МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИ УНІВЕРСИТЕТ

Факультет інформаційних технологій, обліку та фінансів
Кафедра комп'ютерних технологій
і моделювання систем

Кваліфікаційна робота
на правах рукопису

Покоюк Артем Олександрович
(прізвище, ім'я, по батькові здобувача освіти)

УДК 629.312.5:681.3

КВАЛІФІКАЦІЙНА РОБОТА

Розробка інформаційної системи для станції технічного обслуговування автомобілів
(тема роботи)

126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня бакалавр

кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____
(підпис, ініціали та прізвище здобувача вищої освіти)

Науковий керівник – старший викладач
кафедри КТіМС Гіваргізов І.Г., к.е.н

Житомир – 2024

Висновок кафедри_____

за результатами попереднього захисту:_____

Протокол засідання кафедри_____

№ ____ від «_____» _____ 20____ р.

Завідувач кафедри

_____

_____ _____ _____

(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

«_____» _____ 20____ р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _____ захистив (ла)

(прізвище ,ім'я, по батькові)

кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____

за шкалою ECTS _____

за національною шкалою _____

Секретар ЕК

_____ _____ _____

(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

АНОТАЦІЯ

Покоюк А.О. Розробка інформаційної системи для станції технічного обслуговування автомобілів. – Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота присвячена розробленню інформаційної системи для станції технічного обслуговування автомобілів. Ця система призначена для автоматизації ключових процесів обслуговування автомобілів, зокрема, запису клієнтів, планування та контролю робіт, забезпечення якості та звітності.

Робота вирішує актуальну проблему складності технічного обслуговування, сприяючи підвищенню ефективності процесів, скороченню часу очікування клієнтів та поліпшенню загального клієнтського враження.

У роботі детально описано процес підготовки даних та побудову інформаційної системи. Особлива увага приділяється інтеграції системи з різноманітними аспектами технічного обслуговування та підтримки.

Також реалізовано додаток на платформі Windows, написаний мовою програмування C# з використанням технології Windows Forms. Цей додаток допомагає ефективно керувати процесами технічного обслуговування автомобілів, забезпечуючи зручний інтерфейс для реєстрації замовлень, призначення робіт майстрам, ведення обліку витрат тощо.

Ключові слова: інформаційна система, технічне обслуговування автомобілів, автоматизація, якість обслуговування, ефективність.

SUMMARY

Pokoyuk A.O. Development of an information system for a car service station. - Qualification work on manuscript rights.

The qualification work is devoted to the development of an information system for a car service station. This system is designed to automate key car service processes, including customer registration, job planning and control, quality assurance and reporting.

The work solves the pressing problem of technical maintenance complexity, helping to increase the efficiency of processes, reduce customer waiting time and improve the overall customer experience.

The work describes in detail the process of data preparation and the construction of an information system. Special attention is paid to system integration with various aspects of maintenance and support.

An application on the Windows platform, written in the C# programming language using Windows Forms technology, has also been implemented. This application helps to effectively manage car maintenance processes by providing a user-friendly interface for registering orders, assigning work to technicians, keeping track of expenses, and more.

Keywords: information system, car maintenance, automation, service quality, efficiency.

ЗМІСТ

АНОТАЦІЯ 3

ВСТУП 7

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ОСОБЛИВОСТЕЙ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СТАНЦІЇ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ 10

1.1 Аналіз інформаційних потреб і визначень предметної області 10

1.2 Функціональне моделювання інформаційної системи для станції технічного обслуговування автомобілів 14

1.3 Структура інформаційної системи для станції технічного обслуговування автомобілів 16

Висновки до першого розділу 19

РОЗДІЛ 2. РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СТАНЦІЇ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ 20

2.1. Формалізація моделі інформаційної системи для станції технічного обслуговування автомобілів 20

2.2. Реалізація математичної моделі інформаційної системи для станції технічного обслуговування автомобілів 21

Висновки до другого розділу 24

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СТАНЦІЇ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ 25

3.1. Проектування інтерфейсу інформаційної системи для станції технічного обслуговування автомобілів 25

3.2. Керівництво роботи з додатком користувачу 26

Висновки до третього розділу 27

ВИСНОВКИ ТА ПРОПОЗИЦІЇ 28

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ 29

Перелік скорочень та умовних позначень

IDEF0	Icam DEFinition for Function Modeling;
ER	Entity Relationship Diagram;
IDEF3	Integrated DEFinition for Process Description Capture Method;
CRM	Customer relationship management (управління відносинами з клієнтами)
DFD	Data Flow Diagrams
ПЗ	Програмне забезпечення
СТО	Станція технічного обслуговування

ВСТУП

Актуальність теми дослідження. На початку 2024 року, за оцінками експертів авторитетного аналітичного центру Avtosota[1], середній вік цих машин сягнув 23,2 років. Цей фактор, поряд із значною кількістю транспортних засобів та незадовільним станом доріг, робить послуги з ремонту та техобслуговування автомобілів вкрай затребуваними в Україні.

Розробка інформаційної системи для станції технічного обслуговування автомобілів може бути вигідним вкладенням, оскільки якість обслуговування та оперативність вирішення технічних питань певною мірою визначають репутацію сервісних центрів. Перехід до цифрових технологій у всіх сферах життя вимагає від індустрії автосервісу ефективних та інноваційних рішень для забезпечення якісного обслуговування та задоволення потреб клієнтів.

Мета кваліфікаційної роботи полягає у розробці інформаційної системи, яка оптимізує процеси обслуговування автомобілів на станціях технічного обслуговування. Ця має бути спрямована на автоматизацію робочих процесів, поліпшення взаємодії з клієнтами, оптимізацію витрат часу та ресурсів.

Для досягнення сформульованої мети визначено такі **завдання** дослідження: проаналізувати особливості інформаційної системи для станції технічного обслуговування автомобілів; змоделювати бізнес-процеси у сфері технічного обслуговування авто; спроєктувати інтерфейс системи з урахуванням потреби як клієнтів, так і працівників станції; розробити інформаційну систему для СТО; протестувати розробку на відповідність вимогам індустрії автосервісу.

Предметом дослідження є методи, технології та засоби побудови програмного забезпечення для автосервісів, включаючи прийом та реєстрацію транспортних засобів, діагностику, планування робіт, замовлення та облік запчастин, а також взаємодію з клієнтами.

---

[1] https://avtosota.com/41253-ekspert-rozpoviv-chy-vygidno-vidkryvaty-v-ukrayini-sto-u-2024-roczi.html#google_vignette

Об'єктом дослідження є процес розробки інформаційної системи для станції технічного обслуговування автомобілів.

Для досягнення визначеної в роботі мети використані такі **методи** дослідження: *аналізу* (для оцінки інформаційних потреб користувачів та конкурентного середовища на ринку ПЗ для автосервісу); *моделювання* (для розроблення математичної моделі ІС для СТО та опису її складових); *проєктування* (для побудови інтерфейсу ІС); *тестування й аналітики* (для забезпечення ефективності розробленої інформаційної системи для станції технічного обслуговування автомобілів).

Практичне значення отриманих результатів. Отримані результати дослідження відкривають нові можливості для покращення інформаційних систем у сфері технічного обслуговування автомобілів. Впровадження цих вдосконалень сприятиме підвищенню ефективності обслуговування, оптимізації бізнес-процесів та покращенню задоволення як клієнтів, так і персоналу станцій.

Наукова новизна полягає у впровадженні новаторського підходу до створення інформаційної системи для станцій технічного обслуговування автомобілів, що базується на глибокому аналізі потреб ринку та врахуванні специфіки вимог як користувачів, так і фахівців цієї галузі. **Унікальність** кваліфікаційної роботи полягає в розробці та впровадженні графіка обслуговування клієнтів, який автоматично розраховує оптимальну кількість обслуговуваних автомобілів за певний період часу, що сприяє оптимізації робочих процесів і підвищенню задоволення клієнтів.

Роботу апробовано на Міжфакультетській науково-практичній інтернет-конференції здобувачів вищої освіти і молодих вчених «Безпека, технології, інновації: нові горизонти» (м. Житомир, 2023 р.) та Всеукраїнській науково-практичній конференції здобувачів вищої освіти і молодих вчених «Інформаційні технології та моделювання систем» (м. Житомир, 2024 р.), за результатами яких опубліковано тези у збірниках конференцій:

1. Покоюк А., Програмне забезпечення для станції технічного обслуговування автомобілів // Інформаційні технології та моделювання систем : Безпека, технології, інновації: нові горизонти : збірник праць учасників міжфакультетської науково-практичної інтернет-конференції здобувачів вищої освіти і молодих вчених, 15 листопада 2023 р. – Житомир : Поліський національний університет, 2023. – 69 с. – С. 57–58.

2. Покоюк А., Інформаційна система станції технічного обслуговування автомобілів // Інформаційні технології та моделювання систем : збірник праць учасників Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих вчених, 10 квітня 2024 р. Житомир : Поліський національний університет, 2024. 76 с. – С. 29–30.

Структура та обсяг роботи. Кваліфікаційна робота складається з вступу, трьох розділів, висновку та списку використаних джерел. Загальний обсяг роботи становить 35 сторінок.

РОЗДІЛ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ОСОБЛИВОСТЕЙ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СТАНЦІЇ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

1.1 Аналіз інформаційних потреб і визначень предметної області

У даному розділі проведено систематичний аналіз вимог та термінологічних аспектів, спрямований на визначення потреби у інформаційних ресурсах та чітке визначення понять, які використовуються в контексті обслуговування автомобілів. Особлива увага приділена аналізу конкурентного середовища, в рамках якого проведено оцінку інформаційних стратегій конкуруючих станцій технічного обслуговування. Даний розділ також враховує особливості інформаційних потреб у контексті аналізу конкурентоспроможності станції, визначаючи ключові аспекти, які сприятимуть підвищенню якості послуг та задоволенню потреб клієнтів на новий рівень.

Наразі український IT-ринок представлений декількома компаніями, які спеціалізуються на розробці програмного забезпечення для станцій технічного обслуговування (СТО):

1. **CarStore.** Продукт CarStore в першу чергу спрямований на автосервіси, автомагазини та склади, що спеціалізуються на запчастинах для автомобілів та спецтехніки, як вітчизняного, так і зарубіжного виробництва. Робота автомагазину має багато специфічних особливостей, через що використання загальних бізнес-програм часто не дає бажаних результатів, оскільки вони не враховують багато важливих аспектів роботи автомагазину. Основна перевага програми CarStore полягає у її адаптації до специфіки підприємств, які торгують автозапчастинами.

Це програмне забезпечення, завдяки своїй спеціалізації, є високофункціональним і водночас легким у використанні. Простий і зрозумілий інтерфейс сприяє швидкому освоєнню програми, а використання простих форм введення допомагає мінімізувати помилки при реєстрації замовлень та взаємодії з товаром. За рахунок цих особливостей, реєстрація

складних замовлень та формування повного пакета документів вимагає мінімального часу та зусиль.

Ця версія продукту інтегрована з системою TecDoc, яка містить візуалізацію всіх вузлів та деталей автомобілів, дозволяє шукати деталі, визначати їх сумісність і показує аналогії товарів.

Ціна програми на одне перше робоче місце становить 200$. Додаткові робочі місця доступні за 50$ за кожне. Модуль РРО (Реєстратор Розрахункових Операцій) коштує 75$. Модуль інтернет-магазину - 300$.

2. **RemOnline.** Програма для автосервісу та СТО. Автоматизує бізнес-процеси та спрощує контроль. Має можливість попереднього запису на ремонт в автосервіс

За допомогою функції «Планувальника» можна оптимізувати навантаження співробітників та ресурсів перед автосервісом. Це дозволяє ефективно розподіляти навантаження на підйомники та робочі місця. Знаходити вільних фахівців для обслуговування клієнтів у зручний для них час. Рівномірно розподіляти замовлення між працівниками. Відстежувати час виконання робіт для більш ефективного управління часом.

- CRM-система для СТО й автосервісу

Покращувати обслуговування клієнтів, щоб вийти вперед перед конкурентами в автосервісній справі. Реєструвати контакти потенційних клієнтів з різних каналів зв'язку в одній базі та успішно перетворювати їх на реальні замовлення. Вести облік клієнтів автосервісу, відстежувати їх історію замовлень, оплати та виконаних робіт. Приймати вхідні дзвінки, автоматизовано надсилати повідомлення щодо майбутніх записів і збирати відгуки. Встановлювати індивідуальні знижки та підвищувати рівень лояльності клієнтів.

- Облік ТО та ремонту автомобілів

Реєструвати транспортні засоби клієнтів через онлайн-сервіс автосервісу та відстежувати їхню історію обслуговування. Вести облік технічного стану авто та контролювати їх рух, передаючи відповідальним

майстрам для зберігання. Прискорювати процес прийому замовлень завдяки ідентифікації транспортних засобів за VIN-кодом. Розраховувати передбачувану вартість технічного обслуговування та ремонтних робіт. Організовувати фонд автомобілів для надання клієнтам на тимчасове користування.

