Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система технічного обслуговування автомобілів*

Виконав: студент *IV* курсу, групи *СІс-44*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

		Ковач О.С.
	(підпис)	(прізвище та ініціали)
Керівник		*Луцків А.М.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Луцик Н.С.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г.М.*
	(підпис)	(прізвище та ініціали)
Рецензент		*Стадник М.А.*
	(підпис)	(прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« » 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Ковачу Олександру Сергійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система технічного обслуговування автомобілів*

Керівник роботи *Луцків Андрій Мирославович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*10*» *лютого* 2021 року № *4.7-97*

2. Термін подання студентом завершеної роботи *26.06.2021 р.*

3. Вихідні дані до роботи *Дані про працівників, інформація про автомобілі та їх власників, види ремонту*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз вимог та способів організації комп'ютеризованої системи обслуговування автомобілів. 2. Проектування комп'ютеризованої системи обслуговування автомобілів 3. Реалізація і тестування програмного забезпечення комп'ютеризованої системи обслуговування автомобілів. 4. Безпека життєдіяльності, основи охорони праці. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

1. *Діаграма потоків даних*
2. *Діаграма варіантів використання*
3. *Різновиди технології клієнт-сервер*
4. *Архітектура комп'ютеризованої системи*
5. *ER-діаграма бази даних*
6. *Алгоритм модифікації даних*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*Пилипець М.І., д.т.н., проф. каф. МТ*		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка та аналіз вимог технічного завдання*	*10.02-19.02.2021*	
2	*Аналіз способів організації комп'ютеризованих систем*	*19.02-05.03.2021*	
3	*Проектування комп'ютеризованої системи обслуговування автомобілів*	*05.03-26.03.2021*	
4	*Обґрунтування вибору апаратного забезпечення*	*26.03-01.04.2021*	
5	*Реалізація і тестування програмного забезпечення комп'ютеризованої системи*	*01.04-22.04.2021*	
6	*Розробка інструкцій із встановлення та налаштування параметрів комп'ютеризованої системи обслуговування автомобілів*	*22.04-10.05.2021*	
7	*Безпека життєдіяльності, основи охорони праці*	*10.05-18.05.2021*	
8	*Оформлення кваліфікаційної роботи*	*18.05-06.06.2021*	
9	*Попередній захист кваліфікаційної роботи*	*06.06-18.06.2021*	
10	*Захист кваліфікаційної роботи*	*21.06-27.06.2021*	

Студент _____ *Ковач Олександр Сергійович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Луцків Андрій Мирославович*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютеризована система технічного обслуговування автомобілів // Кваліфікаційна робота на здобуття освітнього ступеня бакалавр // Ковач Олександр Сергійович // ТНТУ, спеціальність 123 «Комп’ютерна інженерія»// Тернопіль, 2021 // с.– 63, рис. – 32 , табл. – 21, аркушів А1 – 6, бібліогр. – 23.

Ключові слова: система, обслуговування, автомобіль, ремонт, управління.

У кваліфікацій роботі бакалавра розроблено комп’ютеризовану систему обслуговування автомобілів. На основі аналізу предметної області побудовано діаграму потоків даних та визначено функціональні вимоги до системи, які візуалізовано засобами діаграми варіантів використання. Окрім цього, для зберігання та маніпулювання даними спроектовано схему бази даних, що відображає важливі, з точки зору процесу автоматизації аспекти предметної області.

Базу даних реалізовано із застосуванням реляційного підходу до її проектування та інструментальними засобами СКБД MS SQL Server, зокрема утилітою MS SQL Server Management Studio.

У роботі обгрунтовано доцільність використання архітектурного патерну клієнт-сервер, спроектовано архітектуру програмного забезпечення комп’ютеризованої системи обслуговування автомобілів, враховано вимоги та аспекти до його використання.

Реалізацію програмної складової комп’ютеризованої системи виконано базуючись на підході об’єктно-орієнтованого програмування та за допомогою мови програмування C#, середовищем програмування обрано MS Visual Studio.

ABSTRACT

Car service computer-aided system // Bachelor’s thesis // Kovach Oleksandr Serhiyovych // TNTU, speciality 123 «Computer engineering»// Ternopil, 2021 // p.– 63 , fig. – 32 , tab. – 21, posters A1 – 6, ref. – 23.

Keywords: system, service, car, repair, management.

The computer service system has been developed in the bachelor's qualifications. Based on the analysis of the subject area, a diagram of data flows is constructed and the functional requirements to the system are determined, which are visualized by means of the diagram of use cases. In addition, a database schema has been designed to store and manipulate data, reflecting important aspects of the subject area from the point of view of the automation process.

The database is implemented using a relational approach to its design and tools of the MS SQL Server database, in particular the utility MS SQL Server Management Studio. The expediency of using the client-server architectural pattern is substantiated in the work, the software architecture of the computerized car service system is designed, the requirements and aspects to its use are taken into account.

The implementation of the software component of the computerized system is based on the approach of object-oriented programming and using the C # programming language, the programming environment is MS Visual Studio.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ,
СИМВОЛІВ І СКОРОЧЕНЬ

БД	База даних
КС	Комп’ютерна система
ПЗ	Програмне забезпечення
СКБД	Система керування базами даних
ADO	ActiveX Data Objects
SQL	Structured Query Language
UML	Unified Modelling Language

ВСТУП

Сучасний розвиток сфер обслуговування в Україні вимагає впровадження засобів автоматизації та інформаційних технологій для віддаленого управління процесами бізнес-діяльності. Тому імплементація комп'ютеризованих систем для різних галузей діяльності завжди є актуальною задачею, особливо для областей, які слабо піддаються автоматизації. Це стосується агросектору, обслуговування автомобілів, віддаленого керування технологічними процесами виробництва і т.п.

Враховуючи тенденції щодо здешевлення автомобілів закордоном та низьку купівельну спроможність громадян України, набули широкої популярності ввезення авто, які були в експлуатації у країнах Європи, США, Кореї та ін. Зазвичай, українці купують вживані транспортні засоби на аукціонах по типу Copart, Dashub, Auto Auction Mall, ADESA. При цьому більшість авто потребують ремонту, а це в свою чергу стимулює розвиток та створення нових сервісних центрів та станцій технічного обслуговування автомобілів. Облік діяльності щодо обслуговування та ремонту транспортних засобів є важливим для власників бізнесу та клієнтів, оскільки дозволяє забезпечити процеси щодо сплати податків, сервісного та гарантійного обслуговування. Враховуючи такі тенденції, на сьогодні гостро постає питання автоматизації діяльності станцій технічного обслуговування і вирішення цієї проблеми можна забезпечити шляхом створення і побудови комп'ютеризованих систем обслуговування транспортних засобів.

Кваліфікаційна робота присвячена власне проектуванню та реалізації такої системи, що може ефективно використовуватись сервісними центрами і станціями технічного обслуговування. Метою роботи є розробка проекту системи для підтримки процесу обслуговування автомобілів з відповідним обгрунтуванням апаратного і програмного забезпечення.

РОЗДІЛ 1 АНАЛІЗ ВИМОГ ТА СПОСОБІВ ОРГАНІЗАЦІЇ КОМП'ЮТЕРИЗОВАНОЇ СИСТЕМИ ОБСЛУГОВУВНАНЯ АВТОМОБІЛІВ

1.1 Аналіз технічного завдання при розробці комп'ютеризованої системи обслуговування автомобілів

Комп'ютеризована система обслуговування автомобілів призначена для автоматизації та контролю виконаних робіт із сервісного та гарантійного обслуговування клієнтів станції технічного обслуговування.

Комп'ютеризована система повинна включати в себе як апаратну складову, так і програмну частину. У кваліфікаційній роботі необхідно проаналізувати апаратне забезпечення підприємства, що займається таким видом діяльності, організацію та канали зв'язку між активними та пасивними компонентами комп'ютерної мережі, організаційну структуру підприємства та наявні операційні системи і прикладне програмне забезпечення користувачів.

При цьому потрібно здійснити їх оцінку, провести вибір альтернативних рішень та розробити програмний комплекс для ведення та управління процесом обслуговування автомобілів. При реалізації програмного комплексу необхідно спроектувати базу даних та налаштувати сервер бази даних, а також створити зручний користувацький інтерфейс для роботи з нею. Комп'ютеризована система обслуговування автомобілів повинна генерувати звіти та візуалізувати результати роботи по кожному авто та автослюсарю за вказаний період часу. Користувачами системи є майстри цехів, бухгалтерія та адміністрація підприємства.

Метою створення комп'ютеризованої системи обслуговування автомобілів є автоматизація процесів обліку та управління щодо надання

сервісних послуг власникам автотранспорту. Для досягнення мети потрібно розв’язати наступні завдання:

– автоматизація процесу сервісного та гарантійного обслуговування автомобілів;

– автоматизація процесу контролю використання запчастин;

– автоматизація процесу обліку робочого часу автослюсарів;

– автоматизація процесу нарахування заробітної плати автослюсарам підприємства;

– підвищення ефективності роботи підприємства;

– формування звітів з гарантійного та сервісного обслуговування автомобілів;

– генерування графіків завантаженості автослюсарів та продуктивності їх праці.

Основні задачі, вирішення яких покладається на комп’ютеризовану систему обслуговування автомобілів полягають в наступному:

– забезпечення зв’язку між підрозділами підприємства;

– ведення обліку сервісного обслуговування автомобілів;

– ведення обліку гарантійного обслуговування автомобілів;

– обробка звітної інформації;

– збір та обробка статистичних даних з технічного обслуговування автомобілів.

Функціями комп’ютеризованої системи є автоматизація обліку автомобілів, нарахування заробітної плати, і як наслідок підвищення продуктивності та контролю праці автослюсарів станцій технічного обслуговування.

Вимоги до комп’ютеризованої системи обслуговування автомобілів, які застосовуються станціями технічного обслуговування включають:

– надійність роботи апаратної складової інформаційної системи підприємства;

– відповідність встановленому рівню продуктивності апаратних пристроїв;

– захищеність обладнання;

– захищеність доступу до ресурсів;

– точність обробки даних;

– продуктивність роботи програмного забезпечення;

– паралельний доступ до бази даних;

– розмежування прав доступу;

– часова ефективність;

– ефективність використання ресурсів комп’ютеризованої системи;

– зручність використання програмного забезпечення.

До структури комп’ютеризованої системи обслуговування автомобілів входить: апаратна частина та програмна складова. Вимоги до апаратної частини:

1. Надійність функціонування апаратної складової комп’ютеризованої системи:

– час безвідмовної роботи – 100000 год.;

– наявність системи резервного живлення;

– захист від стрибків напруги;

– надійність роботи складових апаратного забезпечення;

– надійність каналів зв’язку інформаційної системи;

– захист від неавторизованого доступу;

2. Продуктивність апаратних пристроїв:

– пропускна здатність каналів зв’язку – >=100 Мб/с;

– максимальний розмір опрацьовуваних даних – 100 МБ;

3. Захищеність апаратної складової:

– фізичний захист активного обладнання;

– фізичний захист каналів зв’язку;

– фізичний та логічний захист комутаційного обладнання;

4. відновлюваність та резервування:

– аварійне відновлення працездатності після збою – 24 год.;

– резервування каналів зв’язку.

Вимоги до програмної частини комп’ютеризованої системи:

1. Функціональність:

– наповнення та робота з довідником послуг та їх вартості;

– наповнення довідників про автослюсарів;

– наповнення та робота з довідниками авто та їхнє гарантійне обслуговування;

– ввід, редагування даних по виконаних роботах;

– нарахування заробітної плати автослюсарям;

– можливість сортування даних за визначеними критеріями;

– можливість запобігання неавторизованому доступу (логічного);

– можливість формування звітів по облікованому робочому часі;

– можливість керування правами доступу до інформаційних ресурсів;

– облік та контроль робочого часу;

2. Продуктивність:

– час реакції – 1 с;

– час відгуку – 2с;

– ефективність використання оперативної пам’яті;

– ефективність використання дискового простору;

– доступність;

– розмежування та визначення прав доступу;

– масштабованість архітектури;

3. Надійність:

– завершеність;

– стійкість до відмов;

– напрацювання на відмови;

– здатність до відновлення;

4. Зручність використання:

– однорідність стильового оформлення;

– наявність довідки;

– зрозумілість виконання операцій;
– зрозумілість одержаних результатів;
– зручність навчання.

Структура апаратної складової комп’ютеризованої системи – робочі станції, сервер баз даних, канали зв’язку, комутаційне обладнання.

Структура програмного забезпечення комп’ютеризованої системи проектується на основі архітектури клієнт-сервер.

В загальному випадку, інфологічна модель бази даних повинна відображати предметну область, а клієнтська частина – відповідати за можливості обліку даних та забезпечення їх захисту.

Зв’язок між компонентами комп’ютеризованої системи: робочі станції – сервер обробки даних – протокол TCP/IP (рівень операційної системи).

Діагностика системи відбувається згідно з планом супроводу та обслуговування комп’ютеризованої системи в цілому та окремих її складових. Модернізація системи можлива у випадку морального старіння апаратного або програмного забезпечення, або ж у випадку радикальної зміни вимог до інформаційної системи. Перспективи розвитку системи можуть включати зміну на більш сучасну апаратну частину і як наслідок модернізація програмної складової.

Комп’ютеризована система обслуговування автомобілів повинна бути захищена на рівні операційної системи та авторизованого доступу до бази даних. Надійність системи повинна забезпечуватись також і у випадку збою роботи апаратного забезпечення.

Вимоги до функцій та задач, які виконує система, включають:
– забезпечення зв’язку клієнтської частини з базою даних;
– надання точних та адекватних результатів на запит користувачів;
– забезпечення часової ефективності роботи системи;
– забезпечення контролю над доступом до інформації;
– забезпечення зручності використання програмного продукту;
– можливість розгортання бази даних.

Вимоги до сервера:

– процесор – тактова частота не менше 4,0 ГГц;

– об'єм оперативної пам'яті - не менше 4096 Мб;

– об'єм жорсткого диску - не менше 750 Гб.

Вимоги до робочих станцій:

– процесор - тактова частота не менше 2,2 ГГц;

– об'єм оперативної пам'яті - не менше 2048 Мб;

– об'єм жорсткого диску - не менше 250 Гб.

Периферійні пристрої:

– принтер або мультифункціональний пристрій – 5 шт.

Вимоги до програмного забезпечення сервера визначають платформу операційної системи на базі Windows Server 2012 R2.

