МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **«Розробка інформаційної системи планування заготівлі молока із вибором ефективного алгоритму визначення маршрутів»**

Виконав: студент групи Іт-61

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)
Кисіль Сергій Романович
(Прізвище та ініціали)

Керівник: д.т.н., професор Тригуба А.М.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Кригуль Р.Є.
(Прізвище та ініціали)

ДУБЛЯНИ-2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А.М. Тригуба
«____» _____2022 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Кисілю Сергію Романовичу

1. Тема роботи: «Розробка інформаційної системи планування заготівлі молока із вибором ефективного алгоритму визначення маршрутів»

Керівник роботи Тригуба Анатолій Миколайович, професор
затверджені наказом по університету від 12.05.2022 року № 62/к-с.

2. Строк подання студентом роботи 15.12.2022 р.
3. Вихідні дані до роботи: *база даних щодо процесів планування заготівлі молока; алгоритми визначення маршрутів; методика проектування інформаційних систем.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз стану заготівлі молока та завдання кваліфікаційної роботи.
2. Обґрунтування та вибір інструментарію для інформаційної системи планування заготівлі молока.
3. Результати розробки інформаційної системи планування заготівлі молока.
4. Охорона праці та безпека у надзвичайних ситуаціях.
5. Економічна ефективність від використання інформаційної системи планування заготівлі молока.
Висновки та пропозиції.
Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових слайдів): *аналіз стану заготівлі молока та інформаційних систем для планування заготівлі молока; вибір ефективного алгоритму визначення маршрутів; архітектура інформаційної системи планування заготівлі молока; діаграми варіантів використання, діяльності та активності інформаційної системи; архітектура back-end інформаційної системи; програмна реалізація front-end; результати розробки інтерфейсу користувачів та основних функціональних блоків; економічна ефективність.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 5	*Тригуба А.М., завідувач кафедри інформаційних технологій*		
4	*Городецький І.М., доцент кафедри управління проектами та безпеки виробництва*		

7. Дата видачі завдання 12 травня 2022 р.

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*12.05-20.06.22*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*21.06-14.08.22*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*15.08-31.10.22*	
4.	*Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»*	*01.11-10.11.22*	
5.	*Оцінення ефективності запропонованої системи*	*20.11-30.11.22*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*01-04.12.22*	
7.	*Завершення роботи вцілому*	*05-15.12.22*	

Студент _____ Кисіль С.Р.
(підпис)

Керівник роботи _____ Тригуба А.М.
(підпис)

УДК 004.4 : 075.8

Розробка інформаційної системи планування заготівлі молока із вибором ефективного алгоритму визначення маршрутів.

Кисіль С.Р. Кафедра інформаційних технологій – Дубляни, ЛНУП, 2022.

Кваліфікаційна робота: 83 с. текст. част., 18 рис., 2 табл., 15 арк. ілюстраційного матеріалу, 48 джерел.

Виконано аналіз стану заготівлі молока. Проведено аналіз інформаційних систем планування заготівлі молока. Подано особливості заготівлі молока та проектування інформаційних систем її планування. Сформульовано завдання кваліфікаційної роботи.

Наведена модель функціональних можливостей інформаційної системи планування заготівлі молока. Здійснено вибір ефективного алгоритму визначення маршрутів. Виконано порівняння алгоритмів визначення маршрутів. Обґрунтовано доцільність використання Google Maps Platform у інформаційній системі планування заготівлі молока.

Запропонована архітектура інформаційної системи планування заготівлі молока. Розроблено діаграми варіантів використання інформаційної системи, діяльності та активності інформаційної системи, послідовності інформаційної системи. Запропоновано архітектуру back-end інформаційної системи. Виконана програмна реалізація front-end. Подано результати розробки інтерфейсу користувачів та основних функціональних блоків.

Розроблено заходи із охорони праці, а також безпека у надзвичайних ситуаціях. Визначено ефективність інформаційної системи планування заготівлі молока.

ЗМІСТ

ВСТУП

Визначення задач ефективної заготівлі молока включає врахування подвійних критеріїв забезпечення якості заготовленого молока та мінімізації витрат на заготівлю молока. При цьому молоко із окремих господарств виробників молока можна транспортувати безпосередньо на завод або на пункти збору. Виникає інформаційна задача ефективного планування заготівлі молока із виконанням маршрутизації. Іншими словами, зазначена задача є складнішою, ніж окремі вимоги, що зумовлюють якісну заготівлю молока.

У сфері заготівлі та переробки молока інформаційні системи створюють можливості для аналізу та відбору потрібних даних, швидкого доступу та перевірки стану доставки молока, планування процесу та прогнозування витрат на заготівлю молока. Інформаційні системи інтегровані в діяльність молокопереробних підприємств, тому їх проектують відповідно до конкретних виробничих умов заготівлі молока та характеристик сировинної території громад. Використання спеціалізованих інформаційних систем для планування заготівлі молока дозволяє вирішити такі задачі:

- ✓ Забезпечити якість заготовленого молока;
- ✓ Зниження транспортних витрат при доставці молока;
- ✓ Своєчасне інформування про процес доставки молока;
- ✓ Прискорене виконання замовлень на доставку молока.

Сьогодні проектування та розробка інформаційних систем, які призначені для підвищення ефективності планування діяльності окремих об'єктів агропромислового та інших комплексів, є першочерговим завданням та стосується основних напрямків діяльності багатьох IT розробників. У результаті таких розробок замовники на початковому етапі IT проектів побачать ефект, що можна отримувати від таких розробок та впроваджень кожних різновидів інформаційних систем. Окрім того, це забезпечує планування їх бюджетів, коригування виробничих процесів з метою виконання збору та обробки потрібної для користувачів інформації. При цьому, сьогодні

однією із досить актуальних інформаційних задач, вирішення якої потребує досліджень та розробки, є задача проектування інформаційної системи планування заготівлі молока.

Отже, виконана кваліфікаційна робота «Розробка інформаційної системи планування заготівлі молока із вибором ефективного алгоритму визначення маршрутів» є достатньо актуальною як у теоретичному, так і у практичному значеннях.

Об’єктом дослідження є процеси заготівлі молока, алгоритми визначення раціональних маршрутів, які лежать в основі інформаційної системи планування заготівлі молока.

Предмет дослідження є залежність якості планування заготівлі молока прогнозування від вибраних алгоритмів визначення раціональних маршрутів та архітектури інформаційної системи.

Метою роботи є підвищення ефективності планування заготівлі молока завдяки розробці інформаційної системи, яка забезпечує вибір ефективного алгоритму визначення маршрутів.

РОЗДІЛ 1.
АНАЛІЗ СТАНУ ЗАГОТІВЛІ МОЛОКА ТА ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1. Аналіз стану заготівлі молока

За відомими даними Держкомстату, станом на 01.10.2022 року в Україні налічується близько 1 млн 391,1 тис.корів, що на 15,4% менше порівняно із минулим роком. При цьому, 80,3% усього стада (1117,6 тис.корів) розосереджена у районах, де не ведуться бойові дії. У воєнних зонах утримується орієнтовно 19,7% від усього стада (211,2 тис. голів).

Цей рік поголів'я корів у великих сільськогосподарських підприємствах зменшилось орієнтовно на 9,7%, або ж на 380,9 тис. голів, а у господарствах населення цей показник становить 17,4%, або ж 1010,2 тис. голів.

Так, за наявними статистичними даними за 9 місяців 2022 року виробництво молока сировини становить 5803,8 тис. тон, що орієнтовно на 14,8% менше, порівняно із попереднім роком. Також слід зазначити, що 79,2% молока (або ж 4 млн 594 тис. тон) вироблено на територіях, де немає бойових дій.

Великі молочні господарства надоїли близько 1937,6 тис.тон молока, що характеризує зменшення на 6,8% порівняно із минулим роком, а водночас особисті селянські господарства – 3866,2 тис.тон, що на 18,3% менше порівняно із минулим роком.

Станом на 10.10.2022 року середня ціна закупівлі молока становила 11,60 грн/кг, що характеризує зростання її орієнтовно на 62 коп. у кінці вересня цього року (рис. 1.1). Ціна залежить від гатунку молока іколивається у межах 10,20…12 грн/кг.

Спостерігається зростання ціни на молоко сировину екстра-якості, що можна пояснити тенденцією до зростання попиту на цю сировину молокопереробним підприємствами, які постачання молочні продукти у країни

ЄС. Для цього зазначеним підприємствам необхідно мати якісне молоко сировину.

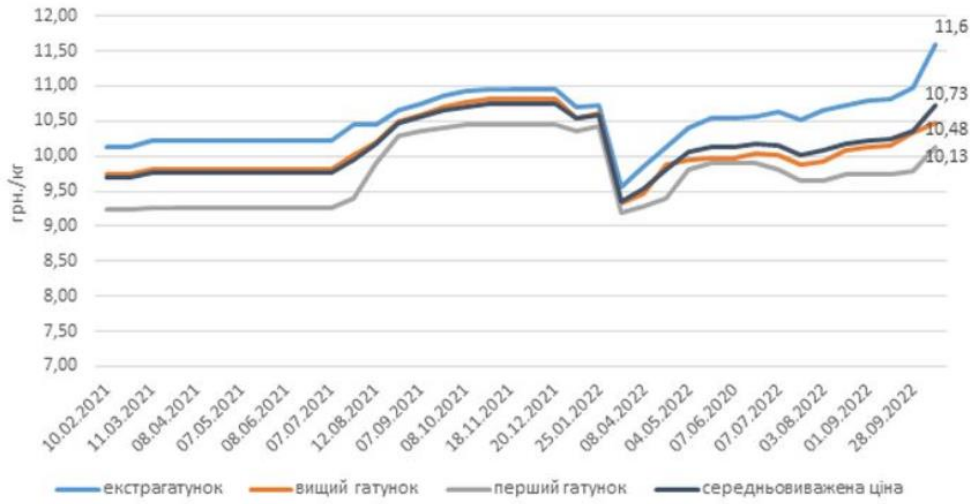

Рис. 1.1. Тенденції зміни закупівельних цін на молоко в Україні

Перший клас молока сировини дорожчає на 35коп. та відповідно становить 10,13грн/кг. Ціна такого молока коливається в межах 9,16…11,0 грн/кг. Середня ціна із взятих трьох сортів молока також зросла на 37 коп. і становить 10,73 грн/кг.

Однак, на підставі врахування падіння світових цін на молоко, впродовж осені 2022 року, очікується прогнозоване падіння цін, у коротко строковій перспективі ціни на молоко сировину не матимуть зростання. Водночас, спостерігається зниження сезонних обсягів виробництва молока.

За відомими даними Держкомстату у січні-вересні 2022 року експорт молока та молочних продуктів у перерахунку на молоко становив 347,3тис.тонн що орієнтовно на 39,4% порівняно із минулим роком. У фінансовому звіті це відповідно становить 252,7 млн. $, що на 66,2 млн. $ більше за минулий рік.

Імпорт молока та молочних продуктів становив 313,4тис.тон, що орієнтовно на 46,2% менше за показники минулого року. Ця вартість склала 172,8 млн.$, що на 98,4 млн.$ менше, ніж за 2021 рік.

У вересні 2022 року було зафіксовано рекордне значення виручки від експорту молочних продуктів. Загалом продано товарів на 35,9 млн.$, що

орієнтовно на 9,2% більше за липень місяць, і орієнтовно на 145% порівнюючи із минулим роком (рис. 1.2).

Рис. 1.2. Тенденції зміни експорту Україною молочних продуктів

До основної категорії експорту належить сухе молоко – 4,9 тис. тонн (+51% до серпня). Його експортовано на 15,8 млн.$., що свідчить про зростання на +39,8%. Окрім того, експорт незбираного молока та вершків у вересні 2022 року впав на чверть, що відповідно становить 2,65 тис. тон, або ж -25,7% та у вартісному виразі це становить 1,37 млн.$, або ж -28,14%.

Величезною проблемою в Україні є заготівля молока сировини від населення. На даний час зруйновано стару систему заготівлі молока – посередницьку, тому що скупники молока фактично займаються здирництвом у населення. Системної роботи щодо покращення якості молока вони не проводять.

Сьогодні заготівлею займаються в основному підприємці або керівництво дрібних молокопереробних підприємств, які створили додаткові організації (про які не знають навіть власники цих підприємств), які збирають молоко самостійно.

Для вирішення непростої ситуації із заготівлею молока спочатку слід провести атестацію підприємств, які збирають молоко від населення, вимоги до них потрібно чітко розписати. Необхідно кардинально змінити спосіб закупівлі

молока у населення. Для цього розробка інформаційних систем планування заготівлі молока дасть можливість частково вирішити існуючі задачі щодо заготівлі молока у дрібних його виробників.

Отже, заготівля молока відіграє важливе значення у ефективності виробництва молочних продуктів та їх якості. Для підвищення ефективності заготівлі молока слід використовувати інформаційні системи, що забезпечать точне планування відповідної діяльності. Це свідчить про доцільність розробки інформаційні системи планування заготівлі молока, що є досить актуальним сьогодні для молокопереробних цехів та підприємств.

1.2. Аналіз інформаційних систем планування заготівлі молока

На даний час у світі спостерігається тенденція до діджиталізації усіх процесів виробничої діяльності, у тому числі і заготівлі молока. Зокрема, у світі створюються молочна кооперативи. Кількість молока, яку заготовляє кооператив «Молочний союз», є одним із найважливіших факторів у його загальному плануванні діяльності. Діяльність із закупівлі молока розповсюджена на великі географічні території та включає велику кількість незалежних молочних кооперативів (DCS) на рівні села та тисячі фермерів, які є членами цих товариств.

Молочний союз може впливати на рівень закупівель через ціну, яку він сплачує своїм асоційованим товариствам, а також ціну та доступність основних технічних ресурсів, які він їм надає, а саме постачання кормів для великої рогатої худоби, послуги штучного осіменіння та ветеринарні послуги.

Географічна інформаційна система (ГІС) – це програма інформаційних технологій, яка включає набір інструментів ГІС із сильним ефектом візуалізації для моніторингу вищезазначених дій, швидкої діагностики проблемних областей, прийняття рішень щодо втручання, необхідних для подолання проблем, з якими стикаються, і для бізнес-планування.

ГІС для молочних союзів була задумана та розроблена виключно для використання кооперативними молочними союзами (рис. 1.3).

Рис. 1.3. Складові ГІС для молочних кооперативів

Однією із складових ГІС для молочних кооперативів є управління транспортуванням молока/експлуатацією автопарку від DCS до холодильного центру/молочного заводу.

Маршрути закупівлі молока для «Молочної спілки» зазвичай плануються з подвійною метою: максимізувати обсяг закупленого молока і в той же час мінімізувати вартість закупівлі молока відповідно до набору поточних обмежень. Крім того, сезонні чи економічні обмеження можуть призвести до змін у цих молочних маршрутах.

У пропонованій ГІС-програмі можна зобразити ці молочні шляхи на оцифрованих картах, і процес візуалізації альтернативних молочних шляхів стане простішим у порівнянні з ручними процесами.

Більшість молочних спілок мають комп’ютеризовану систему виставлення рахунків за молоко для обробки даних про закупівлю молока для здійснення платежів DCS. Молочний союз може використовувати наявну інформацію для створення карти молочних маршрутів на основі своїх існуючих

молочних маршрутів між DCS та охолоджуючим центром/молочним заводом, які потрібно змінювати, коли відповідні зміни будуть внесені на місці.

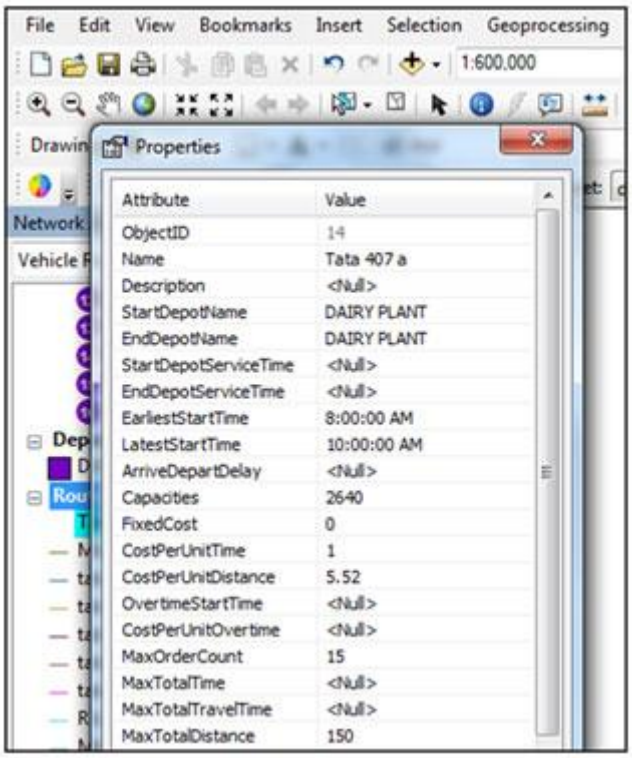

Рис. 1.4. Вхідні дані для маршрутизації транспортного засобу

Наявність оновленої мережі доріг на рівні села є головним вузьким місцем, оскільки це є вирішальним входом для програмного забезпечення для оптимізації маршруту. Хоча намір полягав у тому, щоб використати всю можливу мережу доріг для оптимізації, визначення маршрутів виконується лише для мереж доріг, які могли бути зафіксовані керівниками на рівні села.

Фермерські організації стикаються з більш вираженими проблемами щодо найму та утримання співробітників з досвідом IT. Проте збір даних на рівні окремих господарств за допомогою GPS не є складним завданням, оскільки існуючі наглядачі на рівні села можуть легко використовувати портативний пристрій GPS.

Використання програмного забезпечення для оптимізації маршрутів саме по собі є невеликою частиною проблеми, більшою проблемою є організація належної бази даних і її оновлення для аналізу в режимі реального часу.

Route Name	Vehicle type	Trip distance (km)	Trip cost
1	TATA 407	119	657
2	MAX 207	91	521
3	TATA 407	108	596
4	TATA 407	156	874
5	TATA 407	114	645
6	TATA 407	141	835
7	TATA 407	143	847
8	RIKSHAW	24	112
9	Max Mah	102	574
10	TATA 407	114	612
11	TATA 207	100	588
12	Max Mah	173	862
13	MINI DOR	60	358
	TOTAL	1445	8080

Рис. 1.5. Формування бази даних і її оновлення в режимі реального часу

Результати планування маршрутів представлено на рис. 1.6.

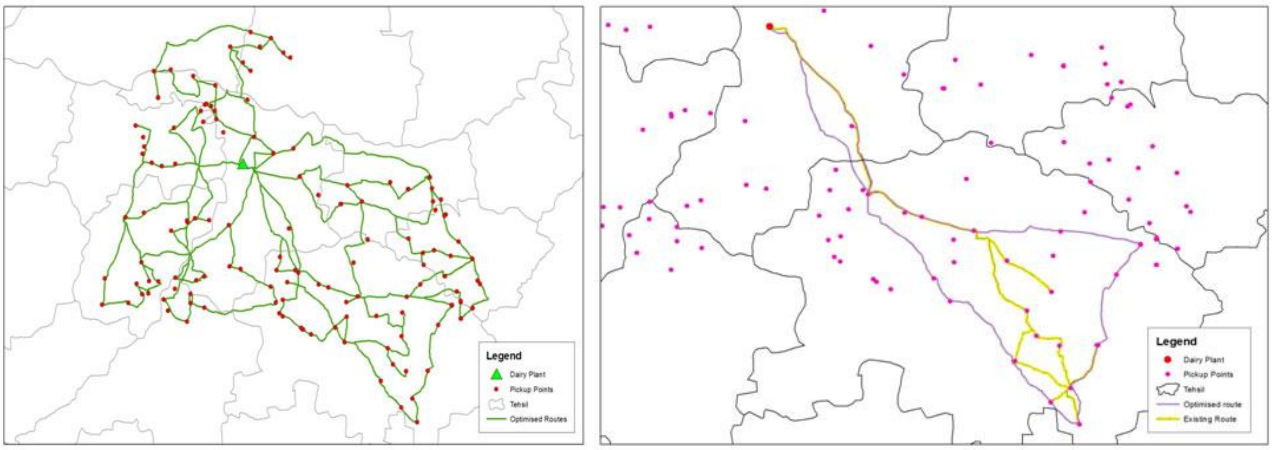

Рис. 1.6. Результати планування маршрутів

Впровадження ГІС-технології в фермерських організаціях було б досить корисним, враховуючи поточний сценарій доступності економічного апаратного та програмного забезпечення, а також враховуючи, що надається належна підтримка та навчання. Потрібна також початкова взаємодія з виконавцями та представниками господарств виробників молока, а також забезпечення розгортання зручного програмного забезпечення.

Фермерські організації можуть ефективно використовувати технологію ГІС для перевірки ефективності та візуалізації своїх існуючих маршрутів збору молока, а також для виявлення можливостей для вдосконалення.