- Автоматизація автосервісу

Автоматизований розрахунок вартості послуг автосервісу з урахуванням виконаних робіт і вартості запчастин. Налаштовує нарахування за різними видами робіт та мотивує співробітників за допомогою встановлення процентних ставок. Веде облік робіт автосервісу та контролює строки виконання замовлень за допомогою часових статусів.

- Облік запчастин в автосервісі

Програма автоматизує складські процеси для організації складу та розміщення товарів на полицях. Контролює місцезнаходження кожної запчастини за допомогою адресного зберігання та унікальних кодів, що допомагає прискорити обслуговування. Забезпечує можливість проводити повні або часткові інвентаризації та миттєво виявляти будь-які недоліки. Організовує своєчасну закупівлю необхідних запчастин для своєчасного виконання сервісних робіт. Також відкриває можливість продажу запчастин і товарів, що додається до послуг, що збільшує товарообіг і прибуток.

- Управління автосервісом

RemOnline спрощує фінансовий та управлінський облік для керівника автосервісу, допомагаючи у прийнятті стратегічних рішень для розвитку бізнесу. Завдяки мобільному додатку можна відстежувати всі показники автосервісу в режимі реального часу та швидко зв'язуватися з персоналом. Формування управлінських звітів за будь-який період стає можливим за декілька кліків, а ключові показники доступні у зручному форматі графіків.

Тарифні плани відображено на рисунку (Рисунок 1.1).

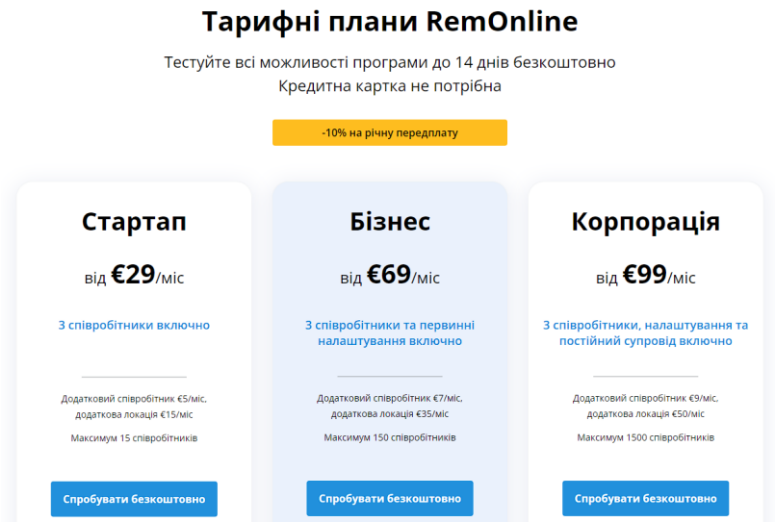

Рисунок 1.1 – Тарифні плани RemOnline

3. ЕкспресСофт. Пропоноване рішення базується на обліковій системі «BAS Малий бізнес» та розширює її функціонал для повноцінного обліку та управління станцією технічного обслуговування (СТО). Система "BAS Малий бізнес" вже має вбудовану функціональність, необхідну для невеликих підприємств, що спрощує початок роботи без глибоких знань в бухгалтерському та податковому обліку.

Це розширене рішення спрямоване на СТО, доповнюючи базовий функціонал системи для задоволення потреб цільової аудиторії. Очікується, що воно може забезпечити не лише основний облік, а й спеціалізовані функції, що характерні для діяльності технічного обслуговування автомобілів.

Вартість СТО програми розраховується виходячи з кількості користувачів, яким необхідно одночасно працювати з конфігурацією. Для одного користувача в якості вихідного рішення використовується “BAS Малий бізнес. ПРОФ“, для п’яти користувачів – “BAS Малий бізнес. ПРОФ. Комплект на 5 користувачів“.

При цьому в кожному з варіантів можна ще більше розширити кількість працюючих користувачів – для цього необхідно додатково придбати BAS клієнтські ліцензії на додаткові робочі місця.

Демо-версія програми для СТО безкоштовно

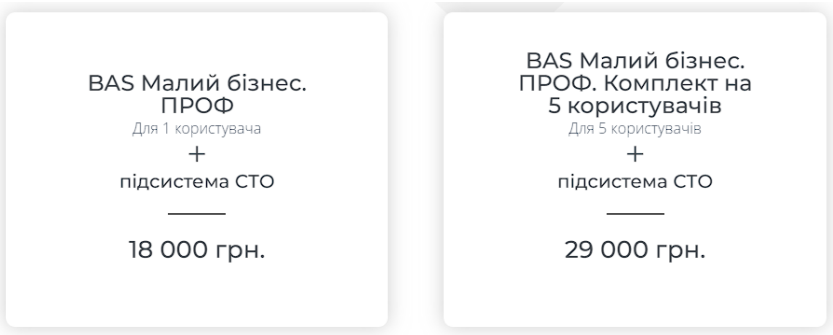

Рисунок 1.2 – Тарифні плани ЕкспресСофт

Ці компанії надають різноманітні рішення для автомобільних сервісних центрів та СТО, спрямовані на полегшення управління робочими процесами та підвищення ефективності обслуговування автомобілів.

Більшість компаній пропонують інтегровані рішення з використанням хмарних технологій, що дозволяє віддалено керувати та моніторити робочі процеси на СТО. Вони надають зручний спосіб керування та оптимізації роботи станцій технічного обслуговування, що спрощує підтримку та обслуговування автомобілів.

Таким чином, нова інформаційна система має поєднувати в собі всі ці характеристики (повне управління роботою СТО; інтеграція з популярними сторонніми системами; адаптація до специфіки роботи СТО; управління персоналом), щоб допомогти СТО покращити ефективність роботи, підвищити рівень обслуговування клієнтів та збільшити прибуток.

1.2 Функціональне моделювання інформаційної системи для станції технічного обслуговування автомобілів

Мета функціонального моделювання інформаційної системи станції технічного обслуговування полягає у створенні детальної та систематизованої картини функцій, які повинна виконувати ця система. Це включає в себе опис робочих процесів, функцій системи, взаємодій між різними компонентами та користувачами для забезпечення ефективного технічного обслуговування автомобілів.

Було створено контекстну діаграму процесу «Діяльність станції технічного обслуговування» (див. рис. 1.3, Дод. А).

До станції технічного обслуговування звертається власник автомобіля зі своїм автомобілем, тому на вході маємо заяву на обслуговування та автомобіль клієнта. Результатом роботи станції технічного обслуговування – відремонтований чи обслужений автомобіль та звіт про проведену роботу.

В діаграмі роль управління буде виконувати внутрішніми правилами СТО та правила обслуговування автомобіля. Механізмом процесу функціонування станції технічного обслуговування буде обладнання, комп’ютер, майстер та бухгалтер.

Проведемо декомпозицію контекстної діаграми на складові процеси (див. рис. 1.4, Дод. А). Процес «Діяльність станції технічного обслуговування» поділимо на такі підпроцеси: прийом заявки; обслуговування авто; збереження історії виконаних робіт.

Розглянемо й проаналізуємо діаграму розкладання процесу "Діяльність станції технічного обслуговування" з використанням стандарту IDEF0.

Проведемо декомпозицію діаграми «Прийом заявки» на складові процеси (див. рис. 1.5, Дод. А). Процес «Прийом заявки» поділимо на такі підпроцеси: додавання заявки; ведення статистики.

Виконаємо декомпозицію процесу «Обслуговування авто» на основі побудованої діаграми з використанням стандарту IDEF0 (див. рис. 1.6, Дод. А).

Діаграма декомпозиції у складі має такі процеси: діагностика авто; замовлення необхідних матеріалів; надання послуги. Вона розкриває структуру та ієрархію основного процесу, розбиваючи його на більш дрібні та конкретні етапи. У цьому контексті визначено три основні підпроцеси: Діагностика авто, Замовлення необхідних матеріалів та Надання послуги. Ця діаграма дозволяє деталізувати та систематизувати кожен етап процесу обслуговування автомобіля для забезпечення більш ефективної та управлінської структури в межах автосервісної діяльності.

1.3 Структура інформаційної системи для станції технічного обслуговування автомобілів

Для того щоб проаналізувати інформаційну систему для СТО, було обрано методологію моделювання процесів за допомогою DFD діаграм. DFD (Data Flow Diagram) – це методологія моделювання процесів, яка використовує графічне зображення для показу потоків даних в інформаційній системі. DFD діаграма відображає обмін даними між процесами, зовнішніми сутностями (такими як користувачі чи інші системи) та даними, які зберігаються в базах даних.

Виконаємо декомпозиції процесу «Додавання заявки в БД» наступним чином (див. рис. 1.7, Дод. А). Опрацювання заявки виконується за такою процедурою:

- Власник автомобіля звертається до менеджера станції технічного обслуговування. Аналізується інформація про причину звернення.
- Менеджер шукає авто в базі СТО або створює нового контрагента.
- Пошук дати та години для обслуговування авто.
- Виконується пошук вільного майстра.

Детальний опис складових процесу наведено в таблиці (Таблиця 1.7).

Таблиця 1.7 – Основні елементи DFD діаграми «Інформаційна система станції технічного обслуговування»

Ключові слова	**Визначення**	**Призначення елементу**
Заявка на обслуговування	Заява власника авто на обслуговування	Зовнішня сутність
З'ясування причини звернення	Процес збору інформації, її систематизація та узагальнення.	Процес
Пошук авто в БД	Процес обміну даними з базою даних, де розміщено облікові записи всіх автомобілів станції технічного обслуговування, з метою знайти конкретне авто	Процес
Пошук дати сервісу	Процес обміну даними з базою даних для пошуку вільної дати та години для сервісу	Процес
Вибір майстра	Процес обміну даними з базою даних, де розміщено облікові записи всіх майстрів з метою підбору вільного майстра по відповідній спеціалізації	Процес

База авто	База даних станції технічного обслуговування, у якій знаходиться інформація автомобіль та його історію відвідування даного СТО.	Сховище даних
БД Календар	База даних календар дозволяє знайти вільну дату та годину прийому на станції технічного обслуговування.	Сховище даних
База майстрів	База даних майстрів даного СТО.	Сховище даних
Дані з БД	Всі необхідні дані про авто власника, та відповідального майстра за обслуговування.	Потік даних

Далі було створено опис процесу «Замовлення необхідних матеріалів» наступним чином (див. рис. 1.8, Дод. А):

Опрацювання звернення виконується за наступною процедурою:

- Пошук витратних матеріалів;
- Узгодження вартості;
- Замовлення зі складу;

Детальний опис складових процесу «Замовлення необхідних матеріалів» наведено в таблиці (Таблиця 1.8).

IDEF3 (Integrated Definition for Function Modeling) - це методологія та графічна нотація, що використовується для моделювання та аналізу функціональних властивостей систем. Розроблена з метою забезпечення структурованого та систематичного підходу до опису взаємозв'язків між функціями в складних системах.

Основна мета – це надати інженерам та аналітикам зручний інструментарій для моделювання і аналізу функціональної взаємодії різних елементів системи. Використовуючи IDEF3 діаграми, можна визначити порядок виконання функцій, їхню взаємодію та інші аспекти, які впливають на функціональні аспекти системи.

Типовою властивістю є використання блок-схем та стрілок для представлення різних функціональних елементів та їхніх зв'язків. Ця методологія може бути використана для уточнення та вдосконалення

Застосування нотації IDEF3 проведемо декомпозицію процесу «Діагностика авто» (див. рис. 1.9, Дод. А).

Процес діагностики автомобіля розпочинається із з’ясування скарг власника авто та переходить до миття автомобіля, що включати в себе різноманітні процеси та процедури миття. У даній декомпозиції процесу використано вид розгалужень, відомий як "асинхронне &". Це розшифровується як асинхронне рішення, що включає в себе одночасне виконання різних функцій чи етапів в процесі. У контексті даної декомпозиції, "асинхронне &" використовується для паралельного виконання трьох блоків, включаючи "Комп'ютерну діагностику", "Діагностику двигуна" та "Діагностику ходової". Це дозволяє одночасно проводити різні види діагностики автомобіля, щоб ефективно визначити його технічний стан. Використання "асинхронного &" сприяє оптимізації часу та ресурсів в рамках обслуговування автотранспортних засобів. Після всіх діагностик потрібно проаналізувати отримані дані. Встановити причину несправності в авто. Далі визначаємо необхідні матеріали для ремонту авто. Всю цю інформацію доводимо до власника автомобіля.

Також застосуємо IDEF3 для опису процесу «Надання послуги» (див. рис. 1.10, Дод. А).