Вимоги до програмного забезпечення робочих станцій включають використання платформи Windows 10 та прикладного програмного забезпечення, що використовується для забезпечення необхідного рівня продуктивності праці.

1.2 Способи організації комп’ютеризованих систем обслуговування автомобілів

При проектуванні комп’ютеризованої системи обслуговування автомобілів та подібних систем необхідно врахувати такі аспекти, як централізоване зберігання та управління даними, а також наявність багатьох користувачів, які можуть одночасно використовувати дані. Тому для вирішення цієї задачі доцільним є застосування архітектури клієнт-сервер.

Враховуючи той факт, що технології клієнт-сервер пройшли декілька стадій свого розвитку, потрібно провести детальний аналіз і визначити переваги кожного з різновидів.

В загальному випадку, архітектура клієнт-сервер характеризується застосування системи керування базами даних (СКБД), що передбачає наявність трьох основних компонент, які забезпечують наступні функції:

- ввід і представлення даних у зручному для кінцевого користувача вигляді;
- забезпечення функцій формування запитів і процедур для одержання даних;
- управління зарезервованими ресурсами.

При проектуванні комп'ютеризованої системи обслуговування автомобілів обов'язковим є наявність таких компонент як:

- засоби представлення даних;
- засоби прикладного рівня;
- засоби управління програмними та апаратними ресурсами.

Для організації взаємодії між визначеними компонентами системи застосовуються відповідні «протоколи взаємодії», що визначають правила комунікації між програмними модулями.

Загальноприйнятою та найпоширенішою класифікацією архітектур «клієнт-сервер» є класифікація за способом взаємодії компонентів та їхнім типом, що формують наступні моделі:

- модель на основі файл-сервера (рис. 1.1, а);
- модель, що передбачає віддалений доступ (рис. 1.1, б);
- модель, що використовує сервер баз даних (рис. 1.1, в);
- модель на основі організації сервера управління додатками (рис. 1.1, г).

Першим різновидом архітектури клієнт-сервер є модель розподіленого доступу і представлення даних, яка була впроваджена на універсальній електро-обчислювальній машині до якої звертались термінали без інтелектуальної складової. Забезпечення доступу до даних та керування ними з позиції кінцевого користувача виконується єдиною програмою, а термінал одержував лише знімок потрібної інформації, що формувалась на центральній машині.

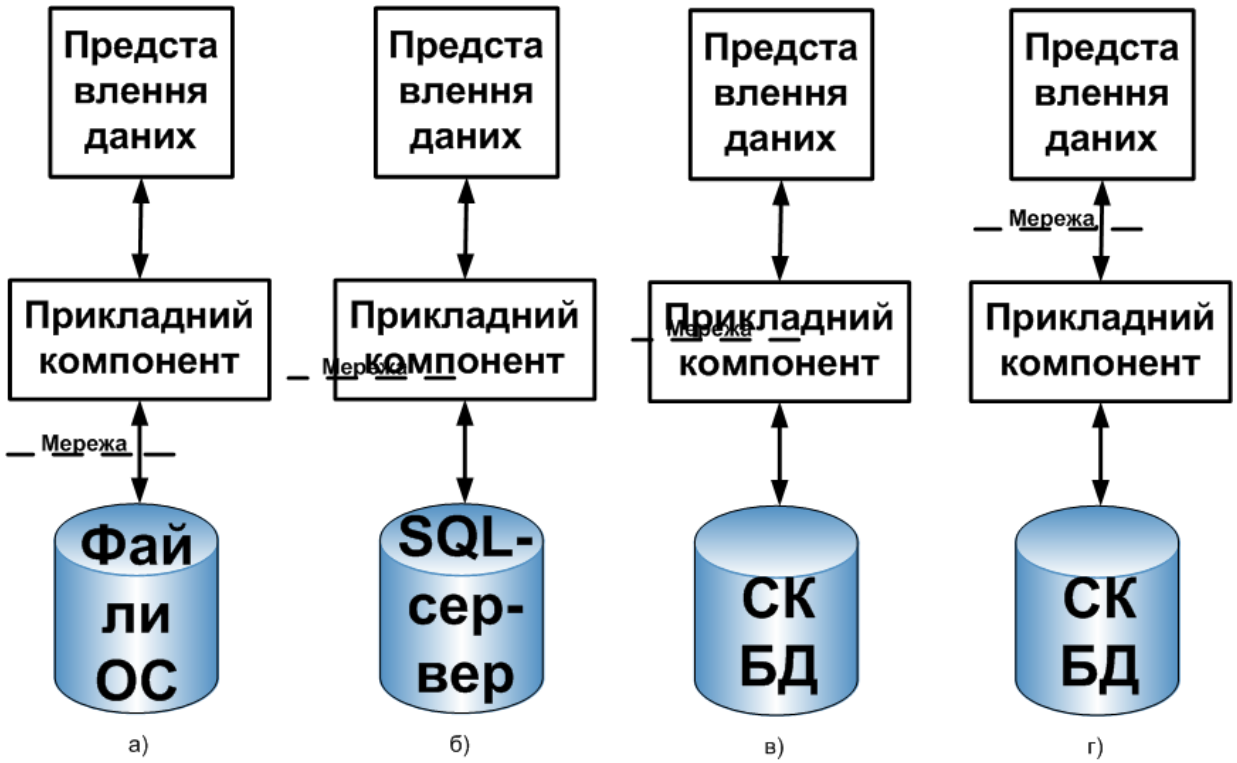

Рисунок 1.1 – Різновиди архітектури «клієнт-сервер»:
а) модель файл-сервер; б) модель віддаленого доступу; в) модель на основі СКБД; г) модель прикладного додатку

Протягом тривалого часу архітектура файлового сервера була найбільш широко використовуваною. При такій організації комп'ютеризованих систем передбачалась наявність файлового сервера та клієнтських станцій на яких виконувались програмні додатки. Характерною особливістю клієнтів є те, що компоненти представлення і прикладного рівня розміщувались на ньому. Протокол взаємодії між файловим сервером і клієнтом забезпечується виконанням сукупності низькорівневих команд на рівні файлової системи.

До недоліків такої архітектури відносять:

– застосування персональних СКБД на клієнтських станціях;
– високе навантаження на трафік комп'ютерної мережі;
– неуніфковані методи доступу до ресурсів комп'ютеризованої системи.

Схема взаємодії при використанні архітектури файл-сервер показана на рис. 1.2.

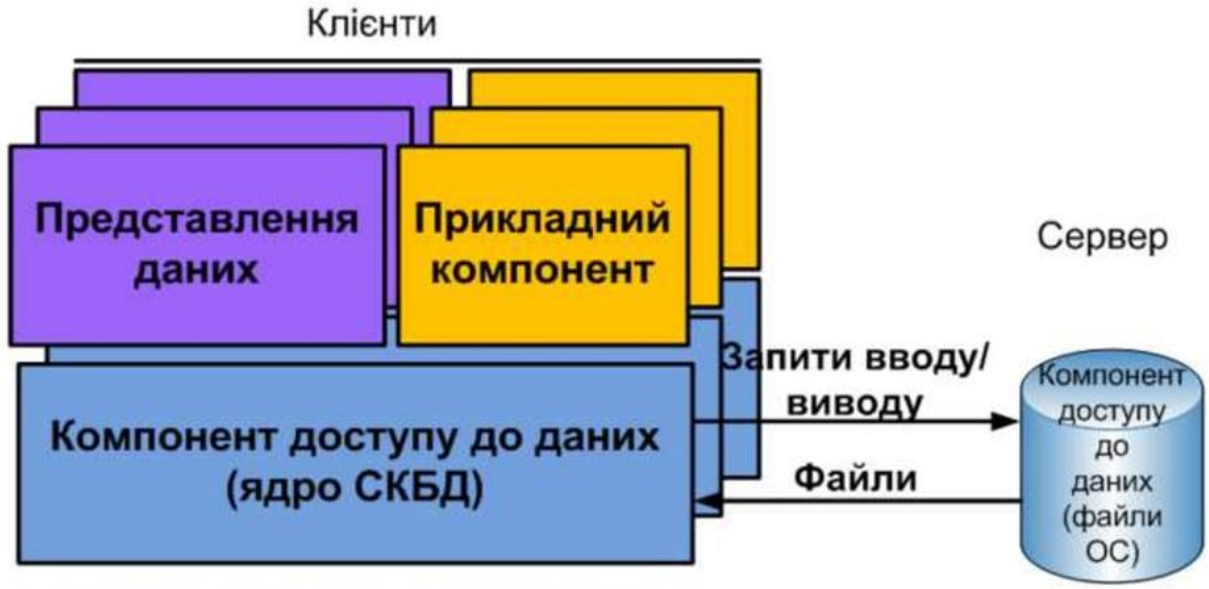

Рисунок 1.2 – Схема взаємодії при використанні архітектури файл-сервер

До особливостей організації систем на основі файлового сервера належать:

- усі важливі компоненти системи розташовуються на клієнтських станціях;
- модель, наведена на рис. 1.2, описує лише загальний підхід щодо взаємодії програмно-апаратних компонентів системи у комп’ютерній мережі, а не спосіб її побудови;
- завжди існує лише один комп'ютер, який виконує функції файлового сервера на якому зберігаються дані різного виду;
- файловий сервер є пасивним.

Пасивність файлового сервера проявляється у тому, що він лише зберігає дані і не виконує жодних додаткових функцій, пов’язаних з опрацюванням звернень та селекцією даних. Функції забезпечення цілісності бази даних, прийому і передачі даних, а також логічного їх опрацювання покладено на прикладний додаток клієнтської станції. Зважаючи на це, кількість даних, які передаються по локальній комп’ютерній мережі значно перевищує обсяг тієї інформації, яка необхідна кінцевому користувачу.

Для прикладу, при організації комп’ютеризованої системи обслуговування автомобілів, користувачів системи цікавить інформація про працівників станції технічного обслуговування та виконанні ними ремонти авто. У цьому випадку одержують дані про всіх працівників, після цього інформацію про усі виконанні ремонти, а вже тоді буде виконано вибірку даних, яка відповідатиме запиту користувача.

Модель file server покладено в основі функціонування таких систем керування базами даних як dBase, Paradox та ін, які були популярними на початку 2000 років. Наведені СКБД не вимагали значних апаратних ресурсів щодо продуктивності, а програмна реалізація не була розподіленою.

До недоліків моделі файл-сервер належать:

- невисока продуктивність, пов’язана з передачею проміжних даних по шині з малою пропускною здатністю та малопотужним прикладним програмним забезпеченням на стороні клієнта;
- високе навантаження на трафік комп’ютерної мережі;
- слабка придатність до масштабування;
- недосконаліть або відсутність засобів організації захисту бази даних;
- відсутність механізмів розподіленого опрацювання транзакцій.

У зв’язку з переліченими недоліками застосування даної моделі не є доцільним при організації комп’ютеризованої системи обслуговування автомобілів та при проектуванні розподілених комп’ютерних систем.

Поява персональних комп’ютерів та широке впровадження локальних комп’ютерних мереж стимулювали розвиток та створення моделі віддаленого доступу. Схема взаємодії компонентів при застосуванні такої технології показана на рис. 1.3.

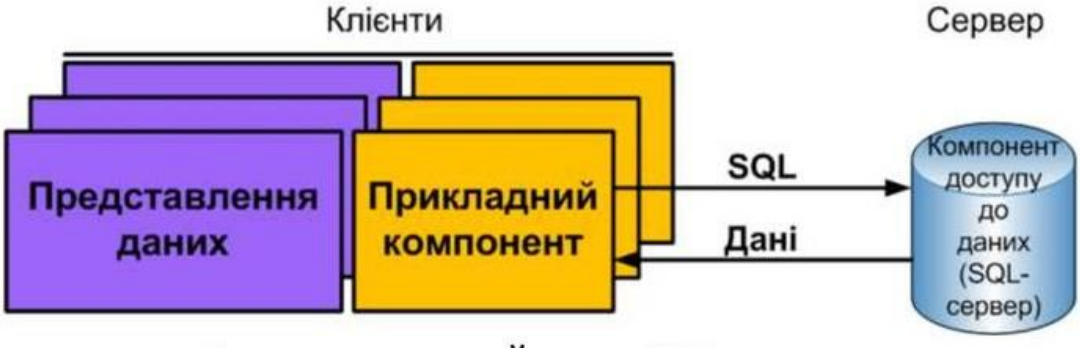

Рисунок 1.3 – Схема взаємодії компонентів при використанні моделі віддаленого доступу

У випадку використання технології віддаленого доступу система керування базами даних розміщена на відповідному сервері, а в якості протоколу обміну даними виступають запити мовою SQL.

Даний підхід до організації комп'ютеризованих систем у порівняні з попереднім (FS), дає змогу знизити навантаження на ресурси комп'ютерної мережі та забезпечити уніфікацію взаємодії між клієнтом і сервером. Проте, незважаючи на зниження інтенсивності трафіку, він все ж залишається доволі високим. Управління та адміністрування програмних додатків забезпечити практично неможливо, оскільки в одній системі інтегровано різні взаємозалежні функції.

Розвитком моделі віддаленого доступу з пасивним сервером є концепція, що передбачає використання активного сервера (рис. 1.4).

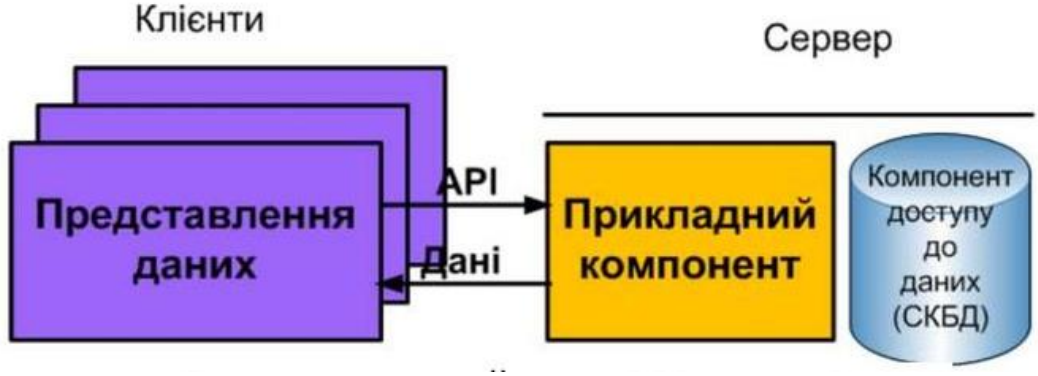

Рисунок 1.4 – Модель клієнт-сервер на основі СКБД

Технологія, наведена на рис. 1.4, дозволяє забезпечити міграцію частини функцій прикладного додатку на серверну сторону і утворює так звану розподілену модель. При такій організації, передбачається застосування процедур функціонального управління, які містяться у довіднику БД та розподіляються між багатьма клієнтами і реалізуються сервером.