Таким чином, використання технології ГІС для оптимізації маршруту закупівлі молока значно сприятиме цим молочним підприємствам з точки зору економії грошей і часу.

1.3. Існуючі особливості заготівлі молока та проектування інформаційних систем її планування

На сьогодні в Україні існує реальна проблема збору сирого молока, з якою зіткнулася громади та цехи з виробництва молочних продуктів. Заготівельники повинні збирати молоко з ряду ферм-виробників, розташованих у великій географічній зоні, і транспортувати його на молокопереробний завод. Все вироблене молоко має бути зібрано, оскільки всі ферми належать до однієї системи.

Оскільки різні ферми виробляють одну з трьох можливих якостей молока, його збір автоцистернами зазвичай здійснюється або різними транспортними засобами різної якості, або одними транспортними засобами з відокремленими відсіками. Передбачається, що молоко різної якості можна змішувати в одному відсіку або одній вантажівці. Після надходження на завод змішаний продукт класифікується відповідно до найнижчої якості молока, що міститься в ньому, таким чином знижуючи його комерційну цінність і тому дохід від кінцевої продукції. З іншого боку, поєднання різних якостей молока таким чином значно знижує витрати на транспортування, тим самим збільшуючи прибуток.

Найпростіший спосіб збору молока – відвідування вантажівками кожного виробника безпосередньо на фермі (забір молока від дверей до дверей). Якщо маршрут збору довгий і має кілька зупинок, транспортні витрати будуть, як

правило, високими. Таким чином, пряме транспортування вантажівкою може бути неефективним, якщо є багато дрібних виробників, розташованих далеко від заводу з переробки молока. Особливо це стосується розділеного збору молока різної якості. У таких ситуаціях процес збору можна було б полегшити шляхом створення пунктів збору, де продукція дрібних виробників, розташованих далеко від заводу, могла б зберігатися в резервуарах. Таким чином, вантажівки можуть зібрати загальну молока від групи таких виробників, незалежно від якості молока, за одну поїздку, таким чином скорочуючи час і вартість транспортування.

Отже, для вирішення задач громад, які займаються виробництвом молока слід створювати інформаційні системи планування заготівлі. Під час проектування інформаційних систем планування слід враховувати особливості заготівлі молока на території громад.

1.4. Завдання кваліфікаційної роботи

Проведений аналіз стану виробництва та заготівлі молока, а також існуючих у світі інформаційних систем планування заготівлі молока вказують на доцільність розроблення та використання їх для громад України, які мають виробників молока. Відповідно до цього, у кваліфікаційній роботі пропонується розробка інформаційної системи планування заготівлі молока із вибором ефективного алгоритму визначення маршрутів. Для цього слід вирішити у роботі слід розв’язати такі завдання:

- провести аналіз стану предметної галузі та використання інформаційних систем для планування заготівлі молока;
- обґрунтувати інструментарій та здійснити вибір ефективного алгоритму визначення маршрутів;
- розробити інформаційну систему планування заготівлі молока;

- розробити заходи стосовно охорони праці та безпеки у надзвичайних ситуаціях;
- визначити економічну ефективність від інформаційної системи.

РОЗДІЛ 2.
ОБҐРУНТУВАННЯ ТА ВИБІР ІНСТРУМЕНТАРІЮ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАНУВАННЯ ЗАГОТІВЛІ МОЛОКА

2.1. Модель функціональних можливостей інформаційної системи планування заготівлі молока

Функціональною частиною інформаційної системи планування заготівлі молока є фактично модель системи управління об’єктами (рис. 2.1). Оскільки складні системи завжди багатофункціональні, інформаційні системи можна класифікувати за різними ознаками:

- ✓ рівень управління (вищий, середній, оперативний);
- ✓ тип керованих ресурсів (матеріальні, трудові, фінансові та інформаційні ресурси);
- ✓ сфера застосування (банківська інформаційна система, статистика, податкова, бухгалтерія, страхування тощо);
- ✓ функції та етапи управління.

Захищена частина інформаційної системи планування заготівлі молока включає види технічної безпеки, інформаційної, технологічної, математичної, організаційної, правової, ергономічної та ін. Характерними особливостями комп’ютерних інформаційних систем великих компаній є:

- ✓ довгий життєвий цикл;
- ✓ різноманітність використовуваного апаратного забезпечення, низький життєвий цикл створюваної системи;
- ✓ різноманітне програмне забезпечення;
- ✓ масштаб і складність розв'язуваного завдання;
- ✓ перетин багатьох різних предметних областей;
- ✓ регіональне поширення та орієнтація на використання локальних і глобальних комп'ютерних мереж для обміну та обробки інформації.

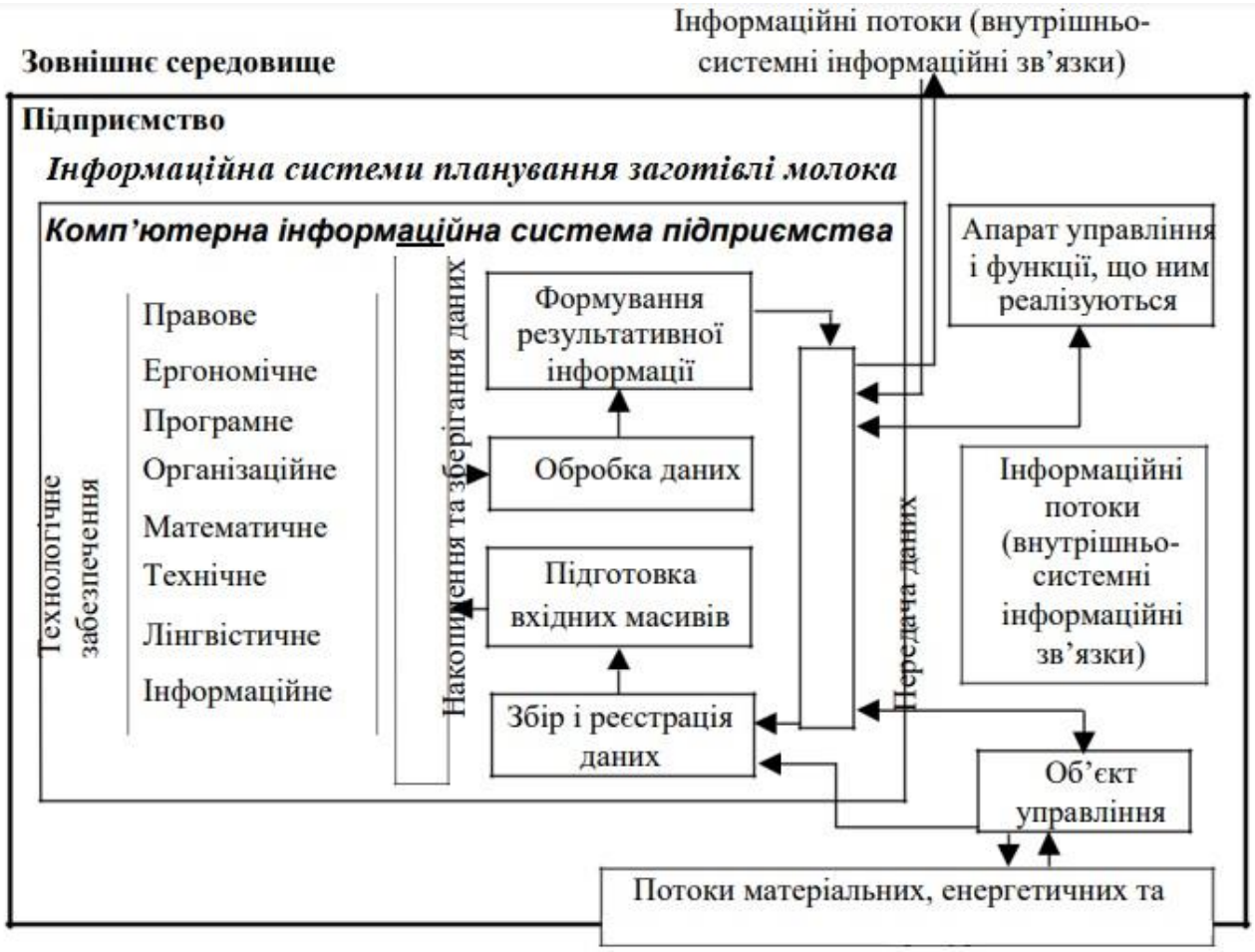

Рис. 2.1. Модель функціональних можливостей інформаційної системи планування заготівлі молока

При управлінні системою виникають питання щодо сумісності програмного забезпечення, інформаційної безпеки, незалежності апаратної та програмної платформ, розмежування доступу до джерел інформації. Залежно від технологічного та функціонального аспектів інформаційну систему підприємства можна розділити на декілька складових.

Із розвитком окремих територіальних громад, збільшенням розподілу посівних площ, виробничих і складських підрозділів, збільшився обсяг заготівлі молока, а також потік матеріальної, фінансової та інформації. Існує потреба для управління процесом заготівлі молока використовувати інформаційні системи, так як цей процес ускладнився. Це призводить до створення комплексної інформаційної системи планування заготівлі молока, здатної підвищити

стійкість громади в ринкових умовах і зробити виробничу діяльність більш передбачуваною.

Незважаючи на очевидні переваги інформаційної системи планування заготівлі молока, її впровадження пов’язане з наступними проблемами:

- ✓ необхідністю серйозної реструктуризації існуючої моделі відносин між виробниками молока та його заготівельниками, яка вироблялася у окремій громаді протягом багатьох років;
- ✓ оптимізації організаційної структури громади;
- ✓ координації виконання робіт із заготівлі молока;
- ✓ системне навчання членів громади використанню інформаційної системи тощо.

На ринку програмного забезпечення для аграрного виробництва представлені різні інформаційні системи, які враховують ідеологію побудови виробничих процесів, функціональні можливості, вимоги до впровадження за заданої предметної галузі, складність розробки та експлуатації, вартість одного робочого місця тощо.

Слід зазначити, що кожна інформаційна система, представлена на ринку, може вирішити окремі задачі для громад. Однак, інформаційної системи для планування заготівлі молока із урахуванням їх особливостей, обсягів заготівлі молока та інших факторів, нажаль ще немає.

2.2. Вибір ефективного алгоритму визначення маршрутів

Одним з ключових аспектів даної роботи є вибір ефективного алгоритму для визначення маршрутів. Існує велика кількість алгоритмів для визначення маршрутів. Нижче наведено порівняння деяких із них, зокрема алгоритму Дейкстри, алгоритму Белмана-Форда та алгоритм Флойда-Уоршела.

2.2.1. Алгоритм Дейкстри

Алгоритм Дейкстри дає можливість знаходити найкращі маршрути для графу, для якого вибирається початкова вершина, а маршрут для усіх інших невідомий (рис. 2.2).

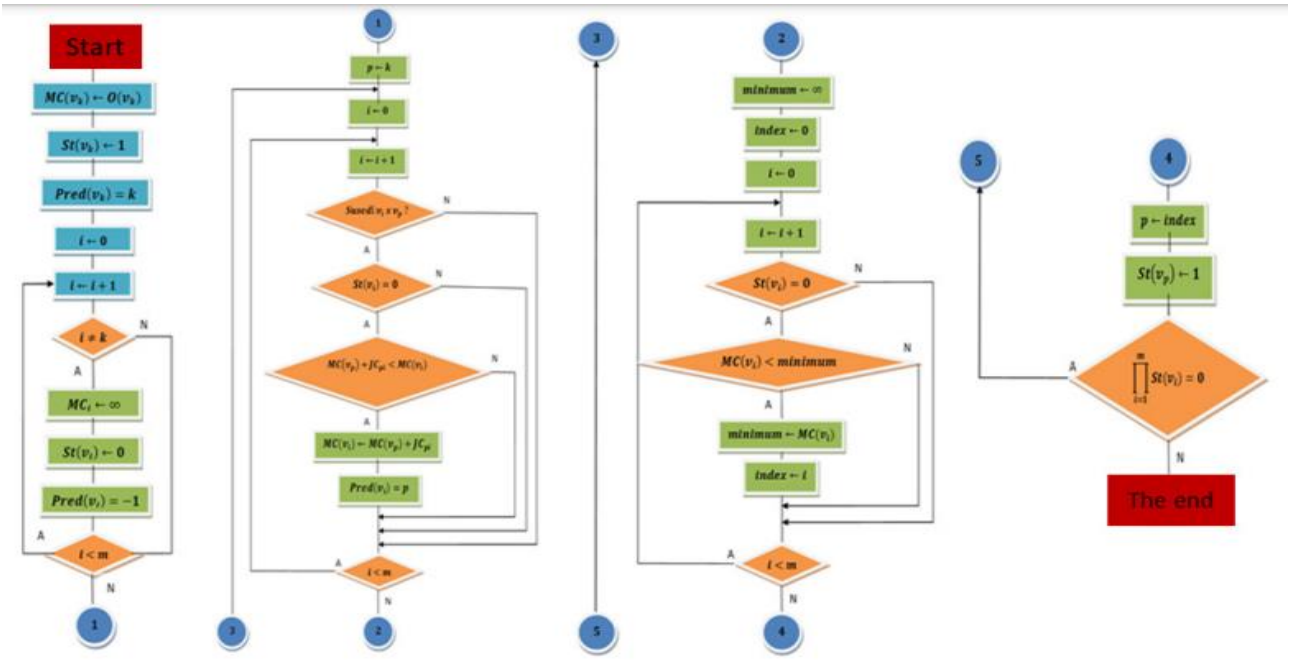

Рис. 2.2. Блок-схема алгоритму Дейкстри

Алгоритм Дейкстри – алгоритм визначення найкоротших шляхів у графі. Знаходить найкоротший шлях від однієї вершини (яка називається вихідною вершиною) до інших вершин. Основним мінусом алгоритму Дейкстри є те, що він не може працювати з від'ємними ребрами, а лише з тими, котрі мають додатне значення, і в такому випадку від даного алгоритму ми змушені відмовитись. Автором алгоритму є голландський вчений Едсгер Дейкстра. Алгоритм використовує жадібний підхід. У кожній ітерації вибирається одна з невідвіданих вершин, яку можна досягти з найменшою ціною. Після визначення шляху до конкретної вершини він не буде змінений протягом решти алгоритму.

Під час виконання алгоритму для кожної вершини визначаються дві величини: вартість досягнення цієї вершини та попередньої вершини на шляху.

На початку алгоритму вартість переходу до вихідної вершини дорівнює 0 (ми вже там), і нескінченність для кожної іншої вершини (ми взагалі не знаємо, як туди потрапити). Усі вершини на початку знаходяться в множині Q (вони не є пройденими вершинами). Тоді алгоритм виглядає наступним чином:

- Поки Q не порожній:
- Знайдіть вершину з найменшою ціною доступу з Q. Позначте його як v і видаліть із Q.
- Для кожного ребра, що виходить з вершини v (назвемо її k), виконайте наступне:
- Позначте вершину на іншому кінці ребра k як u.
- Якщо вартість проїзду до u від v через ребро k менша за фактичну вартість проїзду до u, тоді:
- Призначте вартість переходу до вершини u вартості переходу до вершини v плюс вагу ребра k.
- Встановити вершину v як попередницю вершини u.

```
/// <summary>
/// Алгоритм Дейкстри
/// </summary>
public class Dijkstra
{
    Graph graph;

    List<GraphVertexInfo> infos;

    /// <summary>
    /// Конструктор
    /// </summary>
    /// <param name="graph">Граф</param>
    public Dijkstra(Graph graph)
    {
        this.graph = graph;
    }

    /// <summary>
    /// Ініціалізація інформації
    /// </summary>
    void InitInfo()
    {
        infos = new List<GraphVertexInfo>();
        foreach (var v in graph.Vertices)
        {
            infos.Add(new GraphVertexInfo(v));
        }
    }

    /// <summary>
    /// Отримання інформації про вершину
    /// </summary>
    /// <param name="v">Вершина</param>
    /// <returns>Інформація про вершину</returns>
    GraphVertexInfo GetVertexInfo(GraphVertex v)
    {
```

Рис. 2.3. Фрагмент вихідного коду алгоритму Дейкстри

Позначимо кількість вершин через n, а кількість ребер через e. Кожне ребро аналізуватиметься один раз у випадку орієнтованого графа або двічі у випадку неорієнтованого графа. Окрім аналізу ребер, під час виконання алгоритму n разів шукається множина Q. У разі зберігання множини Q у регулярному масиві пошук вершини відбувається за лінійний час (O(n)). Тоді часова складність алгоритму становить O(n2+e). Враховуючи те, що за відсутності кількох ребер, їх кількість завжди менше n2, можна сказати, що часова складність алгоритму дорівнює O(n2).

Якщо ми зберігаємо набір Q як множину, пошук елемента займе O(logn) часу. Однак кожного разу, коли вартість досягнення вершини змінюється, купа повинна бути перебудована, що також займає O (logn) часу. Тоді обчислювальна складність алгоритму становитиме O(elogn). Тому для розріджених графів (кількість ребер менша за порядком величини, ніж n2/logn) це швидше рішення, але для щільних графів – повільніше. Якщо ми використовуємо купу Фібоначчі для зберігання набору Q, обчислювальна складність алгоритму зменшується до O(nlogn+e).

2.2.2. Алгоритм Белмана-Форда

Якщо шляхи графа мають невід’ємні ваги, то найкращим вирішенням цієї проблеми є алгоритм Дейкстри, представлений вище. У деяких програмах шляхи можуть мати від’ємні ваги. У цьому випадку ми повинні використовувати трохи менш ефективний, але більш універсальний алгоритм Белмана-Форда (рис. 2.4). Алгоритм дає дійсний результат, лише якщо граф не містить від’ємного циклу, тобто циклу, у якому сума ваг ребер від’ємна. Якщо такий цикл існує в графі, то кожен шлях можна «скоротити», пройшовши негативний цикл багато разів. У цьому випадку алгоритм Белмана-Форда повідомляє про помилку.

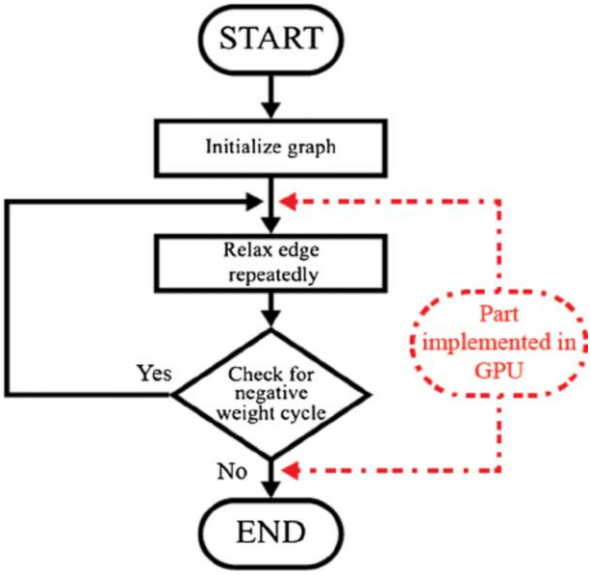

Рис. 2.4. Блок-схема алгоритму Белмана-Форда

Алгоритм, описаний тут, створить два n-елементних масиви даних (n — кількість вершин у графі):

d – і-й елемент містить вартість переходу від початкової вершини до і-ї вершини графа по найкоротшому шляху. Для початкової вершини вартість переходу до неї дорівнює 0.

p – і-й елемент містить номер вершини графа, яка є попередником і-ї вершини на найкоротшому шляху. Для початкової вершини антецедент дорівнює -1.

