МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
ВІДДІЛЕННЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»
ЦИКЛОВА КОМІСІЯ СПЕЦІАЛЬНОСТІ «КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи
освітньо-професійний ступінь «фаховий молодший бакалавр»
на тему:

«Розробка прикладної програми для станції технічного обслуговування»

Виконав студент (ка) _4_ курсу, групи П-41
спеціальності 122 «Комп'ютерні науки»

Лушкін Владислав Валентинович

Керівник Устименко Ярослав Іванович

Рецензент _____

Житомир – 2024 року

Зміст

РЕФЕРАТ

Записка: 53 стор., 34 рис., 1 додаток, 20 джерел.

Ключові слова: СТАНЦІЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ, СТО, ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗАЦІЯ, УПРАВЛІННЯ, БАЗИ ДАНИХ, MICROSOFT VISUAL STUDIO, WINDOWS FORMS, КЛІЄНТИ, АВТОМОБІЛІ, ЗАМОВЛЕННЯ, ПОСЛУГИ, СПІВРОБІТНИКИ, ОПТИМІЗАЦІЯ, ЕФЕКТИВНІСТЬ, НОРМАЛІЗАЦІЯ ДАНИХ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, ТЕСТУВАННЯ, ОБСЛУГОВУВАННЯ, ПРОДУКТИВНІСТЬ.

Дипломна робота на тему "Розробка прикладної програми для станції технічного обслуговування" спрямована на створення програмного забезпечення для оптимізації робочих процесів на станціях технічного обслуговування (СТО). Основною метою є автоматизація управління та координації робіт, що значно підвищить ефективність та продуктивність СТО.

Робота розпочалася з аналізу бізнес-процесів СТО та виявлення основних вимог до інформаційної системи. Було визначено необхідність у створенні системи, яка б дозволяла зручно керувати клієнтами, обліковувати автомобілі, планувати та контролювати виконання замовлень, вести фінансовий облік та забезпечувати безпеку даних.

Проектування бази даних стало наступним важливим етапом. Було створено модель даних, яка включає основні сутності: клієнти, автомобілі, співробітники, замовлення, послуги. Всі дані були нормалізовані до третьої нормальної форми (3NF) для уникнення надлишковості та забезпечення цілісності.

Для розробки програмного забезпечення було обрано Microsoft Visual Studio та технологію Windows Forms. Це забезпечило створення зручного та інтуїтивно зрозумілого інтерфейсу користувача. Програма містить кілька вкладок для управління автомобілями, замовленнями, послугами та співробітниками.

Було реалізовано функціонал для додавання, редагування та видалення записів у різних довідниках. Програма також дозволяє здійснювати фільтрацію та пошук даних, відображати статистику та генерувати звіти.

Тестування програмного забезпечення дозволило виявити та виправити помилки, оптимізувати роботу програми та покращити інтерфейс користувача. Впровадження розробленої системи забезпечить централізоване управління всіма аспектами роботи СТО.

Завдяки новій системі, СТО зможе значно зменшити час на обробку замовлень, покращити організацію робочих процесів, підвищити якість обслуговування клієнтів та ефективність роботи загалом. Це, в свою чергу, сприятиме зростанню задоволеності клієнтів та покращенню репутації станції технічного обслуговування.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1NF	Перша нормальна форма
2NF	Друга нормальна форма
3NF	Третя нормальна форма
CRM	Customer relationship management (управління відносинами з клієнтами)
IDEF1X	Integration DEFinition for information modeling
IP	Internet protocol (Інтернет протокол)
SMS	Short message service (служба коротких повідомлень)
ПЗ	Програмне забезпечення
СТО	Станція програмного забезпечення
ТМЦ	Товарно-матеріальні цінності

ВСТУП

Сучасний світ, насичений швидкістю та динамікою, вимагає відповідного підходу до всіх сфер діяльності, включаючи технічне обслуговування. З точки зору ефективності та точності, станції технічного обслуговування не є винятком. Саме тому розробка та впровадження інформаційної системи, спрямованої на оптимізацію процесів обслуговування техніки, стає надзвичайно важливою.

Мета даної роботи полягає у створенні оптимальної інформаційної системи, яка спростить та покращить управління та координацію процесів станції технічного обслуговування автомобілів.

Об’єктом нашого дослідження є сам процес технічного обслуговування, а предметом - створення інформаційної системи, що полегшить цей процес.

Впровадження такої інформаційної системи має потенціал прискорити обслуговування, зменшити витрати на ресурси та забезпечити більшу точність у веденні документації та контролі за роботою. Це відкриє нові можливості для ефективного функціонування станцій технічного обслуговування в умовах сучасного технологічного прогресу.

Сама станція технічного обслуговування є ключовим елементом в утриманні та функціонуванні технічного обладнання. Її успішна робота базується на комплексі теоретичних знань та практичних навичок. До теоретичних засад роботи станції відносяться технічні аспекти, управління технічним обслуговуванням, використання інформаційних технологій, питання безпеки та екології, стратегічне планування та аспекти комунікації та співпраці.

Отже, розробка та впровадження інформаційної системи для оптимізації процесів технічного обслуговування та підвищення ефективності станцій є кроком у майбутнє, що відкриває нові можливості для розвитку та удосконалення цієї галузі.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробки системи технічного обслуговування передбачає створення інтегрованої інформаційної платформи, яка спростить та оптимізує всі аспекти роботи на станції. Основною метою розробки є забезпечення ефективного управління процесами обслуговування автомобілів, включаючи їх технічний стан, робочі процеси, фінансовий облік та взаємодію з клієнтами.

Ключові аспекти задачі розробки СТО включають:

1. **Управління клієнтами**: Система повинна забезпечувати зручний облік та ведення історії обслуговування клієнтів, управління контактною інформацією та взаємодію з ними через різноманітні канали комунікації.
2. **Управління робочими процесами**: Платформа повинна допомагати в організації графіку роботи, плануванні робіт, контролі за запасами та ефективному розподілі завдань серед працівників.
3. **Фінансовий облік**: Система має забезпечити точний облік витрат на матеріали, робочий час, обладнання та інші фінансові аспекти, а також підтримувати управління фінансами та розрахунок доходів і витрат.
4. **Контроль якості та документування**: СТО повинно мати можливість вести журнали інспекцій, сертифікатів якості та створювати акти обслуговування, що дозволяє забезпечувати високу якість наданих послуг.
5. **Комунікація з клієнтами**: Система повинна підтримувати можливість бронювання часу, надсилання сповіщень про готовність автомобіля, а також забезпечувати підтримку чату або колл-центру для швидкої взаємодії з клієнтами.
6. **Безпека та доступ**: Розроблена платформа повинна гарантувати захист конфіденційної інформації клієнтів та контроль доступу до неї.

Основна задача полягає в створенні інтегрованої та легко налаштовуваної інформаційної системи, яка дозволить станції технічного

обслуговування працювати більш ефективно, забезпечуючи високу якість послуг та задоволення потреб клієнтів.

Визначення потреб станції технічного обслуговування є ключовим етапом у створенні ефективної інформаційної системи. Аналіз потреб включає ретельне вивчення функцій та вимог станції технічного обслуговування до інформаційної системи.

Основні потреби станції технічного обслуговування:

- ведення бази даних клієнтів та їхніх автомобілів.
- збереження історії обслуговування.
- забезпечення ефективної взаємодії з клієнтами.
- організація графіку обслуговування та планування робіт.
- контроль запасів і термінове замовлення деталей.
- моніторинг та управління виконанням завдань.
- ведення обліку витрат на матеріали та робочий час.
- облік витрат на обладнання.
- управління фінансами, включаючи розрахунок доходів і витрат.
- контроль якості та документування:
- ведення журналів інспекцій та сертифікатів якості.
- створення та зберігання актів обслуговування.
- моніторинг відповідності стандартам якості.
- комунікація з клієнтами:
- надання сповіщень про готовність автомобіля.
- забезпечення безпеки та доступу:
- захист даних клієнтів.
- контроль доступу до конфіденційної інформації.
- забезпечення безпеки інформаційної системи від несанкціонованого доступу.

Ці потреби створюють основу для розробки комплексної інформаційної системи, яка допоможе оптимізувати роботу станції технічного

обслуговування, підвищити якість обслуговування клієнтів та забезпечити ефективне управління всіма аспектами її діяльності.

1.2. Огляд та порівняння аналогічних систем.

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

Компанія Soft X: відома своєю експертизою в розробці програмного забезпечення для автосервісів та СТО. Soft X пропонує рішення, що включають управління клієнтами, робочими процесами, фінансовий облік та інші необхідні функції.

Переваги програмного забезпечення від Soft X:

- **Широкий функціонал**: Програмне забезпечення від Soft X має широкий спектр функцій, які включають управління клієнтами, робочими процесами, фінансовий облік, аналітику та інші корисні інструменти. Це дозволяє впоратися з усіма аспектами управління та обслуговування на СТО.
- **Користувацька зручність**: Програмне забезпечення Soft X розроблено з урахуванням потреб користувачів, тому воно має інтуїтивно зрозумілий інтерфейс та легко налаштовується під конкретні вимоги автосервісу.
- **Підтримка клієнтів**: Компанія Soft X надає високоякісну підтримку своїм клієнтам, що допомагає вирішувати будь-які технічні або функціональні питання швидко та ефективно.
- **Постійні оновлення**: Soft X регулярно випускає оновлення та нові версії свого програмного забезпечення, що дозволяє користувачам отримувати доступ до нових функцій та вдосконалень.

- **Безпека даних**: Програмне забезпечення Soft X забезпечує високий рівень безпеки даних, що дозволяє автосервісам зберігати конфіденційну інформацію про своїх клієнтів та робочі процеси.

Недоліки програмного забезпечення від Soft X:

- **Високі витрати**: Ціна програмного забезпечення від Soft X може бути високою, що робить його менш доступним для малих автосервісів або підприємств з обмеженим бюджетом.
- **Необхідність навчання персоналу**: Через широкий функціонал програмного забезпечення Soft X, може знадобитися час на навчання персоналу та адаптацію до нової системи.
- **Залежність від інтернету**: Деякі функції програмного забезпечення можуть вимагати підключення до Інтернету, що може бути недоцільним для автосервісів з обмеженим доступом до мережі.

Компанія Tech Solutions спеціалізується на розробці інформаційних систем для автомобільних сервісних центрів, включаючи СТО. Вони пропонують рішення з автоматизації бізнес-процесів, аналізу даних та управління клієнтами.

Переваги програмного забезпечення від Tech Solutions:

- **Гнучкість налаштувань**: Програмне забезпечення від Tech Solutions має високий рівень гнучкості, що дозволяє користувачам налаштовувати систему під свої потреби та вимоги конкретного автосервісу.
- **Широкий функціонал**: Програмне забезпечення надає широкий спектр функцій для управління клієнтами, робочими процесами, фінансовим обліком та іншими аспектами діяльності автосервісу.
- **Ефективність**: Система розроблена з урахуванням оптимальних робочих процесів та методів управління, що дозволяє підвищити ефективність роботи автосервісу та знизити час виконання завдань.

- **Підтримка клієнтів**: Tech Solutions надає високоякісну підтримку своїм клієнтам, що допомагає вирішувати будь-які технічні або функціональні питання швидко та ефективно.
- **Інтеграція**: Програмне забезпечення може легко інтегруватися з іншими системами управління або бухгалтерськими програмами, що дозволяє забезпечити більшу автоматизацію та зручність в роботі.

Недоліки програмного забезпечення від Tech Solutions:

- **Вартість**: Вартість програмного забезпечення від Tech Solutions може бути високою, що робить його менш доступним для малих автосервісів або підприємств з обмеженим бюджетом.
- **Необхідність навчання персоналу**: Через широкий функціонал програмного забезпечення, може знадобитися час на навчання персоналу та адаптацію до нової системи.
- **Залежність від технічної підтримки**: В разі виникнення проблем або потреби в додаткових налаштуваннях, може знадобитися звернення до технічної підтримки, що може вплинути на продуктивність роботи.

AutoSoft відомий своїми інноваційними рішеннями для автомобільних підприємств, включаючи програмне забезпечення для СТО. Вони пропонують широкий спектр функцій, включаючи управління робочими процесами, складання звітів та аналіз даних.

Переваги програмного забезпечення від AutoSoft:

- **Простота використання**: Програмне забезпечення від AutoSoft має інтуїтивний і легкий у використанні інтерфейс, що дозволяє новим користувачам швидко засвоїти систему та почати працювати з нею.
- **Економія часу**: Завдяки оптимізації робочих процесів та швидкому доступу до необхідної інформації, програмне забезпечення від AutoSoft дозволяє значно зекономити час при веденні обліку та управлінні діяльністю автосервісу.

- **Адаптованість до потреб клієнтів**: AutoSoft надає можливість налаштування системи під конкретні потреби і вимоги кожного автосервісу, що дозволяє забезпечити максимальну відповідність функціоналу програми реальним потребам бізнесу.
- **Вартість**: Порівняно з іншими програмними продуктами на ринку, програмне забезпечення від AutoSoft може мати більш прийнятну вартість, що робить його більш доступним для малих і середніх автосервісів.
- **Підтримка клієнтів**: Компанія AutoSoft надає високоякісну технічну підтримку своїм клієнтам, що допомагає вирішувати будь-які питання та проблеми з використання програмного забезпечення швидко та ефективно.

Недоліки програмного забезпечення від AutoSoft:

- **Обмежений функціонал**: У порівнянні з іншими програмними продуктами, функціонал програмного забезпечення від AutoSoft може бути обмеженим, що може не влаштовувати користувачів з високими вимогами.
- **Масштабованість**: Для великих автосервісів програмне забезпечення може бути недостатньо масштабованим, що може виникнути проблеми з його ефективним використанням в разі збільшення обсягу діяльності.
- **Необхідність регулярних оновлень**: Для підтримки актуальності та безпеки програмне забезпечення вимагає регулярних оновлень, що може створювати додаткові витрати та труднощі для користувачів.

GarageTech: Ця компанія спеціалізується на розробці програмного забезпечення для автосервісів та СТО з урахуванням українських реалій. Вони пропонують рішення, що відповідають потребам малих та середніх підприємств, зокрема управлінням замовленнями, складанням рахунків та обліком клієнтів.

Переваги програмного забезпечення від GarageTech:

- **Широкий функціонал**: Програмне забезпечення від GarageTech може мати широкий набір функцій, що охоплює всі аспекти управління автосервісом, включаючи облік клієнтів, автомобілів, замовлень, фінансовий облік та багато іншого.
- **Гнучкість налаштування**: GarageTech може бути легко налаштований під конкретні потреби та вимоги кожного автосервісу, що дозволяє користувачам налаштовувати програму відповідно до їхніх унікальних потреб.
- **Інтеграція з іншими системами**: Програмне забезпечення від GarageTech може підтримувати інтеграцію з іншими системами, такими як бухгалтерські програми, системи управління запасами та інші, що спрощує обмін даними та підвищує ефективність роботи.
- **Підтримка клієнтів**: Компанія GarageTech може надавати високоякісну підтримку своїм клієнтам, включаючи технічну підтримку та навчання користувачів, що допомагає розв'язувати будь-які питання та проблеми.