До переваг такої технології організації клієнт-серверної архітектури належать:

– наявність механізмів централізованого адміністрування;

– зниження навантаження на мережевий трафік за рахунок виклику stored процедур, а не передачі SQL-запитів.

Основним недоліком технології на основі СКБД є недостатня розвиненість засобів розробки у порівнянні з мовами програмування високого рівня.

З практичної точки зору, для ефективної побудови комп'ютеризованої системи обліку чи управління необхідне застосування гібридних підходів, що передбачає дотримання наступних правил:

- збережені процедури повинні виконувати прості елементарні функції на стороні сервера;
- складна логіка опрацювання даних реалізується на стороні клієнта безпосередньо у прикладному додатку.

Поряд з розглянутими підходами, використовується також модель тонкого клієнта, що передбачає лише віддалене відображення даних.

Окрім цього, ефективними механізмами проектування комп'ютеризованих систем є підходи, базовані на моделі декомпозиції додатку. Характерною особливістю такої моделі є логічна декомпозиція прикладного додатку на кілька рівнів з можливістю виконання на різних клієнтах. Обмін повідомленнями між частинами прикладного застосування відбувається за заздалегідь погодженою схемою і форматом.

Таким чином, дворівнева організація комп'ютеризованої системи перетворюється у трирівневу, або модель із ще більшою кількістю шарів. На рис. 1.5 показано трирівневу організацію архітектури клієнт-сервер.

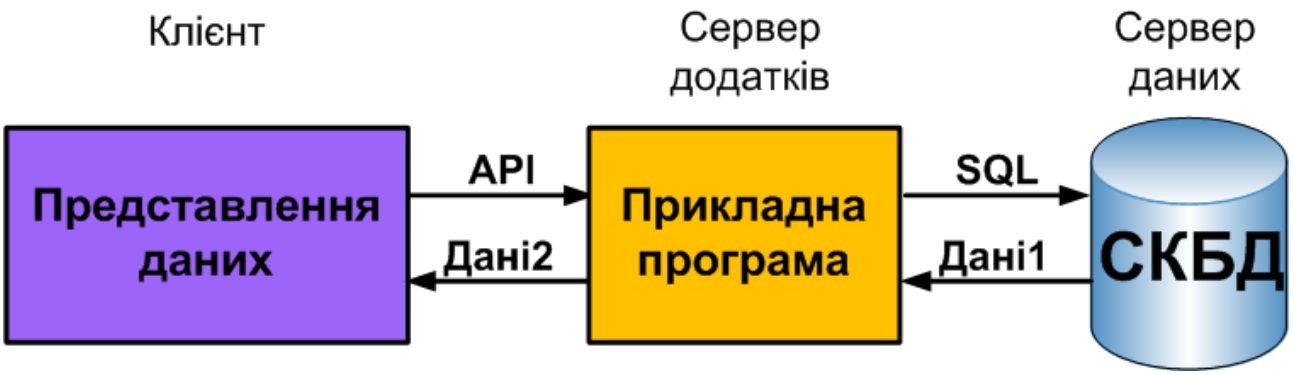

Рисунок 1.5 – Трирівнева клієнт-серверна архітектура

При організації комп’ютеризованих систем опрацювання даних можливе розташування інформації на локальному або віддаленому вузлі. Оскільки, при організації комп’ютеризованої системи обслуговування автомобілів передбачається більше, ніж один користувач, то доцільним є використання трирівневої архітектури клієнт-сервер з віддаленим доступом.

РОЗДІЛ 2 ПРОЕКТУВАННЯ КОМП'ЮТЕРИЗОВАНОЇ СИСТЕМИ ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

2.1 Виявлення та аналіз процесів технічного обслуговування автомобілів

Комп'ютеризована система розробляється з метою автоматизації процесів обліку та аналізу наданих послуг з технічного обслуговування автомобілів, що може бути використана відповідними сервісними станціями. Для досягнення мети роботи необхідно реалізувати програмний засіб, який би задовольняв визначені у технічному завданні вимоги, реалізовував розподіл прав та відповідальності щодо внесених даних і змін.

Основними вимогами до функціональних можливостей програмного засобу є:

– наявність засобів введення та редагування інформації у довідниках деталей, клієнтів, майстрів та підрозділів підприємства;

– прийом та реєстрація автотранспорту для ремонту, реєстрація відомостей про власника та технічний стан транспорту;

– реєстрація залишків деталей на складі, формування замовлень на необхідні деталі;

– супроводження процесу обробки заявки, відстеження статусу ремонту;

– відстеження переміщень авто, що ремонтуються та деталей між складами і пунктами;

– контроль за ресурсною базою та активами підприємства, поповнення необхідних об'ємів ресурсів;

– надання користувачам засобів для швидкого отримання інформації, можливостей її аналізу та досягнення управлінських рішень;

– облік робочого часу майстрів;

– облік наданих послуг та задіяних ресурсів;

– нарахування заробітної плати автослюсарів;

На основі аналізу вимог до функціональних можливостей програмної системи виділено основні користувацькі ролі між якими потрібно розподілити виконання прикладних задач. У табл. 2.1 представлено рольову модель при організації комп'ютеризованої системи обслуговування автомобілів

Таблиця 2.1 – Рольова модель комп'ютеризованої системи

Назва ролі	Опис ролі	Доступні дії
Приймальні пункти	Приймальні пункти виконують прийом транспортних засобів на обслуговування, реєструють їх та вказують характерні дані.	– прийом замовлення
Майстер	Проводить діагностику, сервісне обслуговування та ремонт транспортного засобу	– редагування даних заявки; – замовлення деталі;
Технічний відділ	Технічний відділ виконує управління роботою майстрів, замовляє деталі відповідно до заявок, виконує калькуляцію вартості заявки.	– облік майстрів; – облік та замовлення деталей; – формування підсумків проведених робіт.
Бухгалтерія	Проводить облік та аналіз робочого часу та завантаженості майстрів, розрахунок заробітної плати	– формування звітів; – облік та розрахунок заробітної плати.
Адміністрація	Аналіз інформації, перегляд звітів	– формування звітів – аналіз даних.

Розроблені функціональні вимоги представлено у вигляді UML – моделі прецедентів (рис. 2.1). Відповідно до визначених прикладних задач та

функціональних вимог повинен плануватись та реалізуватись функціонал клієнтського прикладного засобу.

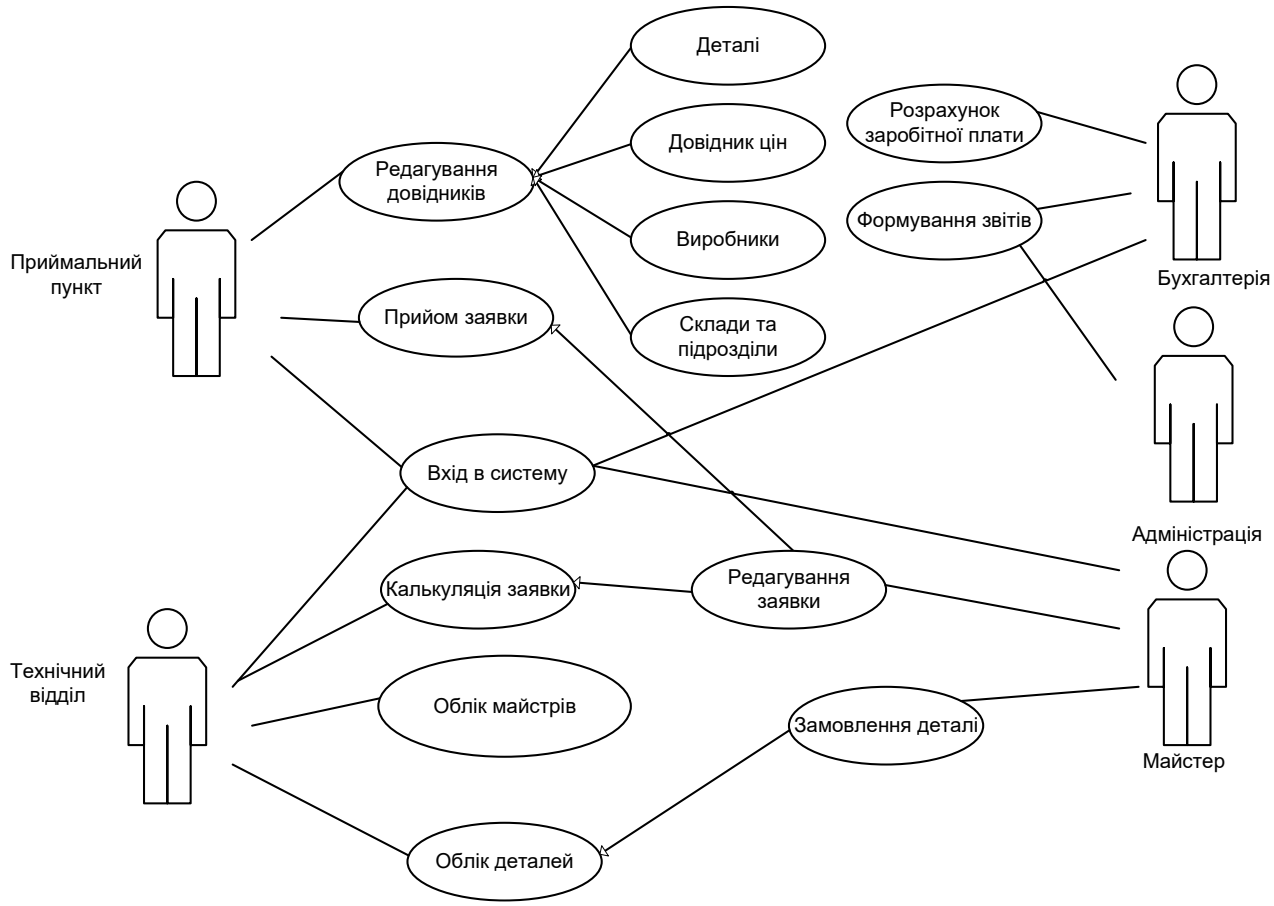

Рисунок 2.1 – Діаграма прецедентів

Побудувавши діаграму варіантів використання та визначивши основні бізнес-процеси, наступний крок полягає в аналізі предметної область для того, щоб спроектувати схему бази даних.

2.2 Визначення сутностей предметної області в процесі обслуговування автомобілів

Аналіз предметної області є важливим етапом проектування комп’ютеризованої системи, оскільки дає змогу виявити важливі сутності у процесах та зв’язки між ними.

Діаграма потоків даних забезпечує наглядність опису процесів, які притаманні сфері технічного обслуговування автомобілів. На рис. 2.2 показано, яким чином циркулює інформація при реалізації комп'ютеризованої системи обслуговування автомобілів.

Рисунок 2.2 – Діаграма потоків даних при проектуванні комп'ютеризованої системи обслуговування автомобілів

У результаті аналізу предметної області визначено основні сутності, інформацію про які необхідно зберігати у базі даних. Представлення сутностей наведено у вигляді таблиць 2.2 – 2.14.

Таблиця 2.2 – Сутність «Клієнт»

Атрибут	Тип атрибута	Примітка
#ID_Клієнт	Цілочисельний	Первісний ключ
Прізвище та ініціали	Текст	
Адреса	Текст	
Телефон	Текст	
Електронна адреса	Текст	

Таблиця 2.3 – Сутність «Майстер»

Атрибут	Тип атрибута	Примітка
#ID_Майстер	Цілочисельний	Первісний ключ
Прізвище та ініціали	Текст	
#ID_Відділ	Цілочисельний	Зовнішній ключ

Таблиця 2.4 – Сутність «Відділ»

Атрибут	Тип атрибута	Примітка
#ID_Відділ	Цілочисельний	Первісний ключ
Назва відділу	Текст	

Таблиця 2.5 – Сутність «Виробник»

Атрибут	Тип атрибута	Примітка
#ID_Виробник	Цілочисельний	Первісний ключ
Назва	Текст	

Таблиця 2.6 – Сутність «Тип транспорту»

Атрибут	Тип атрибута	Примітка
#ID_Тип	Цілочисельний	Первісний ключ
Назва	Текст	

Таблиця 2.7 – Сутність «Транспорт»

Атрибут	Тип атрибута	Примітка
#ID_Транспорт	Цілочисельний	Первісний ключ
#ID_Тип	Цілочисельний	Зовнішній ключ
#ID_Виробник	Цілочисельний	Зовнішній ключ
Модель	Текст	
Номер гарантії	Текст	

Таблиця 2.8 – Сутність «Деталь»

Атрибут	Тип атрибута	Примітка
#ID_Деталь	Цілочисельний	Первісний ключ
#ID_Виробник	Цілочисельний	Зовнішній ключ
Номер деталі	Текст	

Таблиця 2.9 – Сутність «Вид товару»

Атрибут	Тип атрибута	Примітка
#ID_Вид_товару	Цілочисельний	Первісний ключ
Назва	Текст	

Таблиця 2.10 – Сутність «Склад»

Атрибут	Тип атрибута	Примітка
#ID_Склад	Цілочисельний	Первісний ключ
Назва	Текст	
Тип	Текст	

Таблиця 2.11 – Сутність «Статус ремонту»

Атрибут	Тип атрибута	Примітка
#ID_Статус_ремонту	Цілочисельний	Первісний ключ
Назва типу	Текст	

Таблиця 2.12 – Сутність «Ремонт»

Атрибут	Тип атрибута	Примітка
#ID_Ремонт	Цілочисельний	Первісний ключ
#ID_Майстер	Цілочисельний	Зовнішній ключ
#ID_Транспорт	Цілочисельний	Зовнішній ключ
#ID_Клієнт	Цілочисельний	Зовнішній ключ
Дата прийому	Дата	
Дата готовності	Дата	
Дата видачі	Дата	
#ID_Статус_ремонту	Цілочисельний	Зовнішній ключ
Серійний номер	Текст	
Заявлений дефект	Текст	
Діагностика	Текст	
Висновок	Текст	

Таблиця 2.13 – Сутність «Ціна»

Атрибут	Тип атрибута	Примітка
#ID_Ціна	Цілочисельний	Первісний ключ
Ціна	Грошова одиниця	
Назва	Текст	
#ID_Вид товару	Цілочисельний	Зовнішній ключ
#ID_Деталь	Цілочисельний	Зовнішній ключ

Таблиця 2.14 – Сутність «Оплата»

Атрибут	Тип атрибута	Примітка
#ID_Оплата	Цілочисельний	Первісний ключ
Дата оплати	Дата	
Документ про оплату	Текст	
#ID_Ремонт	Цілочисельний	Зовнішній ключ
Сума	Грошова одиниця	

У подальшому аналіз предметної області має також на меті досягти оптимальності при проектуванні сховища даних та оптимізації схеми бази даних. З метою деталізації схеми сховища даних необхідно виконати аналіз зв’язків між сутностями предметної області та з’ясувати їх кратність.