Процес "Надання послуг" в рамках даної схеми є ключовим етапом у відповіді на потреби клієнтів. Починаючи з розібрання автомобіля та виявлення несправностей, процес включає в себе підготовку необхідних запчастин та їхню подальшу заміну чи ремонт. Зібрання автомобіля та виконання обслуговування вузла є завершальними етапами, спрямованими на те, щоб забезпечити оптимальне функціонування транспортного засобу. Також, включає в себе не лише ремонт та технічне обслуговування, але й відповідь на індивідуальні потреби клієнтів, забезпечуючи високий рівень задоволення та довіри до обраного автосервісу. Цей процес є стратегічно важливим для забезпечення надійності та продуктивності автопарку та відображає високий стандарт обслуговування в галузі технічного обслуговування автомобілів.

Висновки до першого розділу

У першому розділі проаналізовано важливість автоматизації процесів управління, ведення обліку та взаємодії з клієнтами для підвищення ефективності та конкурентоспроможності, виявлено комплексність вимог до системи. Досліджено аналоги ІС для станцій технічного обслуговування, на основі чого окреслено необхідність розробки системи, що забезпечить відповідність інформаційних потреб станції та покращить її функціональність: повне управління роботою СТО (планування записів на ремонт; CRM-система для управління клієнтами; облік ТО та ремонту автомобілів; автоматизація автосервісу; управління автосервісом); інтеграція з популярними сторонніми системами (системи онлайн-запису на прийом; системи бухгалтерського обліку; системи управління складом); адаптація до специфіки роботи СТО (облік запчастин; ведення історії обслуговування автомобілів; розрахунок вартості послуг; управління персоналом) тощо.

Також було спроектовано та розроблено модель процесу функціонування станції технічного обслуговування за методологією IDEF0. Ця модель включає деталізацію кожного етапу роботи, який аналізується, разом із глосарієм термінів та ключових концепцій, що застосовуються до кожної декомпозиції.

РОЗДІЛ 2. РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СТАНЦІЇ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

2.1. Формалізація моделі інформаційної системи для станції технічного обслуговування автомобілів

Створення інформаційних систем для обслуговування автомобілів на станціях технічного обслуговування (СТО) стає визначальним фактором в управлінні великим обсягом транспортних засобів. Завдяки цим системам, СТО можуть оптимізувати свою продуктивність та ефективність, обслуговуючи значну кількість автомобілів без втрати якості послуг.

Розробка інформаційної системи для обслуговування автомобілів забезпечують точну інформацію про технічний стан кожного автомобіля, що дозволяє СТО планувати та проводити технічне обслуговування так, щоб максимально використовувати час і ресурси. Це особливо актуально для великих автопарків або мереж автосервісів, які мають великий потік транспортних засобів. Використання інформаційних систем дозволяє збільшити обсяг обслуговування, знижуючи при цьому час, необхідний для кожного автомобіля, що сприяє якості обслуговування та задоволенню клієнтів. Диференційне рівняння методом масового обслуговування:

(1)

$$\frac{dl}{dt} = (\lambda - \mu) * l(t) = \begin{cases} (\lambda - \mu) * \mathrm{l(t)} > 0, \lambda > \mu \\ (\lambda - \mu) * \mathrm{l(t)} < 0, \lambda < \mu \end{cases}$$

де l (t)- кількість об'єктів (автомобілів) в момент часу t, λ - інтенсивність надходження, μ - інтенсивність обслуговування.

Тепер розробим структурну схему цієї формули, вона показана на рисунку (Рисунок 2.1)

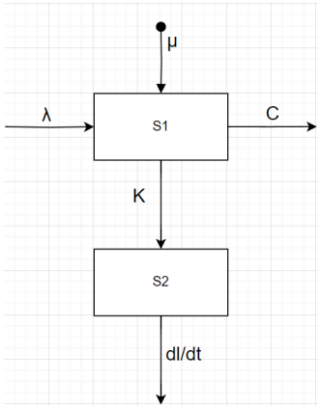

Рисунок 2.1 – Схема числа обслуговуваних автомобілів на станції

Якщо ми розглядаємо рівняння $\frac{dl}{dt} = (\lambda - \mu) * l(t)$ і вважаємо, що λ>μ, то це означає, що вираз (λ−μ) додатній. У цьому випадку, розв'язок буде виглядати як $l(t) = l0 \cdot e^{(\lambda-\mu)t}$, де l_0 - початкова кількість обслуговуваних автомобілів при $t = 0$. Це експоненційне зростання, що означає, що кількість обслуговуваних автомобілів зростатиме пропорційно часу t. Графік буде плавно зростати з клієнтами автомобілів, що обслуговуються, протягом часу.

А якщо $(\lambda - \mu) \cdot l(t) < 0$, це означає, що вираз $(\lambda - \mu)$ має протилежний знак від $l(t)$. Це вказує на те, що кількість обслуговуваних автомобілів $l(t)$ зменшується з часом. Роз'язок виглядатиме так $l(t) = l0 \cdot e^{(\lambda-\mu)t}$, де $l0$ - початкова кількість обслуговуваних автомобілів при $t = 0$. У цьому випадку, графік буде виглядати як експоненційний спад, змінюючи кількість обслуговуваних автомобілів пропорційно зменшенню часу t.

2.2. Реалізація математичної моделі інформаційної системи для станції технічного обслуговування автомобілів

Реалізація моделей була розроблена в редактору коду Visual Studio Code, на інтерпретованій об'єктно-орієнтованій мові програмування Python.

```
def calculate_serviced_cars(l, lambd, mu):
    # Розрахунок кількості обслуговуваних автомобілів
    serviced_cars = (lambd - mu) * l
```

Рисунок 2.2 – Функція

Визначення функції «calculate_serviced_cars», яка приймає три параметри: кількість автомобілів на станції (l), швидкість надходження автомобілів (lambd), та швидкість обслуговування (mu).

```
# Перевірка умови інкременту або декременту
if serviced_cars > 0:
    result = "Більше автомобілів обслуговується (λ > μ)"
elif serviced_cars < 0:
    result = "Менше автомобілів обслуговується (λ < μ)"
else:
    result = "Кількість обслуговуваних автомобілів залишається незмінною (λ = μ)"

return serviced_cars, result
```

Рисунок 2.3 – Перевірка умови

Виведення кількості обслуговуваних автомобілів і відповідної умови, яка вказує, чи більше, менше чи рівне кількість надходження обслуговуванню.

```
# Введення значень для автомобілів на станції
cars_on_station = 6  # Кількість автомобілів на станції технічного обслуговування
incoming_rate = 1  # Швидкість надходження автомобілів (λ)
service_rate = 0.5  # Швидкість обслуговування автомобілів (μ)
```

Рисунок 2.4 – Введення значень

Користувач вводить параметри для кількості автомобілів на станції, швидкості надходження та швидкості обслуговування.

```
# Виклик функції та вивід результатів
serviced_cars_result, condition_result = calculate_serviced_cars(cars_on_station,
print(f"Кількість обслуговуваних автомобілів: {serviced_cars_result}")
print(f"Умова: {condition_result}")
```

Рисунок 2.5 – Вивід результатів

Здійснюється виклик функції «calculate_serviced_cars» з введеними значеннями. Результати виводяться, описуючи, чи більше, менше чи рівне кількість надходження обслуговуванню.

```
Powered by trinket
Кількість обслуговуваних автомобілів: 3.0
Умова: Більше автомобілів обслуговується (λ > μ)
```

Рисунок 2.6 – Результат після компіляції

Ця програма на мові Python реалізовує модель обслуговування автомобілів на станції технічного обслуговування, враховуючи швидкість надходження та обслуговування. Виводиться динаміка зміни кількості обслуговуваних автомобілів, а також умова, чи кількість обслуговуваних

автомобілів перевищує, менше або дорівнює швидкості їхнього надходження на станцію.

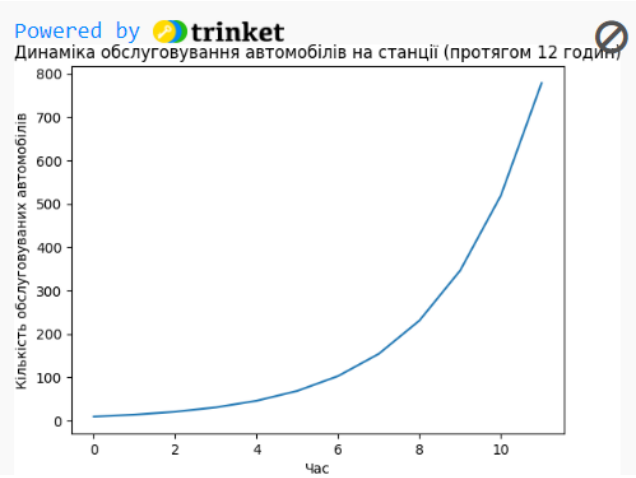

Рисунок 2.7 – Графік динаміки обслуговування автомобілів на станції

Графік динаміки обслуговування автомобілів на станції технічного обслуговування відображає схожість із S-кривою. Починаючи зі стартової кількості автомобілів, процес зростання спочатку відбувається повільно, із помірною темпом, після чого швидкість збільшується, а графік різко піднімається. У середині процес може досягти свого піку, після чого зростання сповільнюється. В кінці графік може демонструвати певний спад, що вказує на можливу деградацію або зменшення кількості обслуговуваних автомобілів. Цей тип зростання характерний для S-кривих, що вказує на зміни темпу в часі відносно кількості обслуговуваних автомобілів.

Дані, отримані з математичної моделі, можуть бути використані для покращення роботи СТО в різних аспектах. Наприклад, результати моделювання обслуговування автомобілів на станції технічного обслуговування, які враховують швидкість надходження та обслуговування, можуть бути використані для оптимізації планування роботи майстрів, розподілу ресурсів та забезпечення ефективного обслуговування клієнтів.

Наприклад, інформаційна система СТО може використовувати дані про динаміку зміни кількості обслуговуваних автомобілів, які отримані з математичної моделі, для прогнозування піку навантаження і планування графіків роботи майстрів. Крім того, дані про темпи зростання та спаду

кількості обслуговуваних автомобілів можуть використовуватися для оптимізації ресурсів, наприклад, планування технічного обслуговування обладнання та закупівлю необхідних запасних частин.

Таким чином, інформація, отримана з математичної моделі, може допомогти інформаційній системі СТО приймати кращі рішення щодо планування роботи майстрів, розподілу ресурсів та забезпечення ефективного обслуговування клієнтів.

Висновки до другого розділу

Впровадження інформаційних систем на станціях технічного обслуговування на сьогодні є одним із ключових факторів для оптимізації роботи та покращення якості послуг. Ці системи дозволяють: отримувати точну інформацію про стан кожного автомобіля; збільшувати обсяг обслуговування та підвищувати якість обслуговування та задоволеність клієнтів.

Математичні моделі, такі що представлені в цьому розділі, дозволяють моделювати роботу СТО та прогнозувати її ефективність залежно від різних параметрів. Це може допомогти СТО приймати кращі управлінські рішення та оптимізувати свою роботу.

Реалізація математичної моделі на мові Python може бути використана для генерування графіків, які показують динаміку кількості обслуговуваних автомобілів. Ці графіки можуть допомогти СТО візуалізувати вплив різних факторів на їхню продуктивність.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СТАНЦІЇ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

3.1. Проектування інтерфейсу інформаційної системи для станції технічного обслуговування автомобілів

Інтерфейс системи – важлива річ будь-якої інформаційної розробки. Інтерфейс повинен демонструвати весь потенціал програми, не відлякувати користувача, бути простим і зрозумілим.

Розглянемо зовнішній вигляд додатку. Першим серед них буде головна форма для роботи з усіма даними СТО. На цій вкладці можна додавати нове авто, відредагувати вибране або видалити. Можна накласти фільтр по власнику або державному номері. (див. рис. 3.1, Дод. Б)

Наступна форма використовується для створення нового автомобіля та редагування вибраного автомобіля в базі СТО. Форма має поля для заповнення та кнопки для виконання певних дій. (див. рис. 3.2, Дод. Б)

Для перегляду детальної інформації про замовлення потрібно вибрати замовлення та перейти на вкладку «Надані послуги по замовленню». Надані послуги стосуються лише вибраного замовлення. Є можливість бачити загальну вартість замовлення. (див. рис. 3.3, Дод. Б)

При натисканні на кнопку «Додати нову послугу» відкриється нова форма для вибору послуги на СТО. Форма дозволяє відкрити довідник послуг для редагування чи то створення нової послуги на СТО. При потребі надану послугу можна видалити або відредагувати. (див. рис. 3.4, Дод. Б)

З меню можна відкрити форму для перегляду інформації про розробника програмного забезпечення. (див. рис. 3.5, Дод. Б)

Стовпчаста діаграма ілюструє кількість обслуговуваних автомобілів протягом певного періоду часу, що дозволяє спостерігати за динамікою обсягів обслуговування. Кожен стовпець діаграми представляє кількість автомобілів, які були обслуговувані протягом конкретного періоду, дозволяючи оперативно оцінити завантаженість станції та розподіл робочого

навантаження. Ця діаграма є корисним інструментом для аналізу роботи системи та прийняття управлінських рішень з метою оптимізації процесів обслуговування автомобілів. (див. рис. 3.6, Дод. Б).