Недоліки програмного забезпечення від GarageTech:

- **Висока вартість**: У порівнянні з іншими програмними рішеннями, вартість програмного забезпечення від GarageTech може бути високою, що робить його менш доступним для малих та середніх автосервісів.
- **Складність використання**: Завдяки широкому функціоналу програмне забезпечення може мати складний інтерфейс та вимагати тривалого часу для навчання користувачів.
- **Необхідність обслуговування**: Програмне забезпечення від GarageTech може вимагати регулярного оновлення та обслуговування, що може створювати додаткові витрати та труднощі для користувачів.

Ці компанії не лише надають програмне забезпечення для автосервісних центрів та СТО, але й активно використовують передові технології, такі як хмарні обчислення. Це дозволяє їм створювати інтегровані рішення, які

забезпечують не лише ефективне управління робочими процесами, а й забезпечують доступ до цих процесів з будь-якого місця і в будь-який час.

Інтеграція з хмарними технологіями дає змогу віддалено керувати та моніторити робочі процеси на СТО, що є надзвичайно важливим в сучасному швидкісному світі. Завдяки цьому, власники автосервісних центрів можуть ефективно керувати своїми бізнесами, навіть якщо вони знаходяться далеко від самого центру.

Такі інноваційні рішення дозволяють спростити підтримку та обслуговування автомобілів, забезпечуючи власникам більшу контроль і можливість реагувати на зміни в реальному часі. Вони допомагають збільшити продуктивність та ефективність роботи станцій технічного обслуговування, що є ключовим для задоволення потреб сучасних вимогливих клієнтів.

1.3. Вибір та обґрунтування технологій розробки

Для створення програмного інтерфейсу я використав середовище розробки Microsoft Visual Studio та технологію Windows Forms. Windows Forms є однією з ключових технологій для створення додатків з графічним інтерфейсом в операційній системі Windows. Ця технологія дозволяє програмістам легко створювати віконні додатки за допомогою ряду стандартних елементів керування, таких як кнопки, поля введення, списки, меню тощо, що робить взаємодію з користувачем інтуїтивно зрозумілою.

Windows Forms надає зручний інструментарій для роботи з графікою та подіями, спрощуючи процес створення та розгортання додатків. Ця технологія також дозволяє розробникам створювати різноманітні додатки з різними рівнями складності та функціональністю. Використання Windows Forms у розробці підтримується Microsoft, що забезпечує доступ до широкого спектру інструментів та ресурсів для розробників програмного забезпечення.

Основні переваги використання Windows Forms:

1. **Інтуїтивно зрозумілий інтерфейс**: Завдяки стандартним елементам керування користувачі можуть легко взаємодіяти з додатками.

2. **Широкий набір елементів керування**: Включає кнопки, текстові поля, списки, меню та інші стандартні елементи для створення зручного користувацького інтерфейсу.
3. **Гнучкість та потужність**: Надає інструменти для створення як простих, так і складних додатків з багатим функціоналом.
4. **Підтримка графіки та подій**: Спрощує роботу з графічними елементами та обробку подій користувача.
5. **Інтеграція з Microsoft Visual Studio**: Зручне середовище розробки з багатьма інструментами для підвищення продуктивності програміста.
6. **Підтримка від Microsoft**: Доступ до технічної підтримки та великої кількості ресурсів, таких як документація, форуми та приклади коду.

Таким чином, використання Windows Forms у поєднанні з Microsoft Visual Studio дозволяє швидко та ефективно створювати високоякісні додатки з графічним інтерфейсом для операційної системи Windows.

Висновки до розділу 1

Вибір технологій розробки є критичним кроком у процесі створення ефективної та функціональної інформаційної системи. У моємі випадку, вважаючи особливості бізнесу та вимоги клієнтів, я вирішив скористатися технологією Microsoft Visual Studio та фреймворком Windows Forms для розробки програмного забезпечення.

Цей вибір був обґрунтований кількома чинниками. По-перше, Microsoft Visual Studio є відомим середовищем розробки з великою кількістю інструментів і ресурсів, що сприяє ефективній розробці та налагодженню програмного забезпечення. Воно також підтримується Microsoft, що гарантує стабільність та актуальність інструментів.

Другим важливим аспектом є використання фреймворку Windows Forms. Цей фреймворк дозволяє швидко створювати графічний інтерфейс користувача, що є ключовим для забезпечення зручності та інтуїтивності взаємодії з програмою. Крім того, Windows Forms має велику кількість стандартних елементів керування, які спрощують розробку та забезпечують консистентний дизайн.

Таким чином, обираючи технології Microsoft Visual Studio та Windows Forms для розробки програмного забезпечення для станцій технічного обслуговування, ми максимізуємо швидкість розробки, зручність використання та надійність програми, що відповідає потребам наших клієнтів і дозволить нам забезпечити їм якісні та ефективні рішення.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Проектування база даних

Для створення ефективної інформаційної системи на станції технічного обслуговування необхідно спроектувати модель даних, яка включає сутності, їх атрибути та взаємозв'язки між ними. Нижче наведені сутності з відповідними атрибутами та зв’язки між ними.

Я виділила наступні **сутності** і для кожної сутності я визначив атрибути:

- **Автомобіль.** Атрибути: марка, модель, рік випуску, номер, фото;
- **Клієнт.** Атрибути: ім'я, адреса, телефон, пошта, фото;
- **Співробітник.** Атрибути: ім'я, посада, адреса, телефон, фото;
- **Посада.** Атрибути: назва посади. Співробітник станції технічного обслуговування працює на посаді. За однією посадою може бути багато співробітників.
- **Послуга:** Атрибути: назва послуги, опис послуги, ціна;
- **Замовлення:** Атрибути: авто, майстер, дата прийому в роботу, дата завершення роботи, статус ремонту.
- **Статус:** Атрибути: назва статусу.

Зв’язки між сутностями

1. **Власник – Автомобіль.** Один клієнт може мати декілька автомобілів. Зв'язок: один до багатьох (Клієнт - Автомобіль).
2. **Автомобіль – Замовлення.** Один автомобіль може мати декілька замовлень у різні дати. Зв'язок: один до багатьох (Автомобіль - Замовлення).
3. **Майстер – Замовлення**. Один майстер може виконувати декілька замовлень. Зв'язок: один до багатьох (Співробітник - Замовлення).
4. **Послуга – Замовлення**. Одна послуга може виконуватися в декількох замовленнях і одне замовлення може мати декілька послуг. Зв'язок: багато до багатьох (Послуга - Замовлення).

На основі вказаних сутностей було створено ER-діаграму (Рисунок 2.1.1). ER-діаграма ілюструє сутності, їх характеристики та зв'язки. Це

графічне відображення структури даних, яке сприяє розумінню взаємозв'язків між об'єктами та їх організації.

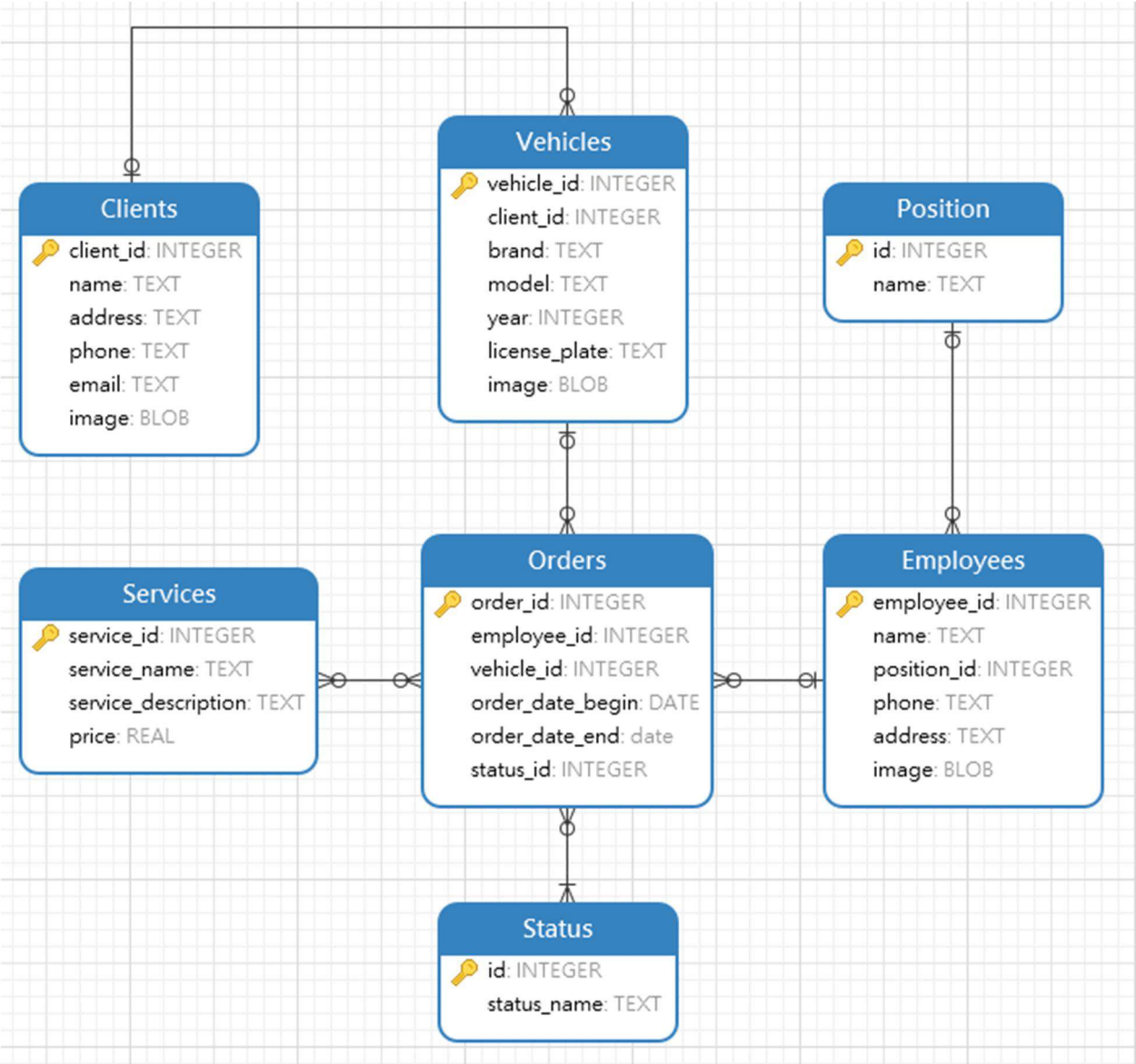


Рисунок 2.1.1 – ER-діаграма станції технічного обслуговування

Отримання ER-діаграми у відповідність до певних нормальних форм допомагає видалити повторювані дані та оптимізувати базу даних для кращої продуктивності. Нормалізація включає в себе переведення інформації до різних рівнів нормалізації, таких як перша (1NF), друга (2NF) та третя (3NF) нормальні форми, а також вищі.

Після аналізу бізнес-процесів я провів нормалізацію діаграми до третьої нормальної форми (3NF) і відобразив її за допомогою мови моделювання Integration DEFinition for information modeling (Рисунок 2.1.2). Ця мова

призначена для створення семантичних моделей даних, що можуть використовуватися для управління даними. IDEF1X є частиною сімейства мов моделювання IDEF у програмній інженерії.

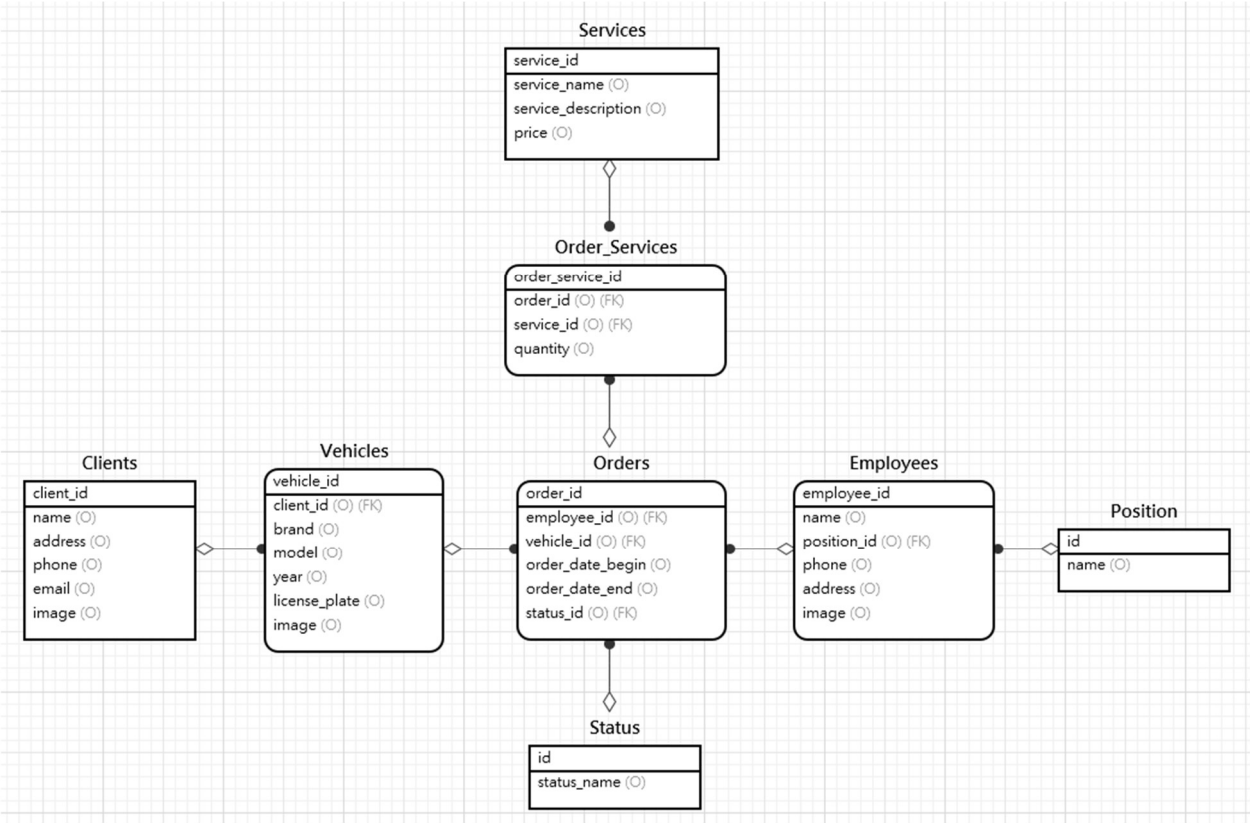


Рисунок 2.1.2 – IDEF1X -діаграма станції технічного обслуговування

Для роботи програми була розроблена база даних на SQLite.

SQLite - це легка, вбудована реляційна система керування базами даних, яка працює без окремого сервера. Вона відзначається своєю простотою в установці та використанні, не вимагає значних ресурсів системи і не потребує налаштування сервера.