Між сутностями предметної області існують відповідні зв’язки, що повинні підтримуватись через зовнішні ключі в фізичній реалізації схеми бази даних. Зв'язок – це поіменована асоціація двох або більше сутностей. Зв'язок між сутностями має наступні властивості:

– ім’я;
– множинність (потужність);
– обов’язковість – зв'язок може бути обов’язковим або факультативним;

Так, наприклад, сутність «Користувач» пов’язана із сутністю «Роль» через однойменну властивість. Підтримка цього зв’язку у фізичній реалізації бази даних повинна включати підтримку відповідного первинного та зовнішнього ключа (рис. 2.3).

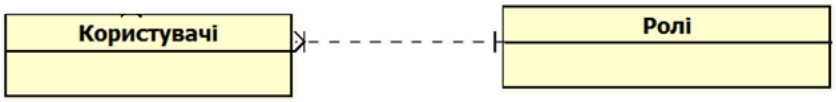

Рисунок 2.3 – Зв'язок між сутностями «Користувач» та «Роль»

Зазвичай, сутності в предметній області утворюють не лише попарні зв’язки, але й складніші групові залежності, коли певні сутності пов’язані не безпосередньо, а утворюють агрегати з допомогою зовнішніх ключів (рис. 2.4).

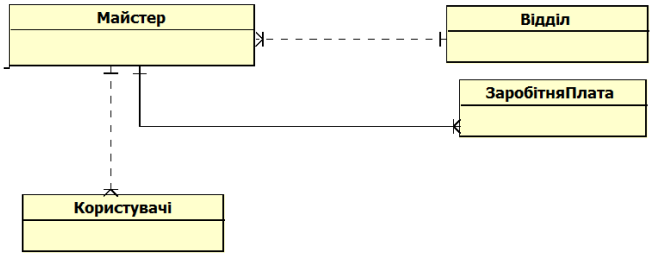

Рисунок 2.4 – Група зв’язків між сутностями, що залежать від сутності «Майстер»

Природно зв’язки між сутностями випливають із їх спільної участі в підтримці певного функціоналу предметної області. Наприклад, сутності «Ремонт», «ДеталіКалькуляції», «Оплата» та «Клієнт» утворюють зв’язки, які дозволяють виконувати реєстрацію факту надання послуг з ремонту автомобілів та розрахунку їх вартості (рис. 2.5).

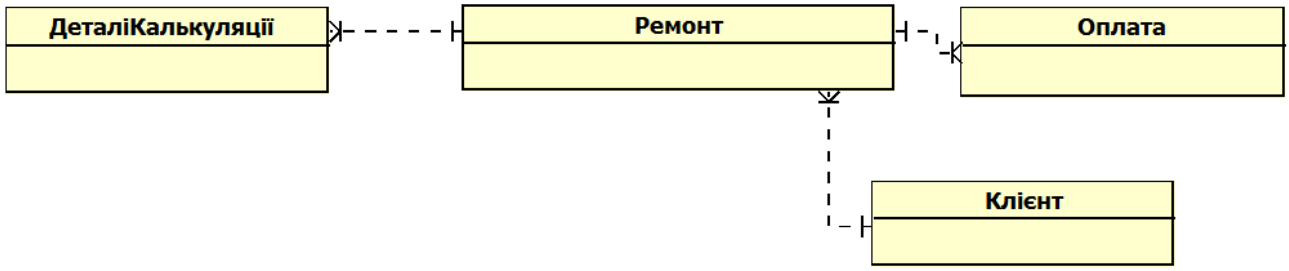

Рисунок 2.5 – Зв’язки між сутностями, що зберігають дані про виконання конкретного ремонту та використовуються для розрахунку вартості робіт

У деяких випадках зв’язки для однієї і тієї ж реляції можуть інтерпретуватись в контексті інших сутностей в різному сенсі. Наприклад, сутність «Виробник» пов’язана із сутностями «Транспорт» та «ОписДеталі», як показано на рис. 2.7. В одному випадку цей зв'язок вказує на виробника авто, в іншому – виробника деталі, при чому сутності «ОписДеталі» та «Транспорт» є взаємонезалежними.

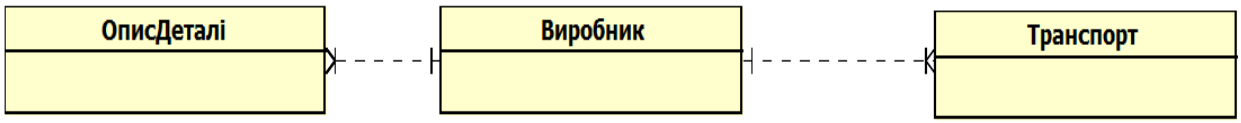

Рисунок 2.7 – Зв'язок між сутностями «Виробник», «ОписДеталі» та «Транспорт»

З’ясування зв’язків між сутностями, їх кратності та обов’язковості має на меті спрощення процедури реалізації бази даних, оптимізації схеми даних, підвищення якості кінцевої програмної системи.

Схема фізичної реалізації бази даних для зберігання інформації про технічне обслуговування автомобілів показана на рис. 2.8.

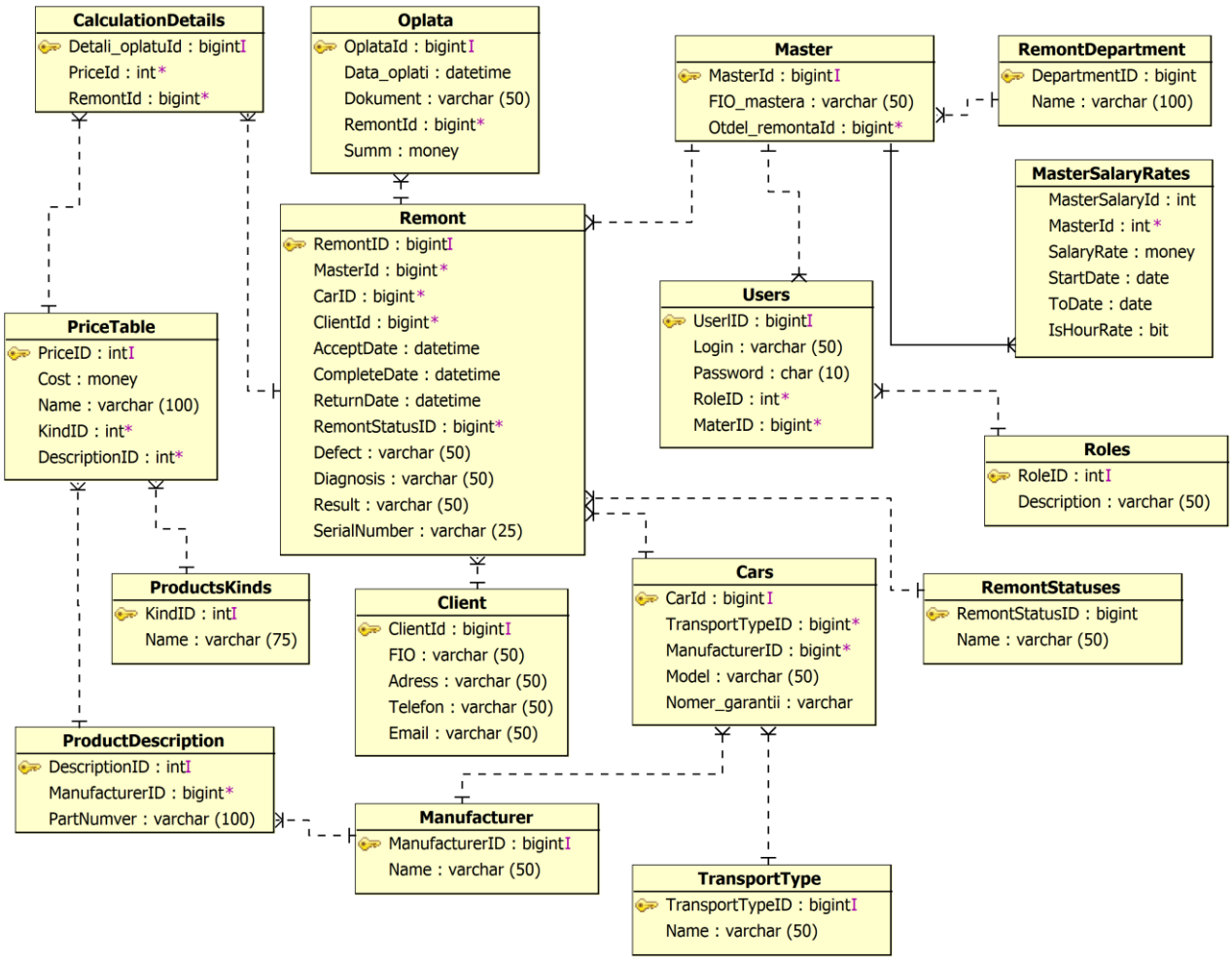

Рисунок 2.8 – ER-діаграма бази даних комп’ютеризованої системи обслуговування автомобілів

Таким чином, спроектовано базу даних для зберігання даних і підтримки бізнес-процесів при обслуговуванні автомобілів.

2.3 Проектування архітектури програмної складової комп’ютеризованої системи обслуговування автомобілів

Архітектура програмного забезпечення комп’ютеризованої системи обслуговування автомобілів є основною компонентою, що забезпечує реалізацію взаємодії між структурними модулями.

На основі визначених вимог спроектовано структурні частини програмного забезпечення та наведено їх на рис. 2.9.

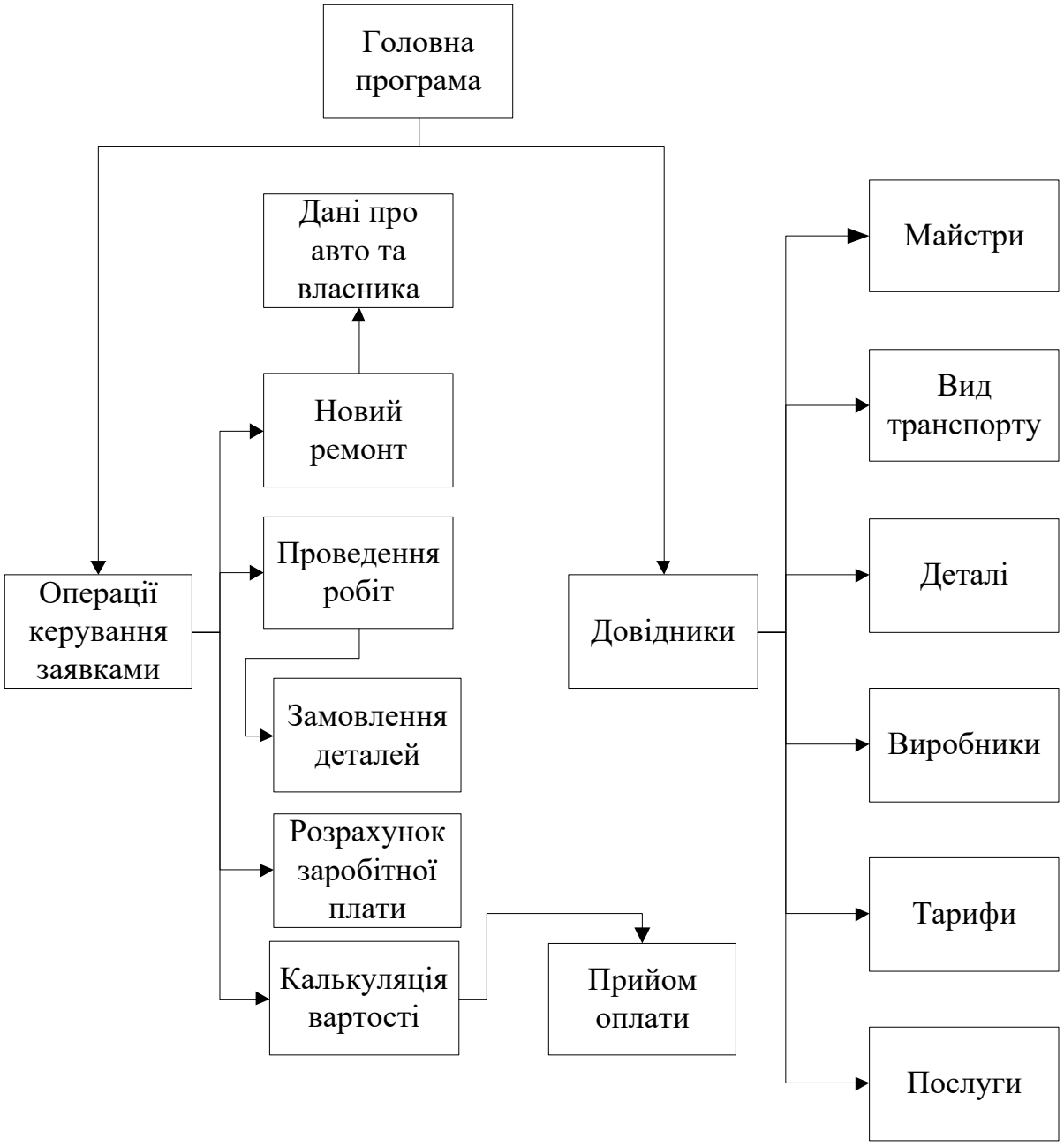

Рисунок 2. 9 – Архітектура програмного забезпечення комп’ютеризованої системи обслуговування автомобілів.

На основі аналізу розроблених вимог до функціональних можливостей програми побудовано модульну структуру комп’ютеризованої системи, що

забезпечуватиме виконання прикладних задач та вимог предметної області, яка показана на рис 2.10.