3.2. Керівництво роботи з додатком користувачу

Після входу в додаток, користувачу буде запропонована форма, яка містить чотири вкладки: "Авто", "Замовлення", "Надані послуги по замовленню" та "Інформація про клієнта". На вкладці "Авто" можна додавати нові автомобілі чи редагувати існуючі, забезпечуючи повну інформацію про них, таку як марка, модель, рік випуску і інше. Важливо натискати "Зберегти" після введення даних.

Вкладка "Надані послуги по замовленню" надає зручний довідник послуг, де можна редагувати чи створювати нові послуги. Крім того, можна вибрати послугу для редагування чи видалення. Вкладка "Інформація про клієнта" дозволяє переглядати дані клієнтів та їхню історію обслуговування.

Додаткові опції, такі як "Інформація про розробника", доступні через меню програми. Це може бути корисно для отримання більш детальної інформації про розробника програмного продукту. Завершіть роботу за допомогою кнопки "Вихід" або відповідного пункту меню.

Було створено відеоінструкцію FAQ, яка ретельно демонструє роботу інформаційної системи станції технічного обслуговування автомобілів. Відео надає вичерпну інструкцію з використання програмного забезпечення, подробиці його функціональності та корисні поради з ефективного використання. Для зручності користувачів, вони мають можливість залишити свої відгуки та пропозиції щодо покращення програми. Ці відгуки надсилаються розробнику, щоб вони могли врахуватись при їх подальшому вдосконаленні програми та внесенні змін.

Це не просто відеоінструкція — це візуальна демонстрація роботи програми. Яка крок за кроком показує, як використовувати різні функції та можливості програмного забезпечення, дозволяючи користувачам з легкістю освоювати його. Відкрийте це посилання «https://youtu.be/L43RahyZ9o0», щоб

отримати доступ до відеоінструкції з демонстрацією роботи інформаційної системи станції технічного обслуговування автомобілів.

Висновки до третього розділу

Інтерфейс користувача розробленої інформаційної системи для СТО відповідає всім сучасним вимогам. Він простий, зрозумілий та зручний у використанні. Інтерфейс не перевантажений зайвою інформацією, що робить роботу з програмою комфортною. Користувач чітко бачить всі доступні функції та можливості програми. Використання візуальних елементів, таких як діаграми, робить інтерфейс більш інформативним та наочним.

Інтерфейс користувача дозволяє виконувати всі необхідні дії з даними СТО: додавати, редагувати та видаляти інформацію про автомобілі; створювати, редагувати та видаляти замовлення; додавати, редагувати та видаляти послуги; переглядати історію обслуговування автомобілів; отримувати інформацію про розробника програми.

Інструкція користувача є якісним доповненням до розробленого додатку для СТО. Вона допомагає користувачам швидко освоїти програму та використовувати всі її можливості.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Автосервіс має значний потенціал для розвитку, проте для досягнення цієї мети потрібно систематично впроваджувати передові інформаційні технології на сервісних станціях. Це включає в себе сучасні засоби діагностики, електронні каталоги запчастин і програмне забезпечення для управління станцією технічного обслуговування.

Для оптимальної функціональності СТО, інформаційна система повинна відповідати стандартам, забезпечуючи автоматизацію бізнес-процесів та надавати зручний інтерфейс користувача. Окрім цього, вона повинна гарантувати високий рівень обслуговування клієнтів, застосовувати сучасні методи мотивації працівників СТО і забезпечувати можливість реального моніторингу діяльності підприємства.

Метою даної кваліфікаційної роботи було розробка інформаційної системи, яка оптимізує процеси обслуговування автомобілів на станціях технічного обслуговування. Крім того, виконання цієї роботи сприяло глибшому розумінню сучасних технологій у сфері інформаційних систем та вдосконаленню навичок у аналізі та проектуванні.

Розроблено моделі процесів діяльності станції технічного обслуговування для систематизації та покращення її функціонування.

Було проаналізовано програмне забезпечення інших компаній на ринку та створена модель процесу діяльності станції технічного обслуговування за методологією IDEF0, включаючи декомпозицію та розгортання цього процесу. Під час написання кваліфікаційної роботи використано DFD для організації потоків документів та обробки інформації, а також виконано кілька декомпозицій. Додатково, було проілюстровано використання нотації IDEF3 для розгляду структурних аспектів процесів.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. El-Mousa, A. H., Muhsin, Z. J., Al-Taee, M. A. (2008). A Web-Based Rapid Prototyping Workflow Management Information System for Computer Repair and Maintenance. Journal of Computer Science, 4 (12), 991–998. ISSN 1549–3636.

2. CarStore - Програмне забезпечення для автомагазинів та автосервісів [Електронний ресурс]. Режим доступу до ресурсу: https://www.carstore.com.ua/.

3. CRM Appointer [Електронний ресурс]. Режим доступу до ресурсу: https://appointer.ua/.

4. EasyWeek [Електронний ресурс]. Режим доступу до ресурсу: https://easyweek.com.ua/programa-dlya-vetklinik.html.

5. Eckerson, W. (2003). Smart Companies in the 21st Century: The Secrets to Creating Successful Business Intelligence Solutions. TDWI Report Series. The Data Warehouse Institute, 40 p.

6. ExpresSsoft - Програма для автосервісу та СТО [Електронний ресурс]. Режим доступу до ресурсу: https://expressoft.com.ua/uk/programa-dlja-avtoservisu-sto/.

7. RemOnline - Програма для автосервісу та СТО [Електронний ресурс]. Режим доступу до ресурсу: https://remonline.ua/autoservice/.

8. SkyService POS [Електронний ресурс]. Режим доступу до ресурсу: https://skyservice.pro/.

9. Андрусенко, С. І., Бугайчук, О. С. (2014). Моделювання бізнес-процесів підприємства автосервісу: монографія. К.: Кафедра, 325 с.

10. Андрусенко, С.І., Бугайчук, О.С. (2017). Технології підвищення ефективності виробничо-технічної бази підприємств автомобільного транспорту: навчальний посібник. К.: НТУ, 190 с.

11. Юрчук, Н. П. Інформаційні системи в управлінні діяльністю підприємства [Текст] / Н. П. Юрчук // Агросвіт. - 2015. - № 19. - С. 53–58.

12. Криворучко, О. В., Дитинюк, О. В. Інформаційні технології моделювання виробничих процесів як інструмент прийняття управлінських рішень [Текст] / О. В. Криворучко, О. В. Дитинюк // Інформаційні технології управління. - 2017. - № 31. - С. 65–70.

13. Kornienko, G., Chabanenko, M., Leheza, Y. Assessment of the economic efficiency of it application at enterprises [Текст] / G. Kornienko, M. Chabanenko, Y. Leheza // Baltic Journal of Economic Studies. - 2018. - Vol. 4, No. 3. - P. 123–132. - DOI: https://doi.org/10.30525/2256-0742/2018-4-3-123-132.

14. Дорохов, О. В. Критерії та методи оцінки ефективності інформаційних систем [Текст] / О. В. Дорохов // Системи обробки інформації. - 2010. - № 1. - С. 219–222.

15. Писарчук, О. О. Оцінювання ефективності інформаційних систем за вектором критеріїв [Текст] / О. О. Писарчук // Журнал Вінницького національного аграрного університету. - 2018. - Вип. 3. - С. 117-123.

16. Верескун, М. В. Методи оцінки ефективності впровадження інформаційних систем на промислових підприємствах [Текст] / М. В. Верескун // Теоретичні і практичні аспекти економіки та інтелектуальної власності. - ПДТУ. - Маріуполь. - 2015. - Вип. 1, Т. 1. - С. 21–26.

17. Сафонов, М. С., Яковенко, О. Є. Прогнозування стану показників об’єктно-орієнтованої моделі в інформаційній системі [Текст] / М. С. Сафонов, О. Є. Яковенко // Інформаційні технології в освіті, науці та виробництві. - Одеса. - 2013. - № 3(4). - С. 92 – 98.

18. Beskorovajnyj, V. V. Systemological analysis of the problem of structural synthesis of geographically distributed systems [Текст] / V. V. Beskorovajnyj // Automated control systems and automation devices. - 2002. - No. 120. - P. 29–37.

19. Буй, Ф. Л., Беляєв, Л. В. Методи і технології реінжинірингу інформаційних систем [Текст] / Ф. Л. Буй, Л. В. Беляєв // Інформаційні технології в економіці і управлінні: зб. наук. студ. праць. - Одеса: ОНЕУ. - 2019. - Вип. 1. - С. 145–152.

20. Іванова, Т. В., Баранов, В. В. Сучасний стан розвитку інформаційних систем [Електронний ресурс]. - URL: www.kntu.kr.ua/doc/nauk_zap_10_1/stat_10_1/64.doc.

21. ДСТУ ISO/IEC 2382:2017 (ISO/IEC 2382:2015, IDT). Інформаційні технології. Словник термінів [Текст]. - К.: УкрНДНЦ. - 2017.

22. Кисіль, Н. М., Гаталяк, З. П., Горбаль, Н. І. Класифікація інформаційних систем [Електронний ресурс]. - URL: base.dnsgb.com.ua/files/journal/Lisove-gospodarstvo-l-p.../242 _Kysil_LG_29.pdf.

23. Литовченко, Д. М. Реінжиніринг інформаційних систем економічних об'єктів. Моделювання регіональної економіки. 2010. № 1. С. 176–184.

24. Черняк, Н.І. Моделювання структури інформаційної системи управління агропромислового комплексу регіону [Текст] / Н.І. Черняк // Оптико-електронні інформаційно-енергетичні технології. - 2008. - № 2. - С. 83-88.

25. Яремко, С.А., Бевз, С.В. Розробка критеріїв оцінювання сучасних інформаційних систем обліку та управління бізнес-процесами підприємств [Текст] / С.А. Яремко, С.В. Бевз // Вісник Хмельницького національного університету. - 2015. - № 1 35 університету. - С. 158–163.

26. Філинюк, М. А., Багацький, В. О., Ліщинська, Л. Б., Войцеховська, О. В. Критеріальне оцінювання ефективності інформаційних пристроїв та систем: навчальний посібник [Текст] / М. А. Філинюк, В. О. Багацький, Л. Б. Ліщинська, О. В. Войцеховська. - Вінниця: ВНТУ, 2014. - 143 с.

27. Zhang Yi Fei, Che Ruhana Isa. Factors Influencing Activity-Based Costing Success: A Research Framework [Текст] / Yi Fei Zhang, Ruhana Isa Che // International Journal of Trade, Economics and Finance. - 2010. - V.1, no.2. - P. 144-150.

28. Evaluation of effectiveness of information systems implementation in organization (by example of ERP-systems) [Текст] / O V Demyanova and other // Journal of Physics Conference Series. - 2018.05. - Vol. 1015. - P. 1-5.

29. Muhamed Saif Q., Mohammed Mohammed Q., Nayl T., Chyrkova K. The Concept of Building a Model of the National Blood Information System [Текст] /

Muhamed Saif Q., Mohammed Mohammed Q., Nayl T., K. Chyrkova // Iraqi Journal for Computers and Informatics (IJCI). - 2017. - 43 (1). - P. 17 – 21.

30. Чиркова, К.С. Особливості функціонування інформаційних систем служби крові [Текст] / К.С. Чиркова // Матеріали 19-ого міжнародного молодіжного форуму «Радіоелектроніка і молодь в XXI в.». - Харків, 2015. - С. 160–161.

31. Beskorovajnyj, V. V. Systemological analysis of the problem of structural synthesis of geographically distributed systems [Текст] / V. V. Beskorovajnyj // Automated control systems and automation devices. - 2002. - No. 120. - P. 29–37.

32. Sander T. Modeling Object-Oriented Software for Reverse Engineering and Refactoring [Текст] / T. Sander // Thesis. - 2001. - University of Bern, Bern, Switzerland. - P. 142 – 149.

33. Ducasse S. Rŕetro-Conception d'Application `a Objets Reengineering ObjectOriented Applications [Текст] / S. Ducasse // Universitŕe Pierre et Marie Curie. - Paris. - 2001. - P. 81–93.

34. Покоюк А., Програмне забезпечення для станції технічного обслуговування автомобілів // Інформаційні технології та моделювання систем : Безпека, технології, інновації: нові горизонти : збірник праць учасників міжфакультетської науково-практичної інтернет-конференції здобувачів вищої освіти і молодих вчених, 15 листопада 2023 р. – Житомир : Поліський національний університет, 2023. – 69 с. – С. 57–58.