SQLite підтримується на різних операційних системах, що забезпечує її крос-платформену сумісність. Вона використовує стандарт SQL, але має свої особливості та обмеження, зокрема відсутність окремого процесу сервера. Дані в SQLite зберігаються в одному файлі, що полегшує перенесення та резервне копіювання. Крім того, вона підтримує транзакції і забезпечує високу швидкодію, особливо при роботі з невеликими обсягами даних. Завдяки своїй

простоті та вбудованій природі, SQLite широко використовується у мобільних додатках та веб-розробці.

Було створено вісім таблиць, які пов'язані між собою за допомогою зв'язків типу "один до багатьох"(Рисунок 2.1.3).

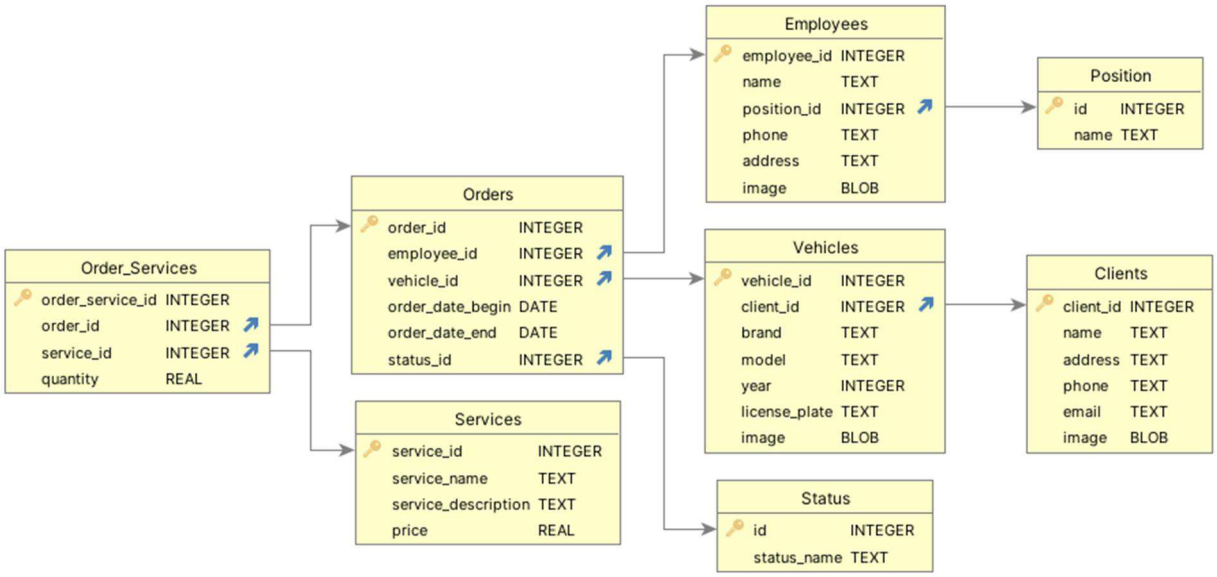


Рисунок 2.1.3 – Схема бази даних станції технічного обслуговування

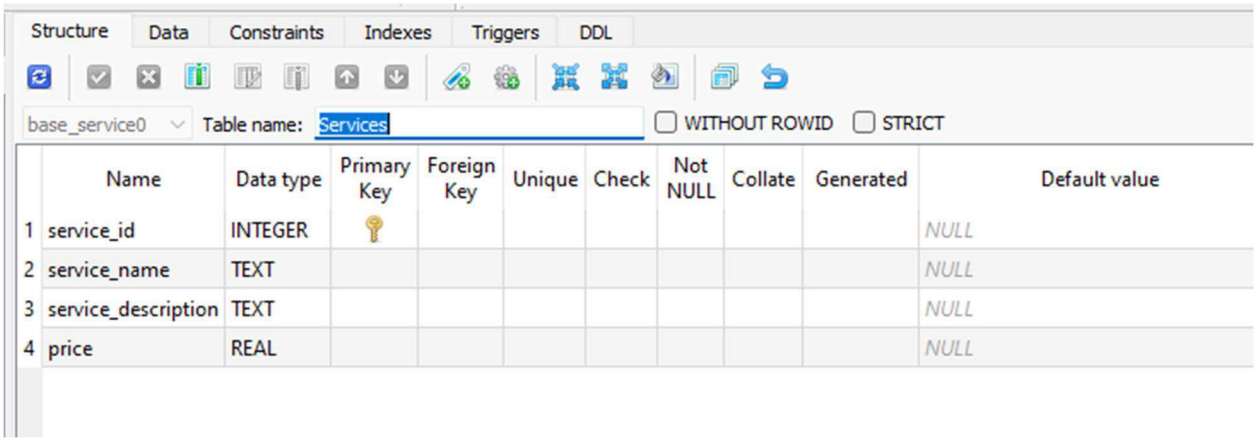


Рисунок 2.1.4 – Структура таблиці Services

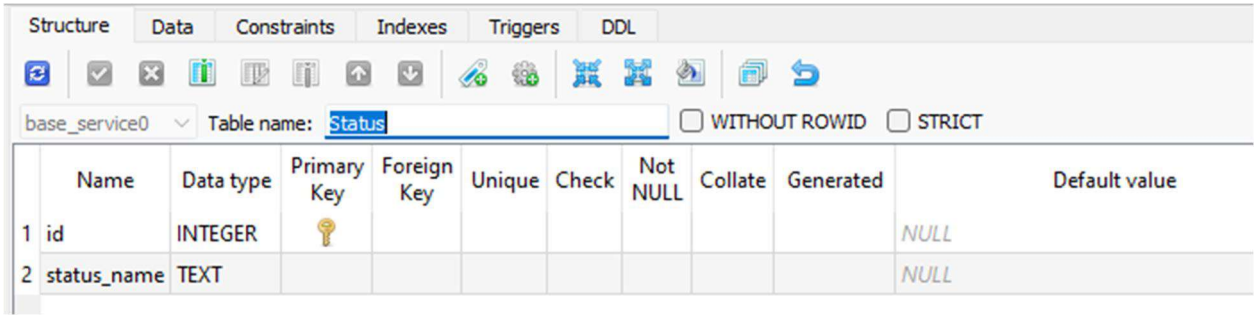


Рисунок 2.1.5 – Структура таблиці Status

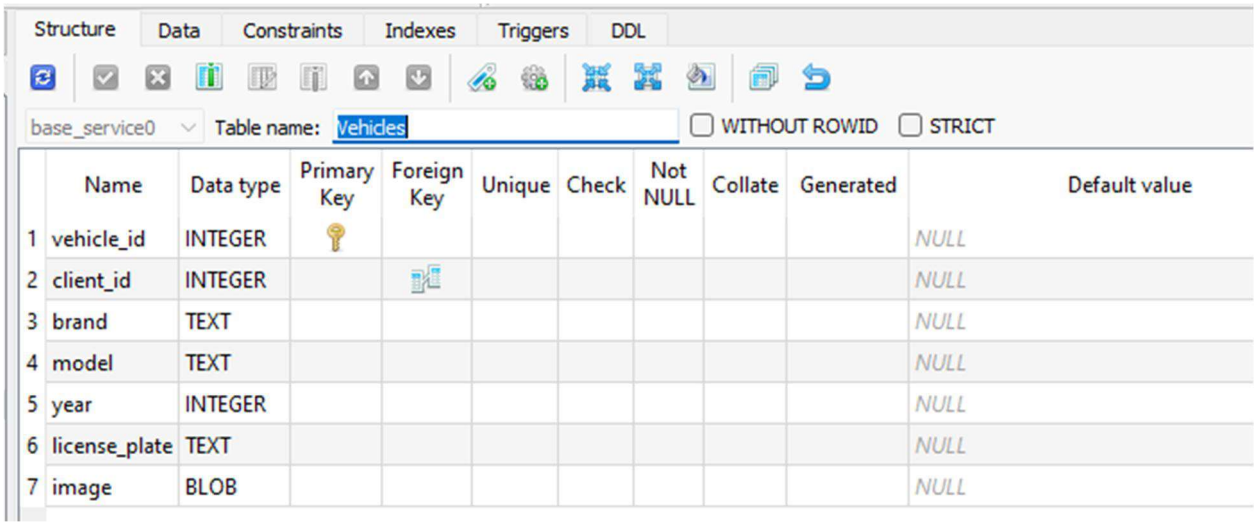


Рисунок 2.1.6 – Структура таблиці Vehicles

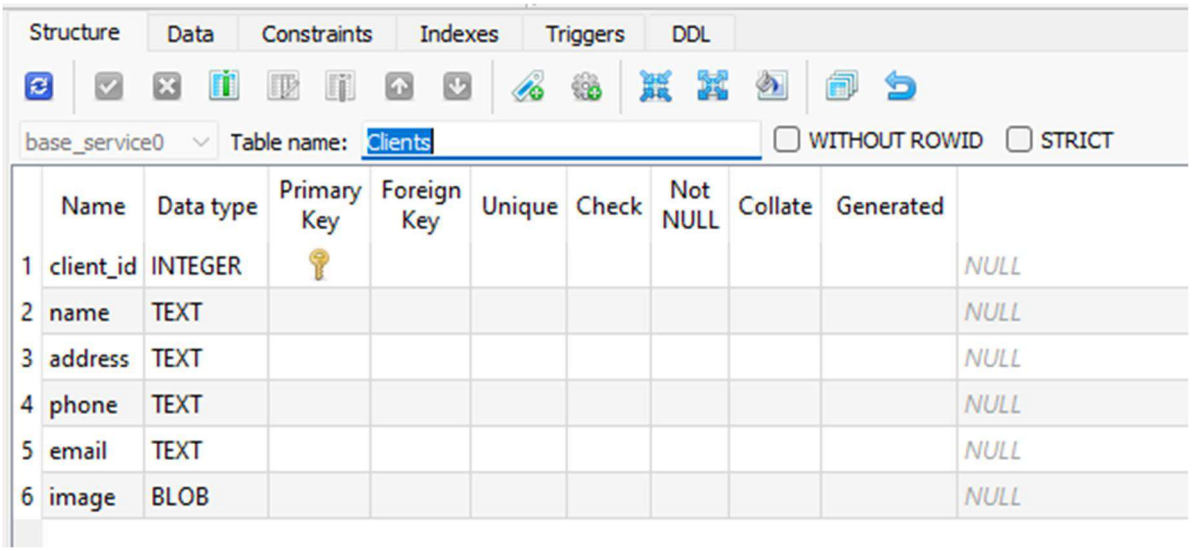


Рисунок 2.1.7 – Структура таблиці Clients

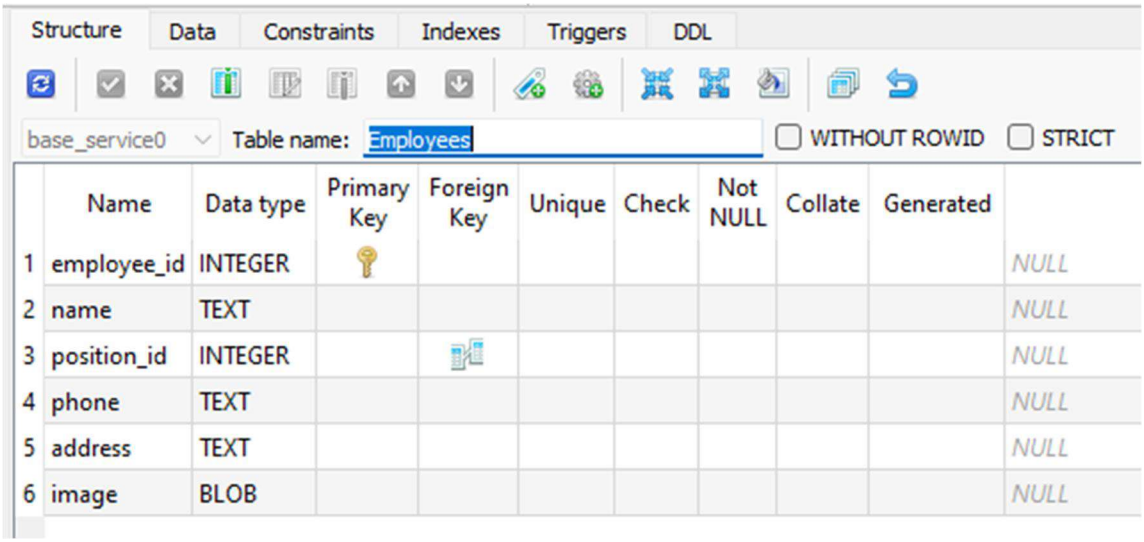


Рисунок 2.1.8 – Структура таблиці Employees

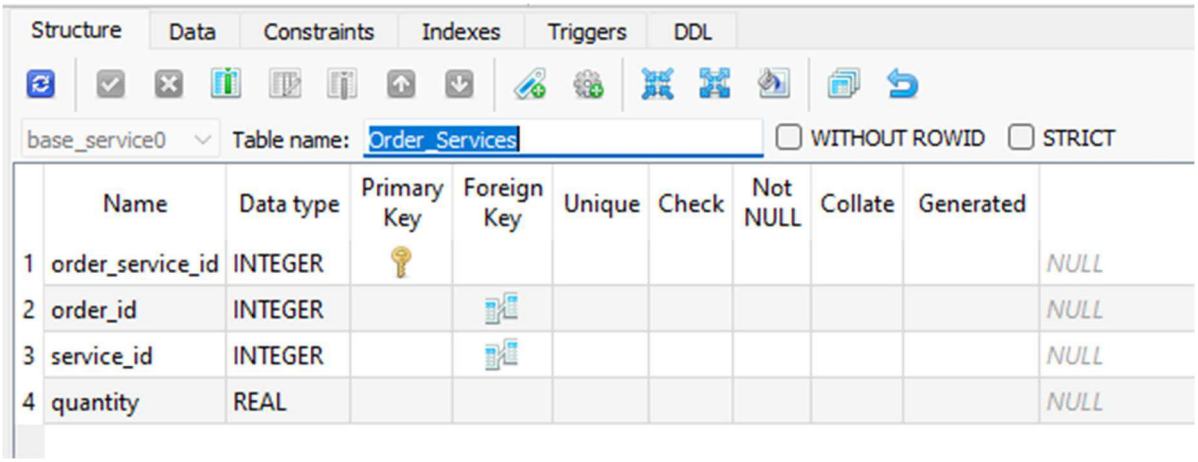


Рисунок 2.1.9 – Структура таблиці Order_Services

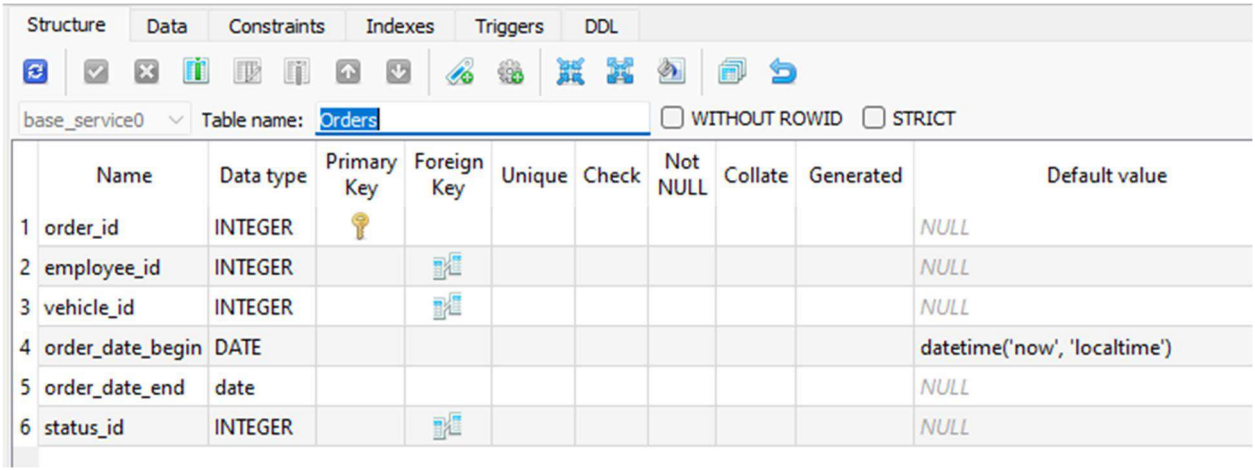


Рисунок 2.1.10 – Структура таблиці Orders

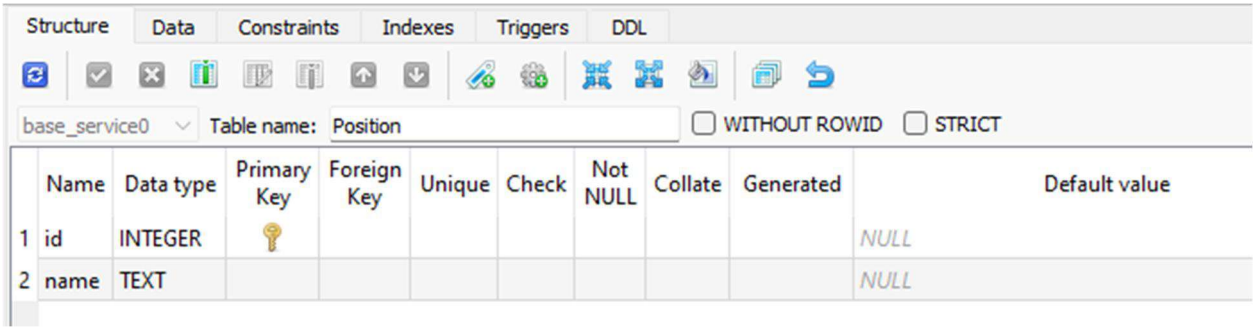


Рисунок 2.1.11 – Структура таблиці Position

2.2. Програмна реалізація

Проект має структуру (Рисунок 2.2.1), визначену у файлі проекту з розширенням ".csproj". Основні складові цієї структури включають визначення типу виводу (OutputType) як Windows Executable (WinExe), цільову версію .NET Framework (TargetFramework), підтримку nullable-типів (Nullable), використання Windows Forms (UseWindowsForms), імпліцитні імпорти (ImplicitUsings) та вказання іконки застосунку (ApplicationIcon).

Далі вказуються файли ресурсів (maintenance.ico) у розділі ItemGroup, що містить інформацію про контент проекту. Також у цьому розділі зазначені посилання на залежності пакетів, які використовуються у проекті, такі як Microsoft.CSharp, System.Data.SqlClient, EntityFramework тощо.

У розділі ItemGroup зазначені файли з кодом (Compile) та ресурси (EmbeddedResource), які включають форми редагування, інформаційні форми та ресурси проекту. Цей перелік містить файли форм редагування (EditClient.cs, EditOrder.cs, EditOrderService.cs, EditVehicle.cs, EditEmployees.cs), файли інформації (InfoClients.cs, InfoEmployees.cs) та файл дизайнера ресурсів (Resources.Designer.cs).

Також визначені файли баз даних (base_service.db) у розділі ItemGroup, які будуть скопійовані в вихідний каталог проекту при компіляції.

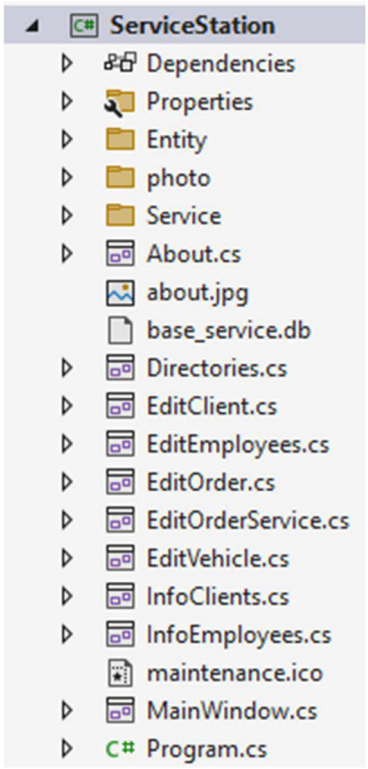


Рисунок 2.2.1 – Структура проєкту

Головне вікно програми має п’ять вкладок з різним функціоналом.

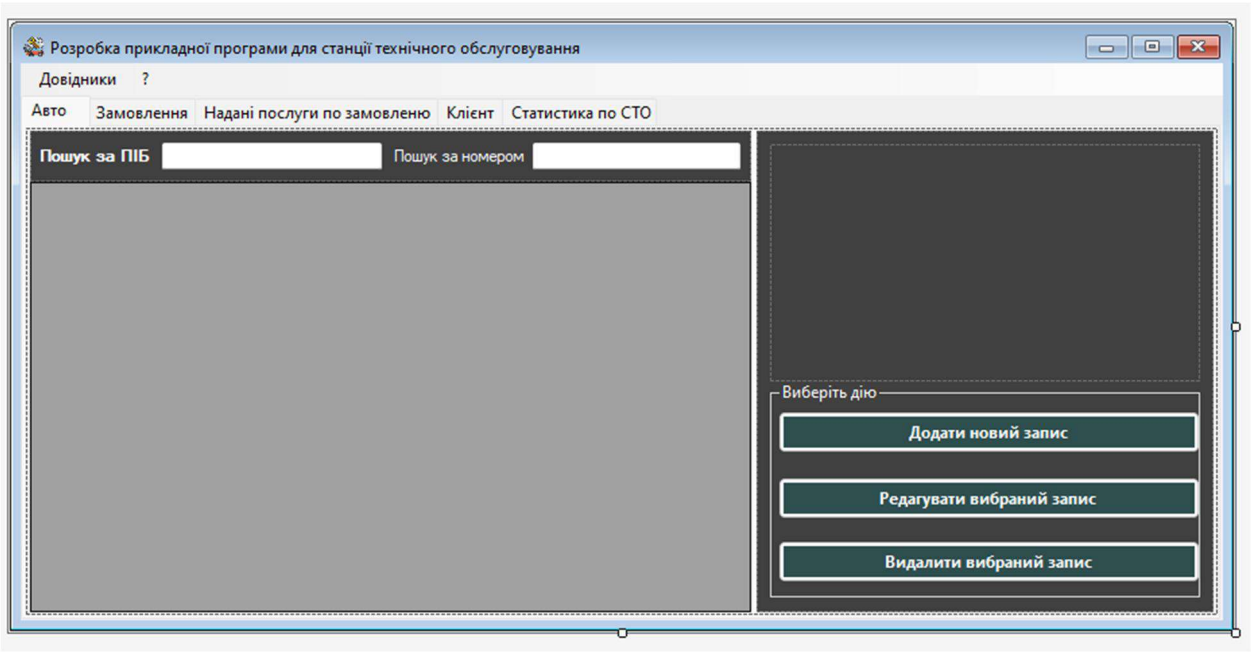


Рисунок 2.2.2 – Головне вікно

Інше вікно це інформація про програму (Рисунок 2.2.3)

Рисунок 2.2.3 – Про програму

2.3. Опис класів

Клас WorkWithImage (Рисунок 2.3.1) містить статичні методи для роботи з зображеннями. Цей клас дозволяє зручно працювати з зображеннями у форматі JPEG, а також забезпечує можливість додавання зображень до програми з різних джерел.

```
21 references
public class WorkWithImage
{
    3 references
    public static byte[]? ImageToByte(Image image)...

    9 references
    public static Image NoPhoto()...

    6 references
    public static Image ByteToImage(byte[] imageBytes)...

    3 references
    public static void LoadPhotoToPicterBox(PictureBox pictureBox)...
}
```

Рисунок 2.3.1 – Клас WorkWithImage

Основні методи цього класу:

Метод ImageToByte: Приймає об'єкт Image і повертає його представлення у вигляді масиву байтів (byte[]). Цей метод конвертує зображення у формат JPEG та зберігає його у вказаному масиві байтів.

Метод NoPhoto: Генерує зображення-замінник у випадку відсутності фото. Він створює нове зображення з білим фоном та текстом "Фото відсутнє", розташованим по центру.

Метод ByteToImage: Приймає масив байтів (byte[]) та повертає об'єкт Image, який є розшифрованим представленням цих байтів.

Метод LoadPhotoToPictureBox: Відкриває діалогове вікно для вибору файлу зображення, а потім завантажує обране зображення в об'єкт PictureBox.