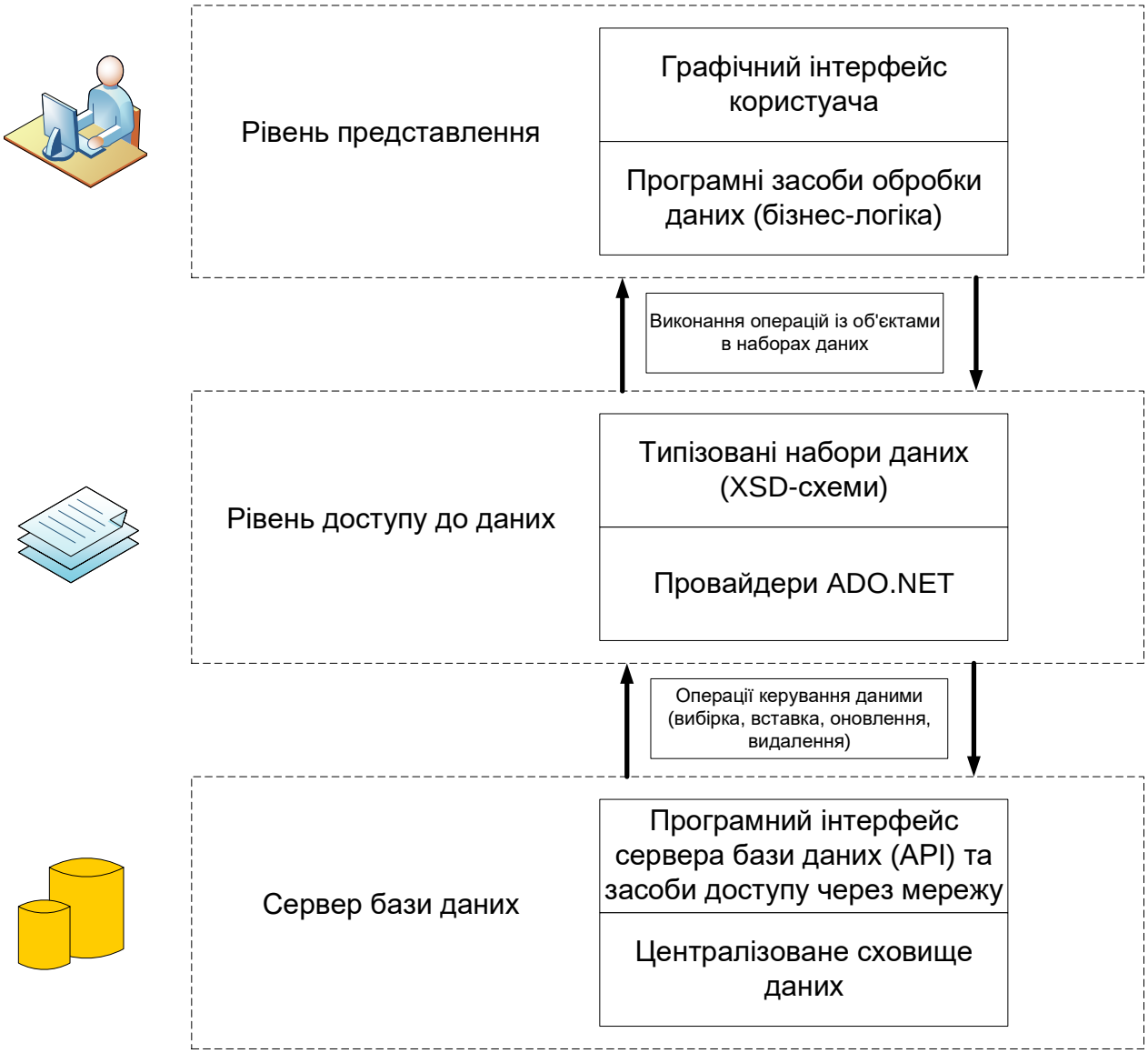

Рисунок 2.10 – Архітектура комп’ютеризованої системи обслуговування автомобілів

Згідно вимог технічного завдання та на основі аналізу прикладних задач предметної області було з’ясовано вимоги до функціональності, що повинна забезпечувати комп’ютеризована система обслуговування автомобілів, а саме:

– обслуговування вводу первинної інформації про стан об’єктів, що беруть участь у процесах – авто, їх види, інформація про клієнтів, дані про наявні деталі;

– надавати можливість редагування та видалення даних із довідників в джерелі даних;

– контролювати введені дані та забезпечувати їх цілісність;

– забезпечувати підтримку операції реєстрації заявки на ремонт, із вказанням даних власника, параметрів авто, даних по діагностиці авто;

– підтримка процесу резервування та використання деталей із деталізацією по кожному найменуванню;

– надання інтерфейсу для проведення процесу калькуляції вартості робіт та оплати праці майстрів;

– забезпечення механізмів пошуку та фільтрації даних.

На основі модульної структури було розроблено алгоритм функціонування програми, що представлений у графічних матеріалах. Основні можливості користувача та їх реалізація при взаємодії із програмним засобом представлено із допомогою діаграми видів діяльності, що дозволяє зобразити та показати на рівні інтерфейсу взаємодію окремих блоків програми та обробку операцій

2.4 Обґрунтування вибору та аналіз технічних характеристик апаратного забезпечення комп’ютеризованої системи

Оскільки, при проектуванні та реалізації комп’ютеризованої системи обслуговування автомобілів запропоновано архітектуру клієнт-сервер як на програмному, так і на апаратному рівні, тому доцільним є обґрунтування технічних характеристик сервера та клієнтських станцій.

При виборі конфігурації сервера доводиться розглядати комбінацію досить складних об’єктів: апаратні засоби, операційна система, СКБД, прикладні додатки. Через складність аналізу комбінації цих об'єктів, зазвичай, неможливо визначити, чи буде здатна дана система підтримувати необхідне навантаження. Проте, часто можливо зробити деякі припущення і потім грубо проаналізувати основні транзакції, щоб визначити, за яких значеннях конфігурація системи не буде здатною опрацювати основні транзакції. Хоча такий підхід корисний тільки для відносно простих додатків, особливо для додатків з однієї або двома

основними транзакціями, він дає непогане початкове наближення і для великих більш складних додатків. У процесі оцінки розглядаються відомі обмеження окремих частин пропонованої системи і потім проводиться їх порівняння з мінімальними потребами відповідно до поставлених завдань. Наприклад, відомо, що диск ємністю 2.1 Гбайт може виконати 62 операції довільного доступу в секунду. На кожну операцію витрачається приблизно 2 мс процесорного часу. Якщо додаток вимагає виконання приблизно 700 операцій довільного читання диску в секунду, то очевидно, що система з одним диском не впорається з таким завданням за необхідний час: необхідно принаймні 12 накопичувачів. Більш того, практично відразу видно, що однопроцесорна система не може впоратися з цим завданням, оскільки для обробки 700 дискових операцій потрібно 1400 мілісекунд процесорного часу в секунду (тобто більше 1 секунди). Хоча цілком зрозуміло, що однопроцесорна система з одним диском буде здатна виконувати цю програму з необхідною швидкістю, зовсім неочевидно, що двопроцесорний система з 12 дисками забезпечить необхідну продуктивність.

Як приклад розглянемо додаток, який протягом чотирьох годин щоночі повинен заново створювати базу даних. База даних складається з кількох таблиць, одна з яких містить 10 мільйонів записів по 2 Кбайт кожна, а інша – 40000 записів розміром приблизно по 1 Мбайт, тобто загальним обсягом 40 Мбайт. Припустимо, що індексація виконується за допомогою окремої операції. Тоді для створення першої таблиці потрібно виконати 10 мільйонів дискових операцій (по 2 Кб) за 4 години, або 695 операцій в секунду. Дисковий накопичувач може забезпечити виконання приблизно 60 операцій (по 2 Кб) довільного доступу в секунду в режимі "чистого" диска і приблизно 400 операцій в секунду в режимі послідовного доступу. Звідси ясно, що необхідно мати принаймні два паралельно працюючих диски, якщо створення таблиці виконується послідовними операціями, або 12 паралельно працюючих дисків, якщо виконуються операції довільного доступу. Зазвичай сама таблиця створюється послідовними операціями, в той час як індекси створюються часто операціями довільного доступу. Створення другої таблиці вимагає розміщення

на диску 40000 записів по 1 Мбайт кожна (всього 40 Гбайт). Хоча конкретні СКБД відрізняються обробкою і розміщенням даних великого обсягу (такі об'єкти часто називаються BLOBs - Binary Large OBjects), всі вони майже напевно будуть використовувати для запису кожного елемента даних тільки одну адресу, і в результаті, такі записи будуть виконуватися послідовними операціями. Таким чином, в нашому випадку кожен запис буде складатися приблизно з 512 фізичних записів на диск, в цій роботі будуть домінувати операції послідовного доступу до диска (навіть, якщо кожен запис зберігається окремо, буде потрібно не більше одного позиціонування головок на все 512 звернень до диска). Для створення таблиці потрібно виконати 20480000 операцій запису на диск зі швидкістю приблизно 1450 операцій в секунду. Для досягнення цієї швидкості принаймні 4 диски повинні здійснювати запис одночасно. Оскільки для розміщення 40 Гбайт даних буде потрібно не менше 14 дискових накопичувачів місткістю 2.9 Гбайт, для досягнення необхідної продуктивності не виникає проблем із забезпеченням достатньої кількості дисків, але вимога паралельного запису на 4 диска диктує, щоб використовувався деякий механізм, що дозволяє розпаралелити доступ по дисках. Тому мінімально необхідна конфігурація повинна забезпечувати чотириразову декомпозицію. Необхідно пам'ятати, що при досягненні таких швидкостей дискового вводу/ виводу, система повинна витрачати також приблизно 1.7 мс процесорного часу на кожну операцію з диском. Створення першої таблиці вимагає виконання 695 дискових операцій в секунду, або 1181 мс на кожну секунду процесорного часу. Для створення другої таблиці потрібно виконати 1425 дискових операцій в секунду, або приблизно 2425 мс на кожну секунду процесорного часу. В сумі це становить приблизно 3606 мс на кожну секунду процесорного часу. Таким чином можна зробити висновок про те, що чотири процесори 50 МГц SuperSPARC повинні займатися виключно обслуговуванням дисків. Крім того, слід зауважити, що всі дані для формування таблиць бази даних звідкись повинні надійти, а обробка приходять даних також вимагає деякого процесорного часу. Припустимо, що дані надходять по мережі FDDI з використанням протоколів TCP / IP і очікується отримати всього 60 Гбайт

даних за 4 години, тобто швидкість надходження даних становить приблизно 4.2 Мбайт / с. Якщо взяти максимальний розмір пересилання по FDDI рівним 4500 байт, це складе приблизно 949 пакетів в секунду. На обробку кожного пакету процесор повинен затратити 400 мікросекунд (0.4 мс), тому обробка всього потоку відбувається протягом 380 мс процесорного часу на кожну секунду. Ці накладні витрати можуть ще збільшитися, якщо дані не займають повністю 4500-байтового пакета (наприклад, СКБД Oracle v.6 просто здійснює пересилку в пакеті одного поля одного запису незалежно від його розміру). Для забезпечення дискового і мережевого вводу / виводу docnfnymj чотирьох процесорів. Однак в конфігурацію системи необхідно включити принаймні ще один додатковий процесор, щоб виконувати будь-яку іншу роботу. Слід зазначити, що для роботи самої CRБД, а також для будь-якої іншої обробки, яка повинна виконуватися під час нічних перезавантажень бази даних, потрібен певний процесорний час. Більш того, необхідно передбачити невеликий поправочний коефіцієнт, оскільки проведений аналіз припускав, що робота може здійснюватися із середньою швидкістю і закінчиться в точно визначений час. Однак це мало ймовірно, так що в системі необхідно передбачити деякі додаткові ресурси через те, що неминуче її середня продуктивність виявиться менше, ніж було обчислено.

Таким чином, до складу кінцевої конфігурація системи слід включити 6 процесорів. Оскільки мінімальний обсяг дискового простору становить 40 Гбайт, цілком розумним кандидатом для вибору системи представляється SPARCcenter 2000. Для зберігання тільки даних потрібні чотирнадцять дисків ємністю 2.9 Гбайт, так що в конфігурацію системи необхідно включити принаймні 16 накопичувачів (окремий диск потрібно для розміщення системи і журналу СКБД). Оскільки в зверненнях до дискової підсистеми домінують послідовні операції, при конфігуруванні дисків СКБД необхідно підключати не більше 4 дисків до одного головного адаптера SCSI. Необхідність 4-кратного розщеплення збільшує загальна кількість дисків до вісімнадцяти. Вони будуть підключатися до п'яти головних адаптерів DWI / S: по 4 диска на 4 адаптера для реалізації розщеплення, а також системний і журнальний диски на окремому

головному адаптері. Конструкція дискових шасі визначає необхідність установки п'яти шасі в одній стійці (кабінеті) розширення. Вісімнадцять дисків могли б бути встановлені і в одній основній стійці, але в ній неможливо настроїти необхідну комбінацію дисків і шин SCSI. Для підключення 5 плат інтерфейсу SCSI і однієї плати інтерфейсу FDDI всього потрібно 7 слотів SBus. Для установки 6 процесорів необхідні три системних плати, що забезпечують 12 слотів SBus. Сам по собі процес завантаження СКБД не пред'являє будь-яких особливих вимог до обсягу основної пам'яті.

Апаратне забезпечення робочих станцій повинно відповідати наступними технічним характеристикам:

- процесор – тактова частота 2,0 ГГц;
- об'єм оперативної пам'яті – 4096 Мб;
- об'єм жорсткого диску – 500 ГБ.

Операційною системою сервера є платформа на базі Windows Server 2008 R2, а програмне забезпечення робочих станцій включають використання платформи сімейства Windows та прикладного програмного забезпечення, що використовується для забезпечення необхідного рівня продуктивності праці.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРИЗОВАНОЇ СИСТЕМИ ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

3.1 Реалізація програмного інтерфейсу комп'ютеризованої системи обслуговування автомобілів

Інтерфейс прикладного клієнтського додатку слугує засобом редагування даних в репозиторії та призначений для підтримки прикладних задач, що виконуються в процесі діяльності станцій технічного обслуговування. Клієнтський інтерфейс програмного засобу реалізується на основі компонентів Windows Forms. Користувацький інтерфейс комп'ютеризованої системи обслуговування автомобілів повинен забезпечувати зручне введення, редагування та перегляд інформації, бути інтуїтивно зрозумілим, забезпечувати перевірку коректності вводу даних. Для реалізації алгоритму роботи програми запропоновано використати мультивіконний інтерфейс із головною та дочірніми формами. Головне вікно інтерфейсу показано на рис 3.1.

