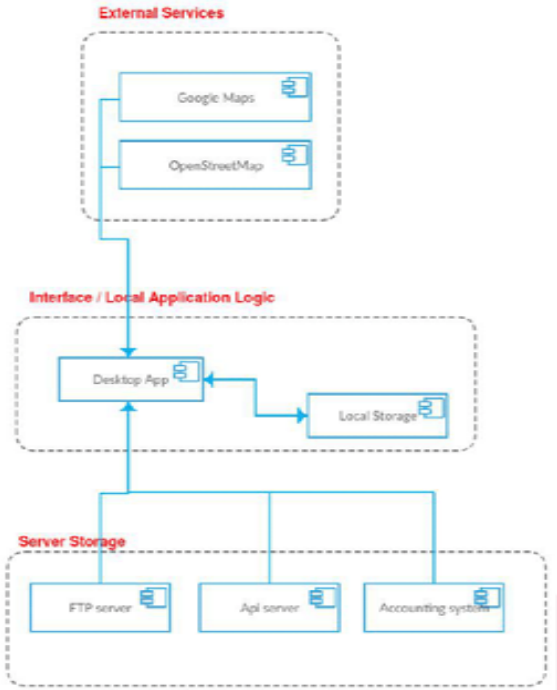
Під час проектування програми була спроектована структура бази даних, щоб зберегти список торгових агентів, їх маршрутів, а також допоміжної інформації під час роботи програми.



Допоміжні матеріали при розробці

Під час розробки програмного засобу було створено декілька UML даграм, наприклад діаграму компонентів.

Це дозволило спроектувати архітектуру майбутньої програми і краще розуміти принцип її роботи

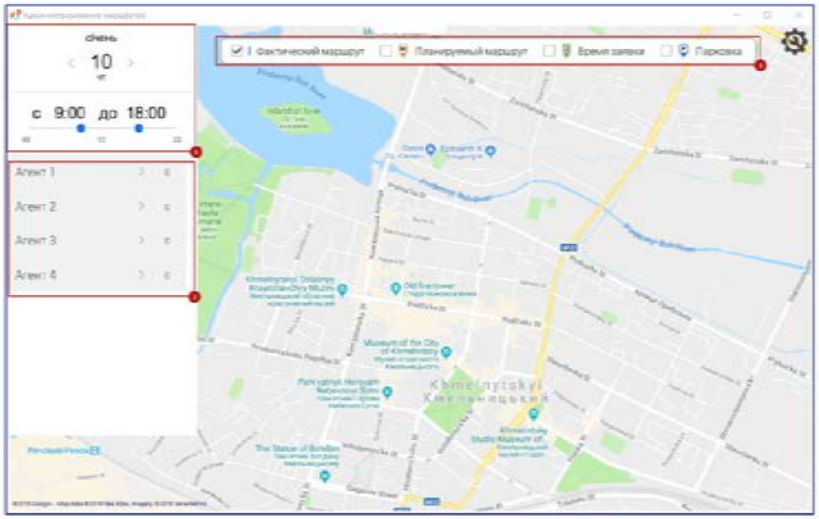


Головне вікно програми

Програма складається з головного вікна, з вікна налаштувань та допоміжних діалогових вікон.

Зліва знаходяться фільтри по даті і часу, які допомогають вибрати точки агента за певний період.

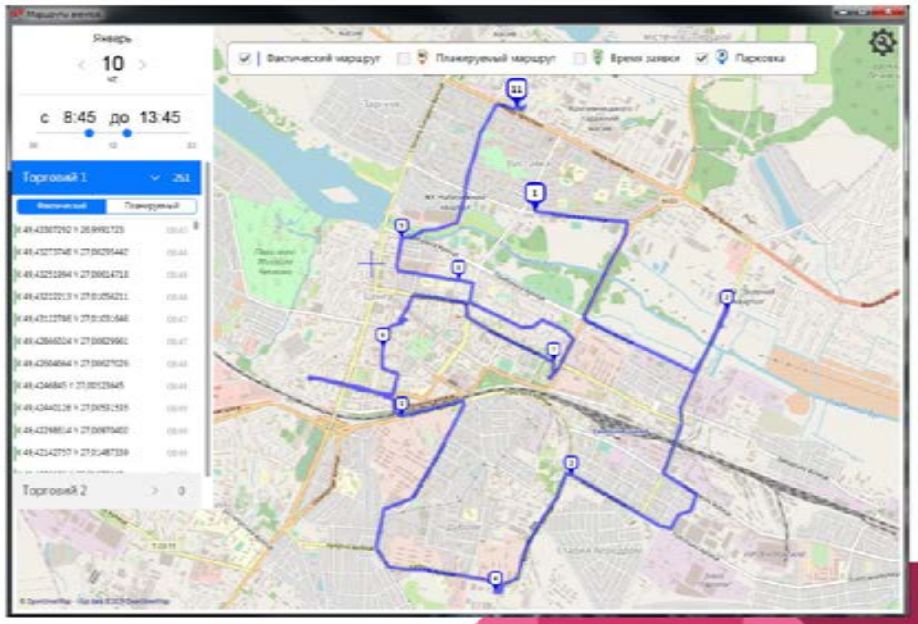
Під фільтром по даті знаходиться список торгових агентів при виборі яких відображаються їх точки.