Клас ItemForCombo (Рисунок 2.3.2) представляє собою модель для об'єктів, які використовуються для заповнення випадаючих списків (ComboBox) у графічному інтерфейсі. Основні характеристики цього класу:

1. Властивість Id: Це унікальний ідентифікатор елемента. Використовується для ідентифікації кожного об'єкта в списку.
2. Властивість Value: Це значення, яке відображається у випадаючому списку для користувача. Вона містить текстове представлення об'єкта.
3. Конструктор за замовчуванням ініціалізує властивість Value пустим рядком.
4. Перевизначений метод ToString(): Цей метод повертає значення властивості Value об'єкта. Його використання дозволяє автоматично відображати значення об'єкта у випадаючому списку.

```
21 references
public class WorkWithImage
{
    3 references
    public static byte[]? ImageToByte(Image image)...

    9 references
    public static Image NoPhoto()...

    6 references
    public static Image ByteToImage(byte[] imageBytes)...

    3 references
    public static void LoadPhotoToPicterBox(PictureBox pictureBox)...
}
```

Рисунок 2.3.2 – Клас ItemForCombo

Висновки до розділу 2

У розділі було ретельно розглянуто процес створення бази даних для станції технічного обслуговування. Була надана докладна інформація про структури всіх таблиць у базі даних та про зв'язки між ними. Крім того, була розглянута структура проекту, що включає в себе не лише базу даних, але й інші компоненти програми. Увагу було зосереджено на класах, які використовуються у програмі для забезпечення її функціональності, надано детальний опис їх призначення та ролі у системі.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

1. **Операційна система**: Windows 7 або новіша.
2. **Процесор**: 1 GHz або швидший процесор (рекомендується 2 GHz або більше).
3. **Пам'ять**: Мінімум 1 GB оперативної пам'яті (рекомендується 2 GB або більше).
4. **Дисковий простір**: Потрібне місце на жорсткому диску для встановлення .NET 8 та програми.
5. **Відеокарта**: Вбудована або зовнішня відеокарта з підтримкою DirectX 9 або новішої версії.
6. **Монітор**: Мінімально рекомендується роздільна здатність екрану 1024x768 пікселів.
7. **.NET 8**: Для запуску програми потрібно встановити .NET 8 або новішу версію.

3.2. Інтерфейс користувача

Програма запускається з папки через файл ServiceStation.exe. Для роботи програми потрібні додаткові файли().

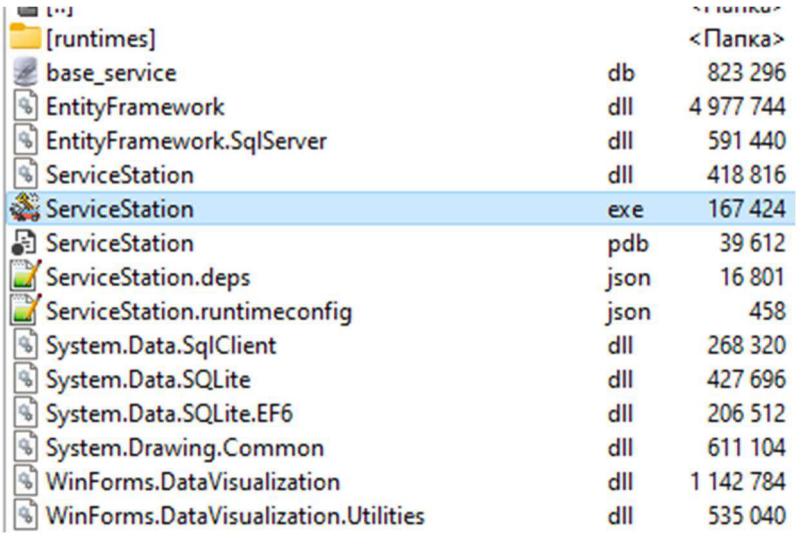


Рисунок 3.2.1 – Файл запуску

При запуску програми відкривається головне вікно, яке служить центральним інтерфейсом для роботи з усіма даними станції технічного обслуговування (СТО). Це головне вікно організовано у вигляді декількох вкладок, що забезпечують зручний доступ до різних функцій та дозволяють економити місце на екрані.

Програма містить п’ять вкладок, кожна з яких призначена для виконання певних завдань. Перша вкладка, яка називається «Авто» (Рисунок 3.2.2), спеціально розроблена для роботи з автомобілями, що обслуговуються на СТО. На цій вкладці користувач може виконувати такі дії:

1. Додавання нового автомобіля: Можна легко додати інформацію про новий автомобіль, включаючи такі дані, як марка, модель, рік випуску, державний номер, та інші необхідні деталі.
2. Редагування інформації: Користувач має можливість вибрати існуючий автомобіль зі списку та внести необхідні зміни до його інформації. Це може включати оновлення даних про власника, зміни у технічних характеристиках або додавання нових фотографій.
3. Видалення автомобіля: Якщо автомобіль більше не обслуговується на СТО, його можна видалити зі списку. Ця функція дозволяє підтримувати базу даних актуальною та позбавленою зайвої інформації.

Фільтрація даних

Для полегшення пошуку та управління даними про автомобілі, вкладка «Авто» також включає функції фільтрації. Користувач може застосувати фільтри для швидкого знаходження необхідного автомобіля:

1. Фільтр за власником: Дозволяє знайти всі автомобілі, що належать конкретному власнику. Це корисно для перегляду історії обслуговування автомобілів певного клієнта.
2. Фільтр за державним номером: Дозволяє швидко знайти автомобіль за його державним номером. Це особливо корисно, коли необхідно знайти конкретний автомобіль серед великої кількості записів.