35. Покоюк А., Інформаційна система станції технічного обслуговування автомобілів // Інформаційні технології та моделювання систем : збірник праць учасників Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих вчених, 10 квітня 2024 р. Житомир : Поліський національний університет, 2024. 76 с. – С. 29–30.

Додаток А

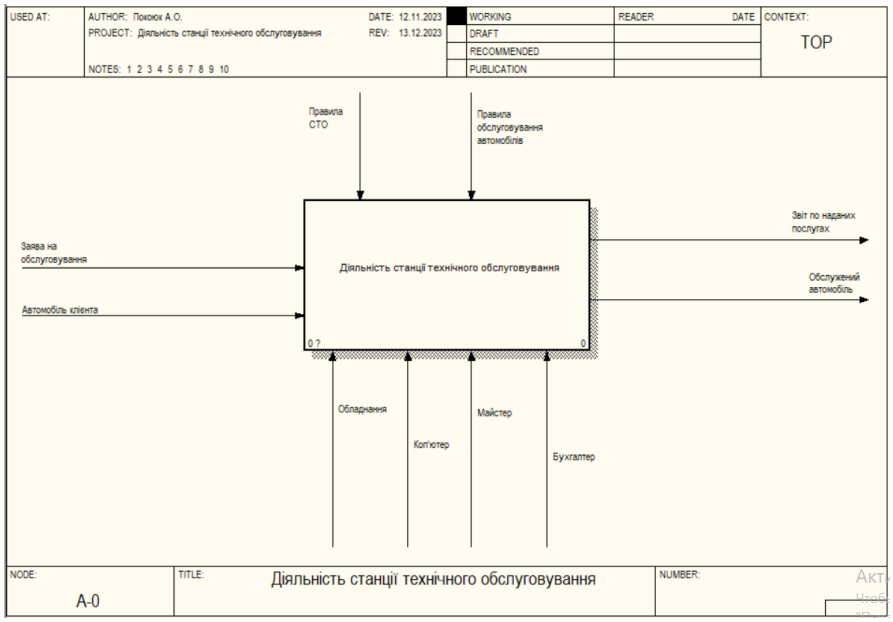

Рисунок 1.3 – Контекстна діаграма процесу «Діяльність станції технічного обслуговування»

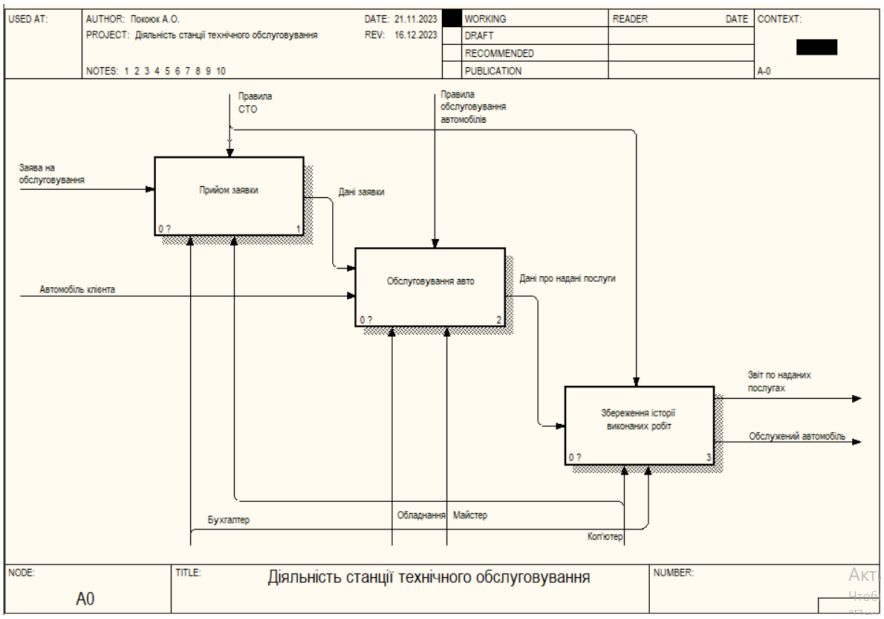

Рисунок 1.4 – Декомпозиція контекстної діаграми процесу «Діяльність станції технічного обслуговування»

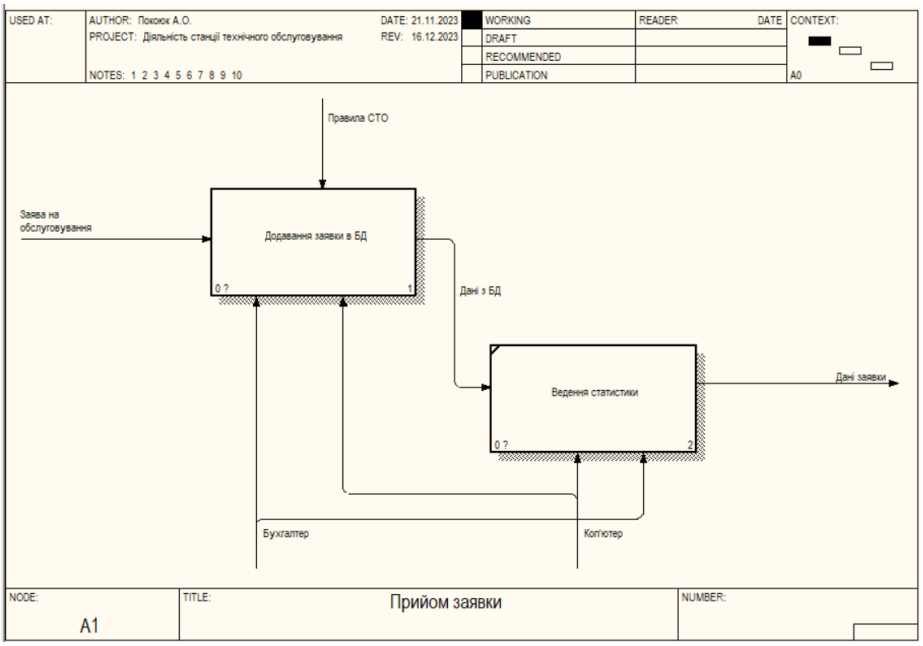

Рисунок 1.5 – Декомпозиція контекстної діаграми процесу «Прийом заявки»

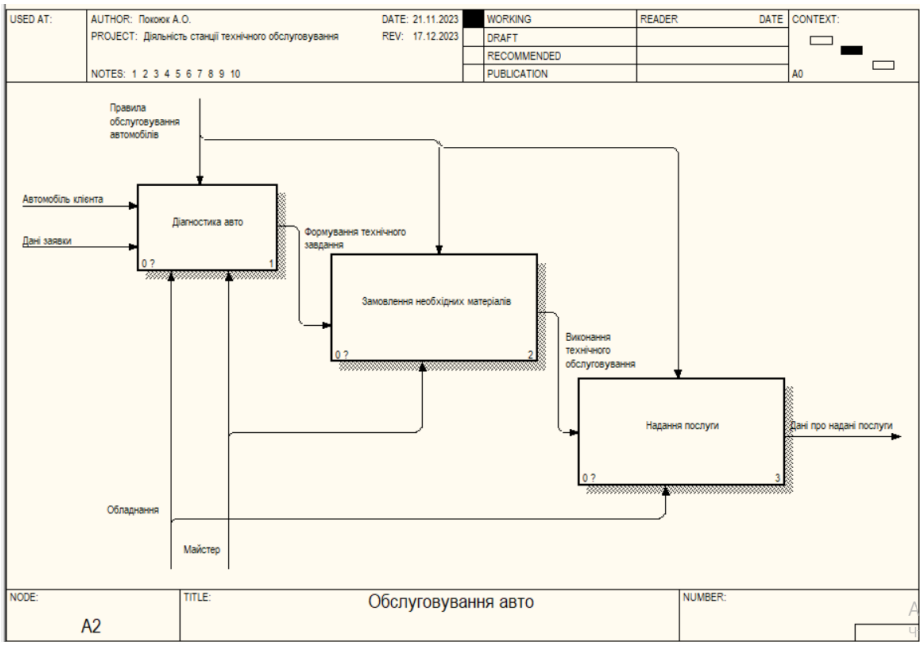

Рисунок 1.6 – Декомпозиція процесу «Обслуговування авто»

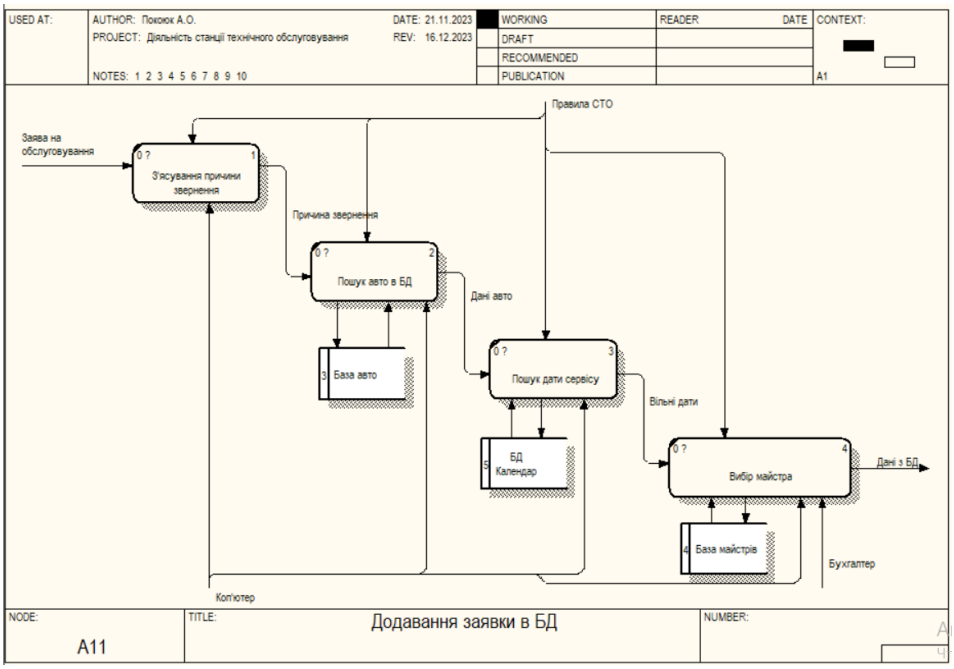

Рисунок 1.7 – DFD «Додавання заявки в БД»

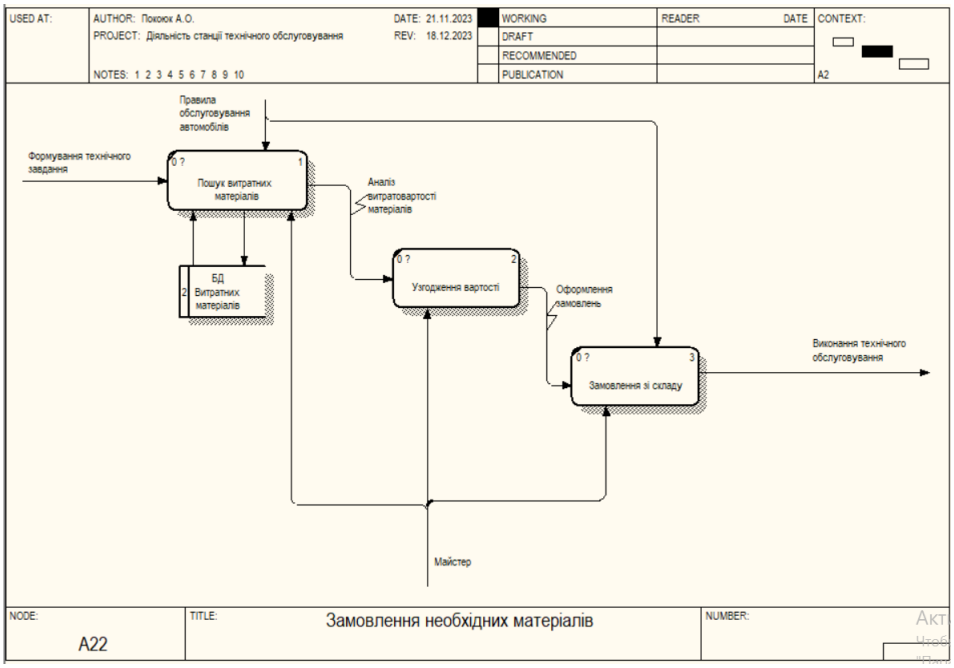

Рисунок 1.8 – Опис процесу «Замовлення необхідних матеріалів»