На початку алгоритму ми встановлюємо для всіх клітинок таблиці d максимально можливе значення (спочатку до нескінченності), за винятком клітинки, що представляє початкову вершину, у якій ми ставимо 0. Однак у всіх клітинках таблиці p , ставимо -1 (у графі немає вершини з номером -1, отже, немає антецеденту).

```
{
    class Ford
    {
        static void Main()
        {
            int n, s, m, k = 0, mod_e; // mod_e - Кількість ребер (у всьому графі)
            mod_e = k; // k - лічильник масивів ребер
            int[] a = new int[200]; // масив почав ребер
            int[] b = new int[200]; // масив кінців ребер
            int[] w = new int[200]; // масив ваг ребер
            int[] d = new int[200]; // масив відстаней

            Console.WriteLine("Ввести кількість вершин:");
            n = int.Parse(Console.ReadLine()); // Количество вершин
            Console.WriteLine("\nВведите номер стартовой вершины:");
            s = int.Parse(Console.ReadLine()); // Стартовая вершина
            for (int i = 1; i <= n; i++) // Задание массивов ребер
            {
                Console.WriteLine("\nВвести кількість ребер, виходячих із вершини " +i);
                m = int.Parse(Console.ReadLine());
                for (int j = 1; j <= m; j++)
                {
                    a[k] = i;
                    Console.WriteLine("\nВведіть номер кінцевої вершини для "+ j + "-го ребра, виходячого з "+ i+ ":");
                    b[k] = int.Parse(Console.ReadLine());
                    Console.WriteLine("\nВведіть вагу "+ j+ "-го ребра, виходячого з "+ i+ ":");
                    w[k] = int.Parse(Console.ReadLine());
                    k++;
                }
                mod_e++;
            }
            for (int i = 1; i <= n; i++) d[i] = 999999;
            d[s] = 0;
            for (int i = 1; i < n; i++)
            {
                if (i == s) i++;
                else
                    {
                        for (int j = 1; j <= mod_e; j++)
                            {
                                if (d[b[j]] > (d[a[j]] + w[j])) d[b[j]] = d[a[j]] + w[j];
```

Рис. 2.5. Вихідний код алгоритму Белмана-Форда

Потім ми робимо n - 1 циклів, у яких розслаблюємо ребра (кожен цикл визначає вартість досягнення принаймні однієї вершини графа, оскільки початкова вершина має вартість 0, нам потрібно визначити вартість лише для n - 1 вершин, тому нам потрібно щонайбільше n - 1 ланцюгів циклу). Він полягає в тому, що ми проходимо по черзі всі ребра графа. Якщо ми натрапляємо на ребро u–v ваги w, для якого вартість отримання d [v] більша за вартість отримання d [u] + w (тобто потрапляння до вершини v із вершини u цим ребром дешевше, ніж раніше знайдені маркери), тоді ми встановлюємо вартість d [v] на d [u] + w і в таблиці антецедентів для p [v] ставимо номер вершини u. Коли цикл робить n - 1 раундів, у таблиці d ми матимемо витрати на досягнення окремих вершин графа по найкоротшим шляхам, а в таблиці p для кожної вершини знаходимо її попередника на найкоротшому шляху від початкової вершини.

Також необхідно перевірити, чи немає на графіку негативного циклу. Для цього ми знову проходимо набір ребер, і якщо ми натрапляємо на ребро u-v з вагою w, для якого подальша вартість доступу d [v] більша ніж d [u] = w, то ми маємо справу з негативний цикл (зазвичай така ситуація не може виникнути,

оскільки релаксація повинна встановити d[v] на d[u] + w. Лише за наявності негативного циклу релаксація не вдається). У цьому випадку алгоритм повинен повідомити про помилку.

2.2.3. Алгоритм Флойда-Уоршела

Алгоритм Флойда-Уоршела – алгоритм, який використовується для визначення найкоротших шляхів між кожною парою вершин у графі. Це алгоритм, заснований на динамічному програмуванні. Алгоритм має часову складність O(n3) і складність пам’яті O(n2), де n – кількість вершин (рис. 2.6).

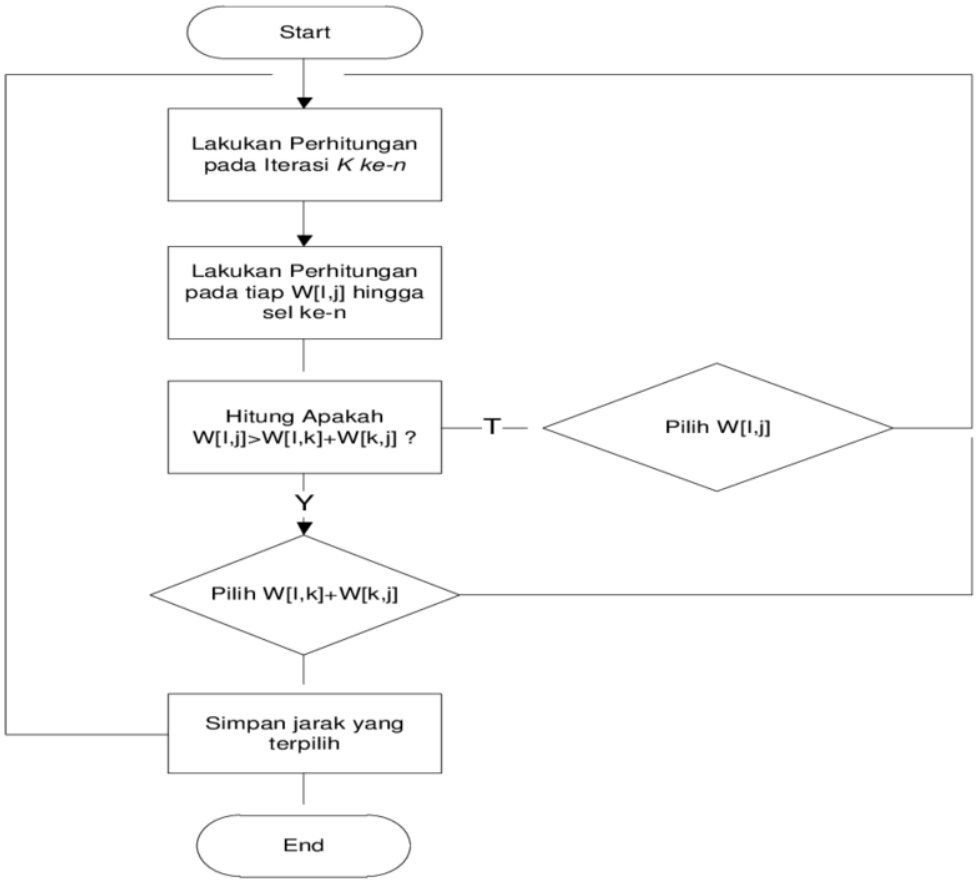

Рис. 2.6. Блок-схема алгоритму Флойда-Уоршела

Алгоритм допускає негативно зважені ребра, якщо вони не створюють негативних циклів. Алгоритм також можна використовувати для пошуку від’ємних циклів на графіку.

Ми створюємо дві матриці з розмірами n на n. У першій матриці ми будемо зберігати відстані між точками, а в другій матриці ми будемо зберігати ідентифікатори передостанніх точок на шляху, що з'єднує точки. Більш формально, di,j представляє відстань від точки i до точки j, а pi,j представляє точку перед точкою j на шляху від точки i до точки j.

```
public const int INF = 99999;

private static void Print(int[,] distance, int verticesCount)
{
        Console.WriteLine("Shortest distances between every pair of vertices:");

        for (int i = 0; i < verticesCount; ++i)
        {
                for (int j = 0; j < verticesCount; ++j)
                {
                        if (distance[i, j] == INF)
                                Console.Write("INF".PadLeft(7));
                        else
                                Console.Write(distance[i, j].ToString().PadLeft(7));
                }

                Console.WriteLine();
        }
}

public static void FloydWarshall(int[,] graph, int verticesCount)
{
        int[,] distance = new int[verticesCount, verticesCount];

        for (int i = 0; i < verticesCount; ++i)
                for (int j = 0; j < verticesCount; ++j)
                        distance[i, j] = graph[i, j];

        for (int k = 0; k < verticesCount; ++k)
        {
                for (int i = 0; i < verticesCount; ++i)
                {
                        for (int j = 0; j < verticesCount; ++j)
                        {
                                if (distance[i, k] + distance[k, j] < distance[i, j])
                                        distance[i, j] = distance[i, k] + distance[k, j];
                        }
                }
        }
```

Рис. 2.7. Фрагмент вихідного коду алгоритму Флойда-Уоршела

На початку ми ініціалізуємо значення в матрицях. Якщо є ребро, що веде від точки i до точки j, ми присвоюємо di,j довжині цього ребра та встановлюємо pi,j рівним i. В іншому випадку ми присвоюємо di,j нескінченності та розглядаємо pi,j як невизначений. Ми встановлюємо довжину шляхів, що ведуть до тієї самої точки, з якої вони починаються, рівним 0.

Далі виконуємо основну частину алгоритму:

Для кожної точки u з множини вершин:

Для кожної точки v1 з множини вершин:

Для кожної точки v2 з множини вершин:

Якщо dv1, v2 > dv1, u + du, v2, то:

dv1, v2 = dv1, u + du, v2,

pv1, v2 = pu, v2.

Після виконання алгоритму матриці містять оптимальні рішення. Шлях між будь-якими двома точками можна прочитати за допомогою значень матриці p. Якщо di,j все ще дорівнює нескінченності після виконання алгоритму, то шляху від точки i до точки j немає.

Якщо головна діагональ матриці відстані менше нуля після виконання алгоритму, графік містить негативний цикл.

Для нашого випадку все трохи складніше, тому що крім вирахування оптимального маршруту заготівлі, нам також потрібно орієнтуватись на місткість транспортних засобів та кількість заготівельної продукції. Це зумовлює собою велику кількість обрахунків які потрібно проводити дуже швидко. Отже маючи дані від клієнтів з координатами їх розташування та кількістю заготівельної продукції ми вже можемо сформувати матрицю відстаней між усіма точками на карті, включаючи наш пункт збору-заготівлі.

Наступним кроком нам потрібно вирахувати загальну кількість продукції для підбору транспортних засобів, котрі в свою чергу вирушать до клієнтів. Маючи необхідні транспортні засоби, ми також добавляємо їх аргументами до нашого алгоритму вирахування.

Провівши ретельні обрахунки відстаней, місткості та кількості продукції, а також базуючись на матриці відстаней, ми отримуємо план заготівлі молока, у якому вказано який транспортний засіб, на який маршрут (за необхідності можливий випадок що один транспортний засіб може здійснити не одну поїздку) він вирушатиме та які пункти збору йому необхідно буде заїхати, щоб зібрати усю необхідну продукцію.

2.2.4. Порівняння алгоритмів визначення маршрутів

З метою вибору ефективного алгоритму визначення маршрутів проведено порівняльний аналіз, які стосуються результатів визначення маршрутів доставки молока від господарств громади до цеху його переробки із маршрутами, виконаними із використанням алгоритму Дейкстри, алгоритму Белмана-Форда та алгоритму Флойда-Уоршела. Результати проведених досліджень нами наведені на рис. 2.8.

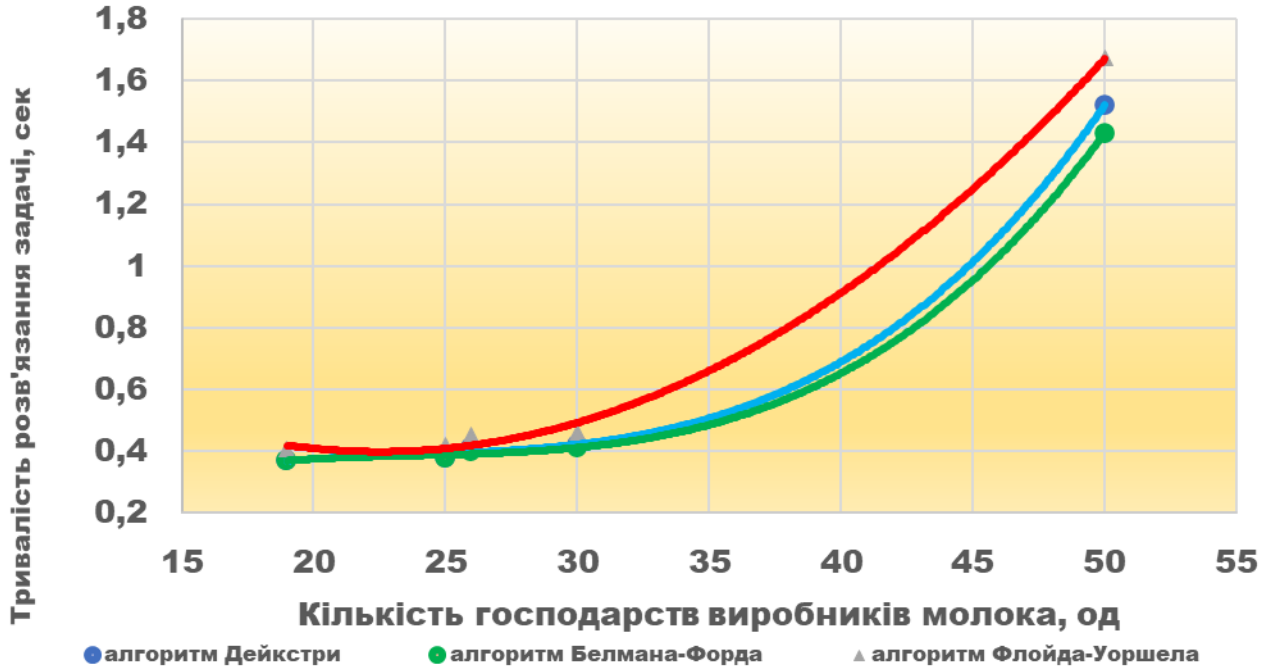

Рис. 2.8. Залежність тривалості визначення маршрутів доставки молока від господарств громади до цеху його переробки від кількості господарств виробників молока

На підставі залежності тривалості визначення маршрутів доставки молока від господарств громади до цеху його переробки від кількості господарств виробників молока можна сказати, що за тривалістю вирішення поставленої задачі найкращим є алгоритм Белмана-Форда, який забезпечує зниження часу на 3,2…6,8%.

Стосовно точності визначення маршрутів транспортування молока, то жоден із них під час їх порівняння не показав однозначно кращого результату.

Тобто, за різної кількості господарств виробників молока та обсягів його заготівлі, раціональний алгоритм змінювався. Це свідчить про те, що проектована інформаційна система заготівлі молока для заданих виробничих умов на території громади повинна передбачати використання усіх зазначених алгоритмів, постійно їх порівнювати між собою та вибирати кращий.

2.3. Обґрунтування доцільності використання Google Maps Platform у інформаційній системі планування заготівлі молока

Платформа Google Maps являє собою набір додатків, які забезпечують створення допомоги для користувачів. Вона дає можливість відобразити місця, де знаходиться розташовані господарства виробники молока, забезпечити ефективне формування маршрутів із проаналізованими нами алгоритмами, а також графічне представлення результатів визначення маршрутів.

2.3.1. Використання служби Directions в Maps JavaScript API

Перш ніж використовувати службу Directions в Maps JavaScript API, спершу слід переконатися, що Directions API увімкнено в Google Cloud Console у тому самому проекті, який налаштували для Maps JavaScript API.

Доступ до сервісу Directions є асинхронним, оскільки Google Maps API потребує виклику зовнішнього сервера. З цієї причини потрібно передати метод *зворотного виклику* для виконання після завершення запиту. Цей метод зворотного виклику має обробляти результати. Відомо, що служба Directions може повертати більше одного можливого маршруту як масив окремих routes[].

Щоб планувати маршрути в Maps JavaScript API, насамперед слід створити об’єкт type DirectionsServiceand call DirectionsService.route(), щоб

ініціювати запит до служби Directions, передавши йому *DirectionsRequest* літерал об’єкта, що містить вхідні терміни та метод зворотного виклику для виконання після отримання відповіді (рис. 2.9).

```
getDirections(points: ICollectionPointReport[]): google.maps.DirectionsResult {
    if (!points) {
        return;
    }

    const depotStart = points[0];
    const depotEnd = points[points.length - 1];

    const request: google.maps.DirectionsRequest = {
        destination: { lat: depotStart.latitude, lng: depotStart.longitude },
        origin: { lat: depotEnd.latitude, lng: depotEnd.longitude },
        waypoints: [],
        travelMode: google.maps.TravelMode.DRIVING
    };

    for (let index = 1; index < points.length - 1; index++) {
        const point = points[index];
        request.waypoints.push({
            location: new google.maps.LatLng(point.latitude, point.longitude),
            stopover: true
        })

        this.directionsResults$ = this._mapDirectionsService.route(request).pipe(map(response => response.result));
    }
}
```

Рис. 2.9. Фрагмент вихідного коду об’єкта *DirectionsRequest*

Ініціювання запиту маршруту до методу DirectionsServicewith route() вимагає передачі зворотного виклику, який виконується після завершення запиту служби. Цей зворотній виклик поверне у відповідь код DirectionsResulti DirectionsStatus.

DirectionsResult містить результат запиту маршрутів, який можна обробити самостійно або передати об’єкту DirectionsRenderer, який може автоматично обробити відображення результату на карті.

Щоб відобразити DirectionsResult за допомогою DirectionsRenderer, слід зробити наступне:

1. Створити об'єкт DirectionsRenderer.
2. Викликати setMap(), щоб прив’язати його до переданої карти.
3. Викликати setDirections(), можна передаючи йому DirectionsResult як зазначено вище. При цьому засіб візуалізації MVCObject автоматично

виявить будь-які зміни у своїх властивостях і оновить карту, коли пов’язані напрямки зміниться.

У наведеному нижче прикладі розраховуються маршрути між двома заданими точками, де вихідний і кінцевий пункти встановлюються значеннями "start" та "end" в розкривних списках. При цьому відображається ламана лінія між вказаними місцями, а DirectionsRenderer забезпечує розміщення маркерів у початковій точці, пункті призначення та будь-яких маршрутних точках, якщо це можливо.

```
<div *ngIf="_dataReport">
    <div *ngFor="let depotReport of _dataReport.depotReports; let depotIndex = depotReportIndex">
        <p-panel [toggleable]="true" [transitionOptions]="'350ms'">
            <ng-template pTemplate="header">
                Depot: {{ depotReport.name }}
            </ng-template>
            <div *ngFor="let routeReport of depotReport.routeReports; let i = routeReportIndex">
                <p-panel [toggleable]="true" [transitionOptions]="'350ms'">
                    <ng-template pTemplate="header">
                        Route for vehicle: {{ routeReport.vehicleName }}. Total distance:
                        {{routeReport.totalDistance}}. Total Amount: {{routeReport.totalLoad}}
                    </ng-template>
                    <ng-template pTemplate="footer">
                        Route: {{routeReport.formattedWay}}
                    </ng-template>
                    <div>
                        <google-map width="100%" [options]="_mapLoader.options">
                            <map-directions-renderer
                                *ngIf="(directionsResults$ | async) as directionsResults"
                                [directions]="directionsResults"></map-directions-renderer>
                        </google-map>
                    </div>
                </p-panel>
            </div>
        </p-panel>
    </div>
</div>
```

Рис. 2.10. Фрагмент вихідного коду об’єкта *DirectionsRenderer*

Відомо, що у DirectionsRequest, можна вказати маршрутні точки (типу DirectionsWaypoint) під час розрахунку маршрутів за допомогою служби Directions для автомобільних маршрутів. Точки маршруту дозволяють розрахувати маршрути через додаткові місця, у цьому випадку повернутий маршрут проходить через задані маршрутні точки.

При цьому waypoint складається з таких полів:

- location(обов’язково) вказує адресу маршрутної точки.
- stopover(необов’язково) вказує, чи ця маршрутна точка є фактичною зупинкою на маршруті (true) чи натомість лише складовою маршруту через вказане місце (false). Зупинки встановлюються true за замовчуванням.

2.3.2. Використання сервісу Google's Distance Matrix

Сервіс Distance Matrix від Google обчислює відстань і тривалість поїздки між кількома пунктами відправлення та призначення з використанням певного маршруту. Ця служба не повертає детальну інформацію про маршрут. Інформацію про маршрут, включно з ламаними лініями та текстовими вказівками, можна отримати, передавши єдиний потрібний вихідний пункт і пункт призначення службі напрямків Google.

Перш ніж використовувати службу Distance Matrix в Maps JavaScript API, слід переконатися, що API Distance Matrix увімкнено в Google Cloud Console у тому самому проекті, який ви налаштували для Maps JavaScript API. Доступ до сервісу Distance Matrix є асинхронним, оскільки Google Maps API потребує звернення до зовнішнього сервера. З цієї причини слід передати метод *зворотного виклику* для виконання після завершення запиту, щоб обробити результати.

```
var distanceMatrixList = new List<long[]>();
var distanceMatrix = new long[,] { };

string origins = string.Empty;

//Add depot
addresses.Add(new AddressDataInput()
{
    Id = depot.Id,
    Latitude = depot.Latitude,
    Longitude = depot.Longitude,
    Address = depot.Address,
    Demand = 0,
    IsDepot = true
});

origins += $"{depot.Latitude},{depot.Longitude}";

var amountToCollect = default(long);

foreach (var depotData in depotDatas)
{
    addresses.Add(new AddressDataInput
    {
        Id = depotData.Id,
        Latitude = depotData.Latitude,
        Longitude = depotData.Longitude,
        Address = depotData.Address,
        Demand = depotData.Amount,
        IsDepot = false
    });

    amountToCollect += depotData.Amount;

    demands.Add(depotData.Amount);

    origins += $"|{depotData.Latitude},{depotData.Longitude}";
}

using var httpClient = new HttpClient
{
    BaseAddress = new Uri($"https://maps.googleapis.com")
};
```

Рис. 2.11. Фрагмент вихідного коду доступу до сервісу *Distance Matrix*

У результаті отримується доступ до сервісу Distance Matrix у своєму коді через об’єкт конструктора Google.maps.DistanceMatrixService. Метод

DistanceMatrixService.getDistanceMatrix() ініціює запит до сервісу Distance Matrix, передаючи йому до об’єкта DistanceMatrixRequest, що містить початкові пункти, пункти призначення та режим виконання маршруту, а також метод зворотного виклику для виконання після отримання відповіді.