Рисунок 3.1 – Вигляд головного вікна клієнтського прикладного додатку

					КС КРБ 123.171.00.00 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Ковач О.С.			Реалізація і тестування комп'ютеризованої системи обслуговування автомобілів	Літ.	Арк.	Аркушів
Перевір.		Луцків А.М.					40	
Реценз.						ТНТУ, каф. КС, гр. СІс-44		
Н. Контр.		Луцик Н.С.						
Затверд.		Осухівська Г.М.						

Створення чи відкриття заявки на ремонт можливе через пункт меню «Заявки» та підпункти «Нова заявка», «Відкрити…» чи «Список заявок». При відкритті необхідно вказати номер заявки яку потрібно відкрити, список заявок підтримує можливості фільтрації за критеріями «Прізвище замовника», «Дата замовлення», «Майстер» (рис. 3.2).

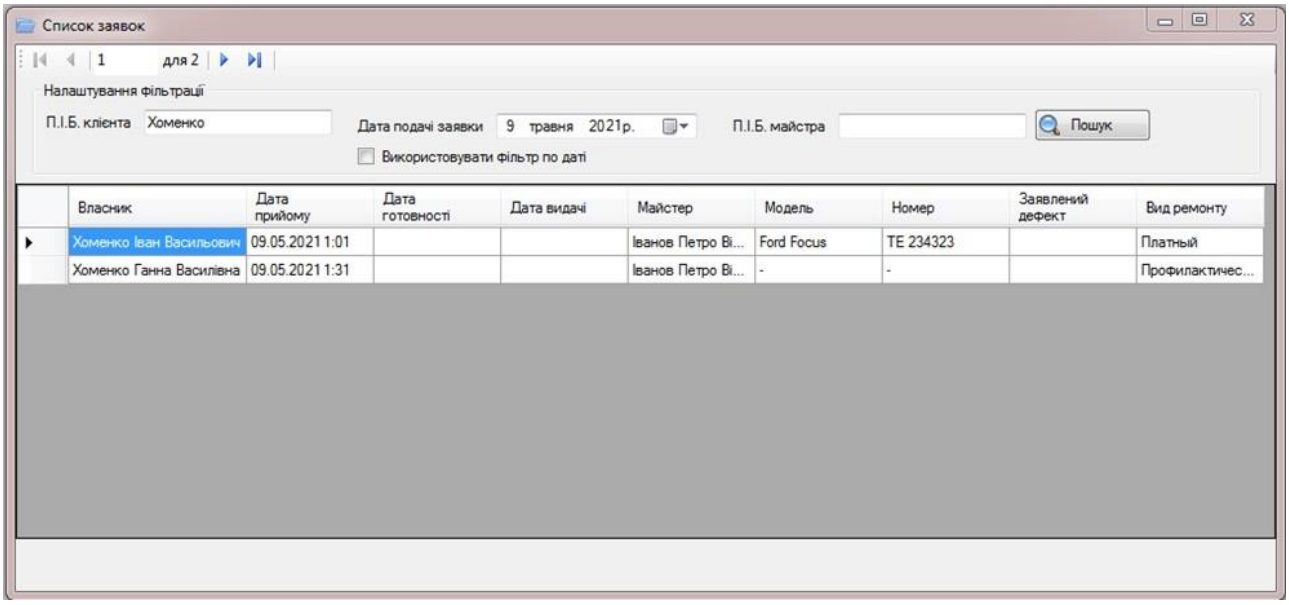

Рисунок 3.2 – Вигляд вікна «Список заявок»

Вікно редагування даних заявок містить поля для вводу:

– інформації про особу, що подає заявку;
– контактні дані особи;
– дані про транспортний засіб;
– опис дефекту;
– перелік необхідних робіт.

Також дана форма містить елементи для відкриття та навігації до інших форм: калькуляції, перегляду оплати, друку бланку накладної замовлення.

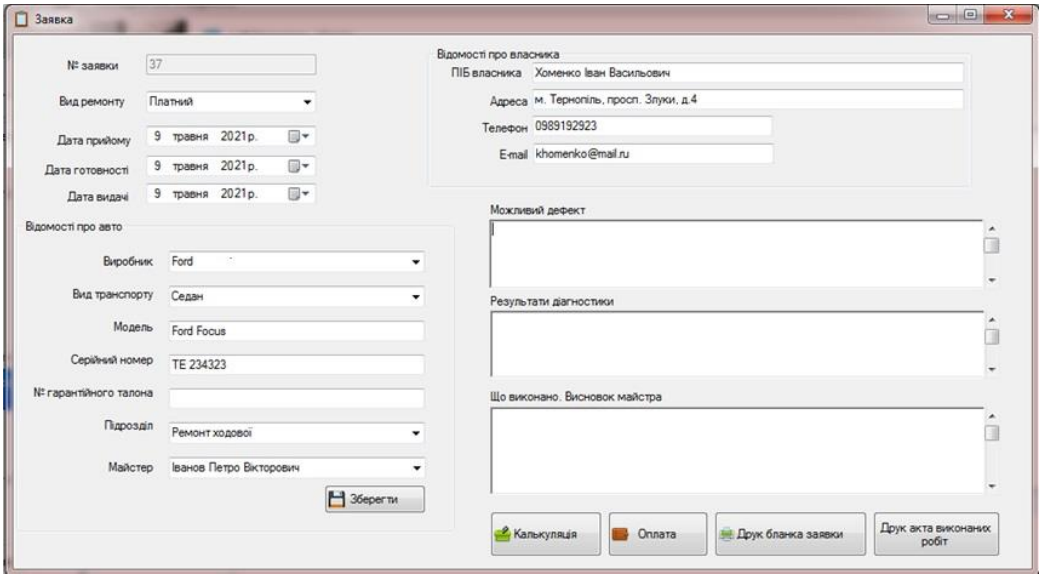

Рисунок 3.3 – Інтерфейс форми внесення та редагування даних про заявку на виконання робіт

Форма калькуляції призначена для визначення переліку послуг та деталей, що були використані в процесі обслуговування транспортного засобу (рис. 3.4). Дана форма дозволяє обчислити примірну кінцеву вартість ремонту та послуг. Крім того дані внесені через форму калькуляції в подальшому використовуються для аналізу ефективності надання послуг, затрачених ресурсів, обліку робочого часу та інших аналітичних цілей.

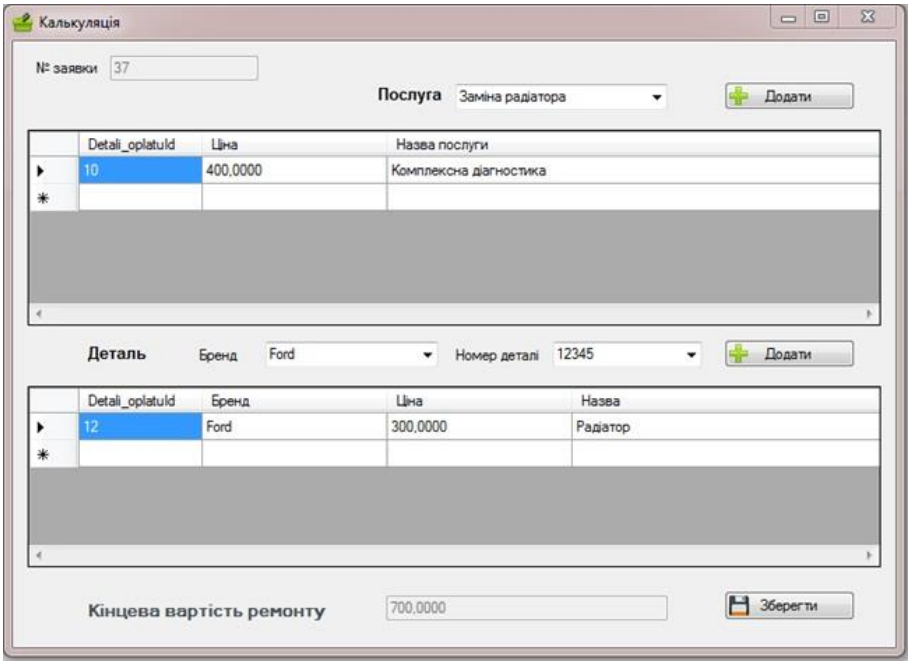

Рисунок 3.3 – Інтерфейс проведення калькуляції вартості наданих послуг

Форма даних про оплату містить записи про факти оплати по даній заявці, дати внесення коштів та номера платіжних документів (як от квитанцій та чеків). Також ця форма дозволяє викликати форму калькуляції для швидкого і зручного внесення даних про надані послуги та використані деталі (рис. 3.5).

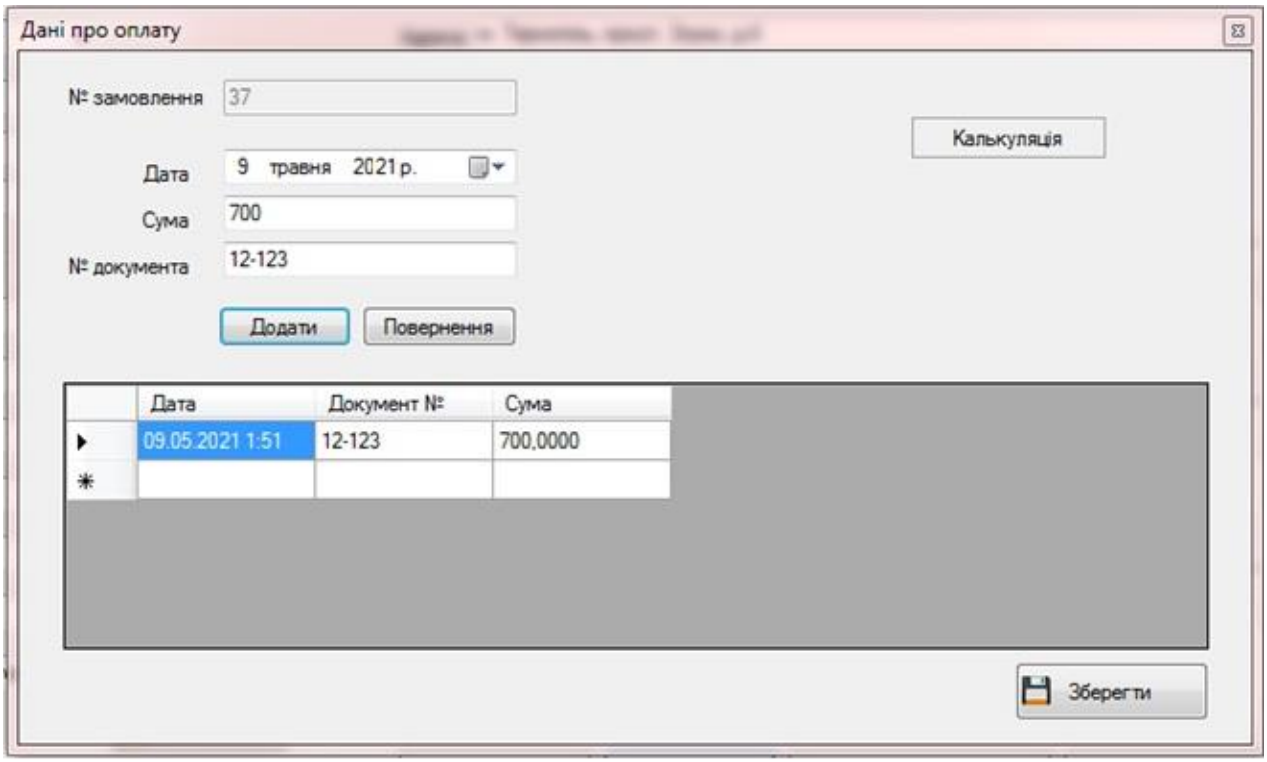

Рисунок 3.5 – Зовнішній вигляд форми внесення даних про оплату

Ефективність використання програмного засобу дуже часто визначається тим наскільки просто та наглядно реалізовано пошук необхідної інформації. Пошук заявок здійснюється з допомогою форми «Пошук», що містить поля для вибору різноманітних фільтрів та налаштувань: номеру заявки, інформації замовника, даних про транспортний засіб. Відкриття знайденої заявки виконується подвійним натисканням на запис в результатах пошуку. На рис. 3.6 показано вигляд вікна для пошуку заявок за визначеними фільтрами.

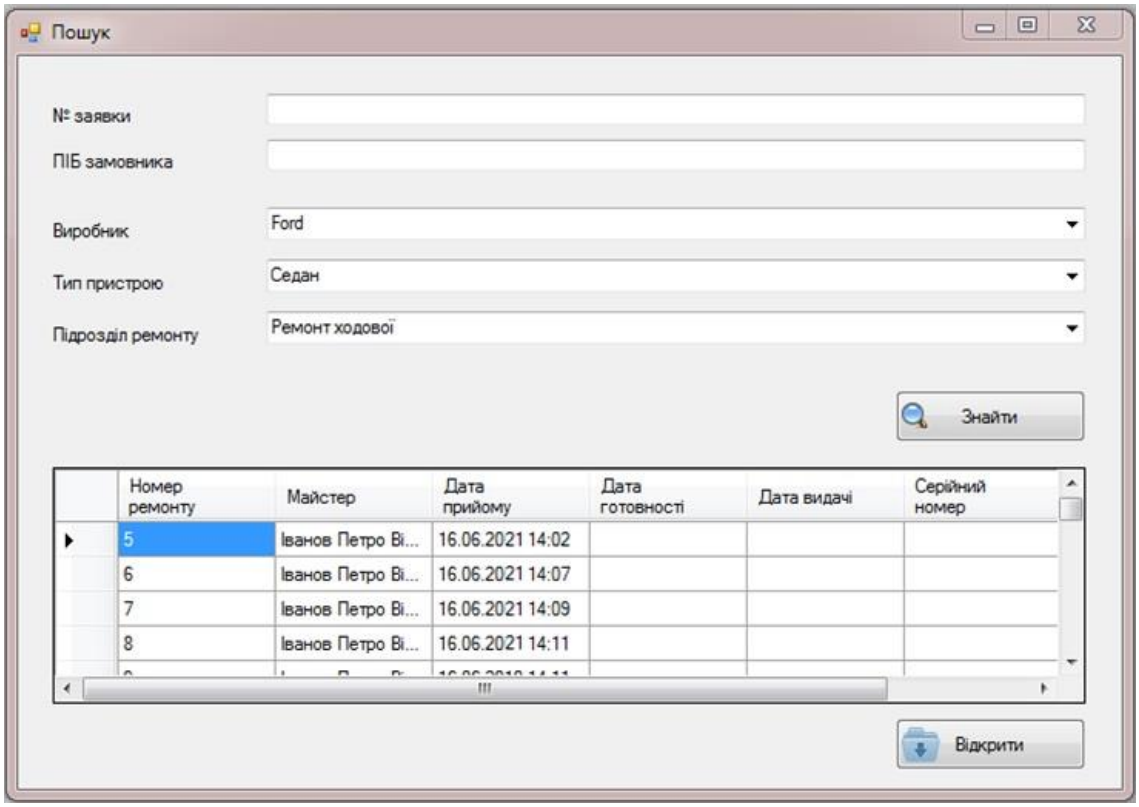

Рисунок 3.6 – Форма пошуку заявки

Програмний засіб для підтримки діяльності станції технічного обслуговування дозволяє також зберігати інформацію про економічну стратегію діяльності підприємства в вигляді засобів формування прайсу запчастин та вартості послуг (рис. 3.7 та рис. 3.8). Засоби редагування прайсу запчастин також інтегровані із засобами контролю за наявністю необхідних запчастин на складі.