Візуалізація маршрута

При виборі торгового агента відображається маршрут поточного агента.

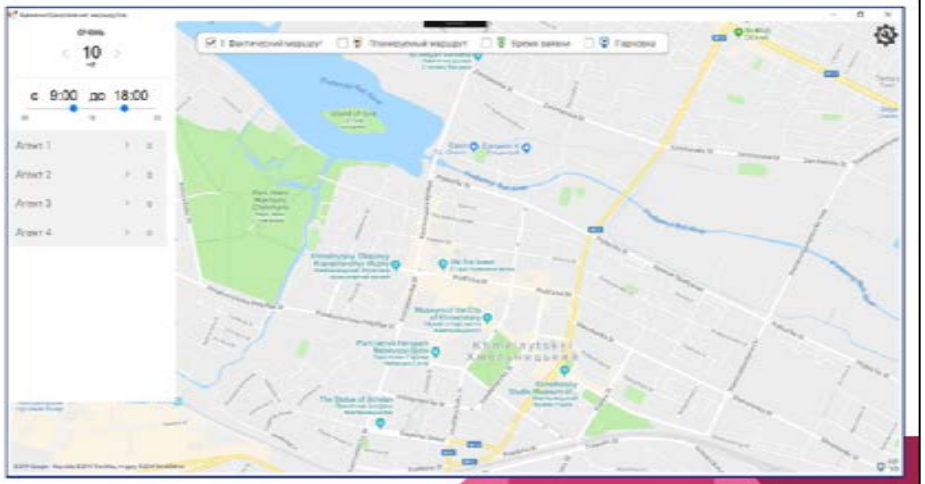
Також на маршруті можна відобразити точки парковок і проставити точки по маршруті кожні декілька хвилин.



Відображення додаткової інформації

Додатково можна відобразити на карті плановий маршрут торгового агента, з інформаціє по торгові точки, які мав відвідати агент.

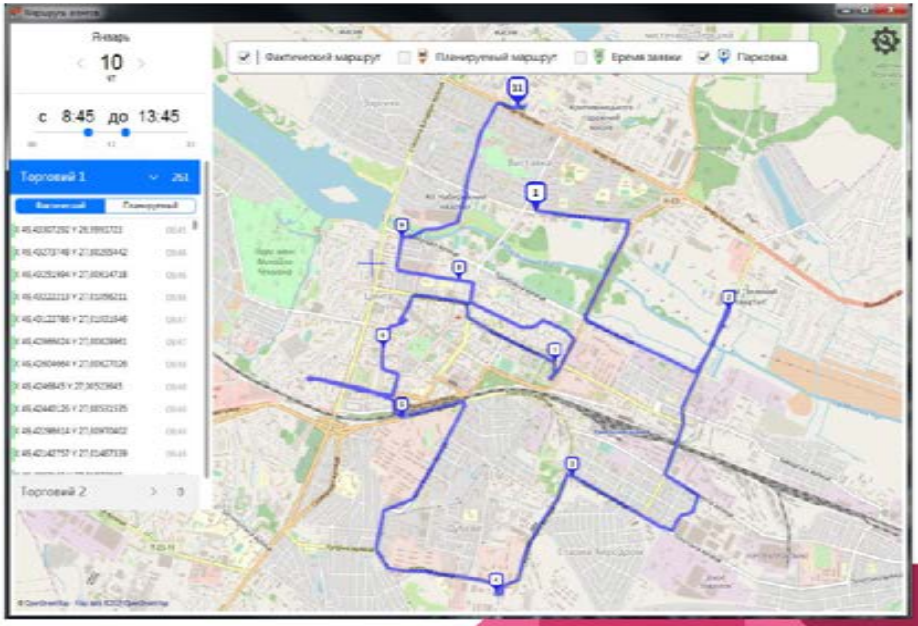
Також є можливість вивести паралельно місцезнаходження, де торговий агент брав замовлення



Зупинки торгового агента

Програма дозволяє відобразити для супервайзера місця, де зупинявся торговий агент, а також час і тривалість зупинки.

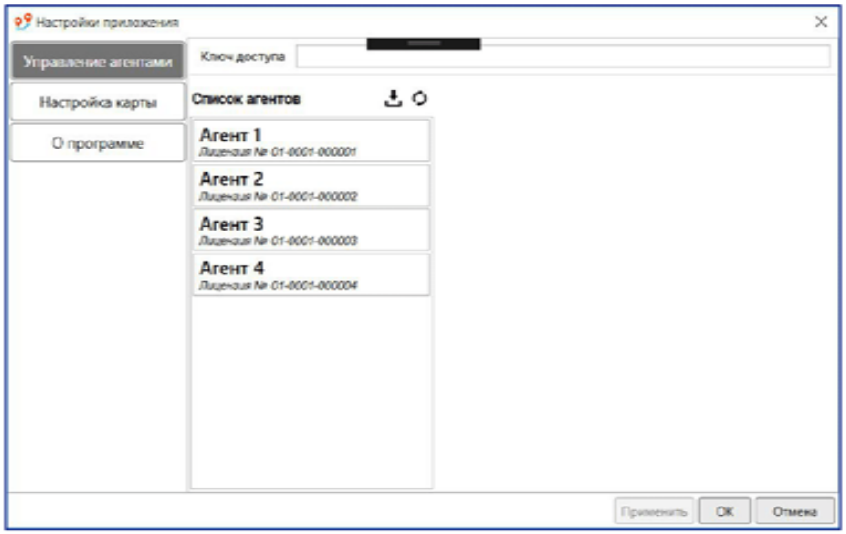
За допомогою цього можна дізнатися, чи агент не проїхав торгову точку.