Успішний виклик сервісу Distance Matrix повертає DistanceMatrixResponseоб’єкт і DistanceMatrixStatusоб’єкт. Вони передаються функції зворотного виклику, яку можна вказали в запиті.

Об’єкт DistanceMatrixResponse містить інформацію про відстань і тривалість руху автомобіля для кожної пари населених пунктів (відправлення/призначення), для яких потрібно розрахувати маршрут.

```
var response = await httpClient.GetAsync($"/maps/api/distancematrix/json?origins={origins}&destinations={origins}&&key=AIzaSyDcq5W9Lh1vDdTg54dFfbb9MpNbyjwVtbg");

var matrixResponse = new DistanceMatrixResponse();

if (response.IsSuccessStatusCode)
{
    var content = await response.Content.ReadAsStringAsync();
    matrixResponse = JsonConvert.DeserializeObject<DistanceMatrixResponse>(content);
}

foreach (var row in matrixResponse.Rows)
{
    var tempLits = new List<long>();
    foreach (var element in row.Elements)
    {
        tempLits.Add(element.Distance.Value);
    }
    distanceMatrixList.Add(tempLits.ToArray());
    tempLits = null;
}

distanceMatrix = CreateRectangularArray(distanceMatrixList);
```

Рис. 2.12. Фрагмент вихідного коду доступу до сервісу *DistanceMatrixResponse*

Відповідь матриці відстані містить код стану для відповіді в цілому, а також статус для кожного елемента. Об’єкт DistanceMatrixResponse містить по одному row для кожного джерела, переданого в запиті. Кожен рядок містить поле element для кожного сполучення цього джерела з наданим пунктом призначення.

2.3.3. Використання сервісу Google Maps Static API

Maps Static API повертає зображення (GIF, PNG або JPEG) у відповідь на HTTP-запит через URL-адресу. Для кожного запиту можна вказати

розташування карти, розмір зображення, рівень масштабування, тип карти та розміщення додаткових маркерів у місцях на карті. Можна додатково позначати маркери за допомогою буквено-цифрових символів.

Зображення Maps Static API вбудовано в атрибут <img> тегу src або його еквівалент в інших мовах програмування.

Цей сервіс забезпечує представлення необхідного формату URL-адрес статичного API Карт і доступні параметри. Він призначений для розробників веб-сайтів і мобільних пристроїв, які хочуть включити зображення Maps Static API на веб-сторінку або мобільну програму.

Перш ніж розпочати розробку за допомогою API Static Maps, слід вивчити вимоги до автентифікації (потрібен ключ API), а також мати інформацію про використання API та платіжну інформацію (потрібно ввімкнути виставлення рахунків у вашому проекті).

URL-адреса API Static Maps має такий вигляд:

```
https://maps.googleapis.com/maps/api/staticmap?parameters
```

Якщо доступ до проектованого веб-сайту (інформаційної системи планування заготівлі молока) здійснюється через HTTPS, то слід завантажувати зображення Maps Static API через HTTPS, щоб уникнути сповіщень безпеки браузера. HTTPS також рекомендується, якщо запити містять конфіденційну інформацію користувача, наприклад місце знаходження господарства виробника молока:

```
https://maps.googleapis.com/maps/api/staticmap?parameters
```

Незалежно від того, чи використовується HTTP чи HTTPS, певні параметри URL є обов’язковими, а деякі необов’язкові. Як це стандартно в URL-адресах, усі параметри розділяються символом амперсанда (&). Перелік параметрів та їх можливі значення наведено в цьому документі.

Maps Static API відносно простий у використанні, оскільки він складається лише з параметризованої URL-адреси. Maps Static API повинен мати можливість точно визначати розташування на карті, як для фокусування

карти на правильному місці (за допомогою параметра center), так і/або для розміщення будь-яких додаткових позначок місця (за допомогою параметра markers) у місцях на карті. Maps Static API використовує числа (значення широти та довготи) або рядки (адреси) для визначення цих місць. Ці значення ідентифікують *геокодоване* розташування.

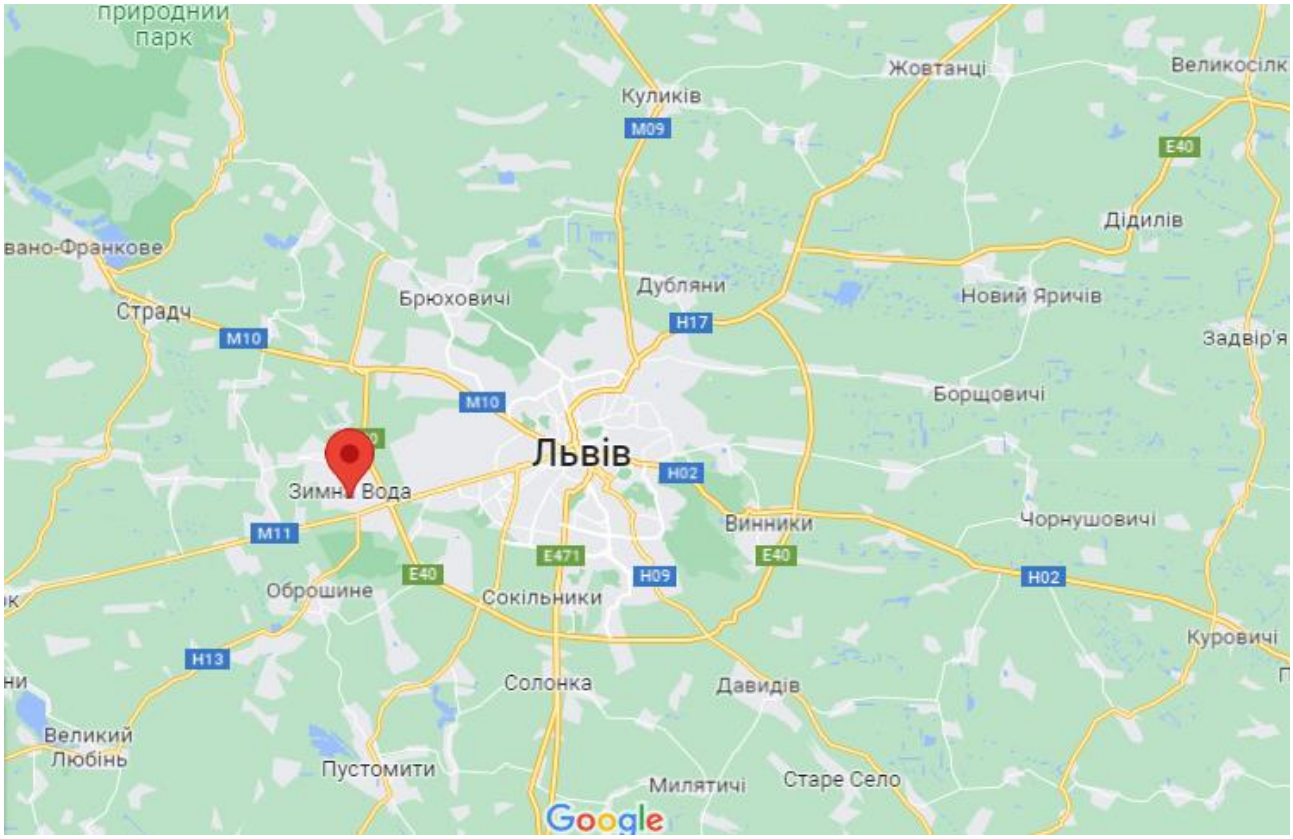

Рис. 2.13. Фрагмент використання сервісу Google Maps Static API

Кілька параметрів (таких як параметри markersi path) мають кілька місць. У таких випадках розташування розділяються вертикальною рискою (|).

Широта та довгота визначаються за допомогою цифр у текстовому рядку, розділеному комами, з точністю до 6 знаків після коми. Наприклад, "49.83547948091561, 23.888801557168563" є дійсним значенням геокоду. Точність понад 6 знаків після коми ігнорується (рис. 2.13).

Значення широти та довготи мають відповідати дійсному положенню на поверхні землі. Широта може приймати будь-яке значення від -90 до 90 тоді як довгота може приймати будь-яке значення від -180 до 180. Якщо вказати

недійсне значення широти чи довготи, то запит буде відхилено як неправильний.

РОЗДІЛ 3.
РЕЗУЛЬТАТИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАНУВАННЯ ЗАГОТІВЛІ МОЛОКА

3.1. Архітектура інформаційної системи планування заготівлі молока

Як архітектуру для інформаційної системи планування заготівлі молока, ми обрали мультиоренду (Multi-tenant) та SaaS рішення. Мультиоренда – це форма хмарної архітектури, де кілька клієнтів одного постачальника хмари спільно використовують ті самі обчислювальні ресурси. Кожен клієнт відомий як орендар. Ця форма спільного використання стосується спільного використання ресурсів програмного забезпечення, а також спільного розміщення на серверах.

Мультиоренда дозволяє кільком примірникам певної програми працювати в спільному середовищі. Таким чином, один екземпляр програмного забезпечення працює на сервері та вміщує численні орендарі. Орендарі інтегруються фізично, але вони розділені логічно. З цією конфігурацією програмна програма, що знаходиться в архітектурі з кількома клієнтами, спільно використовуватиме виділені екземпляри конфігурацій, даних, керування користувачами, серед інших властивостей.

Орендарі мають певну міру налаштування для спільного ресурсу, як-от керування тим, які користувачі можуть отримати доступ до ресурсів або як виглядає та працює програма. Однак вони не можуть налаштувати код.

Це модель публічної хмари – одна інфраструктура, багато серверів, причому кожен сервер має ексклюзивне використання та доступ до відповідних даних та інших хмарних ресурсів.

Багатоквартирність буває трьох ступенів:

- ✓ Низький: IaaS і PaaS є мультитенантними; SaaS є одним орендарем.

✓ Середній: IaaS і PaaS є мультитенантними; невеликі кластери SaaS є мультитенантними.

✓ Високий: IaaS, PaaS і SaaS повністю мультитенантні.

Архітектура з одним орендарем, як не дивно, протилежна архітектурі з кількома клієнтами. Назва «одноосібний орендар» є абсолютно недоречною щодо своєї природи. Єдина оренда означає, що програмне забезпечення та допоміжна інфраструктура обслуговують лише одного клієнта. Кожен клієнт має свою окрему незалежну базу даних або програмне забезпечення, і ніхто не ділиться ними.

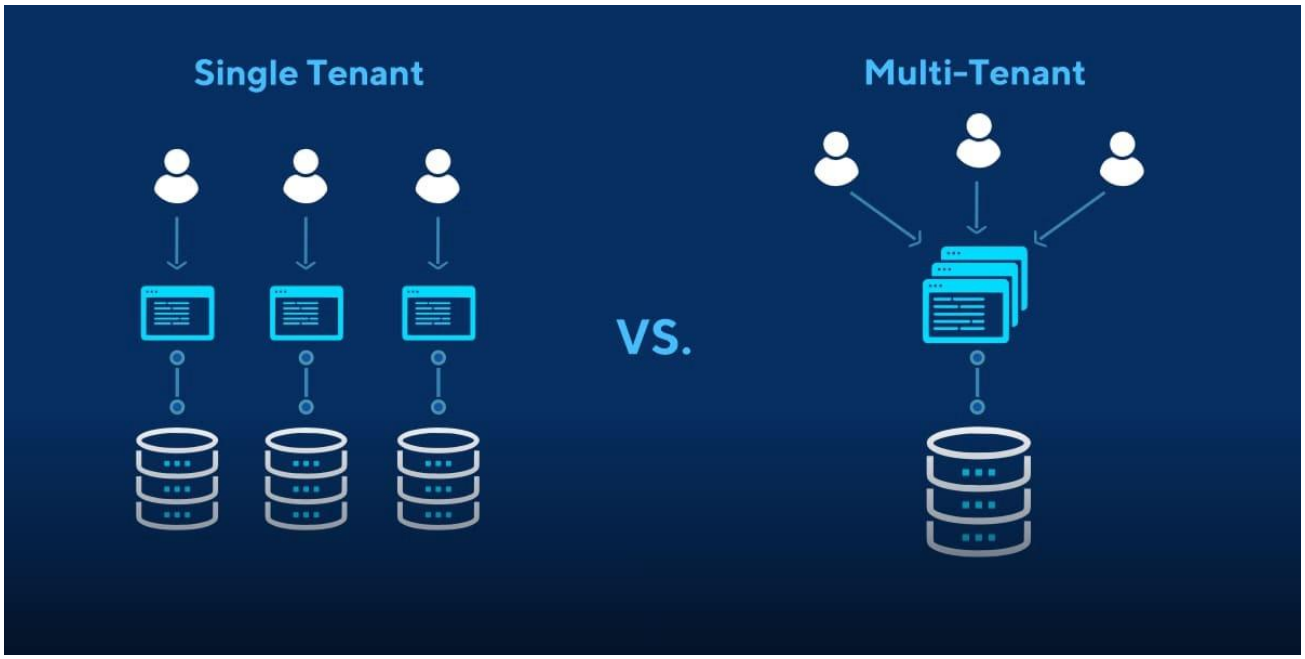

Рис. 3.1. Порівняння мультиорендної та одноорендної архітектур

Модель одного орендаря має свої переваги. Одна оренда пропонує для початківців більш безпечне середовище, оскільки ресурси кожного орендаря повністю відокремлені від ресурсів та інформації інших орендарів. Крім того, клієнт має повне налаштування та контроль функціональності. Нарешті, клієнти-одноорендарі користуються більшою надійністю, оскільки ресурси завжди доступні та в достатку.

Однак є недоліки одноразової оренди. Відсутність спільного використання ресурсів також означає відсутність розподілу витрат, тому одна оренда зазвичай дорожча. Крім того, одиночні орендарі не мають з ким

розділити регулярні обов’язки з обслуговування та налаштування, тому роботи доводиться більше.

Таким чином, єдина оренда пропонує більшу конфіденційність і доступність ресурсів, ніж багатоквартирна. Однак останнє забирає у клієнта менше часу, ресурсів і грошей.

Наведемо переваги мультиорендної архітектури. Насамперед, це економічно вигідно. Подібно до того, як обмінюватися поїздкою з іншими людьми для вас дешевше, обмінюватися хмарними ресурсами дешевше. Клієнти зрештою платять лише за те, що їм потрібно, і нічого більше. Робота та персонал, підключення нових орендарів, технічне обслуговування та розробка, а також оновлення – усім цим займається хмарний хост, лише частина витрат розподіляється між орендарями.

Спираючись на перевагу економічності, наступна перевага означає, що клієнти можуть додавати або видаляти ресурси за потреби. Ця гнучкість ідеальна для організацій, які розвиваються швидко, але непередбачувано.

Наступна перевага мультиорендної архітектури вказує на безпечне та конфіденційне її використання. Одна оренда є більш безпечною, мультиоренда, тим не менш, добре допомагає виявити загрози та зберігати ресурси орендарів окремо один від одного.

Також вона забезпечує краще використання ресурсів. З боку хоста мультиоренда дозволяє краще використовувати інфраструктуру. Доцільніше відкрити доступ до сервера багатьом клієнтам, а не обмежувати його одним клієнтом.

Мультиоренда не потребує обслуговування для клієнта. Клієнти не тільки не повинні оплачувати витрати на моніторинг і адміністрування, але й докладати зусиль. Хост виконує такі завдання, як оновлення та оновлення, технічне обслуговування та інші пов’язані завдання. Розглянемо таку модель, як оренда квартири – орендодавець займається ремонтом, орендар сплачує орендну плату.

Отже, для інформаційної системи планування заготівлі молока нами обрано мультиорендну її архітектуру.

3.2. Діаграма варіантів використання інформаційної системи

Для створення інформаційної системи планування заготівлі молока використовують методи дослідження, які забезпечують обґрунтування основних етапів життєвого циклу розробки програмного забезпечення. Розробка програмного забезпечення для реалізації інформаційної системи планування заготівлі молока використовує метод життєвого циклу розробки системи (SDLC), який у розробці системи eXtreme Programming має чотири етапи процесу, а саме:

1. ***Планування.*** Планування або розробка програмного забезпечення, яке буде створено для досягнення цілей, які були визначені за допомогою діяльності зі збору потреб у даних, які були зібрані шляхом спостереження та інтерв'ю з дослідниками.

2. ***Проектування***. Проектування та моделювання системи з використанням мови уніфікованої моделі після отримання висновків на основі даних та аналізу.

3. ***Кодування.*** Передбачає писання програмного коду, використовуючи різні мови програмування, і обов'язково передбачається виконання зв'язку його з базою даних.

4. ***Тестування.*** Проводиться тестування створеної програми та перевіряється працездатність наявних у програмі функцій.

Система проектування UML (Уніфікована мова моделювання) – це стандартна мова, яка широко використовується у світовій індустрії для визначення вимог, створення аналітиків і проектування, а також опису архітектури в об'єктно-орієнтованому програмуванні. Дизайн цієї програми використовує UML (уніфікована мова моделювання), яка складається з

розробки діаграм варіантів використання, розробки діаграми діяльності та послідовності проектування діаграм.

У запропонованій системі планування заготівлі молока є три ролі для користувачів, кожна з яких має свої можливості – Host Admin, Admin та User (рис. 3.2).

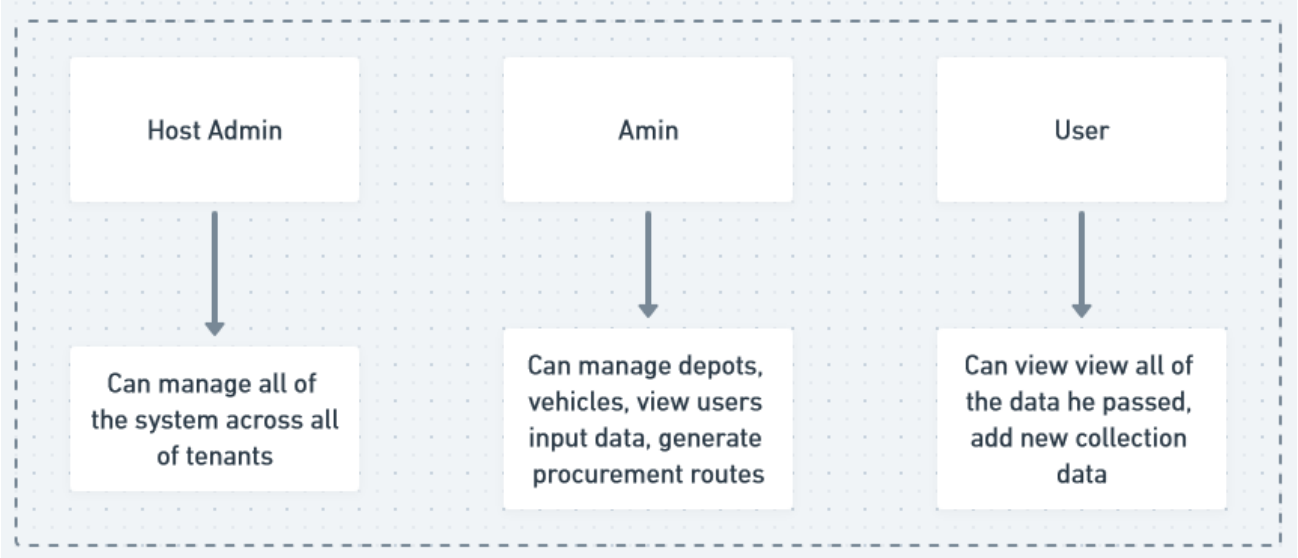

Рис. 3.2. Ключові ролі у інформаційній системі планування заготівлі молока

1. Host Admin – роль головного адміністратора системи. Дана роль є найбільш привілейованою, так як головний адміністратор має доступ до усього функціоналу системи, може керувати усіма користувачами, а у випадку потреби має змогу ввійти за конкретного користувача, та вирішити його задачу.

2. Admin – дана роль призначена для адміністратора громади, на території якої здійснюється заготівлі молока. Він буде отримувати дані від користувачів, про обсяг заготівлі молока, матиме змогу керувати усіма користувачами в системі, керувати кількістю пунктів заготівлі молока, а також транспортними засобами, які забезпечують звезення молока від господарств до пенкту первинної обробки (заготівлі). Окрім того, найголовніше що він виконує – визначати раціональні маршрути заготівлі молока на основі отриманих реальних даних.

3. User – це роль користувача, який буде створений адміністратором громади, та матиме змогу заповнювати дані про обсяг молока, яке продає заготівельникам.