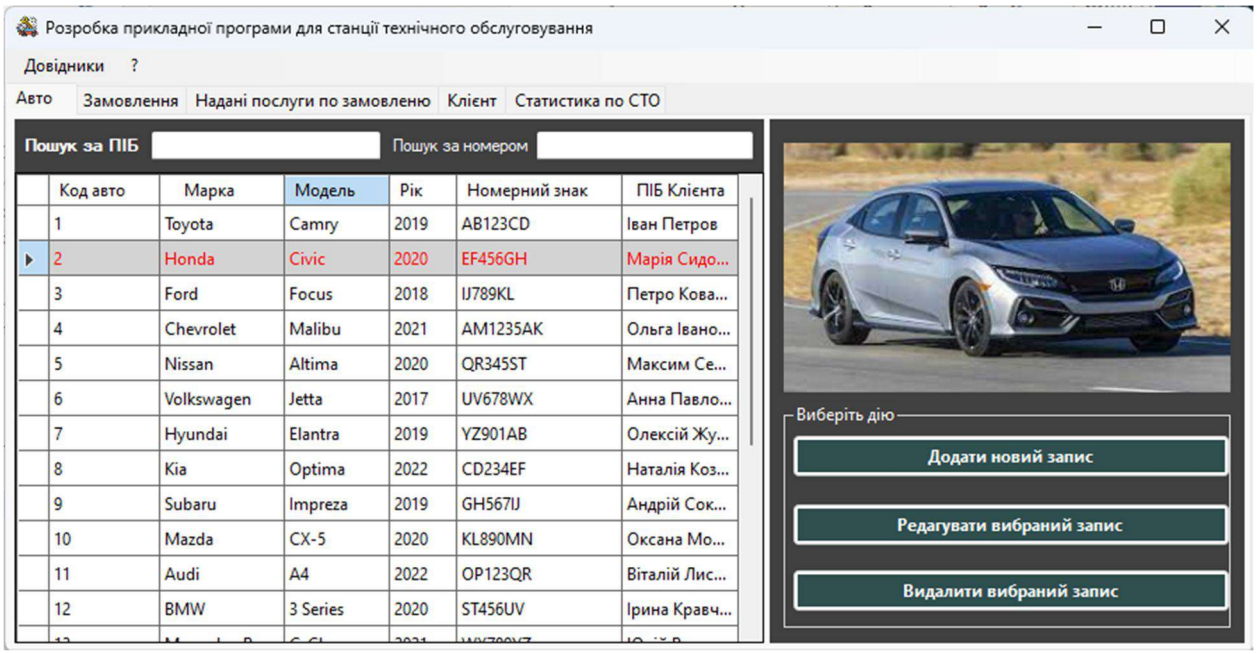


Рисунок 3.2.2 – Інтерфейс програми на вкладці «Авто»

При натисканні на кнопку «Додати новий запис» в головному вікні програми відкриється нове вікно (Рисунок 3.2.3), яке призначене для створення нового запису про автомобіль у базі даних станції технічного обслуговування (СТО). Це вікно спроектоване для зручного та ефективного введення всієї необхідної інформації про новий автомобіль.

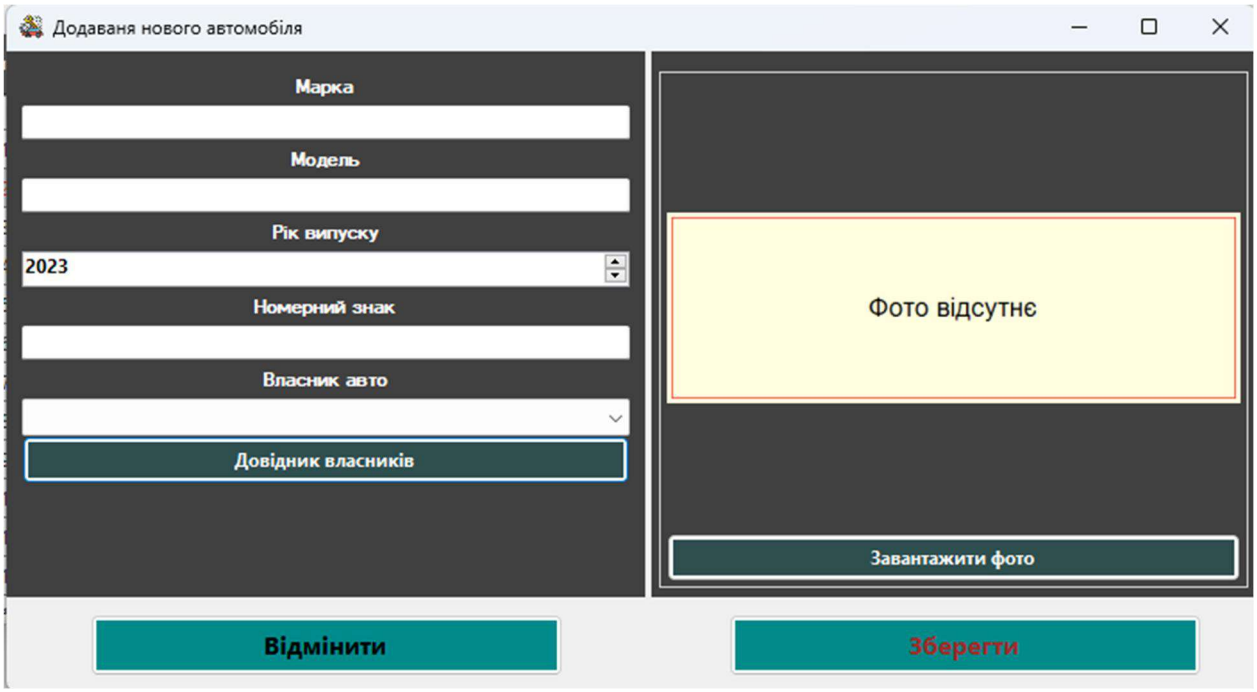


Рисунок 3.2.3 – Вікно «Додавання нового автомобіля»

При натисканні на кнопку «Редагувати вибраний запис» в головному вікні програми відбудеться відкриття нового вікна, спеціально призначеного для редагування інформації про обраний автомобіль в базі даних станції технічного обслуговування(Рисунок 3.2.4). Це вікно забезпечує можливість коригування будь-яких даних, пов'язаних з автомобілем, щоб забезпечити їх актуальність та точність.

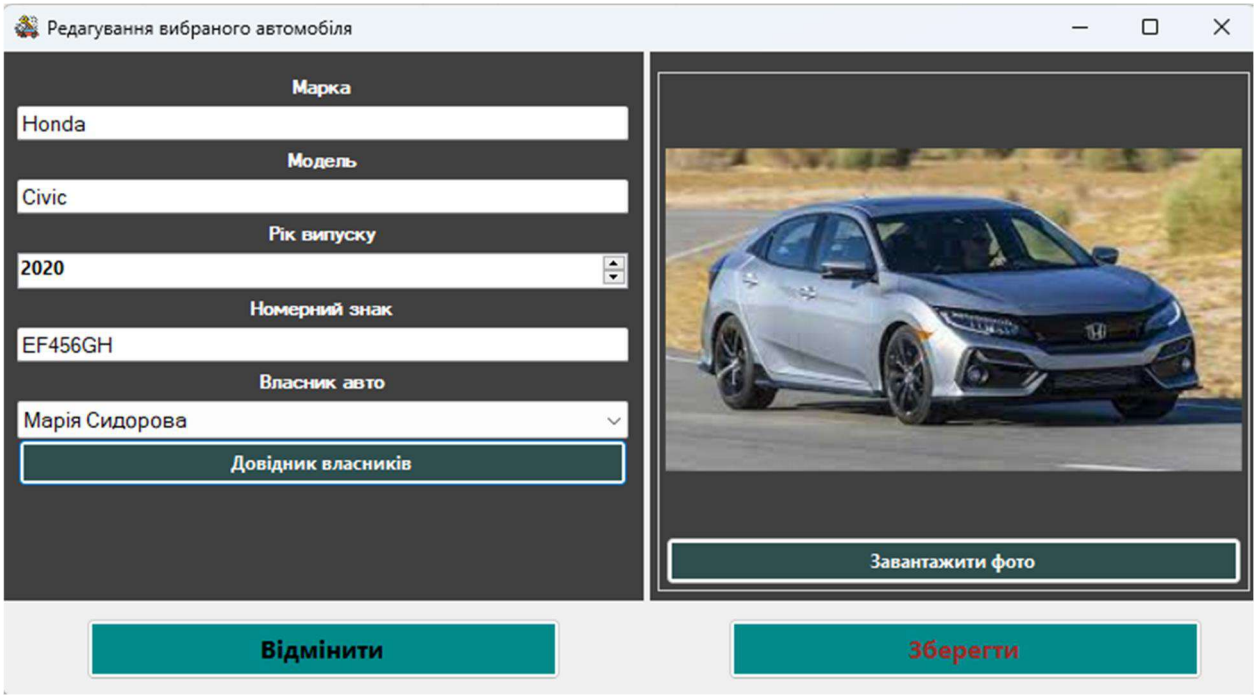


Рисунок 3.2.4 – Довідник «Редагування автомобіля»

Для перегляду або створення нового замовлення необхідно перейти до другої вкаладки «Замовлення» в головному вікні програми(Рисунок 3.2.5). Ця вкладка призначена для управління всіма замовленнями, пов'язаними з конкретним автомобілем, що обслуговується на станції технічного обслуговування. Вона забезпечує зручний доступ до інформації про замовлення та можливість створення нових замовлень в системі.

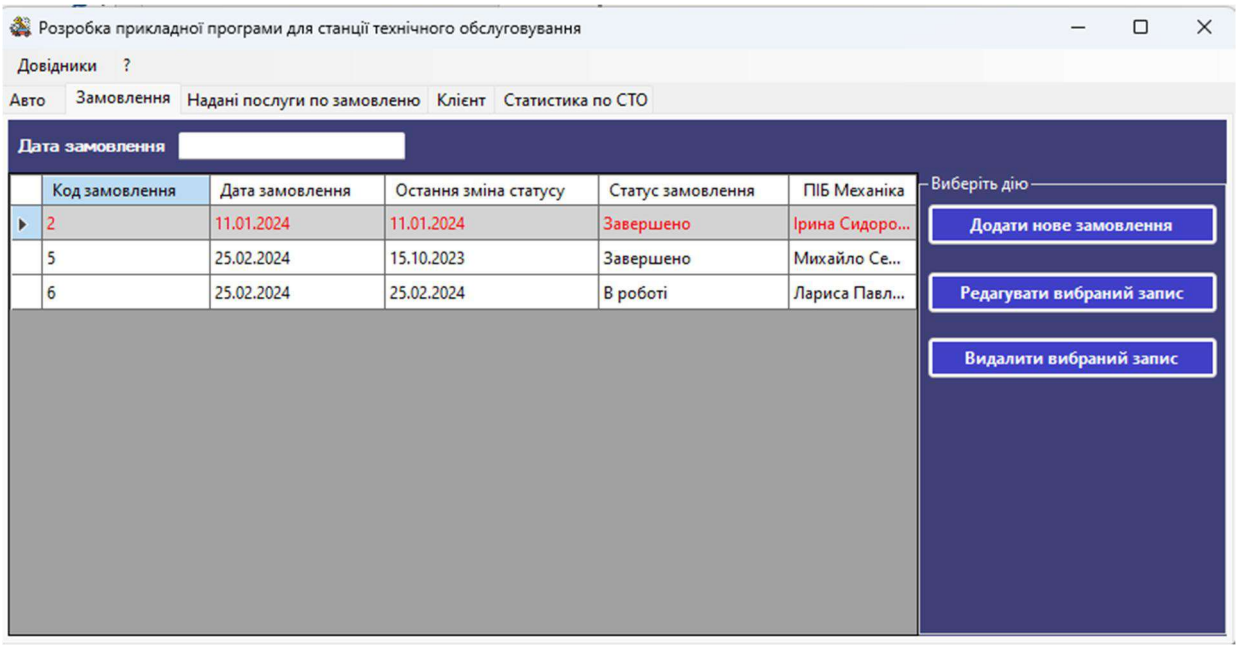


Рисунок 3.2.5 – Інтерфейс програми на вкладці «Замовлення»

При натисканні на кнопку «Додати нове замовлення» в головному вікні програми автоматично відкриється нове вікно (Рисунок 3.2.6), яке призначене для створення нового замовлення на обслуговування конкретного автомобіля на станції технічного обслуговування. Це вікно розроблене з урахуванням зручності та легкості використання для оперативного створення нових замовлень.

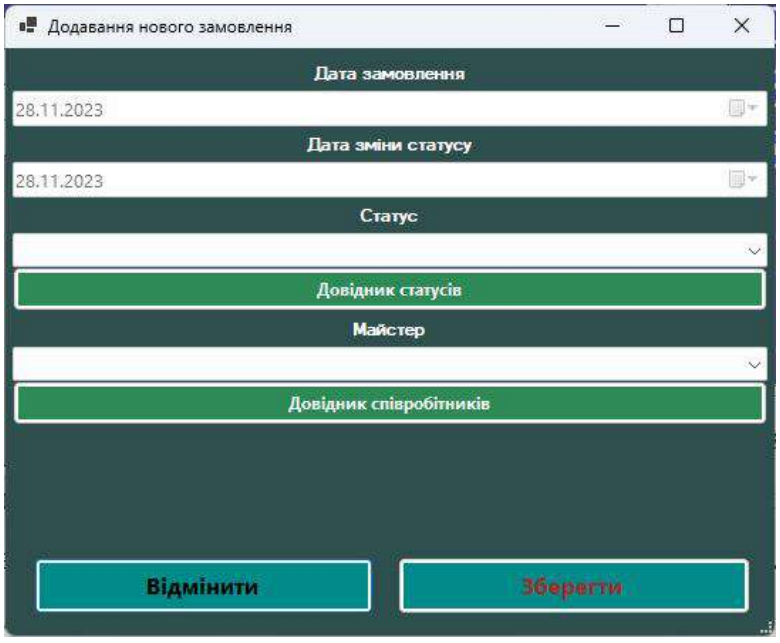


Рисунок 3.2.6 – Вікно «Додавання нового замовлення»

При натисканні на кнопку «Редагування вибраного замовлення» відкриється нове вікно для редагування вибраного замовлення (Рисунок 3.2.7).

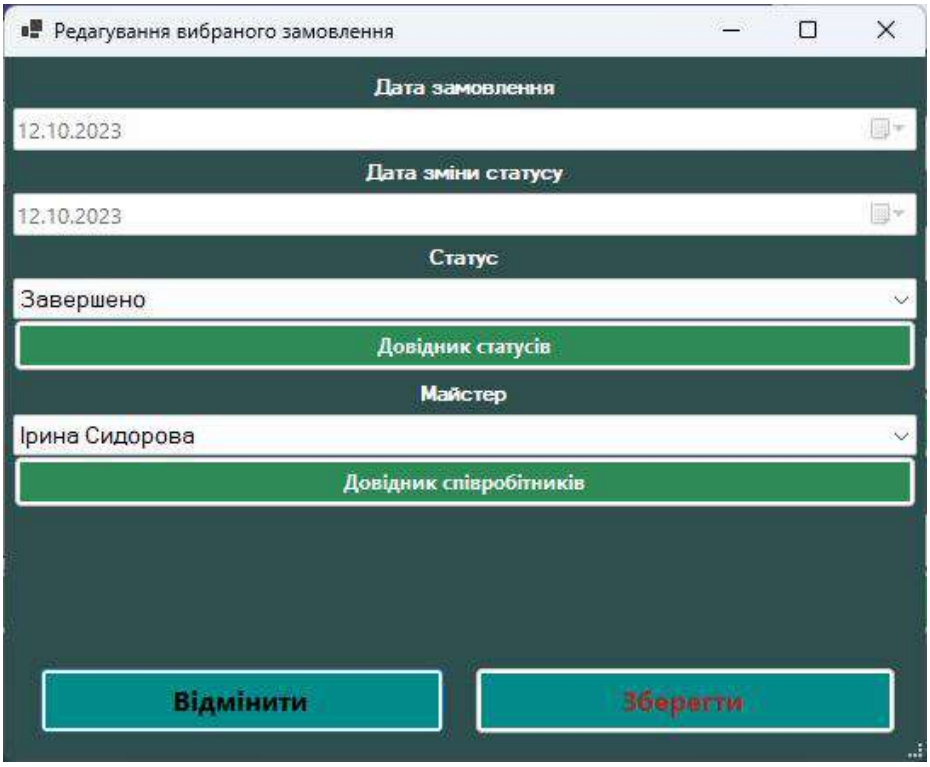


Рисунок 3.2.7 – Вікно «Редагування вибраного замовлення»

Для перегляду детальної інформації про замовлення або створення нового замовлення необхідно вибрати відповідне замовлення і перейти на вкладку «Надані послуги по замовленню» в головному вікні програми (Рисунок 3.2.8). Ця вкладка призначена для надання повної інформації про послуги, які були надані в рамках конкретного замовлення на обслуговування автомобіля на станції технічного обслуговування. Тут користувач може ознайомитися з усіма виконаними роботами, їх вартістю та іншими деталями.