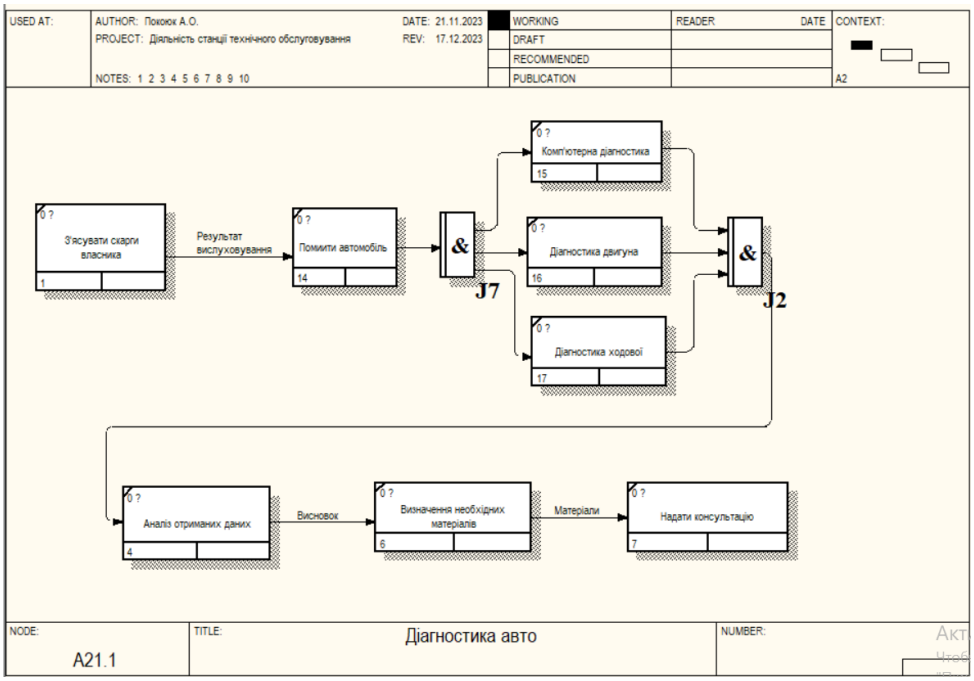

Рисунок 1.9 – IDEF3 «Діагностика авто»

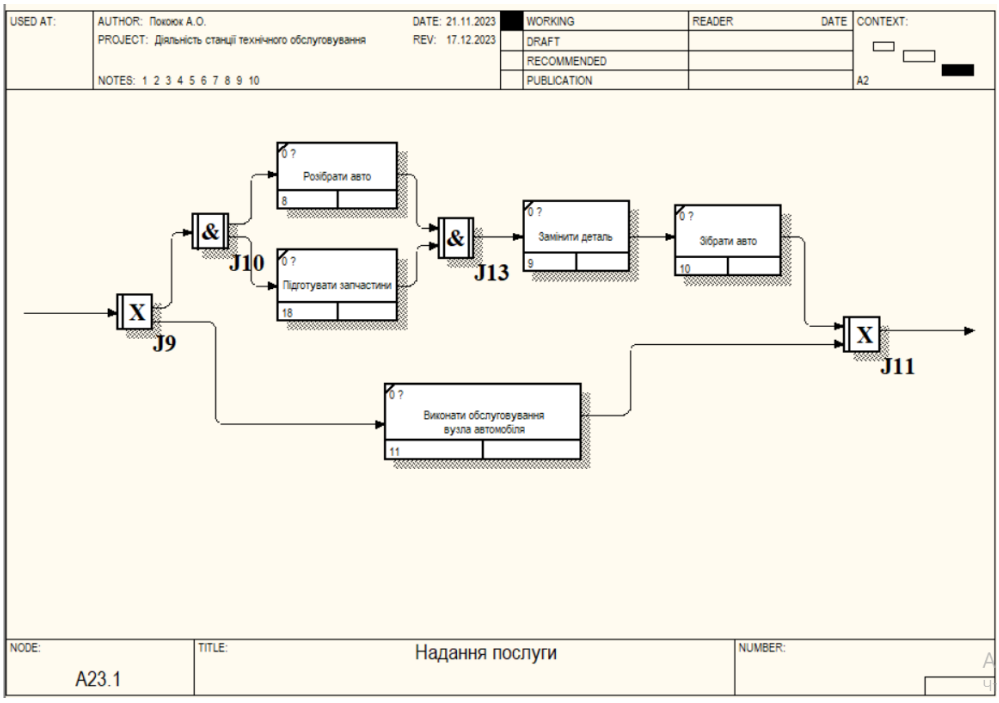

Рисунок 1.10 – Декомпозиція процесу «Надання послуги»

Додаток Б

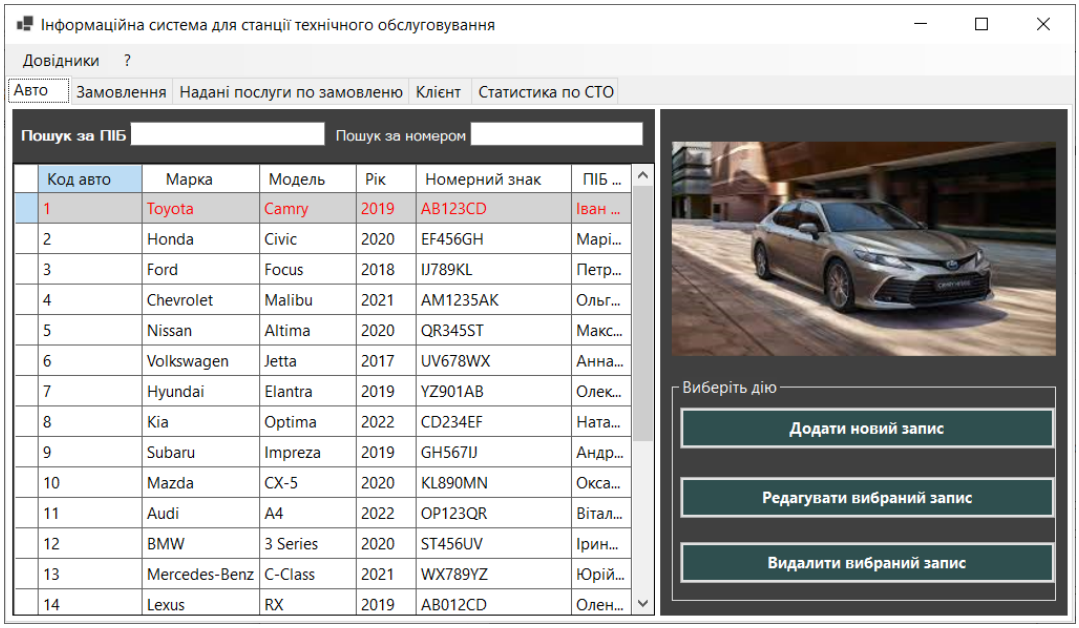

Рисунок 3.1 – Інтерфейс програми на вкладці «Авто»

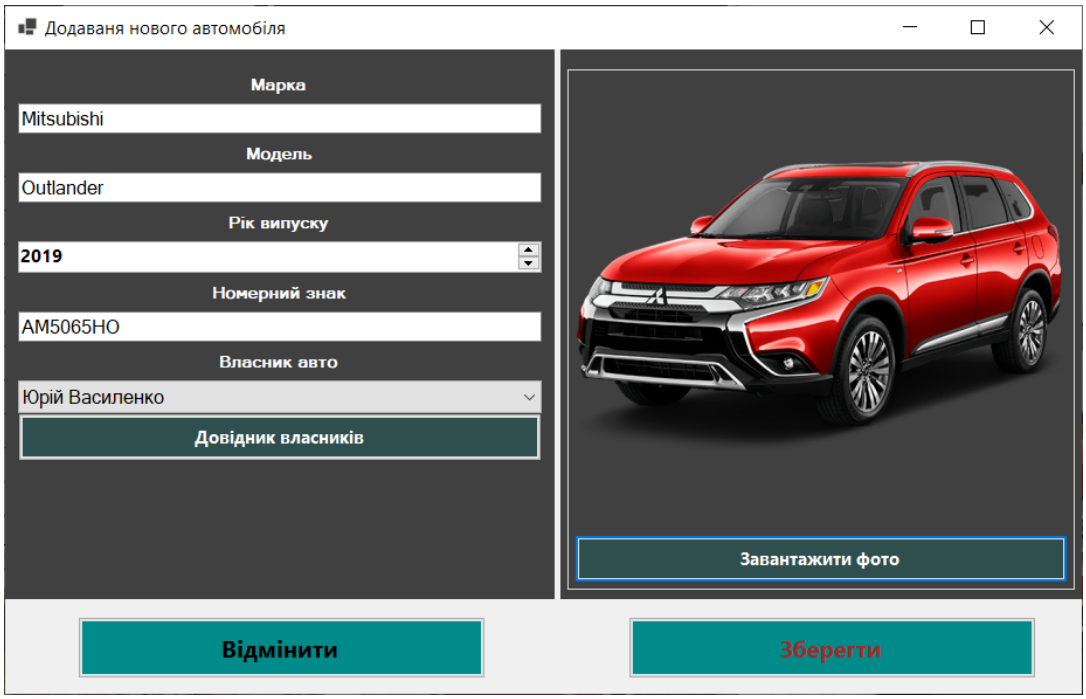

Рисунок 3.2 – Форма «Додавання нового автомобіля»

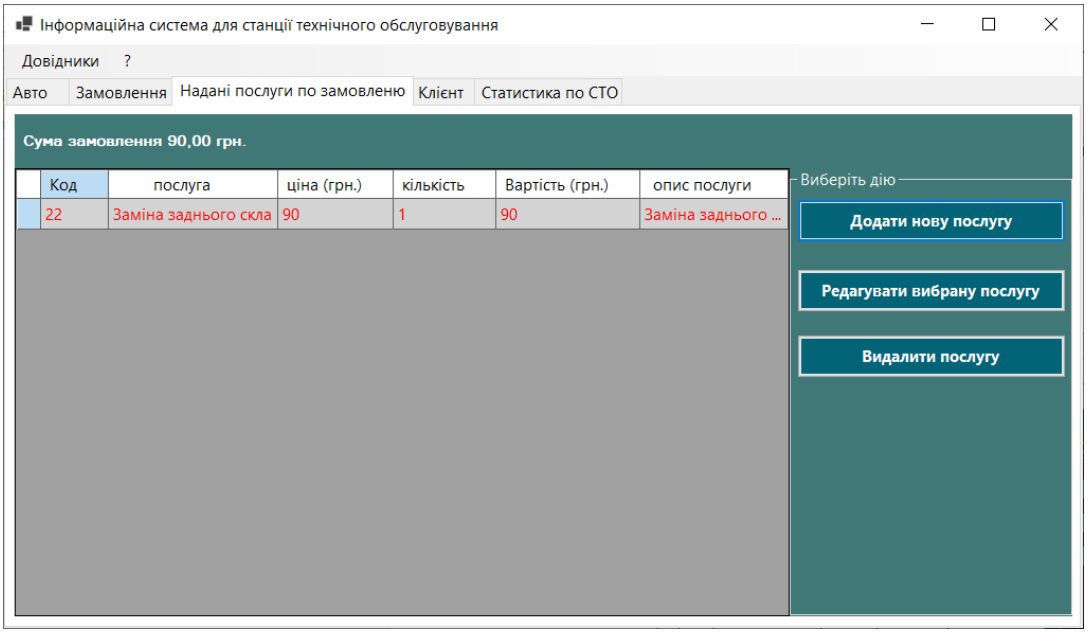

Рисунок 3.3 – Інтерфейс програми на вкладці «Надані послуги по замовленню»

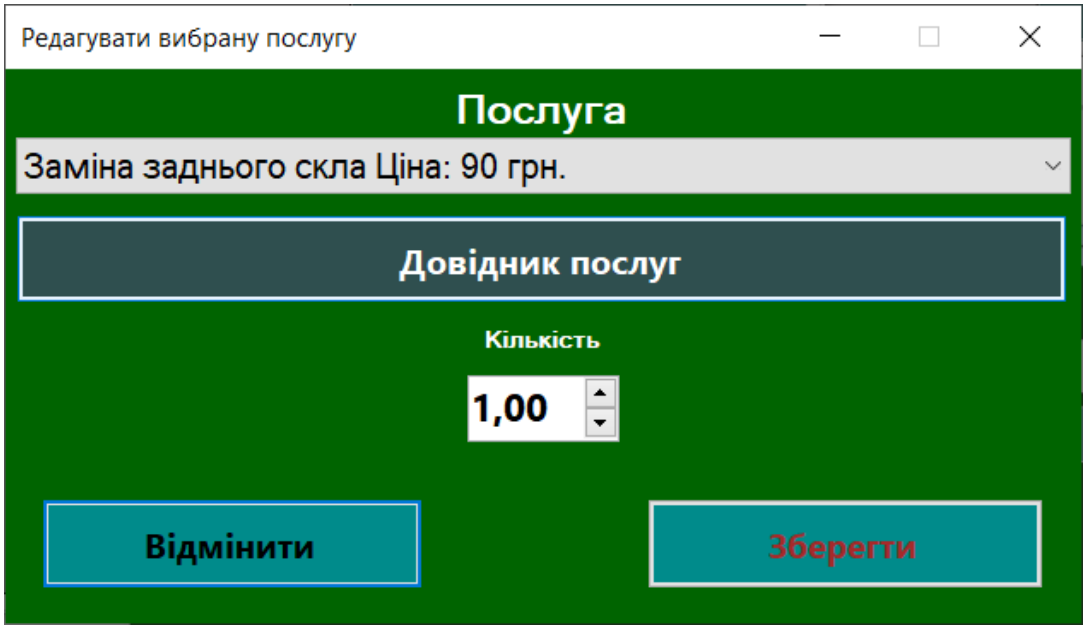

Рисунок 3.4 – Форма «Додати нову послугу»