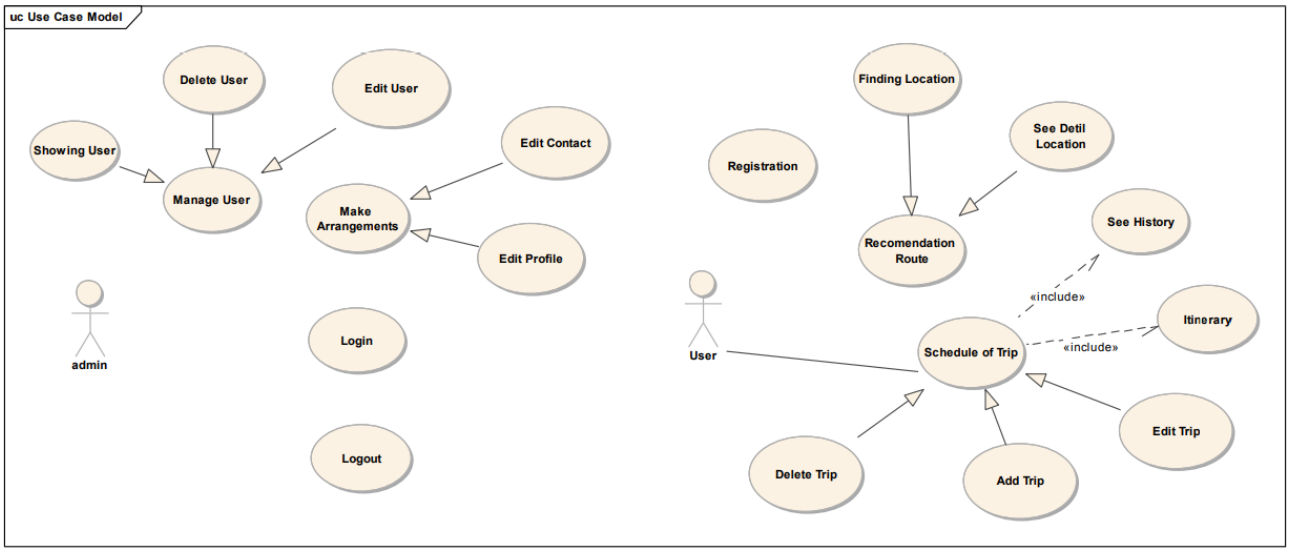

Рис. 3.3. Діаграма варіантів використання

В майбутньому кількість функціональності для ролей буде тільки розширюватись, буде добавлятись нова функціональність як від громади, так і за потреб користувачів.

Діаграма варіантів використання для інформаційної системи планування заготівлі молока має дві групи учасників: користувач і адміністратор (Host Admin та Admin).

3.3. Діаграма діяльності та активності інформаційної системи

Діаграми діяльності ілюструють різноманітні потоки дій у створюваній інформаційній системі планування заготівлі молока. Зокрема, у ній відображається те, як починається кожен потік інформації, які можуть бути прийняті рішення, і як закінчується потік інформації.

Керування користувачами має три функції, які будуть детально описані на діаграмах активності з керуванням користувачами, яке може виконувати лише адміністратор. На рис. 3.4, а показано діаграму активності, яка пояснює процес перегляду деталей користувача, а на рис. 3.4, б можна побачити дію зі зміни даних користувача.

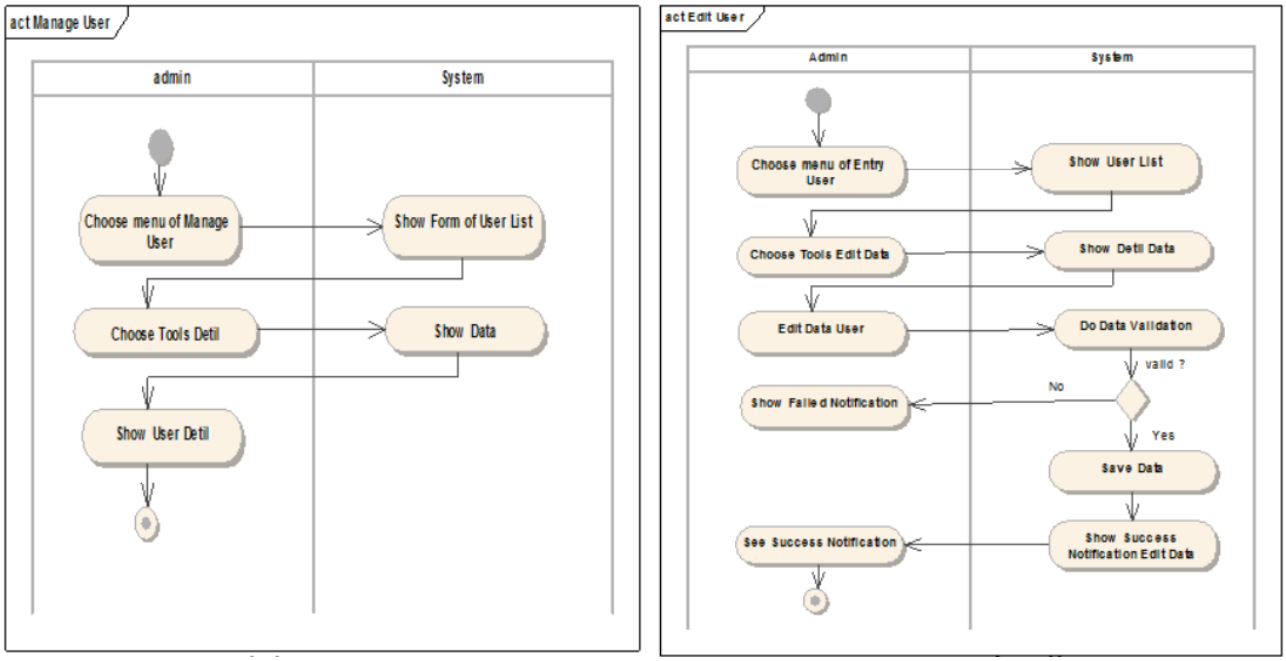

Рис. 3.4. Діаграма активності інформаційної системи

Іншою функцією керування користувачами є видалення облікових записів із програми за допомогою кроків, зображених на рис. 3.5.

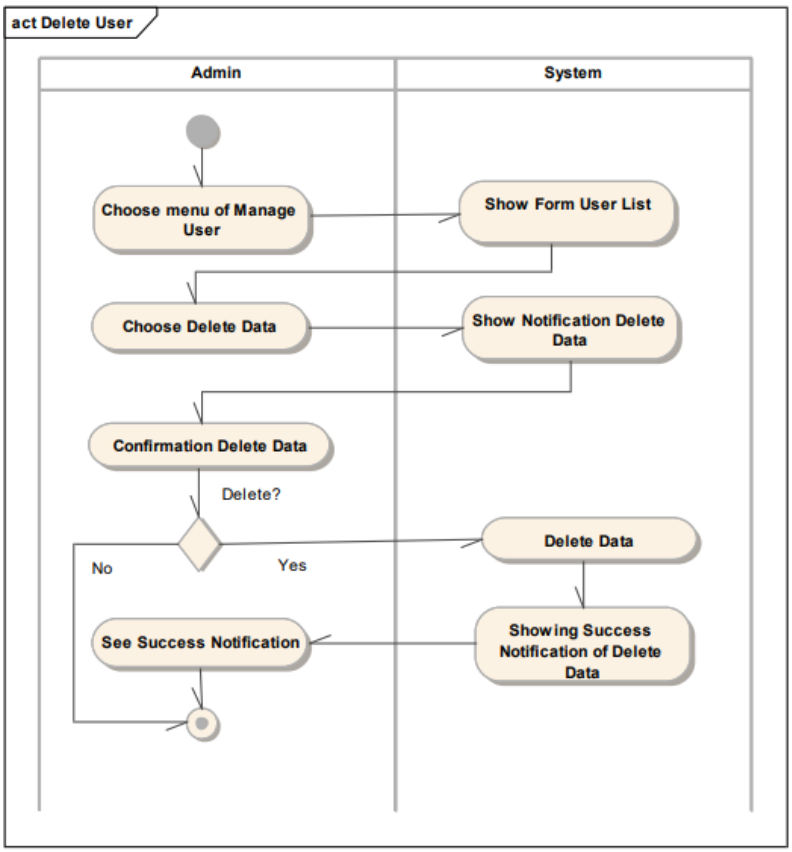

Рис. 3.5. Видалення користувачів

3.4. Діаграма послідовності інформаційної системи

Діаграма послідовності описує взаємодію між об’єктами в системі та навколо неї. На діаграмі послідовності показано серію кроків, які виконуються системою у відповідь на процес для отримання певних результатів. Нами розроблено діаграму послідовності, яка забезпечує у заданій програмі пошук пунктів призначення за найкоротшим маршрутом. На цій діаграмі дії описують процес реєстрації в системі та введення заявки та підтвердження користувачів.

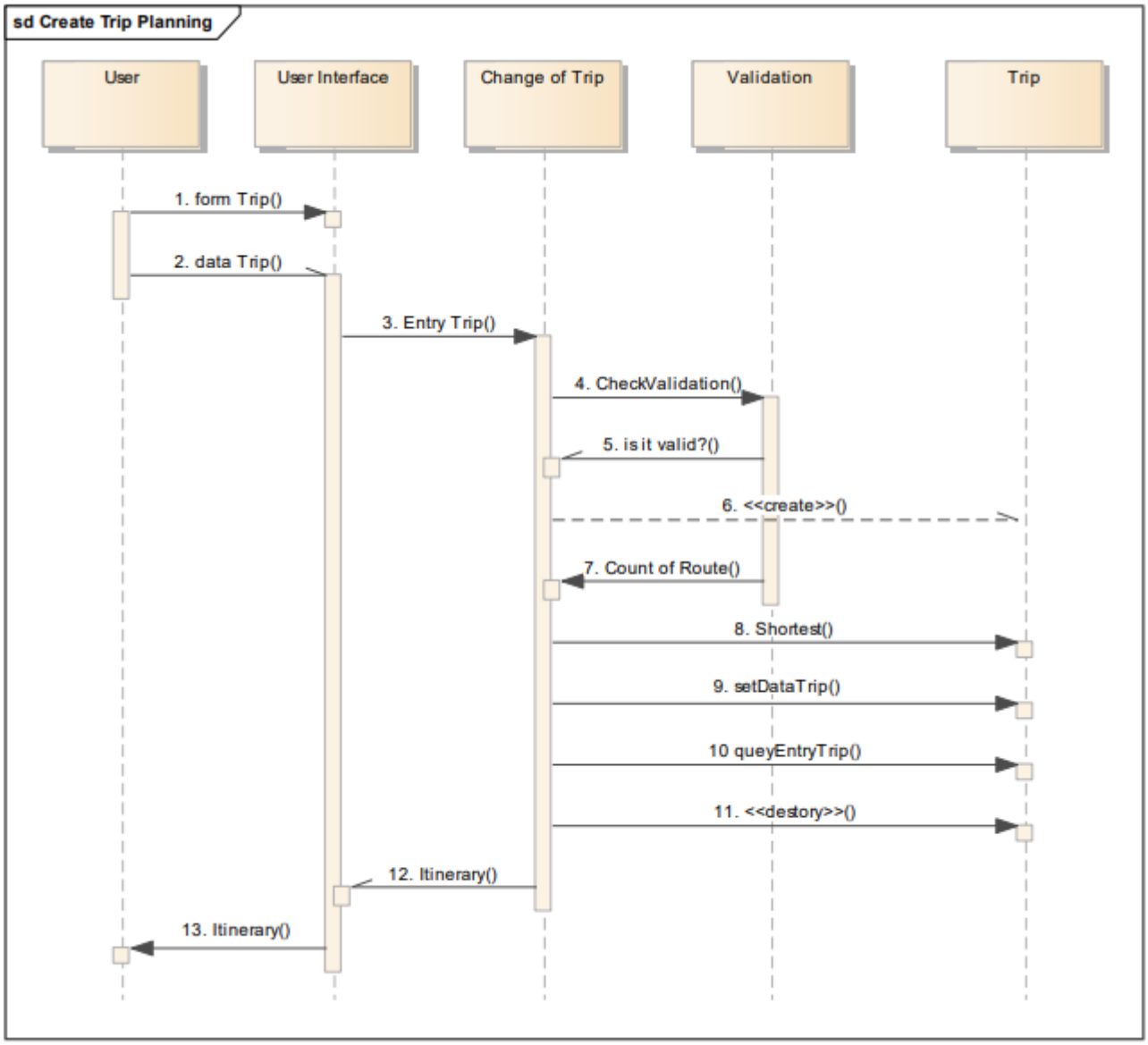

Рис. 3.6. Діаграма послідовності інформаційної системи під час формування маршруту заготівлі молока

Модуль керування користувачами має функції, доступ до яких має лише адміністратор, зокрема внесення змін до даних і статусу користувача та видалення користувачів із програми.

Нами виконано огляд процесу перегляду детальних даних, зміни даних і перегляду статусу користувача. Під час видалення користувача буде видалено лише обліковий запис користувача, де в системі все ще зберігатимуться сформовані дані про господарства виробники молока.

У системі планування заготівлі молока формування маршруту є основною функцією програми, яка дає користувачам рекомендації щодо оптимального порядку об’їзду господарств виробників молока за обґрунтованим маршрутом. Цей модуль має три функції, а саме створення нового маршруту, зміна планів заготівлі молока та видалення планів заготівлі молока.

На рис. 3.6 представлено діаграму послідовності інформаційної системи під час формування маршруту заготівлі молока.

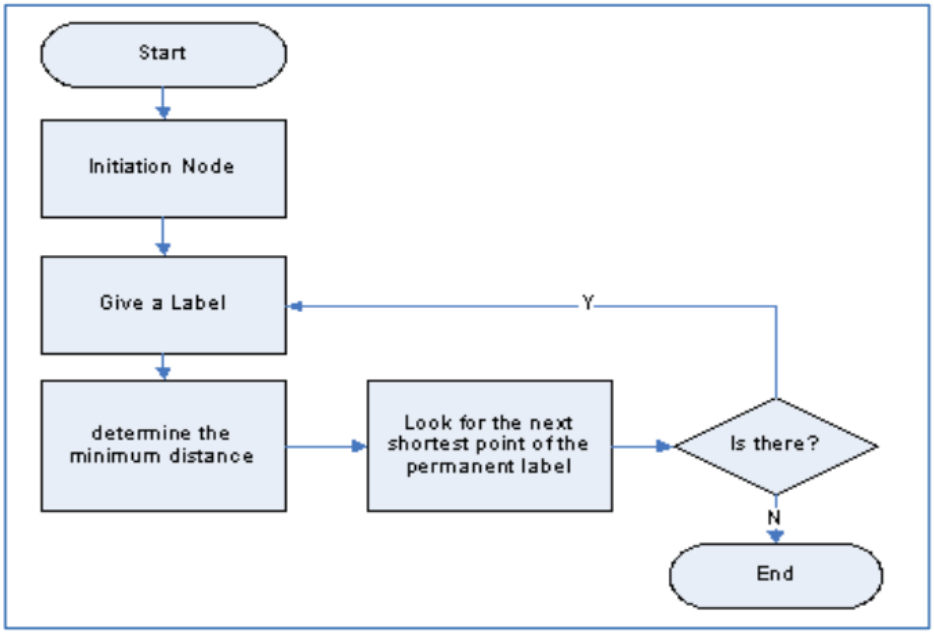

Рис. 3.7. Алгоритм, який знаходить найкоротший маршрут у заданій мережі господарств виробників молока та заданих виробничих умовах

Нами представлено алгоритм, який знаходить найкоротший маршрут у заданій мережі господарств виробників молока та заданих виробничих умовах (рис. 3.7).

3.5. Архітектура back-end інформаційної системи

Для побудови архітектури back-end API було обрано архітектурне рішення DDD (Domain Driven Design). Це є проектування інформаційної системи, орієнтоване на предметну область. Такий підхід до проектування та розробки програмного забезпечення, який спочатку базується на вимогах бізнесу. Компоненти програми (об'єкти, класи, масиви тощо) вказують на галузь, сектор або область, у якій працює бізнес.

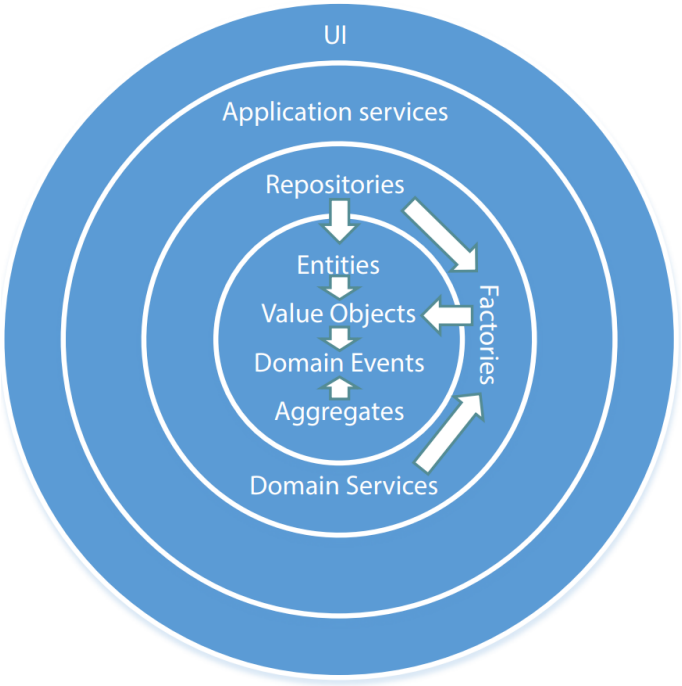

Рис. 3.8. Графічне зображення архітектури DDD

Проектування, кероване доменом (скорочено DDD) – це метод проектування програмного забезпечення, яке створює програмне забезпечення відповідно до вхідних даних домену, вставлених експертами домену. Домен відноситься до характеристик проблеми, яку має вирішити запропоноване програмне забезпечення.

Це означає, що програмне забезпечення буде спеціально розроблено відповідно до характеристик проблеми для її вирішення. Щоб цього досягти, структура та мова використовуваного програмного коду мають відповідати бізнес-сфері. Цей метод також використовує різні принципи та моделі, щоб

усунути розрив між бізнес-реальністю та програмним кодом. Дизайн, орієнтований на домен, є найбільш корисним у вирішенні складності домену, оскільки він працює для збереження основного фокусу проекту як основного домену.

Цей термін відноситься до частини домену, яка реалізує реальні бізнес-правила. Тобто логіка предметної області визначає, як певні дані мають зберігатися, одержувати доступ, маніпулювати або створюватися відповідно до заявлених бізнес-правил.

Одним із основних бізнес-правил є те, що бізнес повинен вітати кожного нового клієнта/відвідувача. Логіка бізнесу/домену диктує маршрути та методи, які різні бізнес-об’єкти використовуватимуть для реалізації цього правила.

3.6. Програмна реалізація Front-end

Програмним засобом для частини візуалізації даних була обрана платформа Angular. Angular – це відкрита структура та платформа для створення односторінкових додатків, написаних на TypeScript, які підтримуються та розробляються Google. Спочатку Angular мав бути версією 2 популярного фреймворку AngularJS. Тим не менш, дизайнерські рішення змусили Google випустити його як окрему сутність, включаючи відсутність зворотної сумісності та простий шлях оновлення програм, написаних на AngularJS, до Angular 2. Angular випущено за ліцензією MIT. Angular має очевидні переваги як фреймворк, а також забезпечує стандартну структуру для роботи розробників. Це дозволяє користувачам створювати великі додатки зручним для обслуговування способом.

Слід зауважити, що Angular є кращим вибором для нашого проекту розробки інформаційної системи планування заготівлі молока. Однією з найбільших переваг Angular є те, що він підтримується Google. Google пропонує свою довгострокову підтримку (LTS) для Angular, яка проливає

світло на план Google щодо дотримання фреймворку та подальшого масштабування екосистеми Angular.

Програми Google також використовують Angular, і їх команда досить оптимістично налаштована щодо його стабільності. Інші розробники Angular також отримують чудову можливість навчатися у сертифікованих спеціалістів Angular від Google.

Додатки Angular створюються з використанням мови TypeScript, верхнього індексу для JavaScript, який забезпечує більш високий рівень безпеки, оскільки підтримує типи (примітиви та інтерфейси). Це допомагає виявляти та усувати помилки на ранніх стадіях процесу під час написання коду або виконання завдань обслуговування.

На відміну від CoffeeScript або Dart, TypeScript не є окремою мовою. За допомогою TypeScript ви можете легко взяти наявний код ES5 або ES2015+ JS, і він скомпілює його на основі того, що ви налаштовуєте. Він повністю підтримує основні функції ES2015 і ES2016/ES2017, такі як декоратори або async/await.

Існує можливість також безпосередньо налагоджувати код TypeScript у браузері чи редакторі, якщо наявні відповідні файли карт, створені під час збирання. Ця мова забезпечує покращену навігацію, рефакторинг і автодоповнення. Ви навіть можете відмовитися від його вбудованих функцій, коли це необхідно.