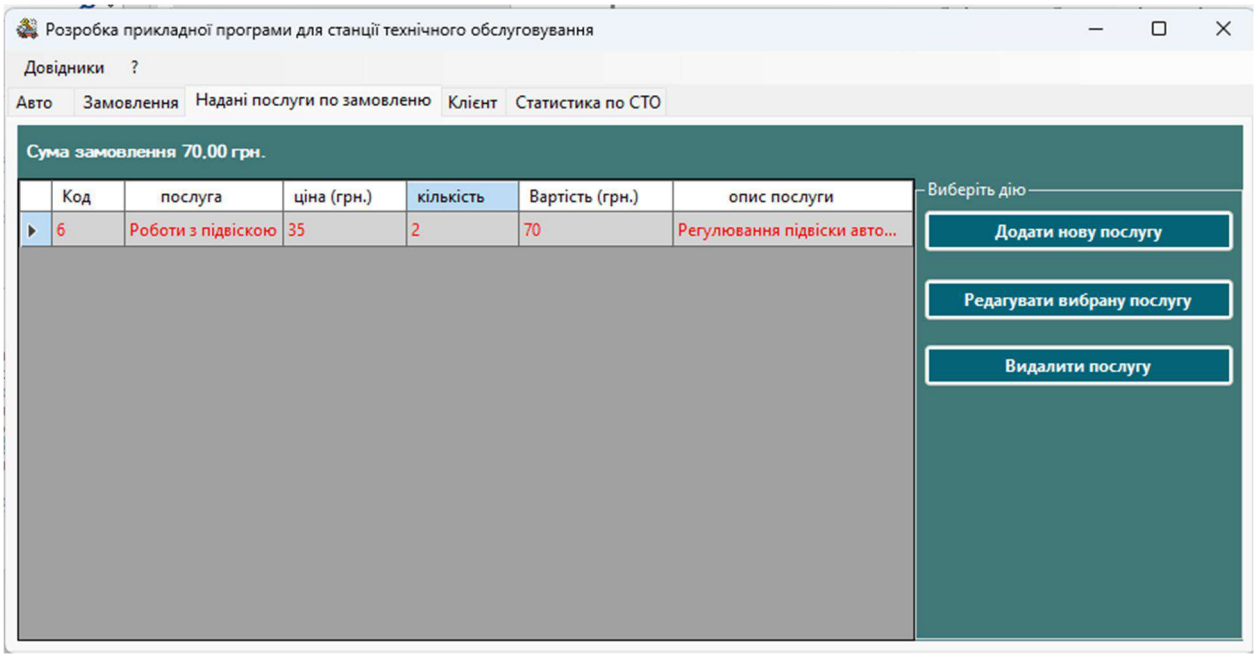


Рисунок 3.2.8 – Інтерфейс програми на вкладці «Надані послуги по замовленню»

При натисканні на кнопку «Додати нову послугу» в головному вікні програми відкриється нове вікно (Рисунок 3.2.9), яке призначене для вибору певної послуги на станції технічного обслуговування. Це вікно дозволяє користувачу взаємодіяти з довідником послуг, додавати нові або редагувати вже існуючі послуги, що надаються на СТО(Рисунок 3.2.10).

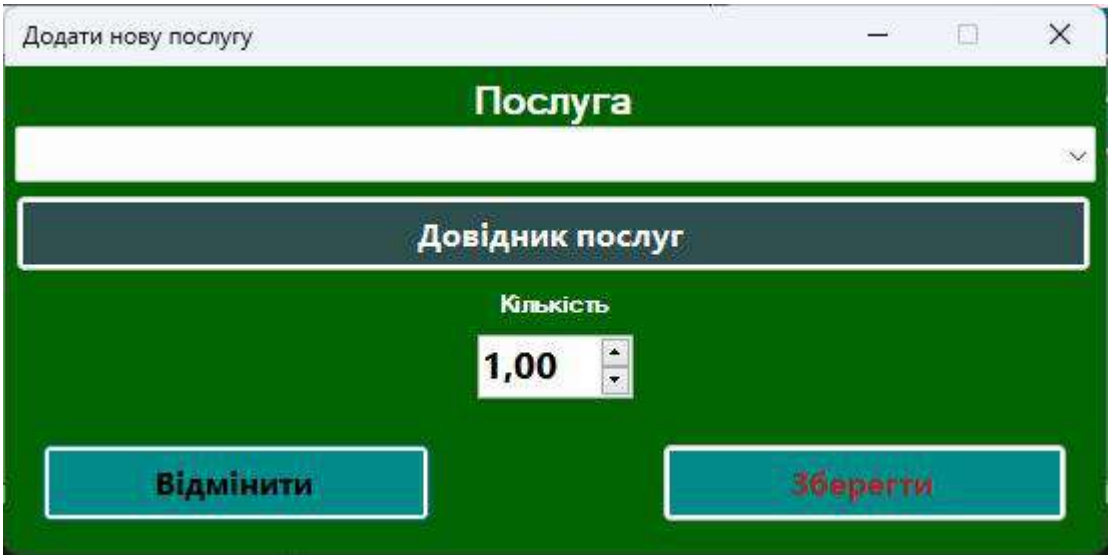


Рисунок 3.2.9 – Вікно «Додати нову послугу»

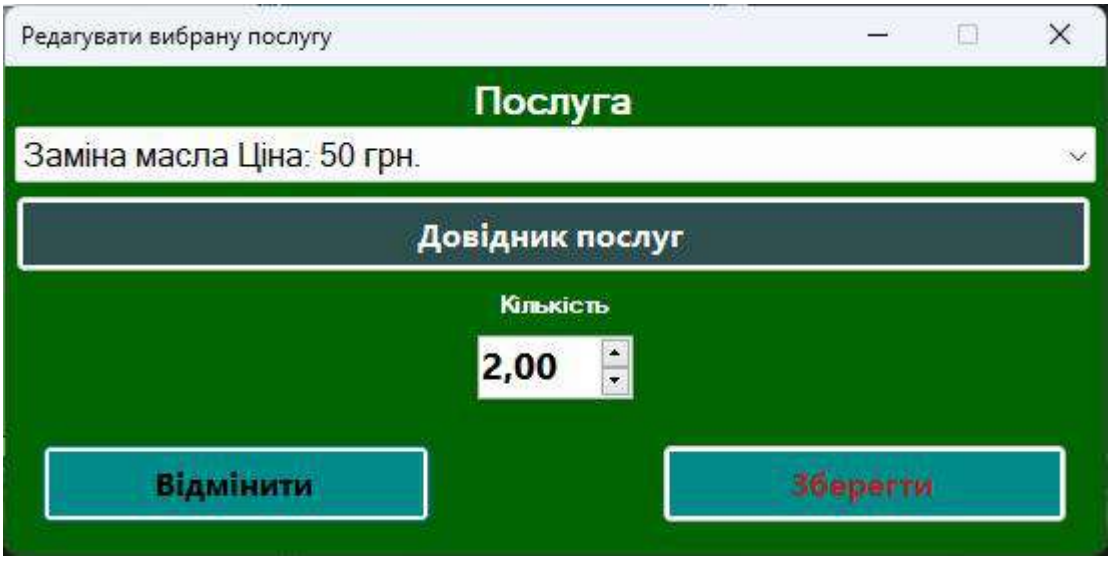


Рисунок 3.2.10 – Вікно «Редагувати вибрану послугу»

Четверта вкладка головного вікна дозволяє переглянути детальну інформацію про власника автомобіля без можливості редагування (Рисунок 3.2.11).

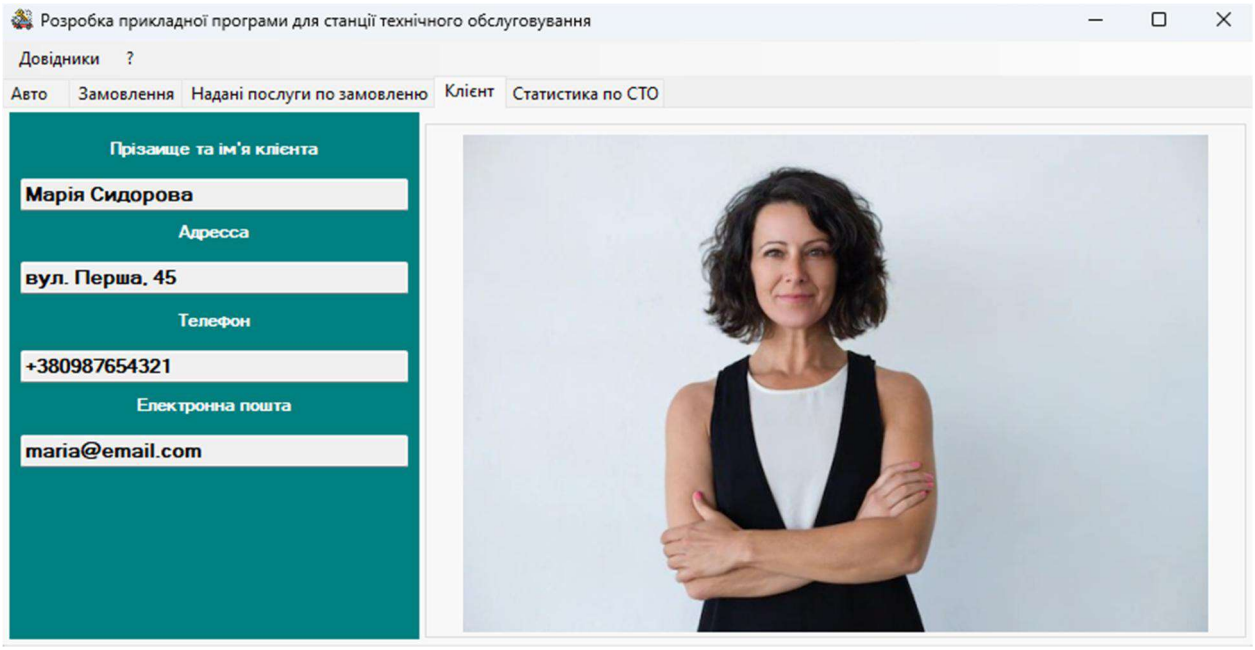


Рисунок 3.2.11 – Інтерфейс програми на вкладці «Клієнт»

Остання вкладка програми станції технічного обслуговування (СТО) призначена для візуалізації діаграми, яка відображає кількість автомобілів, які були обслуговані на СТО протягом певного періоду. Ця функціональність

дозволяє власникам або адміністраторам СТО отримувати важливу статистику та аналізувати динаміку обслуговування автомобілів.

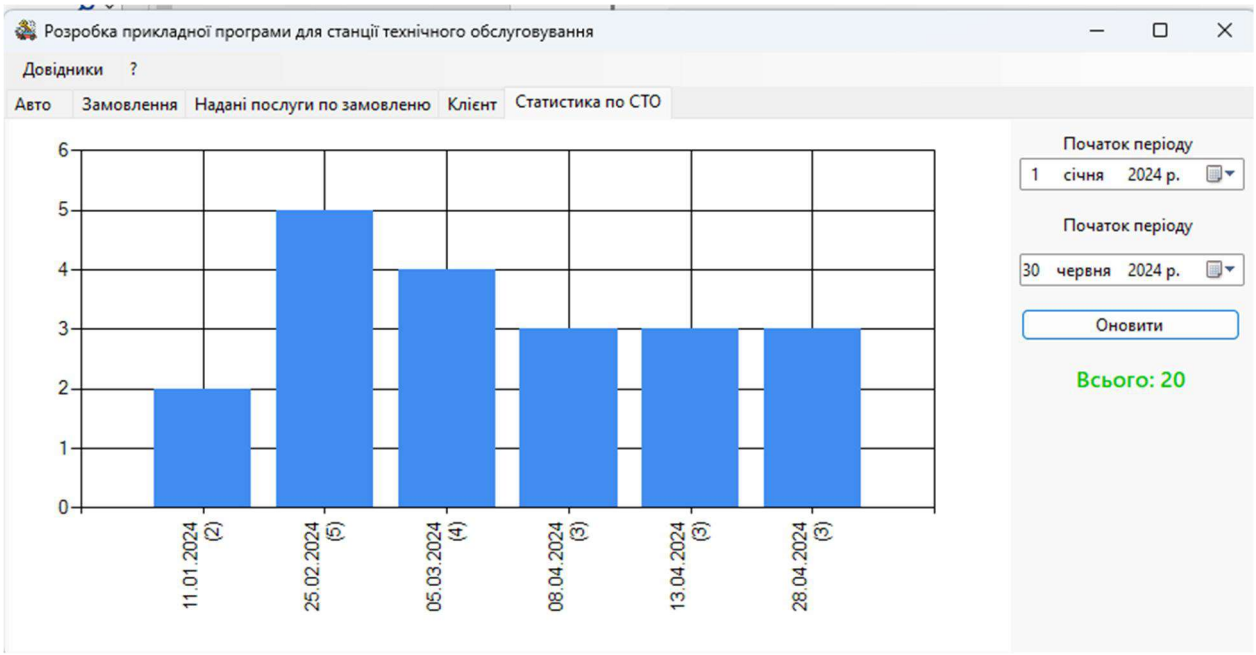


Рисунок 3.2.12 – Інтерфейс програми на вкладці «Статистика СТО»

Програма має вбудовані довідники, до яких можна отримати доступ з головного меню. Ці довідники призначені для управління різними аспектами діяльності станції технічного обслуговування. Кожен довідник надає можливість виконувати операції додавання, редагування або видалення записів у відповідному довіднику.

Довідник посад на станції технічного обслуговування служить для управління посадами та ролями персоналу, які працюють на СТО (Рисунок 3.2.12.). Цей довідник дозволяє визначати і впорядковувати різні посади, необхідні для ефективного функціонування СТО. Користувач може виконувати операції додавання нових посад, редагування існуючих та видалення зайвих посад.