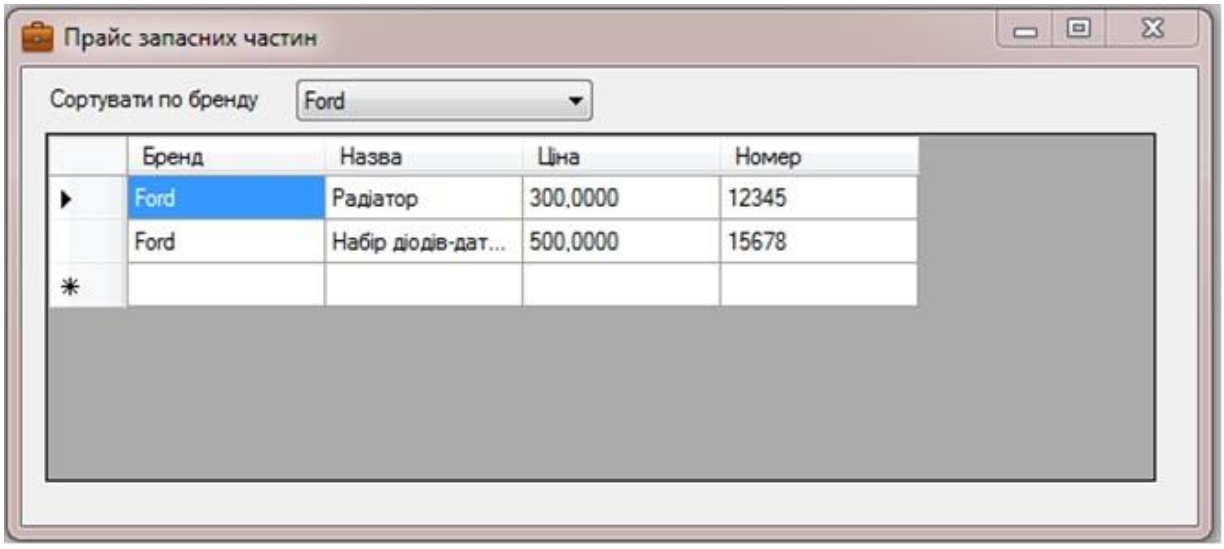

Рисунок 3.7 – Форма редагування прайсу запчастин

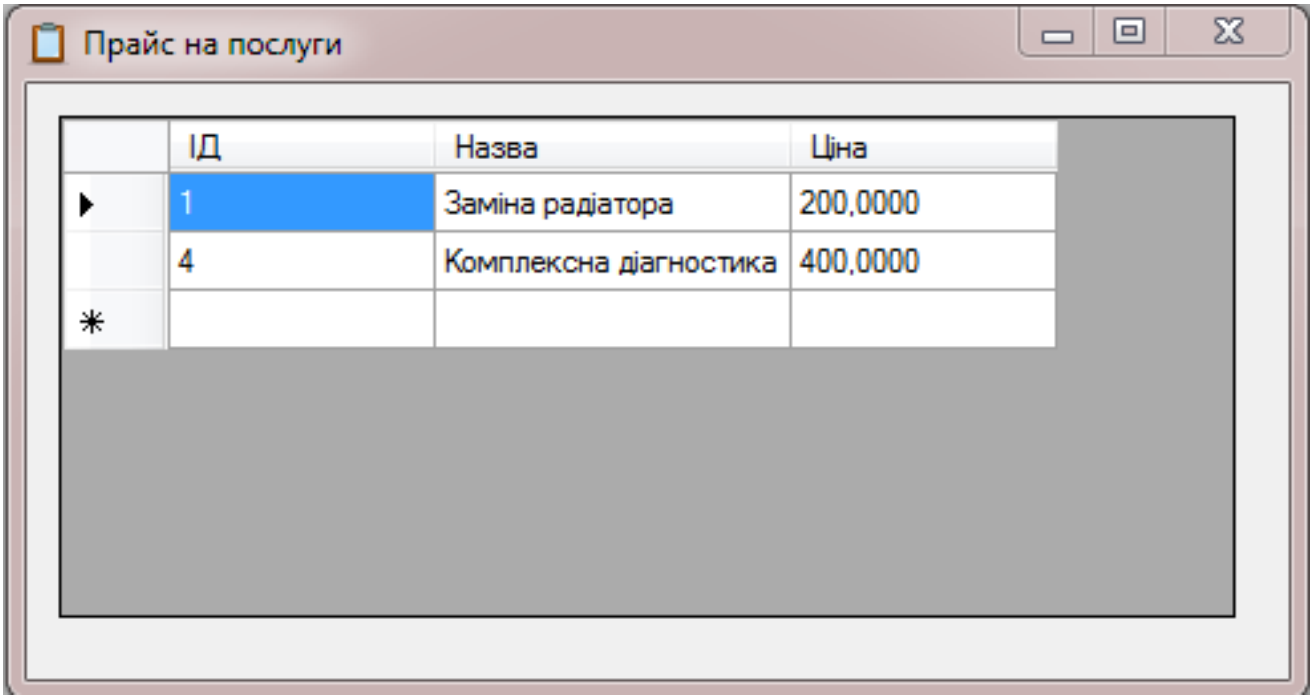

Рисунок 3.8 – Форма редагування вартості послуг, що надаються станцією технічного обслуговування автомобілів

Форма редагування даних заявки дозволяє також генерувати та виводити на друк накладний лист замовлення. Лист-замовлення формується з допомогою компонентів Reporting Services, що є багатофункціональними засобами формування звітів. Інтеграція із системою Reporting Services дозволяє легко розширяти програмну систему з метою формування специфічної документації, аналітичних даних та зручного виводу інформації.

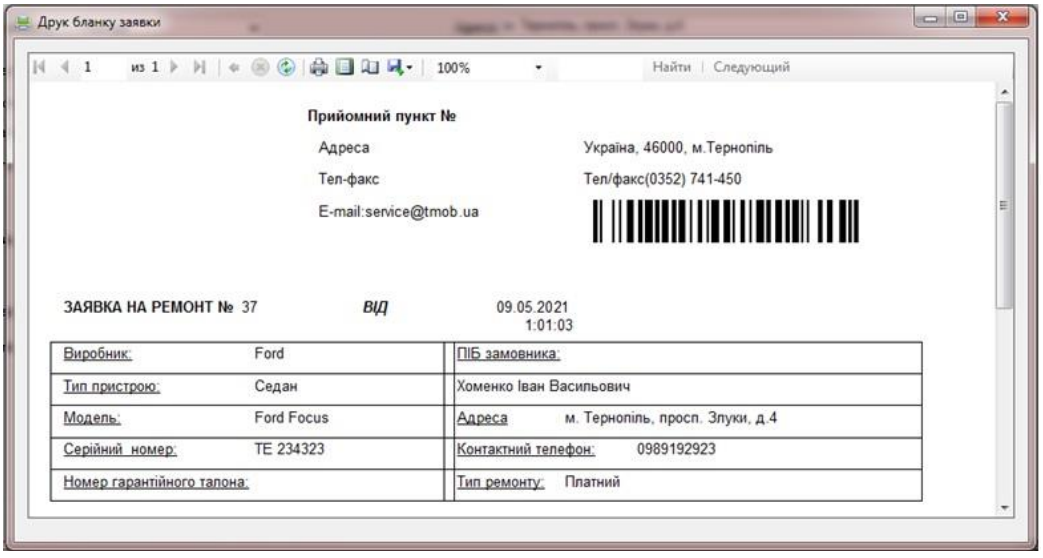

Рисунок 3.9 – Зовнішній вигляд сформованого бланку заявки

Графічний інтерфейс прикладного додатку реалізовано для задоволення вимог, що висунуті до функціональності комп'ютеризованої системи та з врахуванням аспектів якості, ергономіки та позитивного досвіду використання. Для розробки прикладного інтерфейсу було використано платформу Windows Forms, як багатофункціональний, гнучкий та надійний засіб компонування віконного інтерфейсу прикладних додатків.

3.2 Тестування (оцінювання) функціональності роботи комп'ютеризованої системи

Тестування є невід'ємною складовою інженерії комп'ютерних систем та процесу розробки будь-якого програмного засобу. Цей вид робіт проводять з метою виявлення дефектів, що не були виявлені та не усунуті у процесі розробки, перевірки відповідності програмних засобів висунутим вимогам, покращення якості комп'ютеризованої системи.

Тестування програмних складових комп'ютеризованих систем поділяють на різні категорії за певними показниками: об'єктом досліджень, знанням структури об'єкта тестування, ступенем автоматизації, виглядом представлення результатів.

Одним із базових видів тестування відносно початкового коду та структури програми є модульне тестування. Модульне тестування передбачає розбиття об'єкта тестування на деякі окремі компоненти, які формують функціональну одиницю – модуль. Парадигма модульного тестування базується на принципі коректності роботи окремих локальних компонентів, що в кінцевому випадку задовольняє критерії функціональності функціонування цілої системи.

До переваг модульного тестування належить:

– здатність забезпечення детального тестування окремих ізольованих елементів програми;

– можливість відстеження змін, що можуть призвести до регресії системи;

– забезпечення можливості рефакторингу існуючого коду та повторного тестування;

– розділення концепцій тестування інтерфейсу та реалізації;

Найпопулярнішим і найбільш повнофункціональним засобом модульного тестування для програм, що розроблені для виконання платформою .NET є NUnit. NUnit представляє собою середовище для реалізації модульних тестів програмного забезпечення, що використовує платформу .NET і засноване на відкритому коді.

До складу NUnit входять компоненти, які дозволяють візуалізувати етапи процесу тестування. При цьому такі компоненти можуть бути як окремими застосуваннями (Visual NUnit), так і інтегрованими у середовище розробки..

Середовище для модульного тестування NUnit функціонує на базі декількох збірок .NET бібліотек. При використанні мови програмування C# тести представляються у вигляді окремих класів платформи .NET з атрибутами TestFixture. Модульні тести реалізуються як безпараметричні методи із типом повернення void з поміткою Test.

Для можливості запуску тестів необхідно включити в проект збірку .NET nunit.framework, та простір імен NUnit. Це дозволить викликати методи для перевірки виконання тестів та маркувати тести відповідними атрибутами. Окрім цього для успішного виконання тестів цільова збірка, та бібліотека тестування повинні компілюватися без помилок.

Структура тесту NUnit зазвичай включає крок препроцесингу даних, проведення дії, що тестується та перевірку результату із еталонним значенням. Наприклад, якщо необхідно виконати перевірку коректності роботи табличних адаптерів XSD-схеми на прикладі вставки нової заявки на ремонт в базу необхідно реалізувати бібліотеку, як показано у лістингу 3.1.

Лістинг 3.1 – Реалізація бібліотеки перевірки коректності роботи адаптерів

```
namespace TestProject
{
    [TestFixture]
    public class OrdersTest
    {
        [Test]
        public void Add_Order_In_DB()
        {
            newMyDbDataSet.RemontDataTableAdapter cta = new
newMyDbDataSet.RemontDataTableAdapter();
            var ordersTable = cta.GetData();
            var orderID = ordersTable.First().OrderID;

            cta.InserSet("Test", orderID, 1, 1, "TEST", "");

            ordersTable = cta.GetData();
            var modelName = ordersTable.Last().Model;

            Assert.AreEqual("Test", modelName);
        }
    }
}
```

Даний програмний код отримує з бази даних відомості про існуючі заявки, створює новий об’єкт табличного адаптера для таблиці заявок на обслуговування та виконує вставку нового запису про заявку на ремонт із іменем моделі транспортного засобу “Test” та відповідними відомостями. Потім виконується вибірка таблиці заявок та перевірка чи є іменем моделі транспортного засобу останньої заявки на ремонт слово “Test”.

Для виконання тесту в середовищі розробки Visual Studio необхідно виконати компіляцію бібліотеки та запуск утиліти Visual NUnit, скомпілювати збірку із тестом та програмою, обрати її як цільову для тестів та запустити тест. У разі успішного проходження біля тесту появиться відмітка зеленого кольору, у випадку невдачі – червоного, як показано на рис. 3..