Angular використовує HTML для визначення інтерфейсу користувача програми. HTML, порівняно з JavaScript, є менш заплутаною мовою. Це також декларативна та інтуїтивно зрозуміла мова з такими директивами, як ng-app, ng-model, ng-repeat і forms control.

З його допомогою вам не потрібно витрачати час на програмні потоки та вирішувати, що завантажується першим. Просто слід визначити, що у проекті потрібно, і Angular подбає про це.

Із використанням Angular не потрібні додаткові функції отримання та встановлення. Це пояснюється тим, що кожен об’єкт, який використовує

Angular, є POJO (Plain Old JavaScript Object), який дозволяє маніпулювати об’єктами, надаючи всі звичайні функції JavaScript. Ви можете видаляти або додавати властивості до об’єктів, а також зациклювати ці об’єкти, коли потрібно.

Angular Progressive Web Application (PWA) – це економічне рішення, яке дозволяє веб-сайтам працювати як мобільні програми. Це зменшує залежність від мережі, що значно покращує взаємодію з веб-сайтом. Кешування в PWA працює ефективно та зберігає пропускну здатність, коли це можливо. Це мінімізує ризики надання застарілого вмісту. Крім того, оскільки це веб-сайт, його можна оптимізувати для SEO.

Angular також сприяє розробці односторінкових додатків (SPA), які надають можливості рендеринга на стороні сервера, що підвищує рейтинг SEO. Це також допомагає швидко завантажити першу сторінку та підвищити продуктивність веб-сайту на мобільних і малопотужних пристроях.

Фреймворк Angular вбудовано в оригінальну архітектурну програму MVC (Model-View-Controller). Однак це не за встановленими стандартами. Angular не просить розробників розділити додаток на різні компоненти MVC і створити код, який міг би їх об’єднати. Швидше, він лише просить розділити додаток і піклується про все інше. Отже, структури дизайну Angular і MVVM (Model-View-View-Model) досить схожі.

Angular забезпечує легку розробку, оскільки усуває потребу в непотрібному коді. Він має спрощену архітектуру MVC, що робить непотрібним написання геттерів і сеттерів. Директивами може керувати інша команда, оскільки вони не є частиною коду програми. Загалом, розробникам обіцяють менше кодування, а також легші та швидші програми.

Angular організовує код у місткості, будь то компоненти, директиви, канали чи служби. Ті, хто знайомий з Angular, називають ці місткості модулями. Модулі спрощують організацію функціональних можливостей програми, розділяючи її на функції та блоки, які можна багаторазово використовувати. Модулі також дозволяють відкладене завантаження, що

відкриває шлях для завантаження функцій програми у фоновому режимі або на вимогу.

Angular робить досяжною метою розділити роботу між різними членами команди, забезпечуючи впорядкований код. Ви можете використовувати модулі якнайкраще, якщо добре їх розумієте. Розробники також можуть підвищити продуктивність за допомогою відповідних модулів.

Кожна кодова база вимагає послідовного кодування. Це пояснюється тим, що неузгоджене кодування може збільшити ризики затримки запуску або підвищених витрат. З іншого боку, узгоджене кодування може полегшити використання розробленого web додатку та дозволити використання шаблонів або попередньо визначених фрагментів коду.

3.7. Результати розробки інтерфейсу користувачів та основних функціональних блоків

Інтерфейс користувача запропонованої інформаційної системи планування заготівлі молока повинен спочатку забезпечувати їх реєстрацію та вхід, як показано на рис. 3.9.

Рис. 3.9. Інтерфейс користувача інформаційної системи планування заготівлі молока, що забезпечує їх реєстрацію та вхід

Після реєстрації користувач входить у програму, ввівши ім’я користувача та пароль на сторінці входу, як показано на рис. 3.9.

Реалізація окремих користувачів інформаційної системи здійснюється через меню «Реєстрація», вимагаючи ввести своє ім’я, номер телефону, адресу електронної пошти та пароль для входу у програму, як показано на малюнку 3.9. Після завершення реєстрації програма надішле підтвердження на зареєстровану електронну адресу, щоб переконатися, що адреса електронної пошти правильна та новий обліковий запис готовий до використання.

Після цього виконано підключення матриць відстаней, виробничих умов та алгоритмів вибору маршрутів (рис. 3.10).

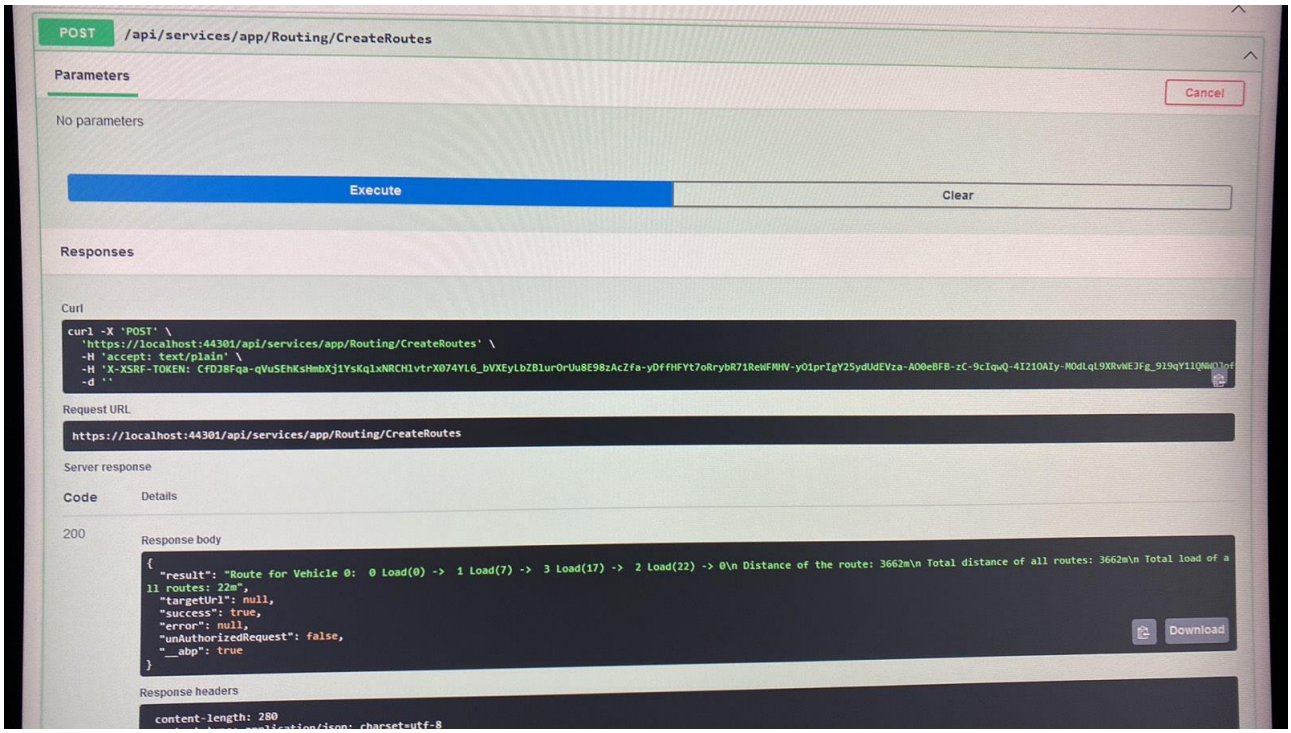

Рис. 3.10. Результати підключення матриць відстаней, виробничих умов та алгоритмів вибору маршрутів

Сторінка, яка відображається на сторінці входу адміністратора, показана на рис. 3.11. Адміністратори можуть вказати характеристики нових користувачів, якщо вони вже зареєстровані та ввійшли в додаток.

Дизайн домашньої сторінки із вибором місця розташування господарства, що виробляє молоко та внесення його точних координат із використанням Maps

Static API показано на 3.11, а користувачі можуть змінювати профілі за допомогою функції налаштувань профілю з відображенням.

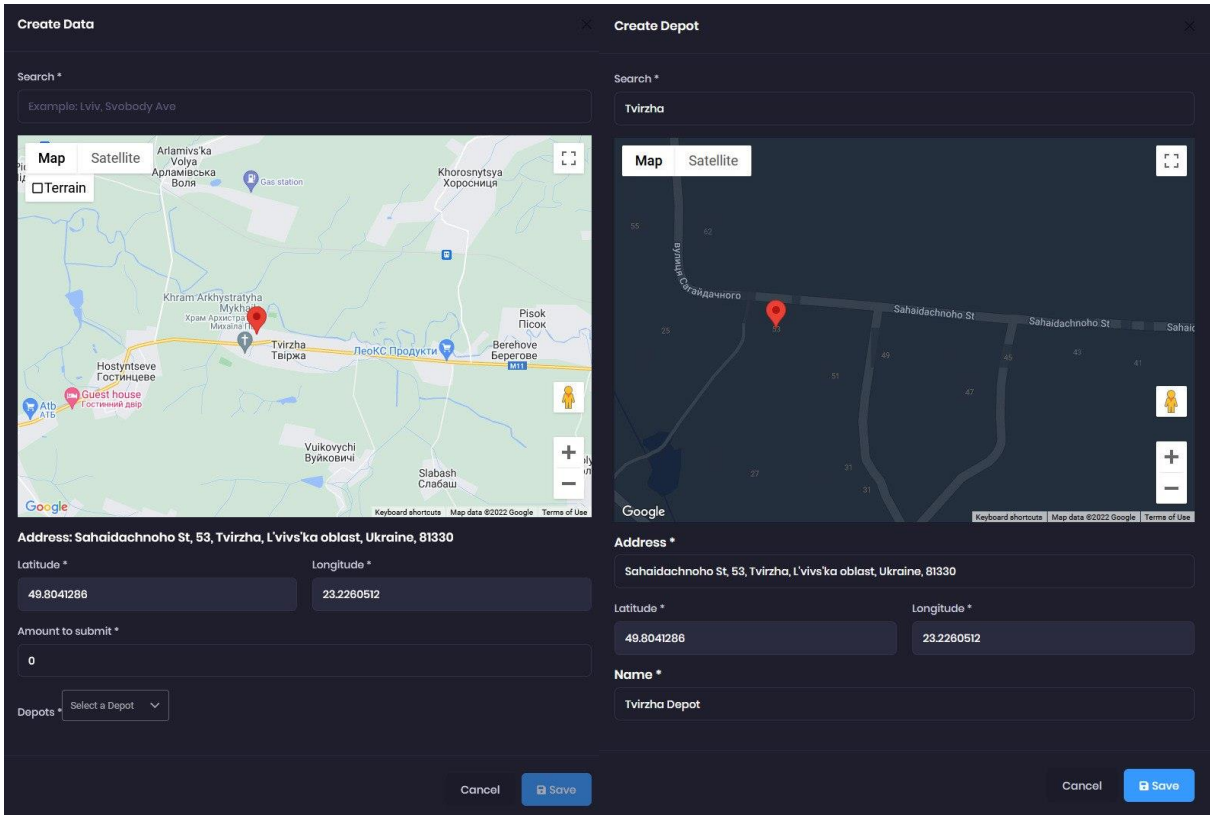

Рис. 3.11. Вибір місця розташування господарства, що виробляє молоко та внесення його точних координат із використанням Maps Static API

Окрім того, можна виконувати видалення господарств, що виробляють молоко (рис. 3.12).

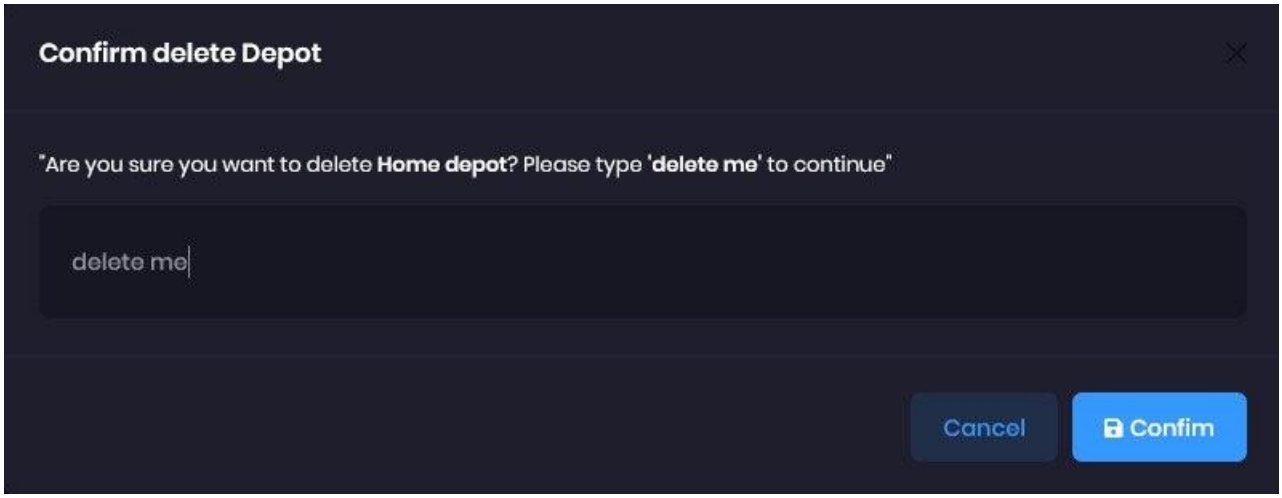

Рис. 3.12. Меню із видалення господарства, що виробляє молоко

На наступному рис. 3.13 представлено результати відображення бази даних щодо територіального розташування господарств виробників молока та пункту його збору.

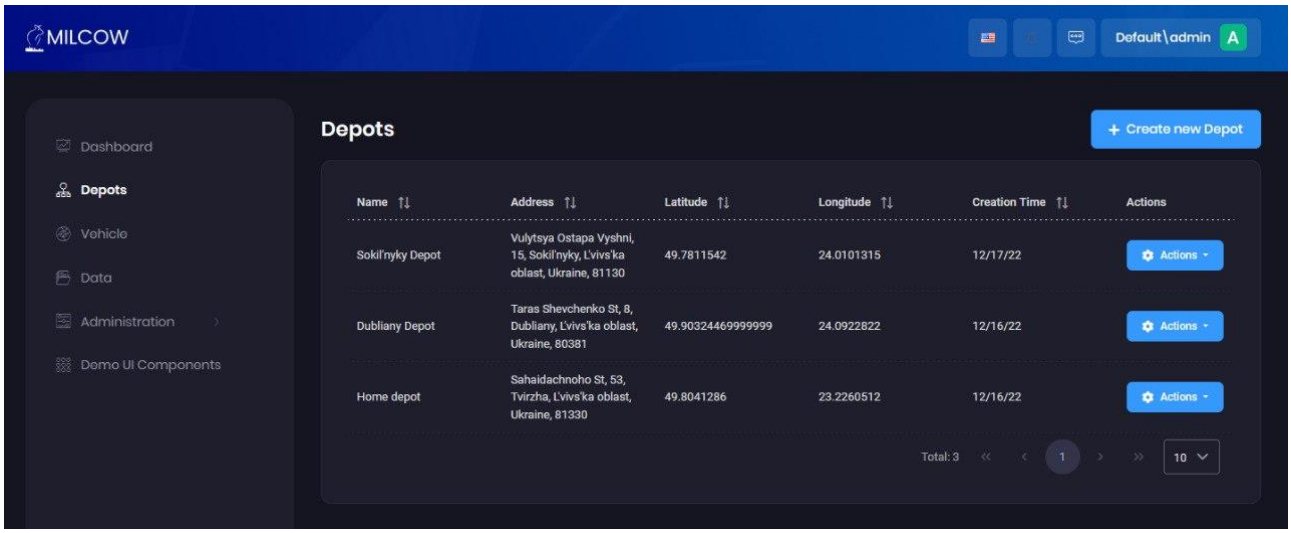

Рис. 3.13. Результати відображення бази даних щодо територіального розташування господарств виробників молока та пункту його збору

На підставі використання Maps Static API отримали можливість точно відобразити територіальне розташування на карті господарств виробників молока та пункту його збору (рис. 3.14). Для цього використано числа (значення широти та довготи) або рядки (адреси) для визначення зазначених місць. Ці значення ідентифікують геокодоване розташування.

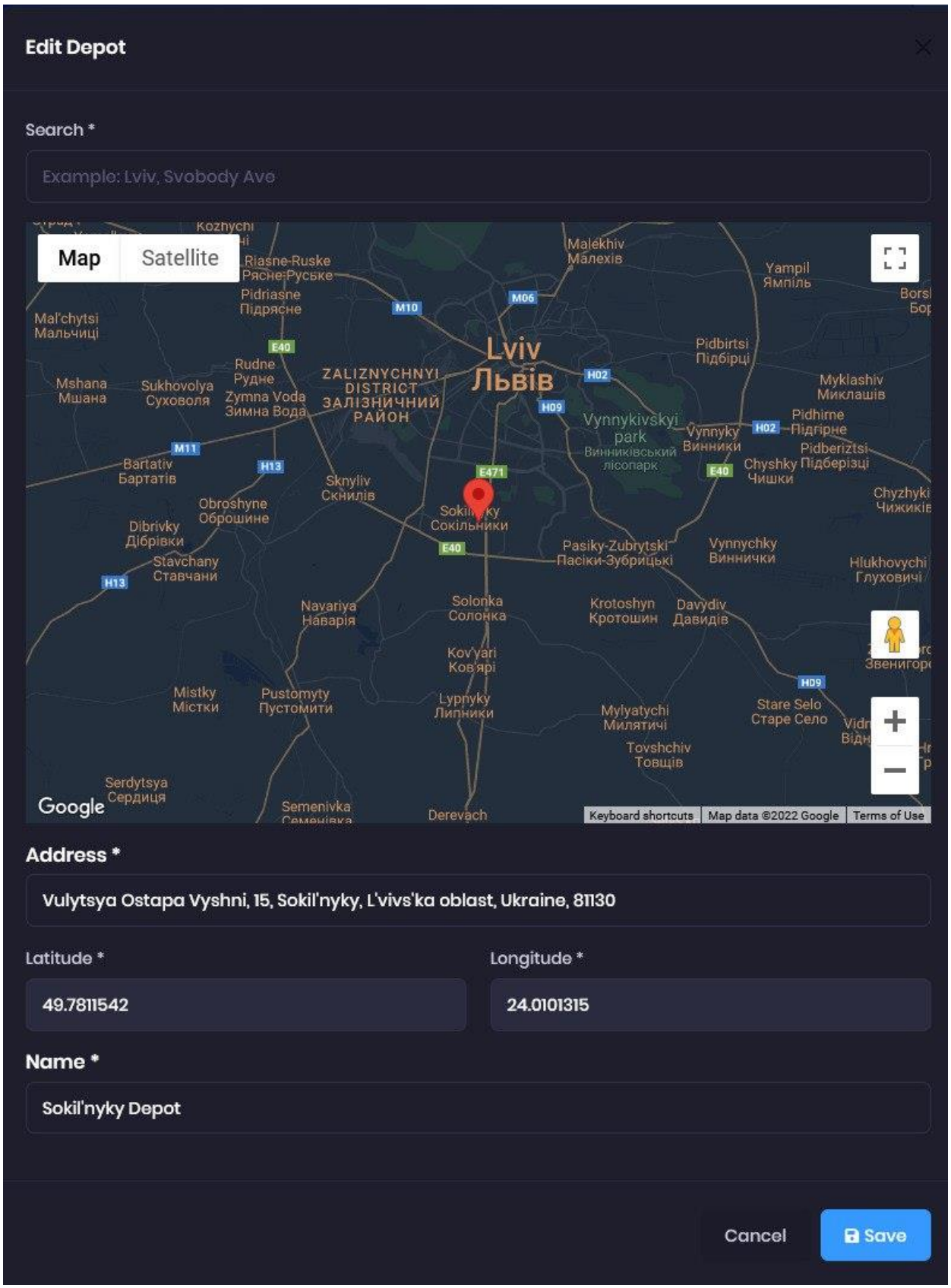

Рис. 3.14. Результати відображення територіального розташування на карті господарств виробників молока та пункту його збору