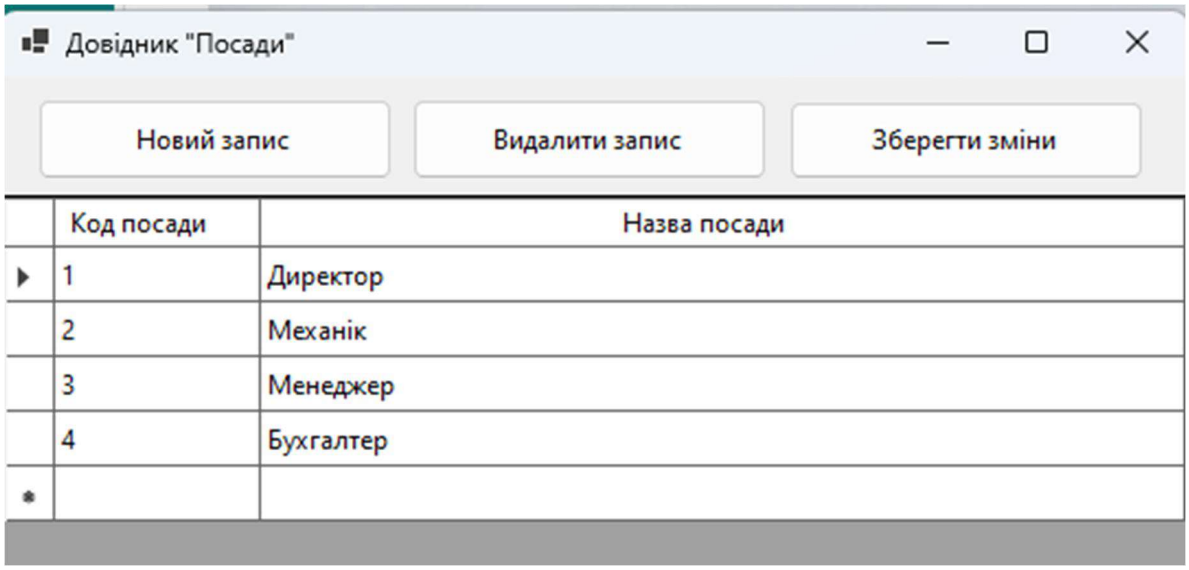


Рисунок 3.2.13 – Довідник «Посади»

Довідник "Статуси замовлення" надає користувачеві можливість керувати статусами замовлень, що забезпечує ефективне управління процесом обслуговування автомобілів на станції технічного обслуговування (Рисунок 3.2.14).

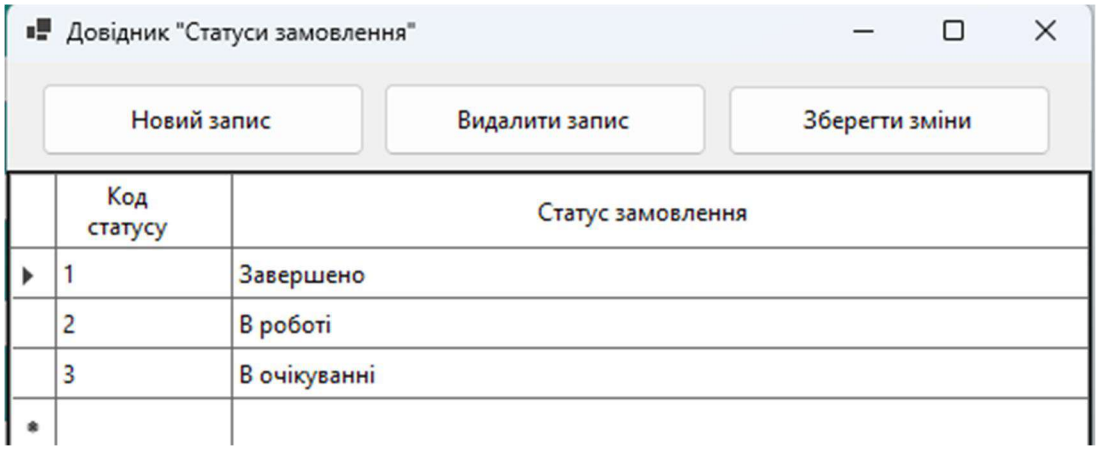
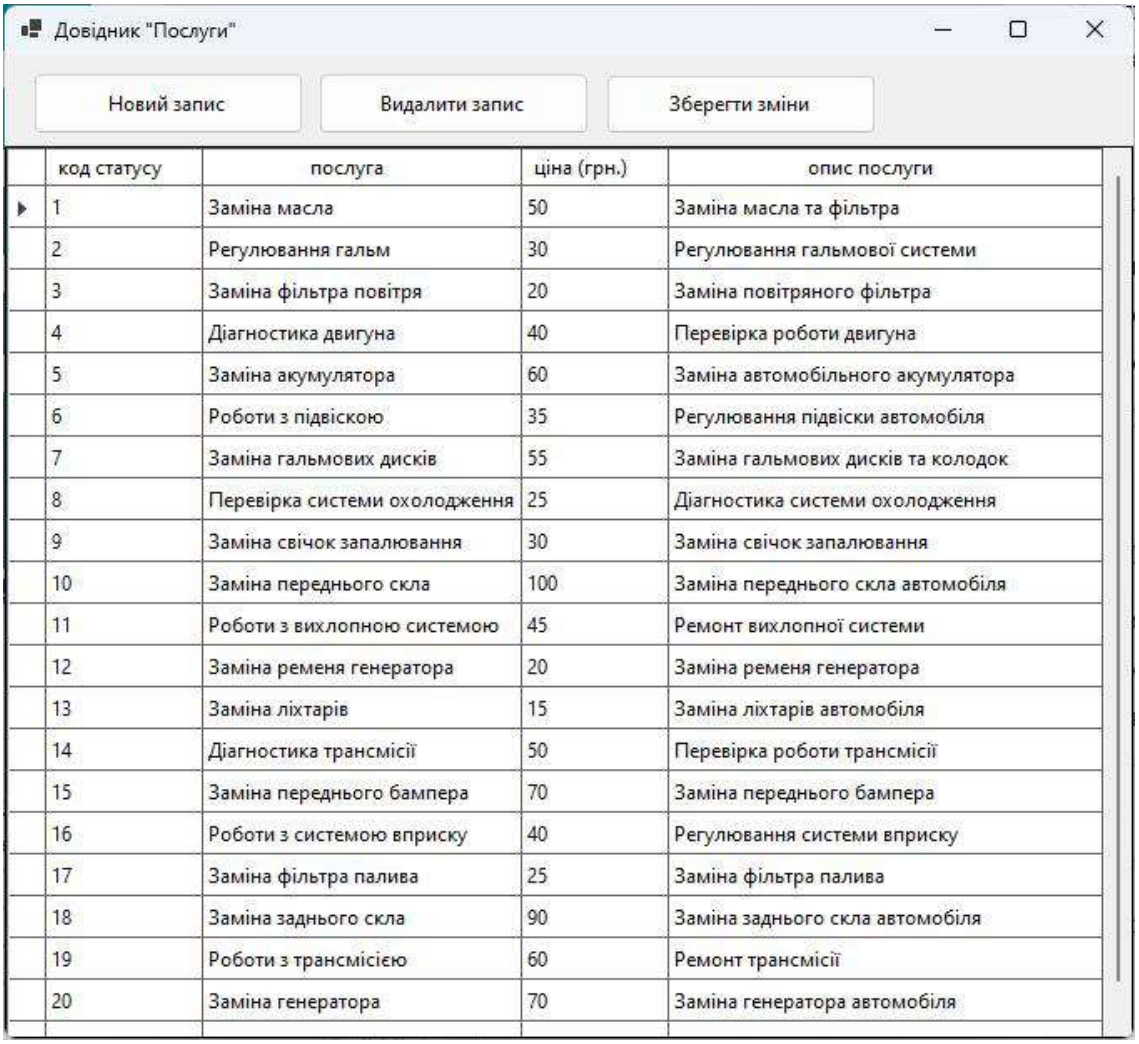


Рисунок 3.2.14 – Довідник «Статуси замовлення»

Зовнішній вигляд довідника "Послуги" на станції технічного обслуговування представлений у вигляді таблиці, що містить інформацію про різні послуги, які надаються СТО ()Рисунок 3.2.15. Такий довідник дозволяє керувати переліком послуг, що доступні для клієнтів, включаючи можливість додавання нових, редагування чи видалення існуючих послуг.



код статусу	послуга	ціна (грн.)	опис послуги
1	Заміна масла	50	Заміна масла та фільтра
2	Регулювання гальм	30	Регулювання гальмової системи
3	Заміна фільтра повітря	20	Заміна повітряного фільтра
4	Діагностика двигуна	40	Перевірка роботи двигуна
5	Заміна акумулятора	60	Заміна автомобільного акумулятора
6	Роботи з підвіскою	35	Регулювання підвіски автомобіля
7	Заміна гальмових дисків	55	Заміна гальмових дисків та колодок
8	Перевірка системи охолодження	25	Діагностика системи охолодження
9	Заміна свічок запалювання	30	Заміна свічок запалювання
10	Заміна переднього скла	100	Заміна переднього скла автомобіля
11	Роботи з вихлопною системою	45	Ремонт вихлопної системи
12	Заміна ременя генератора	20	Заміна ременя генератора
13	Заміна ліхтарів	15	Заміна ліхтарів автомобіля
14	Діагностика трансмісії	50	Перевірка роботи трансмісії
15	Заміна переднього бампера	70	Заміна переднього бампера
16	Роботи з системою вприску	40	Регулювання системи вприску
17	Заміна фільтра палива	25	Заміна фільтра палива
18	Заміна заднього скла	90	Заміна заднього скла автомобіля
19	Роботи з трансмісією	60	Ремонт трансмісії
20	Заміна генератора	70	Заміна генератора автомобіля

Рисунок 3.2.15 – Довідник «Послуги»

Зовнішній вигляд вікна "Довідник працівників" на станції технічного обслуговування спрямований на зручну та ефективну навігацію між списком працівників та виконанням різних операцій з ними (Рисунок 3.2.16). Вікно містить елементи інтерфейсу, які дозволяють фільтрувати та керувати списком працівників.

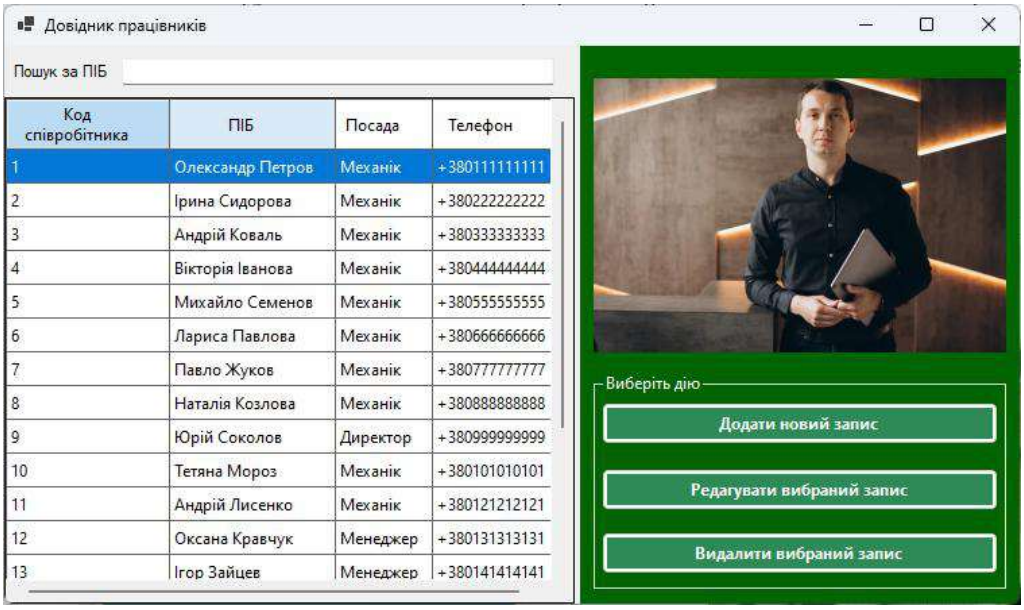


Рисунок 3.2.16 – Інтерфейс вікна «Довідник працівників»

На вікні "Довідник клієнтів" представлений зручний інтерфейс для роботи з базою даних клієнтів станції технічного обслуговування (Рисунок 3.2.17). Це вікно дозволяє користувачеві виконувати операції, такі як пошук, додавання, редагування та видалення клієнтів.

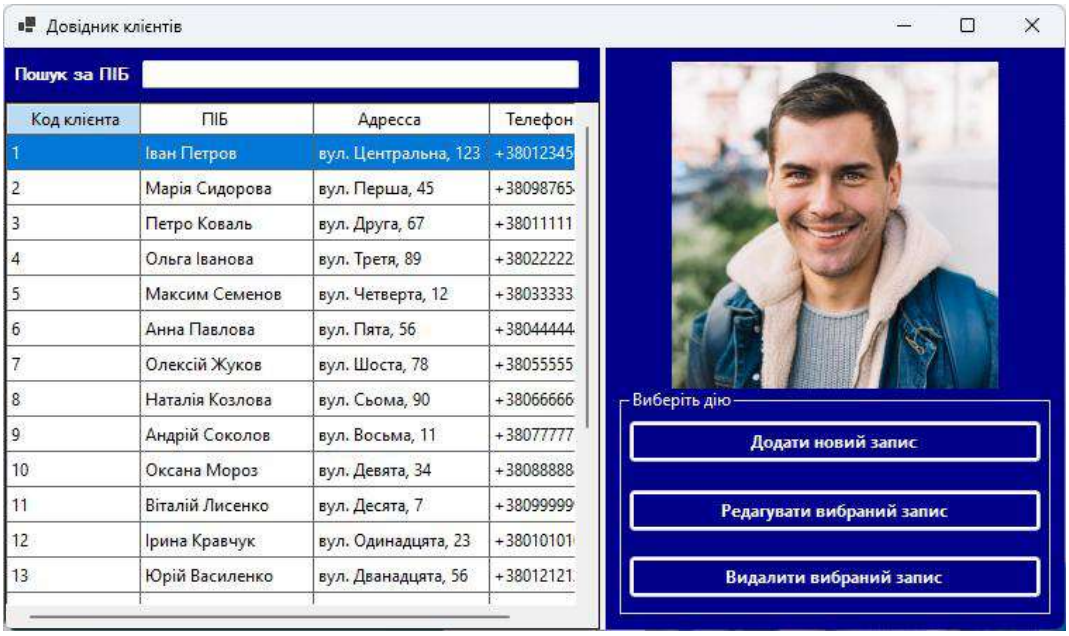


Рисунок 3.2.17 – Інтерфейс вікна «Довідник клієнтів»

Меню програми містить пункт "?", за допомогою якого можна відкрити вікно, що містить інформацію про автора програми ().