Налаштування списку агентів

В додатку можна налаштувати імпорт торгових агентів і їхніх планових і фактичних маршрутів, для цього потрібно вказати «Ключ доступа» у формі.

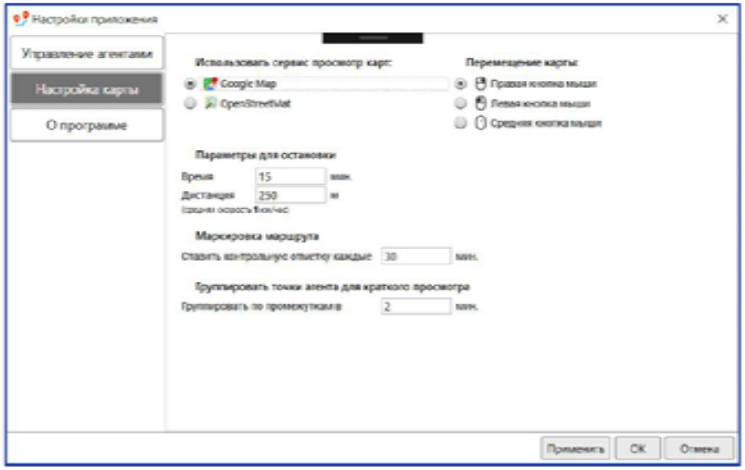
Також можна перейменувати торгових агентів вибравши необхідного агента, і можна приховувати неактивних агентів.



Налаштування карти

Для налаштування параметрів карти потрібно перейти на вкладку «Налаштування карти» там можна змінити наступні параметри для зручнішої роботи з картою:

- тип карти(GoogleMap або OpenStreetMap);
- клавіша миші для переміщення карти;
- параметри для зупинки (при яких налаштуваннях буде ставитися маркер із зупинкою);
- маркування маршрута (встановлюється періодичність маркерів на маршруті);
- групування точок, для зручнішого перегляду.



Дякую за увагу!

Thu Dec 03 13:10:30 EET 2020, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 47.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 5%**

	Документ		Сумарний збіг по Базі Даних	
ID: 82305 Назва: Інформаційна система оптимізації маршрутів торгових агентів Додано в БД: 2020-12-03 Автора: Буров А.Ю. Керівники: Петровський С.С. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	95292	707	51119 (54%)	374 (53%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
81769	Назва: Інформаційна система оптимізації маршрутів торгових агентів Додано в БД: 2020-12-01 Автора: Буров Андрій Юрійович Керівники: Петровський С.С. Консультанти: Опоненти:	45239 (47.0%)	322 (46.0%)

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 оптимізації маршрутів торгових агентів**
Автор: **Буров А.Ю.**
Спеціальність: **122 Комп'ютерні науки**
Науковий керівник: **к.п.н., доцент Петровський С.С.**

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 (є власні терміни, визначення тощо), складають 18,4% та мають посилання на приведений список літературних джерел.

04.12.2020
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 Бурова Андрія Юрійовича*

На тему: *Інформаційна система оптимізації маршрутів торгових агентів.*

1. Актуальність і значення теми

В умовах запеклої конкуренції велике значення в успішній діяльності підприємства набуває управління просуванням виробів на ринок, що є однією з функцій маркетингу. Це потрібно для того, щоб створити систему управління просуванням виробів підприємства на ринок. Ця проблема ставить завдання з просування виробів фірми на ринок, тобто завдання логістичні. І тому з використанням торгових агентів і ефективним їх управлінням можна значно збільшити товарообіг і тим самим збільшити прибутки компанії.

2.Оцінка якості та достовірності проведених досліджень.

Отримані результати добре співвідносяться з результатами, наведеними в наукових роботах і довідниках.

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

Проведені дослідження представляють науково-технічну цінність, є ефективним дослідженням у сфері управління торговими агентами.

4. Загальний висновок та оцінка

Робота виконана в повному обсязі. Досліджені та проаналізовані дані за допомогою комплексу входять в рамки допустимих відхилень. 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 Буров А. Ю. заслуговує присвоєння кваліфікації магістра з комп'ютерних наук та інформаційних технологій.

Робота заслуговує на оцінку «Задовільно».

Опонент Дихо О.В., д.т.н., проф., зав. каф. ТАНХіТУ