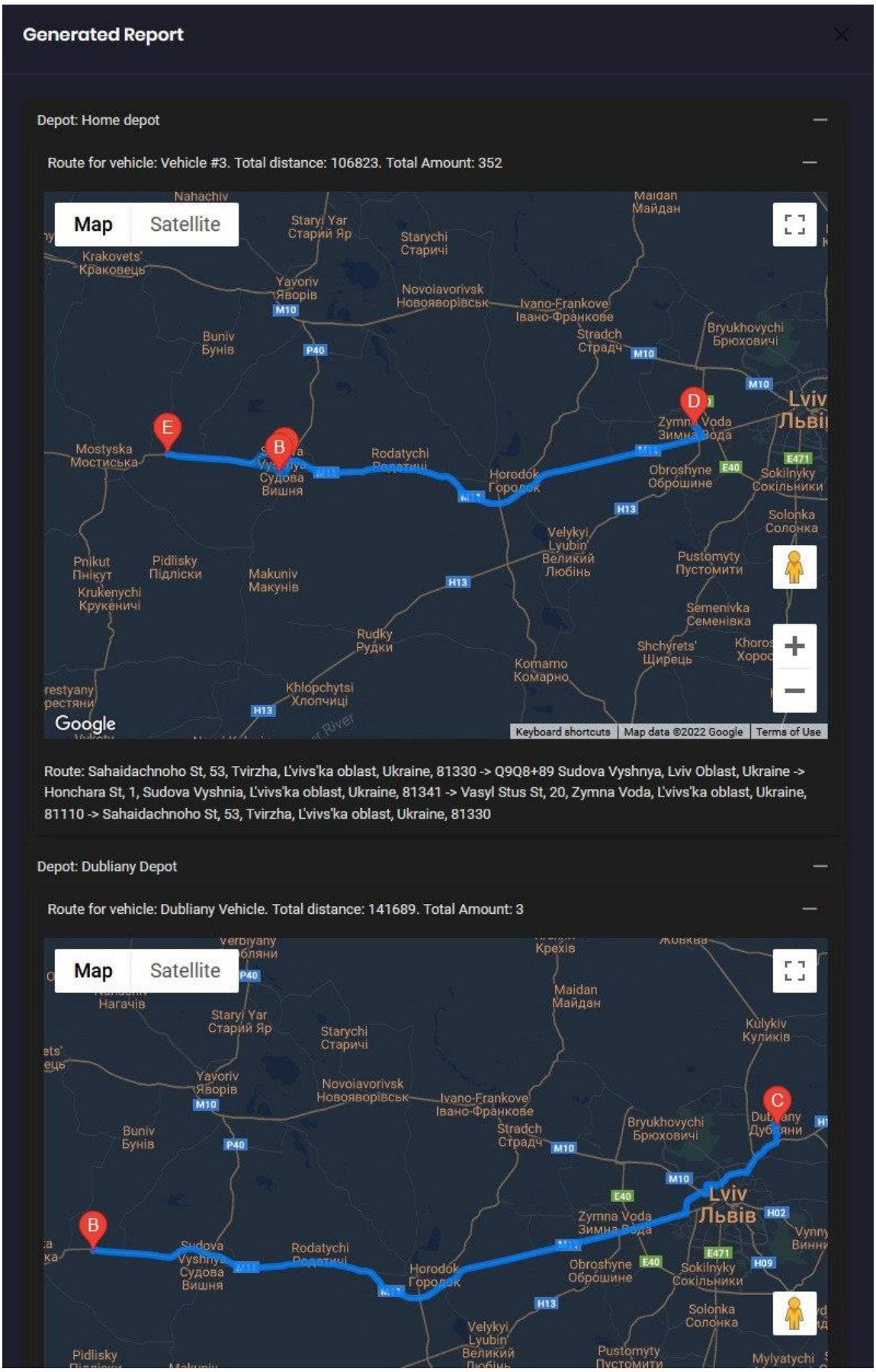

Рис. 3.15. Результати визначення раціонального маршруту заготівлі молока

На рис. 3.15 представлено дизайн сторінки із результатами визначення раціонального маршруту заготівлі молока, виконаний у три етапи для отримання результатів розрахунку ефективного маршруту. У той час як дизайн інтерфейсу сторінки маршруту є результатом розрахунку маршруту, у вигляді карти представлено господарства заготівлі молока, а також черговість їх об їзду та доставки молока до пункту первинної обробки.

У меню маршруту користувач спочатку побачить історію складеного маршруту, чи був він виконаний, або план на поточну дату із заготівлі молока, як показано на рис. 3.15. Для того, щоб створити новий маршрут можна, натиснувши кнопку у меню, а потім вибравши господаства виробники молока, задавши його обсяг у кожному із них, виконати побудову маршруту. Кожне вибране господарство виробник молока буде відображати детальну інформацію, починаючи з точки на карті, адреси тошо. Окрім того, інформація синхронізується з картами Google, як показано на рис. 3.15.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Аналіз процесу заготівлі молока та прогнозування травмонебезпечних ситуацій

Охорона праці – це система законодавчих актів, соціально-економічних, організаційних, технічних, гігієнічних і лікувально-профілактичних заходів і засобів, спрямованих на створення безпечних умов збереження здоров'я і працездатності людей в процесах праці. Складовими охорони праці є законодавство про працю, виробнича санітарія і безпека застосування різних технічних засобів у технологічних процесах транспортування с.-г. сировини, включаючи і пожежну безпеку [5].

Технологічний процес транспортування молока включає такі види технологічних фаз і операцій [12]:

- заповнення цистерни молоком;
- транспортування молока;
- злив молока з цистерни.

Можливими травмонебезпечними чинниками є:

- механічне ураження рухомими частинами машини;
- несправність органів керування, гальм;
- перевищення швидкості руху;
- відмова одного з вузлів автомобіля;
- аварійно-небезпечний стан доріг;
- недотримання правил пожежної безпеки;
- алкогольне сп'яніння.

4.2. Розробка логічно-імітаційної моделі виникнення небезпечних ситуацій

Проаналізувавши кожну із логічних моделей процесів формування та можливого виникнення травмонебезпечних та аварійних ситуацій, завжди можна знайти подію з якої починається небезпечний процес ще до виникнення небезпечних наслідків [12].

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта. Таким показником вибрана ймовірність виникнення аварії, травми залежно від досліджуваного явища [12].

Для оцінки рівня небезпеки певного об'єкта чи явища можна застосувати метод обчислення ймовірності виникнення будь-якого випадкового явища, який широко застосовують в зарубіжній інженерній практиці. Основні його принципи полягають в тому, що на основі обстеження робочого місця чи окремої машини виявляють виробничі небезпеки, можливі аварійні або травматичні ситуації. При оцінці ситуацій визначають події, які можуть стати головною подією при побудові логічно-імітаційної моделі травми. Після цього будують модель «дерева відмов і помилок оператора». При цьому важливе значення має правильний вибір головної події.

Головну подію (травма), модель якої нам необхідно побудувати, вибирають виходячи з оцінки відповідного об'єкта, виробництва чи окремої одиниці обладнання і змісту його найбільш небезпечного явища, яке за певних умов виробництва може виникнути.

Після вибору головного випадкового явища (події) розпочинаємо побудову моделі («дерева»). Використовуючи оператора «і» та «або», використовуємо набір ситуацій (відомих до цього), які можуть призвести до подій, вибраної як головна.

Після визначення відповідних травмонебезпечних ситуацій та їх кількості, визначаємо інші події, що входять до кожної такої ситуації, логічним

аналізом із застосуванням операторів «і», «або» та інших. Процес побудови моделі триває, поки не будуть знайдені усі базові події, що визначають межу моделі.

Слід мати на увазі, що кожна випадкова подія, до якої входять базові події, може формуватися й виникати при входженні у неї двох, трьох і більше базових подій за допомогою відповідних операторів.

Повністю побудована і перевірена модель підлягає математичній обробці для визначення ймовірності кожної випадкової події, що увійшла до моделі, починаючи з базових і закінчуючи головною.

Ймовірність базових подій визначаємо за даними виробництва. Наприклад, базова подія «стан контролю з охорони праці». Для визначення ймовірності ми повинні встановити, наскільки (у відсотках) від ідеального рівня здійснюється відповідний контроль на об'єкті. Якщо буде встановлено, що такий рівень контролю становить 50% або 30%, то ймовірність відповідно дорівнює 0,5 і 0,3. При відсутності контролю ймовірність «не здійснення контролю» становитиме 1, якщо контроль ідеальний, то відповідно ймовірність дорівнює 0.

Після обчислення ймовірності всіх подій, розміщених у ромбах, і базових подій, починаючи з лівої нижньої гілки «дерева», позначаємо номерами всі випадкові події, що увійшли до моделі.

На цьому можна вважати, що певна модель підготовлена до математичних обчислень ймовірностей випадкових подій логічно-імітаційної моделі

Отже, для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми для випадку технологічного процесу розробки інформаційної системи планування заготівлі молока складемо список базових подій. Вони лежатимуть у основі даної моделі. Кожному пункту списку присвоюємо певне значення ймовірності виникнення. Нижче подано сам список:

1. Стан контролю з охорони праці $P_1 = 0,2$;

2. Несерйозне відношення до проходження ТО $P_2 = 0,1$;

3. Відсутність комплектуючих $P_3 = 0,2$;

4. Невисока міцність $P_4 = 0,03$;

5. Застаріле обладнання $P_6 = 0,02$;

6. Попадання предметів з навколишнього середовища $P_7 = 0,4$;

7. Досвід роботи $P_{12} = 0,35$.

8. Професійний рівень розробника $P13 = 0,5$;

9. Психофізіологічний стан розробника $P14 = 0,083$;

На основі даного списку будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічне представлення якої зображено на рис. 4.1.

Розрахуємо ймовірності виникнення подій [12], що входять у дану логіко-імітаційну модель процесу розробки інформаційної системи планування заготівлі молока (на прикладі ймовірності травми розробника, пов'язаної з коротким замиканням електромережі).

Ймовірність виникнення події P_5 визначаємо наступним чином:

$$P_5 = 02 + 0,1 + 0,2 + 0,003 - 0,2 \cdot 0,1 - 0,2 \cdot 0,03 - 0,2 \cdot 0,03 - 0,1 \cdot 0,2 - 0,1 \cdot 0,03 - \\ -0,2 \cdot 0,03 + 0,2 \cdot 0,1 \cdot 0,2 + 0,1 \cdot 0,2 \cdot 0,03 + 0,2 \cdot 0,1 \cdot 0,2 + 0,2 \cdot 0,1 \cdot 0,03 - \\ -0,2 \cdot 0,1 \cdot 0,2 \cdot 0,03 = 0,314$$

Ймовірність виникнення події P_{10} визначаємо так:

$$P_{10} = 0,2 + 0,1 = 0,3.$$

Ймовірність виникнення події P_{11} визначаємо:

$$P_{11} = 0,02 \cdot 0,314 \cdot 0,4 \cdot 0,3 = 0,00075.$$

Ймовірність виникнення події P_{15} визначаємо наступним чином:

$$P_{15} = 0,35 \cdot 0,5 \cdot 0,083 = 0,0145.$$

Ймовірність події P_{18}:

$$P_{18} = 0,5 + 0,083 = 0,58.$$

Ймовірність події P_{19}:

$$P_{19} = 0,0145 \cdot 0,083 = 0,0012.$$

Ймовірність події P_{20}:

$$P_{20} = 0,00075 + 0,0012 = 0,00195.$$

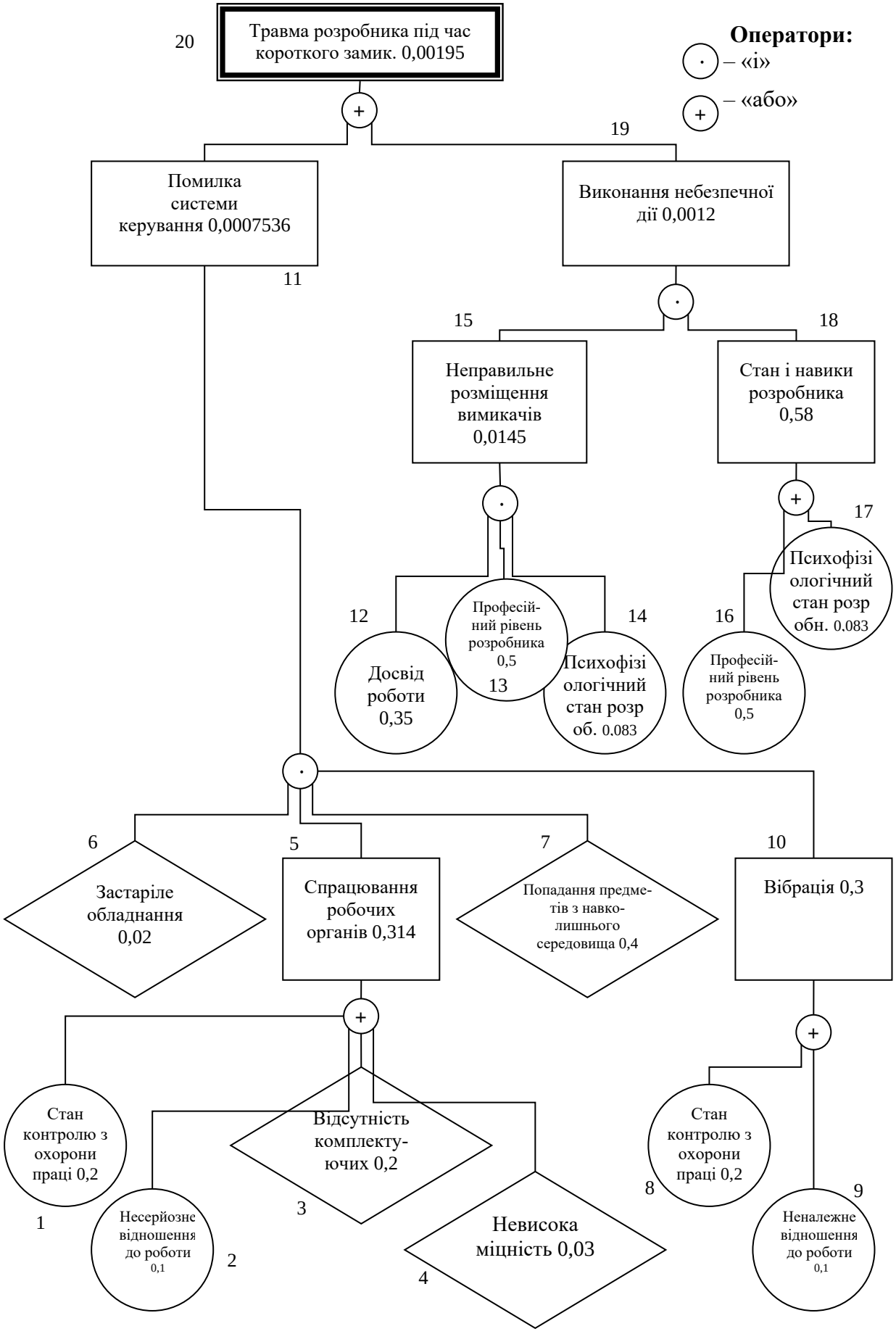

Рис. 4.1. Логіко-імітаційна модель процесу виникнення аварії та травми під час процесу розробки інформаційної системи планування заготівлі молока

Ймовірність травми рівна ймовірності виникнення аварії, бо остання можлива лише за умови виконання робіт людиною.

Логіко-імітаційні моделі аварій і травм допомагають зменшити ймовірність виникнення аварійних та травмонебезпечних ситуацій. Якщо необхідно оцінити рівень небезпеки будь-якого робочого місця, слід уважно вивчити і побудувати логічні моделі можливих небезпечних ситуацій, які охоплюють як стан обладнання і самого робочого місця, так і поведінку працюючого і обчислити ймовірність виникнення травми [12].

Після аналізу результатів моделювання ймовірність виникнення травми можна звести до дуже малої величини – достатньо зменшити вплив ймовірностей вихідних факторів, які до неї призводять.

4.3. Заходи безпеки у надзвичайних ситуаціях

Під надзвичайною ситуацією розуміють порушення нормальних умов життя і діяльності людей, об'єктів або територій унаслідок аварій, катастроф, стихійних лих або інших чинників, що спричинили або можуть спричинити загибель людей та значні матеріальні втрати.

Головною функцією адміністрації підприємства у разі виникнення надзвичайної ситуації є захист населення та організації його життєзабезпечення.

Заходи щодо захисту цивільного населення плануються проводяться по населених пунктах де розміщені господарства і охоплюють населення навколишніх сіл. Водночас характер та зміст захисних засобів встановлюються вид ступеня загрози, місцевих умов з урахуванням важливості виробництва для безпеки населення і інших економічних і соціальних чинників.

Основні заходи щодо захисту населення плануються та здійснюються завчасно і мають випереджувальний характер, це стосується насамперед підготовки, підтримання у постійній готовності індивідуальних та колективних

засобів захисту, їх накопичення, а також підготовки до проведення евакуації населення із зон підвищеного ризику.

Керівництво підприємства є безпосередніми виконавцями цих заходів, у нашому господарстві розробляються завчасно, проводиться навчання робітників та службовців способам захисту та діям в умовах надзвичайних ситуацій.

Також раз в три роки проводяться навчання по підготовці близьких до військових дій, що в разі небезпеки могло би не дістати людину зненацька. Керівництво докладає максимум зусиль, щоб працівники господарства були хоча би мінімально захищенні в разі будь-якої небезпеки пов’язаної з тими чи іншими обставинами.

РОЗДІЛ 5.
ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ВІД ВИКОРИСТАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАНУВАННЯ ЗАГОТІВЛІ МОЛОКА

Ефективність використання інформаційної системи планування заготівлі молока визначається на підставі показників їх використання менеджерами із заготівлі молока. Зокрема, це стосується витрати часу на збір інформації, передачу, обробку, зберігання та передачу користувачеві потрібної інформації. При цьому зазначені показники повинні бути мінімальними.

Особливістю розробленої інформаційної системи планування заготівлі молока є можливість вибору ефективних маршрутів за різними алгоритмами, які надає платформа Google із вище описаними сервісами.

До вартості інформаційної системи входять як технічні засоби, так і витрати на розробку програмних продуктів системи планування заготівлі молока. Основні витрати стосуються розробки програмного забезпечення інформаційної системи, придбання основних технічних компонентів та проведення монтажних робіт. На підставі проведених нами розрахунків визначено вартість програмно-технічних засобів інформаційної системи планування заготівлі молока, які становлять 48600 грн.

Розроблена інформаційна система для планування заготівлі молока повинна забезпечувати використання накладних витрат не менше $P_м = 10\%$, тоді її вартість становить:

$$Ц_м = C_п + C_п \cdot (P_м / 100). \qquad (5.1)$$

На підставі формули (5.1) визначаємо:

$$Ц_м = 48600 + 48600 \cdot (10 / 100) = 53460 \text{ грн.}$$

Для визначення балансової вартості інформаційної системи планування заготівлі молока враховуємо її монтаж та налаштування:

$$C_{бал} = Ц \cdot K_{тп}. \quad (5.2)$$

де $Ц$ – вартість компонент інформаційної системи планування заготівлі молока; $K_{тп}$ – коефіцієнт витрат на монтаж і налаштування ($K_{тп} = 1.1$).

На підставі формули (5.1) визначаємо балансову вартість інформаційної системи планування заготівлі молока із врахуванням її монтажу та налаштування:

$$C_{бал} = 53460 \cdot 1{,}1 = 58806 \text{ грн.}$$

Витрати на обслуговування інформаційної системи планування заготівлі молока становлять 38462 грн. Вартість виконання робіт щодо розробки та тестування інформаційної системи – 51200 грн.

Загальна вартість інформаційної системи планування заготівлі молока для окремого переробного підприємства або громади становить 104660 грн.

Економічна ефективність від використання інформаційної системи планування заготівлі молока для окремого переробного підприємства або громади за формулою:

$$E_{ІСП} = (П_1 - П_2) - З_{ІСП}, \quad (5.3)$$

де $П_1$ – обсяг втрат переробного підприємства або громади із інформаційною системою планування заготівлі молока, грн.; $П_2$ – обсяг втрат переробного підприємства або громади без використання інформаційної системи планування заготівлі молока, грн.; $З_{ІСП}$ – річні приведені витрати на інформаційну систему планування заготівлі молока, грн/рік.

Таблиця 5.1. Результати визначення економічної ефективності від використання інформаційних планування заготівлі молока для переробного підприємства або громади

№ п/п	Показники	Одиниця виміру	Значення показників
1	Вартість програмних та технічних засобів інформаційної системи	Грн.	48600
2	Експлуатаційні витрати інформаційної системи	Грн.	38462
3	Вартість розробки, проведення тестування	Грн.	51200
4	Собівартість пропонованої інформаційної системи	Грн.	104660
5	Приведені витрати на функціонування інформаційної системи	Грн./рік.	48928
6	Економічна ефективність	Грн./рік.	70112
7	Термін окупності капіталовкладень	Років	1,49

Річні приведені витрати від використання інформаційних планування заготівлі молока для переробного підприємства або громади становлять:

$$З_{ІСП} = E_н \cdot C_{бал} + B_р, \qquad (5.4)$$

Підставивши потрібні значення у формулу (5.4) зможемо визначити річні приведені витрати від використання інформаційних планування заготівлі молока для переробного підприємства або громади:

$$З_{ІСП} = 0,1 \cdot 104660 + 38462 = 48928 \text{ грн.}$$

На підставі формули (5.3) визначаємо економічну ефективність від використання переробним підприємством або громадою інформаційної системи планування заготівлі молока:

$$E_{ІСП} = (484250 - 365210) - 48928 = 70112 \text{ грн.}$$

Термін окупності інформаційної системи планування заготівлі молока визначається за формулою:

$$T_{ок} = \frac{З_{ІСП}}{E_{ІСП}}. \qquad (5.5)$$

За формулою (5.5) визначаємо термін окупності капіталовкладень у інформаційну систему планування заготівлі молока:

$$T_{ок} = \frac{104660}{70112} = 1,49 \text{ року.}$$

ВИСНОВКИ І ПРОПОЗИЦІЇ

Проведений аналіз стану заготівлі молока, а також існуючих у світі інформаційних систем планування заготівлі молока вказують на доцільність розроблення та використання інформаційної системи планування заготівлі молока для окремих громад України. Відповідно до цього, у кваліфікаційній роботі пропонується розробка інформаційної системи планування заготівлі молока із вибором ефективного алгоритму визначення маршрутів. Для цього слід вирішити у роботі слід розв’язати такі завдання.