Рисунок 3.5 – Форма «Про програму»

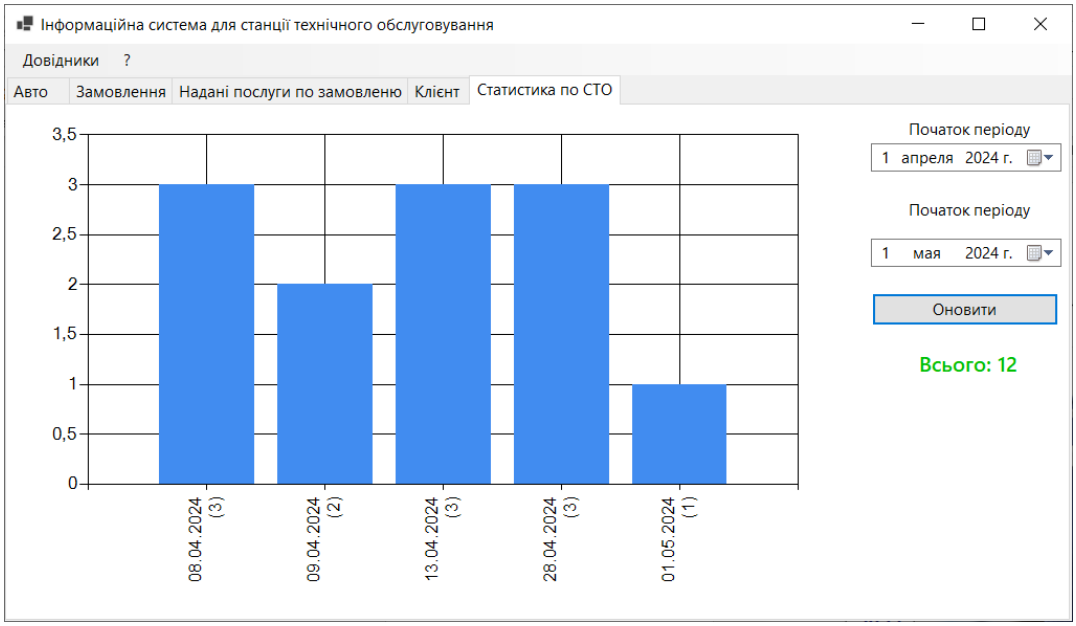

Рисунок 3.6 – Форма «Статистика по СТО»