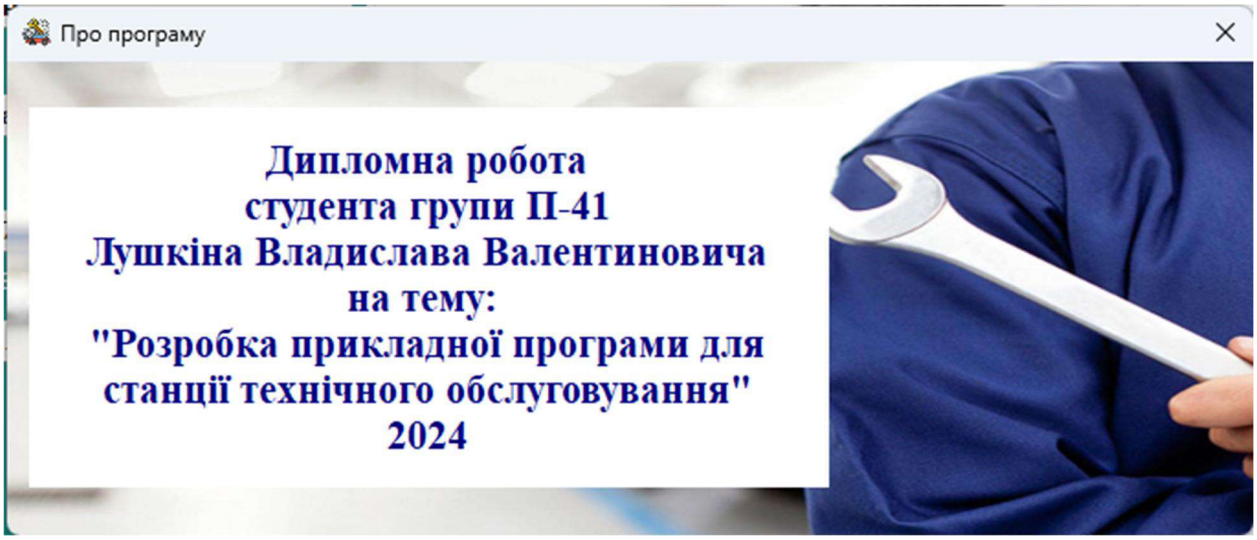


Рисунок 3.2.18 – Вікно «Про програму»

Висновки до розділу 3

Проведено взаємодію з інтерфейсом програми, я продемонстрував роботу з різними довідниками. Користувацький інтерфейс вийшов дуже зручним для використання системи. Програма функціонує на базі даних SQLite, що дозволяє проводити пошук, зміну та відображення інформації через виконання запитів. SQLite має обмеження у великому обсязі даних та операціях, тому вона ідеально підходить для проєктів із невеликими обсягами даних та нескладними вимогами, але цього достатньо для станції технічного обслуговування. Під час розробки програми було проведено тестування для перевірки її ефективності та працездатності.

ВИСНОВКИ

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

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

Дипломній роботі було розглянуто ключові аспекти створення інформаційної системи для станцій технічного обслуговування автомобілів. Було проведено дослідження існуючих програмних рішень на ринку, включаючи розгляд їхніх переваг та недоліків. Це дозволило зробити обґрунтований вибір на користь власної розробки, що відповідає конкретним потребам та вимогам бізнесу.

У процесі дослідження було визначено, що для досягнення цієї мети оптимальним варіантом є використання технологій Microsoft Visual Studio та фреймворку Windows Forms для розробки програмного забезпечення. Цей вибір обґрунтований їхньою потужністю, зручністю використання та підтримкою від відомого виробника.

Таким чином, ця дипломна робота висвітлює важливість та актуальність розробки інформаційних систем для станцій технічного обслуговування автомобілів, а також надає конкретний шлях досягнення цієї мети шляхом використання відповідних технологій та підходів до розробки програмного забезпечення. Крім того, виконання цієї роботи сприяло глибшому розумінню сучасних технологій у сфері інформаційних систем та вдосконаленню навичок у аналізі, проектуванні та розробці програмного забезпечення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ali H. El-Mousa. A Web-Based Rapid Prototyping Workflow Management Information System for Computer Repair and Maintenance / Ali H. El-Mousa, Zahra J. Muhsin and Majid A. Al-Taee // Journal of Computer Science. – 4 (12), 991–998, 2008 – ISSN 1549–3636 – 6 p.
2. CA ERwin Process Modeler [Електронний ресурс] – Режим доступу до ресурсу: https://www.architect-design.ru/item.1011.html.
3. CarStore - Програмне забезпечення для автомагазинів та автосервісів [Електронний ресурс] – Режим доступу до ресурсу: https://www.carstore.com.ua/.
4. CASE-технологии [Електронний ресурс] – Режим доступу до ресурсу: https://piter-soft.ru/knowledge/glossary/process/case-tehnology.html.
5. CRM Appointer [Електронний ресурс] – Режим доступу до ресурсу: https://appointer.ua/.
6. EasyWeek [Електронний ресурс] – Режим доступу до ресурсу: https://easyweek.com.ua/programa-dlya-vetklinik.html.
7. Eckerson W. Smart Companies in the 21st Century: The Secrets to Creating Successful Business Intelligence Solutions / W. Eckerson // TDWI Report Series. – The Data Warehouse Institute, 2003. – 40 p. \
8. ExpresSsoft - Програма для автосервісу та СТО [Електронний ресурс] – Режим доступу до ресурсу: https://expresssoft.com.ua/uk/programa-dlja-avtoservisu-sto/
9. Rational Rose [Електронний ресурс] – Режим доступу до ресурсу: https://www.kpms.ru/Automatization/Rational_Rose.htm.
10. RemOnline - Програма для автосервісу та СТО [Електронний ресурс] – Режим доступу до ресурсу: https://remonline.ua/autoservice/.
11. SkyService POS [Електронний ресурс] – Режим доступу до ресурсу: https://skyservice.pro/.

12. Андрусенко С. І. Моделювання бізнес-процесів підприємства автосервісу: монографія / С. І. Андрусенко, О. С. Бугайчук. – К.: Кафедра, 2014. – 325 с.
13. Андрусенко С.І. Технології підвищення ефективності виробничо-технічної бази підприємств автомобільного транспорту: навчальний посібник / Андрусенко С.І., Бугайчук О.С. – К.: НТУ, 2017. – 190 с.
14. ДСТУ 2392-94. Інформація та документація. Базові поняття. Терміни та визначення.
15. Інформаційні технології та моделювання бізнес-процесів / О. М.Томашевський, Г. Г. Цегелик, М. Б. Вітер, В. І. Дубук. – Київ: Видавництво «Центр учбової літератури», 2012. – 296 с.
16. Литвишко Л.О. Порівняльна характеристика особливостей надання автосервісних послуг у США та Європі / Литвишко Л.О., Васьківська Н.В. // Вісник Національного транспортного університету. Серія «Економічні науки». – Випуск 3 (33), 2015. – С.197 – 204.
17. Марков О. Д. Станции технического обслуживания автомобилей. – К.: Кондор, 2018. – 536 с.CA ERwin Data Modeler [Електронний ресурс] – Режим доступу до ресурсу: https://www.administrator-pro.ru/cat/soft/ca_technologies/ca_erwin_data_modeler/.
18. МЕТОДОЛОГИЯ DFD [Електронний ресурс] – Режим доступу до ресурсу: https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_3.
19. Методология моделирования процессов IDEF3 [Електронний ресурс] – Режим доступу до ресурсу: http://asu.ugatu.ac.ru/library/56c1644b436bb/80ecddf06ad1fd912d8d973b0ec27b40.pdf.
20. Основные методологии обследования организаций. Стандарт IDEF0. [Електронний ресурс] – Режим доступу до ресурсу: https://www.cfin.ru/vernikov/idef/idef9.shtml.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace Service_Station
{
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        splitContainer1 = new SplitContainer();
        DgridMain = new DataGridView();
        panel4 = new Panel();
        label4 = new Label();
        textBoxFiltrByName = new TextBox();
        pictureBoxPhoto = new PictureBox();
        groupBox3 = new GroupBox();
        buttonNew = new Button();
        btnEdit = new Button();
        btnDelete = new Button();
        ((System.ComponentModel.ISupportInitialize)splitContainer1).BeginInit();
        splitContainer1.Panel1.SuspendLayout();
        splitContainer1.Panel2.SuspendLayout();
        splitContainer1.SuspendLayout();
        ((System.ComponentModel.ISupportInitialize)DgridMain).BeginInit();
        panel4.SuspendLayout();
    ((System.ComponentModel.ISupportInitialize)pictureBoxPhoto).BeginInit();
        groupBox3.SuspendLayout();
        SuspendLayout();
        splitContainer1.Dock = DockStyle.Fill;
```

```
splitContainer1.Location = new Point(0, 0);
splitContainer1.Name = "splitContainer1";
splitContainer1.Panel1.Controls.Add(DgridMain);
splitContainer1.Panel1.Controls.Add(panel4);
splitContainer1.Panel2.BackColor = Color.DarkBlue;
splitContainer1.Panel2.Controls.Add(pictureBoxPhoto);
splitContainer1.Panel2.Controls.Add(groupBox3);
splitContainer1.Panel2.Padding = new Padding(10);
splitContainer1.Size = new Size(753, 413);
splitContainer1.SplitterDistance = 423;
splitContainer1.TabIndex = 44;
label4.AutoSize = true;
label4.Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Bold, GraphicsUnit.Point);
label4.ForeColor = Color.White;
label4.Location = new Point(4, 12);
label4.Name = "label4";
label4.Size = new Size(88, 13);
label4.TabIndex = 1;
label4.Text = "Пошук за ПІБ";
textBoxFiltrByName.Anchor = AnchorStyles.Top | AnchorStyles.Left | AnchorStyles.Right;
textBoxFiltrByName.Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Regular, GraphicsUnit.Point);
textBoxFiltrByName.Location = new Point(98, 9);
textBoxFiltrByName.Margin = new Padding(3, 2, 3, 2);
textBoxFiltrByName.Name = "textBoxFiltrByName";
textBoxFiltrByName.Size = new Size(310, 20);
textBoxFiltrByName.TabIndex = 0;
textBoxFiltrByName.TextChanged += textBoxFiltrByName_TextChanged;
```

```
            pictureBoxPhoto.Dock = DockStyle.Fill;
            pictureBoxPhoto.Location = new Point(10, 10);
            pictureBoxPhoto.Margin = new Padding(3, 2, 3, 2);
            pictureBoxPhoto.Name = "pictureBoxPhoto";
            pictureBoxPhoto.Size = new Size(306, 232);
            pictureBoxPhoto.SizeMode = PictureBoxSizeMode.Zoom;
            pictureBoxPhoto.TabIndex = 46;
            pictureBoxPhoto.TabStop = false;
            // btnDelete
            btnDelete.Anchor = AnchorStyles.Top | AnchorStyles.Left |
AnchorStyles.Right;
            btnDelete.BackColor = Color.DarkBlue;
            btnDelete.Font = new Font("Segoe UI", 9F, FontStyle.Bold,
GraphicsUnit.Point);
            btnDelete.Location = new Point(6, 118);
            btnDelete.Name = "btnDelete";
            btnDelete.Size = new Size(294, 31);
            btnDelete.TabIndex = 45;
            btnDelete.Text = "Видалити вибраний запис";
            btnDelete.UseVisualStyleBackColor = false;
            btnDelete.Click += btnDelete_Click;
            AutoScaleDimensions = new SizeF(7F, 15F);
            AutoScaleMode = AutoScaleMode.Font;
            ClientSize = new Size(753, 413);
            Controls.Add(splitContainer1);
            Margin = new Padding(4);
            MinimumSize = new Size(643, 452);
            Name = "InfoClients";
            StartPosition = FormStartPosition.CenterScreen;
            Text = "Довідник клієнтів";
```

```
            Load += InfoClients_Load;
            splitContainer1.Panel1.ResumeLayout(false);
            splitContainer1.Panel2.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)splitContainer1).EndInit();
            splitContainer1.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)DgridMain).EndInit();
            panel4.ResumeLayout(false);
            panel4.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)pictureBoxPhoto).EndInit();
            groupBox3.ResumeLayout(false);
            ResumeLayout(false);
    using System.Data;
    using System.Data.SQLite;
    using System.Windows.Forms;
    using Service_Station.Service;
    namespace Service_Station
    {
       public partial class InfoClients : Form
       {
          SQLiteConnection? connection;
          SQLiteCommand? command;
          DataSet DSet;
          }
          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.Yellow;
            //Grid.DefaultCellStyle.SelectionForeColor = Color.Black;
            Grid.RowHeadersVisible = false;
            Grid.AllowUserToAddRows = false;
            Grid.ReadOnly = true;
            Grid.AutoResizeColumns();
        private void Set_conect()
        {
            try
            {
                if (connection is not null) connection.Close();
                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();
        private void InfoClients_Load(object sender, EventArgs e)
        {
            LoadData();
```

```
        if (DgridMain.Rows.Count > 0)
            DgridMain.Rows[0].Selected = true;
    private void textBoxFiltrByName_TextChanged(object sender, EventArgs e)
    {
        LoadData();
    int curPositionGrid(DataGridView Grid)
    {
        int row = -1;
        if (Grid.DataSource != null && Grid.BindingContext != null)
            row = Grid.BindingContext[Grid.DataSource].Position;
        return row;
    }
    int IdRow
    {
        get
        {
            int index = -1;
            if (DgridMain.SelectedRows.Count > 0)
            {
                DataGridViewRow selectedRow = DgridMain.SelectedRows[0];
                if (selectedRow != null)
                {
                    int.TryParse(selectedRow.Cells[0].Value.ToString(), out index);
                }
            }
            return index;
    private void LoadImageFromBaseToPicterBox(PictureBox pictureBox, string
selectQuery)
    {
        try
```

```
        {
            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); //
Встановлення зображення в PictureBox
                else
                    pictureBox.Image = WorkWithImage.NoPhoto();
            }
            connection.Close();
        }
        catch (Exception)
        {
            pictureBox.Image = WorkWithImage.NoPhoto();
    private void mTable_SelectionChanged(object sender, EventArgs e)
        LoadImageFromBaseToPicterBox(pictureBoxPhoto, $"SELECT image
FROM Clients WHERE client_id = {IdRow}");
    private void btnDelete_Click(object sender, EventArgs e)
    {
        int indexDel = IdRow;
        if (indexDel != -1)
        {
            if (MessageBox.Show("Видалити запис", "Видалення",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2) == DialogResult.No)
                return;
```

```
            Set_conect();
            if (connection == null) { return; }
            command = connection.CreateCommand();
            command.CommandText = $"DELETE FROM Clients WHERE client_id
= '{indexDel}'";
            object result = command.ExecuteScalar();
            connection.Close();
            if (result is int)
            {
          int rowSelect = curPositionGrid(DgridMain);
          int indexSelect = IdRow;
          if (indexSelect == -1)
          {
            MessageBox.Show("Не вибрано запис для редагування",
"Редагування", MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
          using (var childFormDirect = new EditClient(indexSelect))
          {
            childFormDirect.Text = "Редагування вибраного клієнта";
            if (childFormDirect.ShowDialog() == DialogResult.Yes)
            {
              LoadData();
              if (DgridMain.Rows.Count > 0)
              {
                DgridMain.ClearSelection();
                DgridMain.Rows[rowSelect].Selected = true;
                DgridMain.FirstDisplayedScrollingRowIndex = rowSelect;
    }
}
```