Нами виконано аналіз стану заготівлі молока в Україні та його цінової політики. Встановлено, що великою проблемою в Україні є заготівля молока сировини від населення. Вцілому заготівля молока відіграє важливе значення у ефективності виробництва молочних продуктів та їх якості. Для підвищення ефективності заготівлі молока слід використовувати інформаційні системи.

На даний час у світі спостерігається тенденція до діджиталізації усіх процесів виробничої діяльності, у тому числі і заготівлі молока. Зокрема, у світі створюються молочні кооперативи. У пропонованій для них геоінформаційній системі можна зобразити маршрути на оцифрованих картах, і процес візуалізації альтернативних маршрутів доставки молока стане простішим у порівнянні з ручними процесами планування. Використання технології ГІС для оптимізації маршруту закупівлі молока значно сприятиме цим молочним підприємствам з точки зору економії грошей і часу.

В Україні для вирішення задач громад, які займаються виробництвом молока слід створювати інформаційні системи планування заготівлі. Під час проектування інформаційних систем планування слід враховувати особливості заготівлі молока на території громад.

Функціональною частиною інформаційної системи планування заготівлі молока є фактично модель системи управління об’єктами (рис. 2.1). Оскільки складні системи завжди багатофункціональні, інформаційні системи можна класифікувати за різними ознаками. Кожна інформаційна система,

представлена на ринку, може вирішити окремі задачі для громад. Однак, інформаційної системи для планування заготівлі молока із урахуванням їх особливостей, обсягів заготівлі молока та інших факторів, нажаль ще немає.

Одним з ключових аспектів даної роботи є вибір ефективного алгоритму для визначення маршрутів. Існує велика кількість алгоритмів для визначення маршрутів. Нами виконано порівняння деяких із них, зокрема алгоритму Дейкстри, алгоритму Белмана-Форда та алгоритм Флойда-Уоршела.

Алгоритм Дейкстри дає можливість знаходити найкращі маршрути для графу, для якого вибирається початкова вершина, а маршрут для усіх інших невідомий (рис. 2.2). Основним мінусом алгоритму Дейкстри є те, що він не може працювати з від'ємними ребрами, а лише з тими, котрі мають додатне значення, і в такому випадку даний алгоритм має обмежене використання.

Нами проаналізовано доцільність використання для визначення маршрутів заготівлі молока трохи менш ефективний, але більш універсальний алгоритм Белмана-Форда (рис. 2.4). Алгоритм дає дійсний результат, лише якщо граф не містить від'ємного циклу, тобто циклу, у якому сума ваг ребер від'ємна. Якщо такий цикл існує в графі, то кожен маршрут можна «скоротити», пройшовши негативний цикл багато разів. У цьому випадку алгоритм Белмана-Форда вказує про помилку.

Алгоритм Флойда-Уоршела являє собою алгоритм, який використовується для визначення найкоротших шляхів між кожною парою вершин у графі. Це алгоритм, заснований на динамічному програмуванні. Алгоритм має часову складність O(n3) і складність пам'яті O(n2) із заданою кількістю вершин (рис. 2.6). Алгоритм також можна використовувати для пошуку від'ємних циклів на графіку.

З метою вибору ефективного алгоритму визначення маршрутів проведено порівняльний аналіз, які стосуються результатів визначення маршрутів доставки молока від господарств громади до цеху його переробки із маршрутами, виконаними із використанням алгоритму Дейкстри, алгоритму Белмана-Форда та алгоритму Флойда-Уоршела. Встановлено, що за тривалістю

вирішення поставленої задачі найкращим є алгоритм Белмана-Форда, який забезпечує зниження часу на 3,2…6,8%.

Стосовно точності визначення маршрутів транспортування молока, то жоден із них під час їх порівняння не показав однозначно кращого результату. Тобто, за різної кількості господарств виробників молока та обсягів його заготівлі, раціональний алгоритм змінювався. Це свідчить про те, що проектована інформаційна система заготівлі молока для заданих виробничих умов на території громади повинна передбачати використання усіх зазначених алгоритмів, постійно їх порівнювати між собою та вибирати кращий.

Нами пропонується у проектованій інфорамційній системі використовувати додатки платформи Google Maps. Вона дає можливість відобразити місця, де знаходиться розташовані господарства виробники молока, забезпечити ефективне формування маршрутів із проаналізованими нами алгоритмами, а також графічне представлення результатів визначення маршрутів.

Для планування маршрутів у Maps JavaScript API, насамперед слід створити об’єкт type DirectionsServiceand call DirectionsService.route(), щоб ініціювати запит до служби Directions, передавши йому DirectionsRequest літерал об’єкта, що містить вхідні терміни та метод зворотного виклику для виконання після отримання відповіді (рис. 2.9). Точки маршруту дозволяють розрахувати маршрути через додаткові місця, у цьому випадку повернутий маршрут проходить через задані маршрутні точки.

Сервіс Distance Matrix від Google обчислює відстань і тривалість поїздки між кількома пунктами відправлення та призначення з використанням певного маршруту. Ця служба не повертає детальну інформацію про маршрут. Інформацію про маршрут, включно з ламаними лініями та текстовими вказівками, можна отримати, передавши єдиний потрібний вихідний пункт і пункт призначення службі напрямків Google.

Maps Static API повертає зображення (GIF, PNG або JPEG) у відповідь на HTTP-запит через URL-адресу. Для кожного запиту можна вказати

розташування карти, розмір зображення, рівень масштабування, тип карти та розміщення додаткових маркерів у місцях на карті. Можна додатково позначати маркери за допомогою буквено-цифрових символів.

Архітектуру інформаційної системи планування заготівлі молока ми обрали мультиоренду (Multi-tenant) та SaaS рішення. Мультиоренда – це форма хмарної архітектури, де кілька клієнтів одного постачальника хмари спільно використовують ті самі обчислювальні ресурси. Кожен клієнт відомий як орендар. Ця форма спільного використання стосується спільного використання ресурсів програмного забезпечення, а також спільного розміщення на серверах.

Нами побудовано діаграму варіантів використання інформаційної системи. У запропонованій системі планування заготівлі молока є три ролі для користувачів, кожна з яких має свої можливості – Host Admin, Admin та User (рис. 3.2).

Запропоновано, що керування користувачами має три функції, які детально описані на діаграмах активності з керуванням користувачами, яке може виконувати лише адміністратор. На рис. 3.4, а показано діаграму активності, яка пояснює процес перегляду деталей користувача, а на рис. 3.4, б можна побачити дію зі зміни даних користувача. Іншою функцією керування користувачами є видалення облікових записів із програми за допомогою кроків, зображених на рис. 3.5.

Діаграма послідовності описує взаємодію між об’єктами в системі та навколо неї. На діаграмі послідовності показано серію кроків, які виконуються системою у відповідь на процес для отримання певних результатів. Нами розроблено діаграму послідовності, яка забезпечує у заданій програмі пошук пунктів призначення за найкоротшим маршрутом. На цій діаграмі дії описують процес реєстрації в системі та введення заявки та підтвердження користувачів.

Нами представлено алгоритм, який знаходить найкоротший маршрут у заданій мережі господарств виробників молока та заданих виробничих умовах (рис. 3.7).

Для побудови архітектури back-end API було обрано архітектурне рішення DDD (Domain Driven Design). Це є проектування інформаційної системи, орієнтоване на предметну область. Такий підхід до проектування та розробки програмного забезпечення, який спочатку базується на вимогах бізнесу.

Програмним засобом для частини візуалізації даних була обрана платформа Angular. Angular – це відкрита структура та платформа для створення односторінкових додатків, написаних на TypeScript, які підтримуються та розробляються Google. Спочатку Angular мав бути версією 2 популярного фреймворку AngularJS.

Реєстрація окремих користувачів інформаційної системи (рис. 3.9.) здійснюється через меню «Реєстрація», вимагаючи ввести своє ім’я, номер телефону, адресу електронної пошти та пароль для входу у програму. Після завершення реєстрації програма надішле підтвердження на зареєстровану електронну адресу, щоб переконатися, що адреса електронної пошти правильна та новий обліковий запис готовий до використання.

Після цього виконано підключення матриць відстаней, виробничих умов та алгоритмів вибору маршрутів (рис. 3.10). Сторінка, яка відображається на сторінці входу адміністратора, показана на рис. 3.11. Адміністратори можуть вказати характеристики нових користувачів, якщо вони вже зареєстровані та ввійшли в додаток. Дизайн домашньої сторінки із вибором місця розташування господарства, що виробляє молоко та внесення його точних координат із використанням Maps Static API показано на 3.11, а користувачі можуть змінювати профілі за допомогою функції налаштувань профілю з відображенням.

На наступному рис. 3.13 представлено результати відображення бази даних щодо територіального розташування господарств виробників молока та пункту його збору.

На рис. 3.15 представлено дизайн сторінки із результатами визначення раціонального маршруту заготівлі молока, виконаний у три етапи для

отримання результатів розрахунку ефективного маршруту. У той час як дизайн інтерфейсу сторінки маршруту є результатом розрахунку маршруту, у вигляді карти представлено господарства заготівлі молока, а також черговість їх об їзду та доставки молока до пункту первинної обробки.

У роботі розроблено заходи із охорони праці та безпеки у надзвичайних ситуаціях під час заготівлі молока на території громади.

Ефективність використання інформаційної системи планування заготівлі молока визначається на підставі показників їх використання менеджерами із заготівлі молока. Зокрема, встановлено що переробні підприємства та громади використовуючи запропоновану систему зможуть щороку економити 70112 грн/рік. Термін окупності цієї інформаційної системи становить 1,49 року.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритм Беллмана-Форда [Електронний ресурс]. Wikipedia. 2022. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Bellman–Ford_algorithm

2. Брацький, В. О., М'якшило, О. М. Дослідження особливостей застосування реляційних і нереляційних баз даних на прикладі SQL Server та MongoDb. Наукові праці Національного університету харчових технологій, 2016. 22, № 5, С. 15-24.

3. Бродкевич В. М., Ремесло В. Я. Алгоритми машинного навчання та глибокого навчання і їх використання в прикладних додатках. Інтернаука. 2018. №11. С. 65-71.

4. Використання інформаційно-комунікаційних технологій на підприємствах [Електронний ресурс]. Державна служба статистики України. Режим доступу: https://ukrstat.gov.ua/operativ/operativ2018/zv/ikt/arh_ikt_u.html (дата звернення 01.09.2022).

5. Жидецький В.Ц., Джигирей В.С., Мельников О.В. Основи охорони праці. Підручник. Вид. 5-е, доповнене. Львів: Афіша, 2012. 350с.

6. Засоби технічного забезпечення управління інформаційними ресурсами. URL: http://um.co.ua/8/8-12/8-127157.html (дата звернення:19.10.2022).

7. Інформаційна система та програмне забезпечення інформаційної. URL: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/95/95.html (дата звернення:19.10.2022).

8. Коваль Н., Кисіль С., Тригуба А. Розробка бази даних інформаційної системи планування заготівлі молока. Інформаційна безпека та інформаційні технології: збірник тез доповідей V Всеукраїнської науково-практичної конференції молодих учених, студентів і курсантів, м. Львів, 26 листопада 2021 року. Львів, ЛДУ БЖД, 2021, С.137-139.

9. Коваль, Н. Особливості створення інформаційної технології оперативного планування заготівлі молока на території громад. Вісник

Львівського державного університету безпеки життєдіяльності, 2021. 24, С. 48-56. https://doi.org/https://doi.org/10.32447/20784643.24.2021.06

10. Коваль, Н. Функціональні моделі інформаційної технології та архітектура інформаційної системи оперативного планування заготівлі молока на території громад. Вісник Львівського національного аграрного університету. Агроінженерні дослідження, 2021. 25, С. 157–166. https://doi.org/10.31734/agroengineering2021.25.157

11. Кормен, Чарльз, Лейзерсон та ін. Алгоритми. Побудова та аналіз. USA: MIT Press, «Вильямс», 2019. 1296 с.

12. Лехман С.Д., Рублев В.І., Рябцев Б.І. Запобігання аварійності і травматизму у сільському господарстві. К.: Урожай, 1993. 267 с.

13. Мельник, К. В. Моделювання процесу інтелектуальної обробки медичних даних. Системи обробки інформації, 2017. (4), С. 237-244.

14. Міжнародні вантажоперевезення онлайн Lardi-Trans [Електронний ресурс]: [Веб-сайт]. Електронні дані. Режим доступу: https://lardi-trans.com/(дата звернення 01.11.2022)

15. Обсяги перевезених вантажів за видами транспорту за 2021 рік [Електронний ресурс]. Державна служба статистики України. Режим доступу: https://ukrstat.gov.ua/operativ/opera-tiv2022/tr/tr_rik/opvvt_22_ue.xlsx (дата звернення 22.08.2022).

16. Опис стандарту IDEF0. URL: http://easy-code.com.ua/2011/03/opis-standartu-idef (дата звернення:19.10.2022).

17. Опис стандарту IDEF3. URL: https://www.conceptdraw.com/examples/ (дата звернення:19.10.2022).

18. Основні поняття баз даних. Відомості про інформаційні системи URL: https://sites.google.com/view/ddkbmta-info/лекції/системи (дата звернення:19.10.2022).

19. Офіційна документація Google Maps JavaScript API. Google. 2022. Режим доступу до ресурсу: https://developers.google.com/maps/documentation/javascript.

20. Плескач В.Л., Рогушина Ю.В., Кустова Н.П. Інформаційні технології та системи. К.: Книга, 2004. 519 с.

21. Про затвердження Змін до Правил планування перевезень вантажів :Наказ; Мінтрансзв'язку України від 21.06.2007 №552 [Електронний ресурс] // База даних «Законодавство України» / Верховна Рада України. Режим доступу: https://zakon.rada.gov.ua/go/z0787-07 (дата звернення 10.10.2022)

22. Про схвалення Національної транспортної стратегії України на період до 2030 року : Розпорядження Кабінету Міністрів України; Стратегія від 30.05.2018 No 430-р[Електронний ресурс]. База даних «Законодавство України» / Верховна Рада України. Режим доступу: https://zakon.rada.gov.ua/go/430-2018-%D1%80 (дата звернення: 01.09.2022).

23. Програмне забезпечення та послуги в галузі вантажних перевезень [Електронний ресурс]: [Веб-сайт]. Електронні дані. Режим доступу: http://kpd-uz.com/ua/products/arm.php (дата звернення 21.10.2022)

24. Система GPS-моніторингу транспорту та вантажів [Електронний ресурс]: [Веб-сайт]. Електронні дані. Режим доступу: https://smartgps.com (дата звернення 11.10.2022)

25. Тригуба А.М., Коваль Н.Я. Алгоритм прогнозування добових обсягів молока на території громад. Вчені Львівського національного аграрного університету виробництву: каталог інноваційних розробок. За заг. ред. В. В. Снітинського, І. Б. Яціва. Вип. 21. Львів: Львів. нац. аграр. ун-т, 2021. С. 51.

26. Тригуба А.М., Коваль Н.Я., Татомир А.В., Тригуба І.Л. Інформаційна підтримка прийняття рішень під час планування проєктів заготівлі сировини. Інформаційні технології в енергетиці та агропромисловому комплексі: матеріали конференції XI-ї міжнародної наукової конференції Львівського НУП. Львів-Дубляни, 2022, С. 111–113.

27. Тригуба А.М., Коваль Н.Я., Тригуба І.Л., Кисіль С.Р. Архітектура інформаційної системи оперативного планування заготівлі молока на території громад. Інформаційні технології в енергетиці та агропромисловому комплексі:

матеріали конференції X-ї міжнародної наукової конференції присвяченої 165-річчю університету. Львів-Дубляни, 2021, С. 111–113.

28. Функціональні можливості системи «ТМкарта» [Електронний ресурс]: [Веб-сайт]. Електронні дані. Режим доступу: http://tmkarta.com/uk/about/long.php (дата звернення 01.10.2022)

29. Ashokkumar, P., Arunkumar, N., & Don, S. Intelligent optimal route recommendation among heterogeneous objects with keywords. Computers & Electrical Engineering, 2018. 68, 526-535.

30. Chen, X., Zhou, Y., Tang, Z., & Luo, Q. A hybrid algorithm combining glowworm swarm optimization and complete 2-opt algorithm for spherical travelling salesman problems. Applied Soft Computing, 2017. 58, 104-114.

31. Definition - What does IntelliJ IDEA mean? URL: https://www.conceptdraw.com/examples/idef3-diagram-software (дата звернення:19.10.2022).

32. Duque, D., Lozano, L., Medaglia, A. L. An exact method for the biobjective shortest path problem for large-scale road networks. European Journal of Operational Research, 2015. 242(3), 788-797.

33. Goyal A et al 2014 Path Finding: A* Or Dijkstra's? Int.J. of Innovative Trends in Engineering, 2, 1, pp. 1-15.

34. Kairanbay M, and Mat Jani H. A Review And Evaluations Of Shortest Path Algorithms. Int. J. of Sci. & Tech.Res. 2013, 2. 6. P. 99.

35. Malinin A., Korovko Y. How to Start ReactJS Development Fast: 3 Solid Tools and Best Practices. Codica blog. 2019. Режим доступу до ресурсу: https://www.codica.com/blog/how-to-start-reactjs-development-fast-3-solid-tools-and-best-practices.

36. Maulana, G. G. Pembelajaran Dasar Algoritma dan Pemrograman Menggunakan Elgoritma Berbasis Web. Jurnal Teknik Mesin Mercu Buana, 2017. 6(2), 69-73.

37. Nazari M., Oroojlooy A., Snyder L. V., Takác M. Reinforcement Learning for Solving the Vehicle Routing Problem. 18015. 2019. Режим доступу до

ресурсу: https://papers.nips.cc/paper/8190-reinforcement-learning-for-solving-the-vehicle-routing-problem.pdf.

38. Nugroho, A., & Afandi, R. T. Application of Broadcast Position Android-Based Tourist Group Using Google API. Int J Comput Neural Eng, 2018. 5(1), 98-104.

39. Patel, V., & Bagar, C. A survey paper of Bellman-Ford algorithm and Dijkstra algorithm for finding shortest path in GIS application. International Journal of P2P Network Trends and Technology, 2014. 5, 1-4.

40. Sedeno-Noda, A., & Raith, A. A Dijkstra-like method computing all extreme supported non-dominated solutions of the biobjective shortest path problem. Computers & Operations Research, 2015. 57, 83-94.

41. Sharma P, and Khurana N. Study of Optimal Path Finding Techniques. Int. J. of Adv.in Tech. 2013, 4, 2, P. 124.

42. Smilkov D., Thorat N. Tensorflow.js: machine learning for the web and beyond. Режим доступу до ресурсу: https://mlsys.org/Conferences/2019/doc/2019/154.pdf.

43. Talan K, and Bamnote G.R. Shortest Path Finding Using a Star Algorithm and Minimum Weight Node First Principle. Int. J. of Innovative Res. in Computer and Communication Engineering, 2015, 3, p. 1258.

44. Thombre, Multi-objective Path Finding Using Reinforcement Learning [Електронний ресурс] / Thombre, Prashant // Master's Projects. 2018. 643. Режим доступу до ресурсу: https://doi.org/10.31979/etd.2ntb-4j8q, https://scholarworks.sjsu.edu/etd_projects/643.

45. Victor Teixeira de Almeida. Using Dijkstra’s Algorithm to Incrementally Find the K-Nearest Neighbors in Spatial Network Databases. Praktische Informatik IV Fernuniversitat Hagen, D-58084 Hagen, Germany.

46. Wahyuningsih D and Syahreza E. Shortest Path Search Futsal Field Location With Dijkstra Algorithm. Indonesian J. of Computing and Cybernetics Systems. 2018, 12, 2. pp. 161-170.

47. What is REST. Guilding Principles of REST. URL: https://restfulapi.net (дата звернення:19.10.2022).

48. Windarto, S. W. Indratno, N. Nuraini, and E. Soewono. A comparison of binary and continuous genetic algorithm in parameter estimation of a logistic growth model AIP Conf. Proc. 2019, 1587. pp. 139–142 doi: 10.1063/1.4866550.