Додаток В

```
using System.Data.SQLite;
using System.Data;
using workfile;
using ServiceStation.Service;
using System.Data.SqlClient;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Drawing;
using System.Windows.Forms.DataVisualization.Charting;
using System.Windows.Forms;

namespace ServiceStation
{
    public partial class MainWindow : Form
    {
        //   SQLiteConnection? connection;

        SQLiteDataAdapter? dataAdapterCar;
        DataSet DSetCars = new DataSet();
        DataTable DTableCars = new DataTable();
        DataSet DSetOrderService = new DataSet();
        DataSet DSetOrder = new DataSet();



        int OldPozCar = 0;
        int OldPosOrder = 0;
        int OldPosOrderService = 0;

        public MainWindow()
        {
            InitializeComponent();
            dateTimePickerStart.Value = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 1);

            DateTime firstDayOfNextMonth = new DateTime(DateTime.Now.Year,
DateTime.Now.Month, 1).AddMonths(1);
            DateTime lastDayOfThisMonth = firstDayOfNextMonth.AddDays(-1);
            dateTimePickerEnd.Value = lastDayOfThisMonth;

        }
```

```
        void SettingGrid(DataGridView Grid)
        {
            foreach (DataGridViewColumn column in Grid.Columns)
                column.AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            Grid.Columns[0].ReadOnly = true;
            Grid.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.DisplayedCells;
            Grid.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
            Grid.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
            Grid.RowHeadersWidth = 24;
            Grid.ColumnHeadersDefaultCellStyle.WrapMode =
DataGridViewTriState.False;

            Grid.Columns[Grid.Columns.Count - 1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
            Grid.DefaultCellStyle.SelectionBackColor = Color.LightGray;
            Grid.DefaultCellStyle.SelectionForeColor = Color.Red;
            //Grid.RowHeadersVisible = false;
            Grid.AllowUserToAddRows = false;
            Grid.ReadOnly = true;
            Grid.AutoResizeColumns();
        }
        int curPositionGrid(DataGridView Grid)
        {
            int row = -1;
            if (Grid.DataSource != null)
                row = Grid.BindingContext[Grid.DataSource].Position;
            return row;
        }


        private int GetId(string query)
        {
            int id = -1;
            var connection = Set_conect();
            if (connection == null) { return id; }
            SQLiteCommand command = connection.CreateCommand();
            command.CommandText = query;
```

```
            // Виконання SQL-запиту і отримання результату
            object result = command.ExecuteScalar();
            // Перевірка на null та конвертація в int
            if (result != null && result != DBNull.Value)
                id = Convert.ToInt32(result);
            connection.Close();
            return id;
        }

        private double GetDouble(string query)
        {
            double value = 0;
            var connection = Set_conect();
            if (connection == null) { return value; }
            SQLiteCommand command = connection.CreateCommand();
            command.CommandText = query;
            // Виконання SQL-запиту і отримання результату
            object result = command.ExecuteScalar();
            // Перевірка на null та конвертація в int
            if (result != null && result != DBNull.Value)
                value = Convert.ToDouble(result);
            connection.Close();
            return value;
        }


        int IdClient => GetId($"SELECT client_id FROM Vehicles WHERE
vehicle_id = {IdVehicles}");
        double GetAllPrice => GetDouble($"SELECT SUM(price* quantity) FROM
Order_Services " +
                                        $"JOIN  Services ON Services.service_id
= Order_Services.service_id " +
                                        $"WHERE order_id = {IdOrder}");

        int GetIdFromGrid(DataGridView dataGrid)
        {
            int ID = -1;
            if (dataGrid.SelectedRows.Count > 0)
            {
                DataGridViewRow selectedRow = dataGrid.SelectedRows[0];
                if (selectedRow != null)
```

```
                    {
                        int.TryParse(selectedRow.Cells[0].Value.ToString(), out
ID);
                    }
                }
                return ID;
            }

            int IdVehicles => GetIdFromGrid(DGCars);

            int IdOrder => GetIdFromGrid(dataGridOrder);

            int IdOrderService => GetIdFromGrid(dataGridOrderService);


            private void LoadDataVehicle(string filtrPIB = "", string
filtrNumber = "")
            {
                var connection = Set_conect();
                if (connection == null) { return; }
                SQLiteCommand command = connection.CreateCommand();
                string query =
                            @"SELECT
                                vehicle_id as 'Код авто',
                                brand as 'Марка',
                                model as 'Модель',
                                year as 'Рік',
                                license_plate as 'Номерний знак',
                                Clients.Name as 'ПІБ Клієнта'
                            FROM
                                Vehicles
                            JOIN
                                Clients ON Vehicles.client_id =
Clients.client_id" +
                            $" WHERE Clients.Name Like '%{filtrPIB}%' and
license_plate Like '%{filtrNumber}%'";

                command.CommandText = query;
                dataAdapterCar = new SQLiteDataAdapter(command.CommandText,
connection);
                DSetCars.Reset();
```

```
                dataAdapterCar.Fill(DSetCars);
                DTableCars = DSetCars.Tables[0];
                DGCars.DataSource = DTableCars;
                connection.Close();
                SettingGrid(DGCars);

                if (DGCars.DataSource != null)
                    DGCars.BindingContext[DGCars.DataSource].Position =
OldPozCar;

            }

            private void LoadDataOrders(string filtrByDateOrder = "")
            {

                var connection = Set_conect();
                if (connection == null) { return; }
                SQLiteCommand command = connection.CreateCommand();
                string query = @"
      SELECT
            order_id as 'Код замовлення',
            order_date_begin as 'Дата замовлення',
            order_date_end as 'Остання зміна статусу',
            Status.status_name as 'Статус замовлення',
            Employees.name as 'ПІБ Механіка'
      FROM
            Orders
      JOIN
            Employees ON Employees.employee_id = Orders.employee_id
      JOIN
            Status ON Status.id = Orders.status_id
      " + $" WHERE vehicle_id = {IdVehicles} and strftime('%d.%m.%Y',
order_date_begin) Like '%{filtrByDateOrder}%'";

                command.CommandText = query;
                var dataAdapter = new SQLiteDataAdapter(command.CommandText,
connection);
                DSetOrder.Reset();
                dataAdapter.Fill(DSetOrder);
                dataGridOrder.DataSource = DSetOrder.Tables[0];
                connection.Close();
```

```
                SettingGrid(dataGridOrder);

                if (dataGridOrder.DataSource != null)

dataGridOrder.BindingContext[dataGridOrder.DataSource].Position = OldPosOrder;

            }

            private void LoadDataOrdersService()
            {
                labeAllPriceOrder.Text = $"Сума замовлення {GetAllPrice:N2}
грн.";

                var connection = Set_conect();
                if (connection == null) { return; }
                SQLiteCommand command = connection.CreateCommand();
                string query = @"
      SELECT
          order_service_id as 'Код',
          service_name as 'послуга',
          price as 'ціна (грн.)',
          quantity as 'кількість',
          price* quantity as 'Вартість (грн.)',
          service_description  as 'опис послуги'
      FROM
          Order_Services
      JOIN
          Services ON Services.service_id = Order_Services.service_id
      WHERE "
      + $" order_id = {IdOrder}";

                command.CommandText = query;
                var dataAdapter = new SQLiteDataAdapter(command.CommandText,
connection);
                DSetOrderService.Reset();
                dataAdapter.Fill(DSetOrderService);
                dataGridOrderService.DataSource = DSetOrderService.Tables[0];


                connection.Close();
                SettingGrid(dataGridOrderService);
```

```
                if (dataGridOrderService.DataSource != null)

dataGridOrderService.BindingContext[dataGridOrderService.DataSource].Position =
OldPosOrderService;

            }

            private void filterByNameOrNumber_TextChanged(object sender,
EventArgs e)
            {
                LoadDataVehicle(textBoxFiltrByName.Text,
textBoxFiltrByNumber.Text);
            }

            private void tabControl1_SelectedIndexChanged(object sender,
EventArgs e)
            {

                if (tabControl1.SelectedTab == tabPageClients)
                {
                    LoadDataClients();
                }
                else if (tabControl1.SelectedTab == tabPageOrder)
                {
                    LoadDataOrders(textBoxFilderDateBegin.Text);
                }
                else if (tabControl1.SelectedTab == tabPageService)
                {
                    LoadDataOrdersService();
                }
                else if (tabControl1.SelectedTab == tabPageStatistics)
                {
                    buttonRefreshStatistic_Click(sender, e);
                }


            }
            private void LoadDataClients()
            {
                int id = IdClient;
                var connection = Set_conect();
```

```
                if (connection is null) { return; }
                SQLiteCommand command = connection.CreateCommand();
                command.CommandText = $"SELECT name, address,  phone,  email
FROM Clients WHERE client_id ='{id}'";
                using (var reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        textBoxClientName.Text = reader.GetString("name");
                        textBoxClientAddress.Text = reader.GetString("address");
                        textBoxClientPhone.Text = reader.GetString("phone");
                        textBoxClientEmail.Text = reader.GetString("email");
                    }
                }
                connection.Close();
                LoadImageFromBaseToPicterBox(pictureBoxPhotoClient, $"SELECT
image FROM Clients WHERE client_id ='{id}'");
            }

            private SQLiteConnection? Set_conect()
            {
                SQLiteConnection? connection = null;
                try
                {
                    connection = new SQLiteConnection("Data
Source=base_service.db; Version=3;");
                    connection.Open();
                }
                catch (Exception ex)
                {

                    MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    Close();
                }
                return connection;
            }

            private void toolStripMenuItemAbout_Click(object sender, EventArgs
e)
            {
```

```
            About about = new About();
            about.ShowDialog();

        }


        private void співробітникиToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            using (var infoEmployees = new InfoEmployees())
            {
                infoEmployees.ShowDialog();
            };
        }

        private void ClientsToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            using (var infoClients = new InfoClients())
            {
                infoClients.ShowDialog();
            };
        }

        private void MainWindow_Load(object sender, EventArgs e)
        {
            TestTime.TestNow(1724962184);
            LoadDataVehicle();
            if (DGCars.DataSource != null)
            {
                DGCars.Rows[0].Selected = true;
            }
        }




        private void DGCars_SelectionChanged(object sender, EventArgs e)
        {
            LoadImageFromBaseToPicterBox(pictureBoxPhoto, $"SELECT image
FROM Vehicles WHERE vehicle_id = {IdVehicles}");
```

```
            }


            private void LoadImageFromBaseToPicterBox(PictureBox pictureBox,
string selectQuery)
            {

                try
                {

                    var connection = Set_conect();
                    if (connection == null) { return; }
                    using (SQLiteCommand command = new
SQLiteCommand(selectQuery, connection))
                    {
                        var res = command.ExecuteScalar();
                        if (res is byte[] photoByte)
                            pictureBox.Image =
WorkWithImage.ByteToImage(photoByte);
                        else
                            pictureBox.Image = WorkWithImage.NoPhoto();
                    }
                    connection.Close();
                }
                catch (Exception)
                {
                    pictureBox.Image = WorkWithImage.NoPhoto();
                }


            }

            private void buttonNew_Click(object sender, EventArgs e)
            {
                using (var editVehicle = new EditVehicle(-1))
                {
                    editVehicle.Text = "Додаваня нового автомобіля";

                    if (editVehicle.ShowDialog() == DialogResult.Yes)
                    {
```

```
                        LoadDataVehicle(textBoxFiltrByName.Text,
textBoxFiltrByNumber.Text);
                        if (DGCars.Rows.Count > 0)
                        {
                            DGCars.ClearSelection();
                            DGCars.Rows[DGCars.Rows.Count - 1].Selected = true;
                            DGCars.FirstDisplayedScrollingRowIndex =
DGCars.Rows.Count - 1;
                        }

                    }
                }
            }

            private void btnEdit_Click(object sender, EventArgs e)
            {
                int rowSelect = curPositionGrid(DGCars);
                int indexSelect = IdVehicles;
                if (indexSelect == -1)
                {
                    MessageBox.Show("Не вибрано запис для редагування",
"Редагування", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    return;
                }
                using (var childFormDirect = new EditVehicle(indexSelect))
                {
                    childFormDirect.Text = "Редагування вибраного автомобіля";
                    if (childFormDirect.ShowDialog() == DialogResult.Yes)
                    {
                        LoadDataVehicle(textBoxFiltrByName.Text,
textBoxFiltrByNumber.Text);
                        if (DGCars.Rows.Count > 0)
                        {
                            DGCars.ClearSelection();
                            DGCars.Rows[rowSelect].Selected = true;
                            DGCars.FirstDisplayedScrollingRowIndex = rowSelect;
                        }

                    }
                }
            }
```

```
        private void btnDelete_Click(object sender, EventArgs e)
        {
            int indexDel = IdVehicles;

            if (indexDel != -1)
            {
                if (MessageBox.Show("Видалити запис", "Видалення",
MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2)
== DialogResult.No)
                    return;
                var connection = Set_conect();
                if (connection == null) { return; }
                SQLiteCommand command = connection.CreateCommand();
                command.CommandText = $"DELETE FROM Vehicles WHERE
vehicle_id = '{indexDel}'";
                object result = command.ExecuteScalar();
                connection.Close();
                if (result is int)
                {
                    MessageBox.Show("Запис видалено", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                LoadDataVehicle(textBoxFiltrByName.Text,
textBoxFiltrByNumber.Text);

            }
            else MessageBox.Show("Не вибрано рядок для видалення",
"Видалення", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        private void PositionToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            using (Directories childFormDirect = new Directories())
            {
                childFormDirect.Text = "Довідник \"Посади\"";
                childFormDirect.Query = "SELECT id as 'Код посади', name as
'Назва посади'  FROM Position";
                childFormDirect.ShowDialog();
            };
```

```
        }

        private void StatusToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            using (Directories childFormDirect = new Directories())
            {
                childFormDirect.Text = "Довідник \"Статуси замовлення\"";
                childFormDirect.Query = "SELECT id as 'Код статусу',
status_name as 'Статус замовлення'  FROM Status";
                childFormDirect.ShowDialog();
            };
        }

        private void ServicesToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            using (Directories childFormDirect = new Directories())
            {
                childFormDirect.Text = "Довідник \"Послуги\"";
                childFormDirect.Query = "SELECT " +
                    "service_id as 'код статусу', " +
                    "service_name as 'послуга', " +
                    "price as 'ціна (грн.)', " +
                    "service_description as 'опис послуги' " +
                    "  FROM Services";
                childFormDirect.ShowDialog();
            };
        }

        private void textBoxFilterDateBegin_TextChanged(object sender,
EventArgs e)
        {
            LoadDataOrders(textBoxFilderDateBegin.Text);
        }

        private void buttonNewOrder_Click(object sender, EventArgs e)
        {
            var idCar = IdVehicles;
            using (var editFormOrder = new EditOrder(idCar, -1))
            {
```

```
                    editFormOrder.Text = "Додавання нового замовлення";
                    if (editFormOrder.ShowDialog() == DialogResult.Yes)
                    {
                        LoadDataOrders(textBoxFilderDateBegin.Text);
                        if (dataGridOrder.Rows.Count > 0)
                        {
                            dataGridOrder.ClearSelection();
                            dataGridOrder.Rows[dataGridOrder.Rows.Count -
1].Selected = true;
                            dataGridOrder.FirstDisplayedScrollingRowIndex =
dataGridOrder.Rows.Count - 1;
                            OldPosOrder = dataGridOrder.Rows.Count - 1;
                        }
                    }
                }

                private void buttonEditOrder_Click(object sender, EventArgs e)
                {
                    var idCar = IdVehicles;
                    int indexSelect = IdOrder;
                    if (indexSelect == -1)
                    {
                        MessageBox.Show("Не вибрано запис для редагування",
"Редагування", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        return;
                    }
                    using (var editFormOrder = new EditOrder(idCar, IdOrder))
                    {
                        editFormOrder.Text = "Редагування вибраного замовлення";
                        var rowSelect = dataGridOrder.SelectedRows[0].Index;
                        if (editFormOrder.ShowDialog() == DialogResult.Yes)
                        {

                            LoadDataOrders(textBoxFilderDateBegin.Text);
                            if (dataGridOrder.Rows.Count > 0)
                            {
                                dataGridOrder.ClearSelection();
                                dataGridOrder.Rows[rowSelect].Selected = true;
                                dataGridOrder.FirstDisplayedScrollingRowIndex =
rowSelect;
```

```
                }

            }
        }
    }

    private void buttonDeleteOrder_Click(object sender, EventArgs e)
    {

        int indexDel = IdOrder;

        if (indexDel != -1)
        {
            if (MessageBox.Show("Видалити запис", "Видалення",
MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2)
== DialogResult.No)
                return;
            var connection = Set_conect();
            if (connection == null) { return; }
            SQLiteCommand command = connection.CreateCommand();
            command.CommandText = $"DELETE FROM Orders WHERE order_id =
'{indexDel}'";
            object result = command.ExecuteScalar();
            connection.Close();
            if (result is int)
            {
                MessageBox.Show("Запис видалено", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            LoadDataOrders(textBoxFilderDateBegin.Text);

        }
        else MessageBox.Show("Не вибрано рядок для видалення",
"Видалення", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    private void buttonNewService_Click(object sender, EventArgs e)
    {

        var idOrder = IdOrder;
        var idOrderService = IdOrderService;
```

```
                if (IdOrder == -1)
                {
                    MessageBox.Show("Не вибрано замовлення", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    return;
                }
                using (var editOrderService = new EditOrderService(-1, IdOrder))
                {
                    editOrderService.Text = "Додати нову послугу";
                    // var rowSelect =
dataGridOrderService.SelectedRows[0].Index;
                    if (editOrderService.ShowDialog() == DialogResult.Yes)
                    {

                        LoadDataOrdersService();
                        if (dataGridOrderService.Rows.Count > 0)
                        {
                            dataGridOrderService.ClearSelection();

dataGridOrderService.Rows[dataGridOrderService.Rows.Count - 1].Selected = true;
                            dataGridOrderService.FirstDisplayedScrollingRowIndex
= dataGridOrderService.Rows.Count - 1;
                            OldPosOrderService = dataGridOrderService.Rows.Count
- 1;
                        }

                    }
                }
            }

            private void buttonEditService_Click(object sender, EventArgs e)
            {
                var idOrder = IdOrder;
                var idOrderService = IdOrderService;
                if (idOrderService == -1)
                {
                    MessageBox.Show("Не вибрано запис для редагування",
"Редагування", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    return;
                }
```

```
                using (var editOrderService = new
EditOrderService(idOrderService, idOrder))
                {
                    editOrderService.Text = "Редагувати вибрану послугу";
                    var rowSelect = dataGridOrderService.SelectedRows[0].Index;
                    if (editOrderService.ShowDialog() == DialogResult.Yes)
                    {

                        LoadDataOrdersService();
                        if (dataGridOrderService.Rows.Count > 0)
                        {
                            dataGridOrderService.ClearSelection();
                            dataGridOrderService.Rows[rowSelect].Selected =
true;
                            dataGridOrderService.FirstDisplayedScrollingRowIndex
= rowSelect;
                        }

                    }
                }
            }

            private void buttonDeleteService_Click(object sender, EventArgs e)
            {
                int indexDel = IdOrderService;

                if (indexDel != -1)
                {
                    if (MessageBox.Show("Видалити запис", "Видалення",
MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2)
== DialogResult.No)
                        return;
                    var connection = Set_conect();
                    if (connection == null) { return; }
                    SQLiteCommand command = connection.CreateCommand();
                    command.CommandText = $"DELETE FROM Order_Services WHERE
order_service_id  = '{indexDel}'";
                    object result = command.ExecuteScalar();
                    connection.Close();
                    if (result is int)
                    {
```

```
                        MessageBox.Show("Запис видалено", "Інформація",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    LoadDataOrdersService();

                }
                else MessageBox.Show("Не вибрано рядок для видалення",
"Видалення", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            private void button2_Click1(object sender, EventArgs e)
            {
                var connection = Set_conect();
                if (connection == null) { return; }
                SQLiteCommand command = connection.CreateCommand();
                string query = "SELECT order_date_begin as date,count(*) as
couter" +
                    "  FROM Orders group by order_date_begin";
                command.CommandText = query;
                SQLiteDataAdapter da = new
SQLiteDataAdapter(command.CommandText, connection);
                DataSet ds = new DataSet();
                da.Fill(ds);

                this.chart1.DataSource = ds.Tables[0];

                //Mapping a field with x-value of chart
                this.chart1.Series[0].XValueMember = "date";

                //Mapping a field with y-value of Chart
                this.chart1.Series[0].YValueMembers = "couter";

                //Bind the DataTable with Chart
                this.chart1.DataBind();

                connection.Close();

            }
```

```
            private void buttonRefreshStatistic_Click(object sender, EventArgs
e)
            {
                var connection = Set_conect();
                if (connection == null) { return; }
                SQLiteCommand command = connection.CreateCommand();
                string query = "SELECT order_date_begin as date, count(*) as
couter " +
                    "FROM Orders " +
                     $"WHERE order_date_begin BETWEEN
'{dateTimePickerStart.Value:yyyy-MM-dd}' and  '{dateTimePickerEnd.Value:yyyy-MM-
dd}' " +
                    "group by order_date_begin ";
                command.CommandText = query;
                var reader = command.ExecuteReader();
                int i = 0;
                chart1.Series[0].Points.Clear();
                while (reader.Read())
                {
                    DateTime date = reader.GetDateTime(0);
                    Int64 counter = reader.GetInt64(1);
                    DataPoint point = new DataPoint(i, counter);
                    point.AxisLabel = $"{date:d}\n({counter})";
                    //point.AxisLabel = $"123456666 -{counter}";
                    chart1.Series[0].Points.Add(point);
                    i++;
                }
                connection.Close();
                chart1.ChartAreas[0].AxisX.LabelStyle.Angle = -90;
                chart1.ChartAreas[0].AxisX.LabelStyle.Font = new Font("Arial",
10);
                //
                labelAllStatistic.Text = "Всього: "+ GetId($"SELECT count(*) as
couter FROM Orders  " +
                    $"WHERE order_date_begin BETWEEN
'{dateTimePickerStart.Value:yyyy-MM-dd}' and  '{dateTimePickerEnd.Value:yyyy-MM-
dd}' ");

            }
        }
    }
```