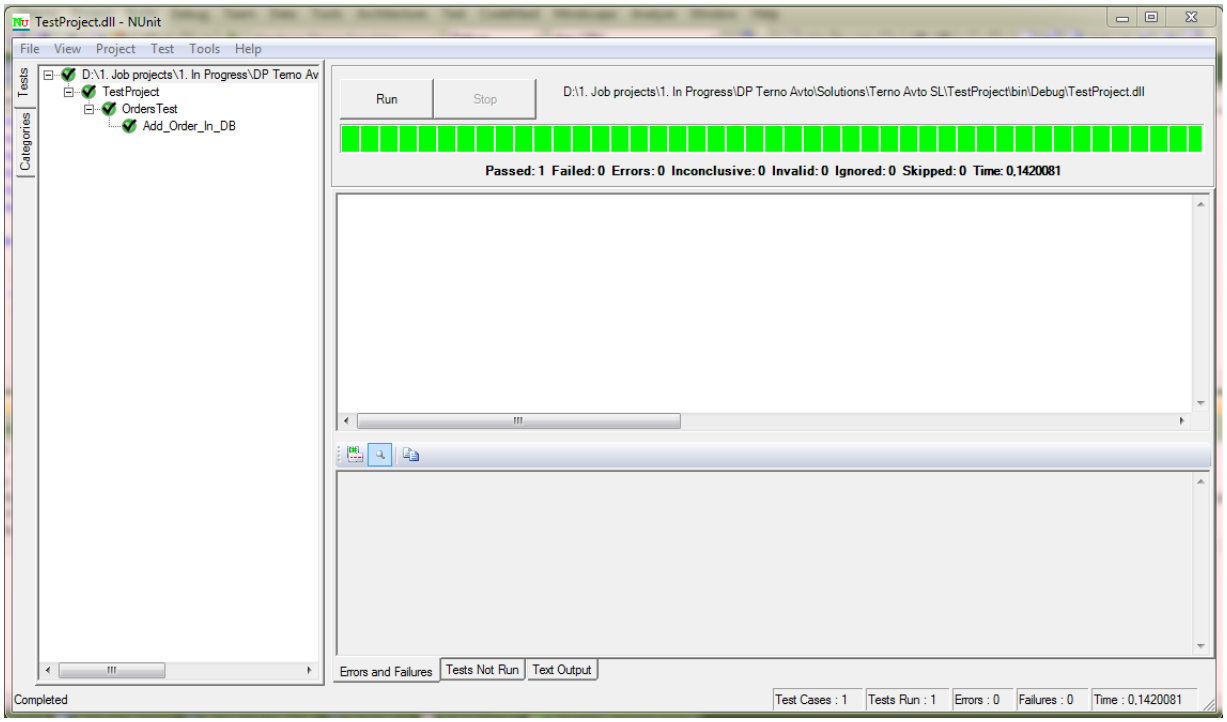

Рисунок 3.10 – Успішне проходження тесту в середовищі Visual NUnit

Для підтвердження працездатності та коректності функціонування програмного засобу підтримки діяльності станції технічного обслуговування проведено написання модульних тестів для усіх нетривіальних та ресурсоємних методів – таких як вставка, оновлення та видалення записів, зв'язування записів в головних та детальних таблицях, проведення процесів перевірки залишків при резервуванні товару на складі.

Успішне проходження модульних тестів засвідчило функціональну повноту, надійність та безпечність використання програмної системи.

3.3 Інструкції з інсталяції та налаштування комп'ютеризованої системи

Платформа .NET та середовище Visual Studio надають широкий спектр варіантів розгортання програм в клієнтському середовищі – засоби утиліти xcopy, проекти пакетів інсталяторів (Setup Project), майстри інсталяції (Setup Project Wizard), швидка інсталяція з допомогою технології ClickOnce.

Кожен з цих варіантів розгортання володіє власними перевагами та недоліками використання в певних сценаріях інсталяції. Стандартним

рекомендованим способом розгортання клієнтських настільних додатків є проект інсталяційного пакету (Setup Project), що дозволяє виконувати розгортання прикладного додатку з допомогою покрокового діалогу.

Для початку інсталяції прикладного користувацького додатку, що призначений для підтримки діяльності станцій технічного обслуговування необхідно виконати запуск пакету Microsoft Installer (рис. 3.11).

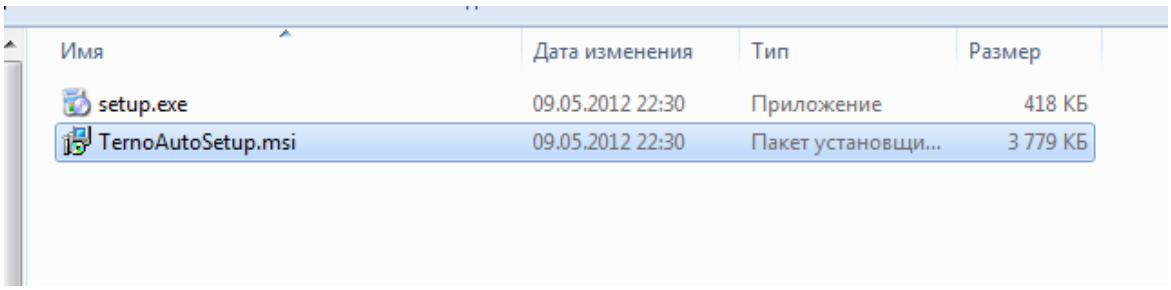

Рисунок 3.11 – Запуск інсталяційного пакету

На екрані з’явиться початкове вікно інсталятора з пропозицією розпочати процес встановлення клієнтського додатку (рис. 3.12).

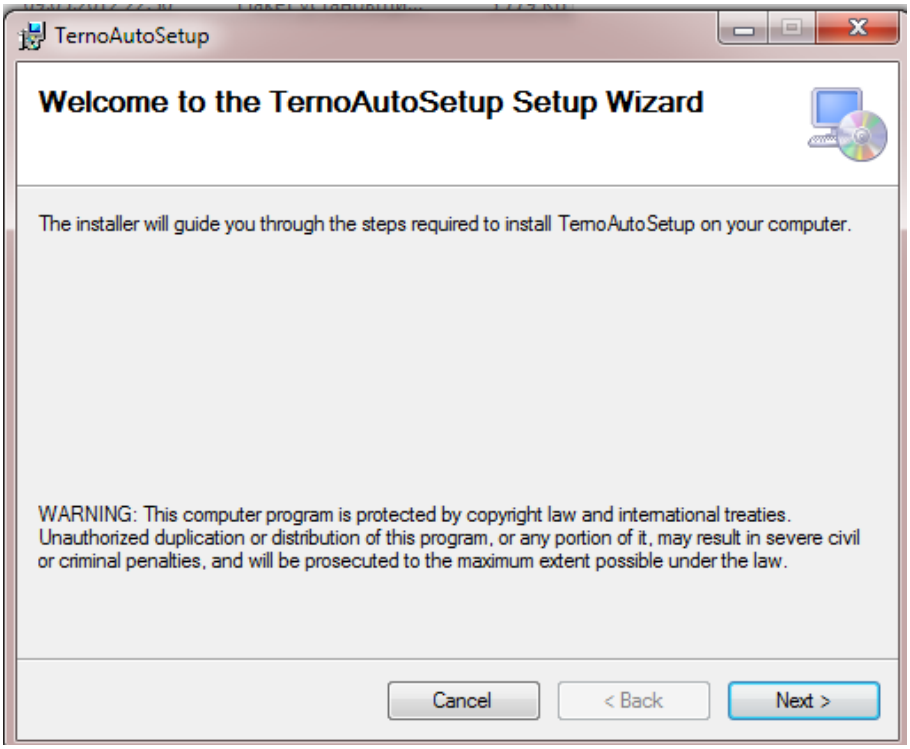

Рисунок 3.12 – Початкове вікно діалогу інсталятора

При натисканні кнопки «Далі» наступним кроком діалогу буде пропозиція вибрати розташування в файловій системі куди буде встановлюватись програма та опції доступності програми для запуску (рис. 3.13).

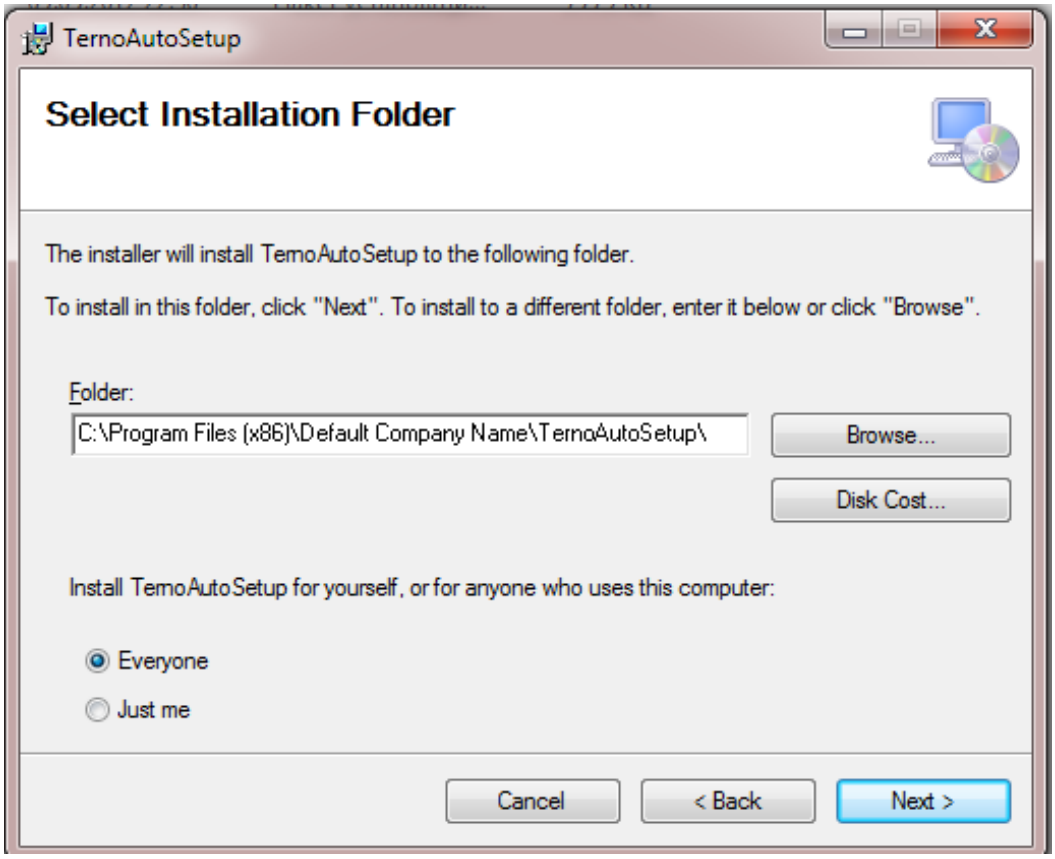

Рисунок 3.13 – Вибір місця розташування в файловій системі для встановлення програми

Після вибору необхідного місця розташування та опцій доступу до встановленої програми можна перейти до наступного кроку встановлення з допомогою кнопки «Далі».

Наступне діалогове вікно попереджує про те що далі майстер встановлення розпочне інсталяцію програми і що на даний момент ще є можливість змінити опції інсталяції (рис. 3.14).

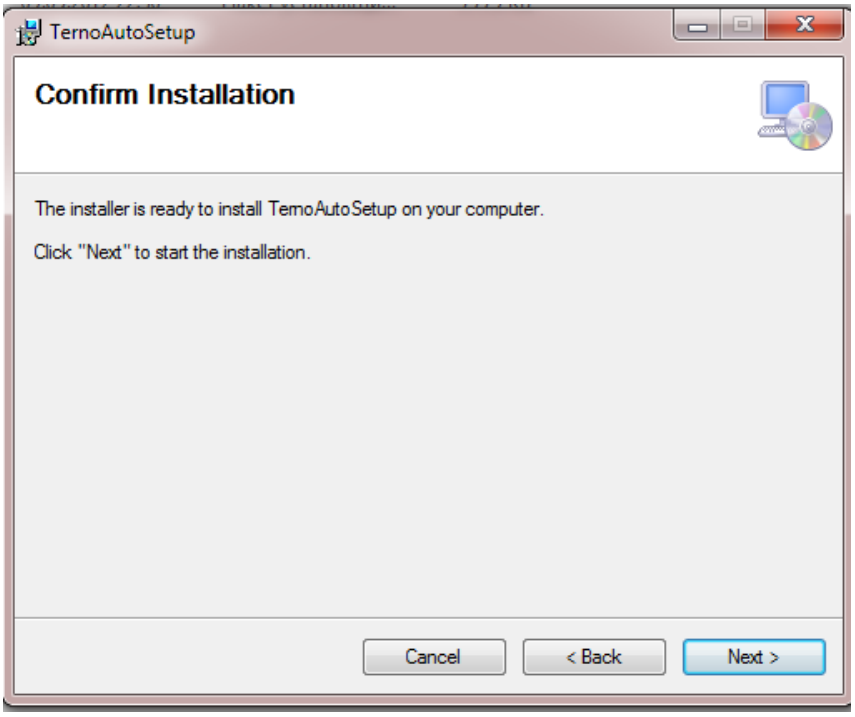

Рисунок 3.14 – Екран погодження для початку інсталяції

Для початку процесу копіювання файлів в файлову систему та розгортання необхідних компонентів необхідно натиснути кнопку «Далі». Майстер інсталяції розпочне автоматичне копіювання файлів, реєстрацію та розгортання компонентів, це може зайняти деякий час (рис. 3.15).

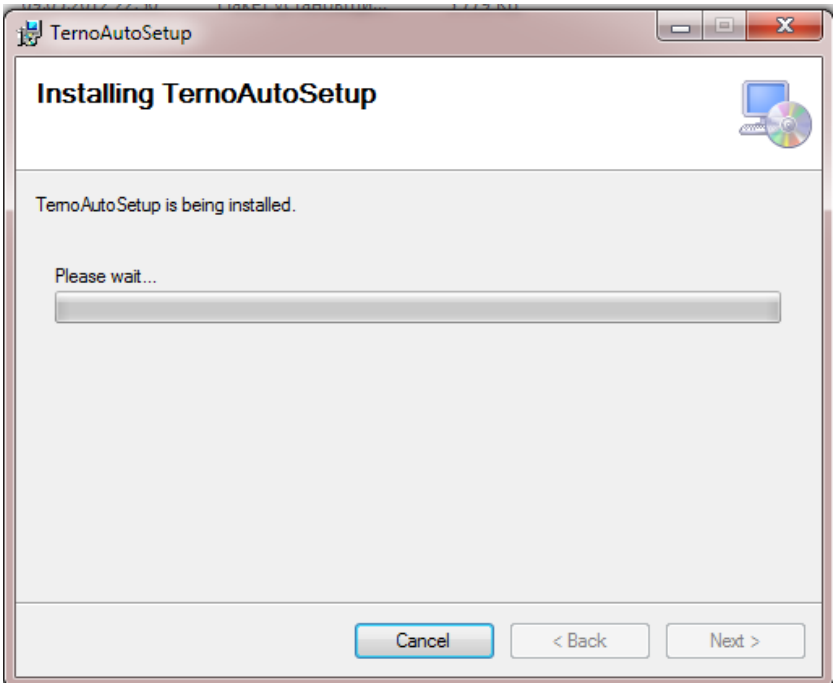

Рисунок 3.15 – Процес копіювання та реєстрації компонентів програми

Про вдале завершення установки програмних компонентів буде повідомлено з допомогою спеціального вікна діалогу. Процес установки можна завершити натиснувши кнопку «Вихід» (рис. 3.16).

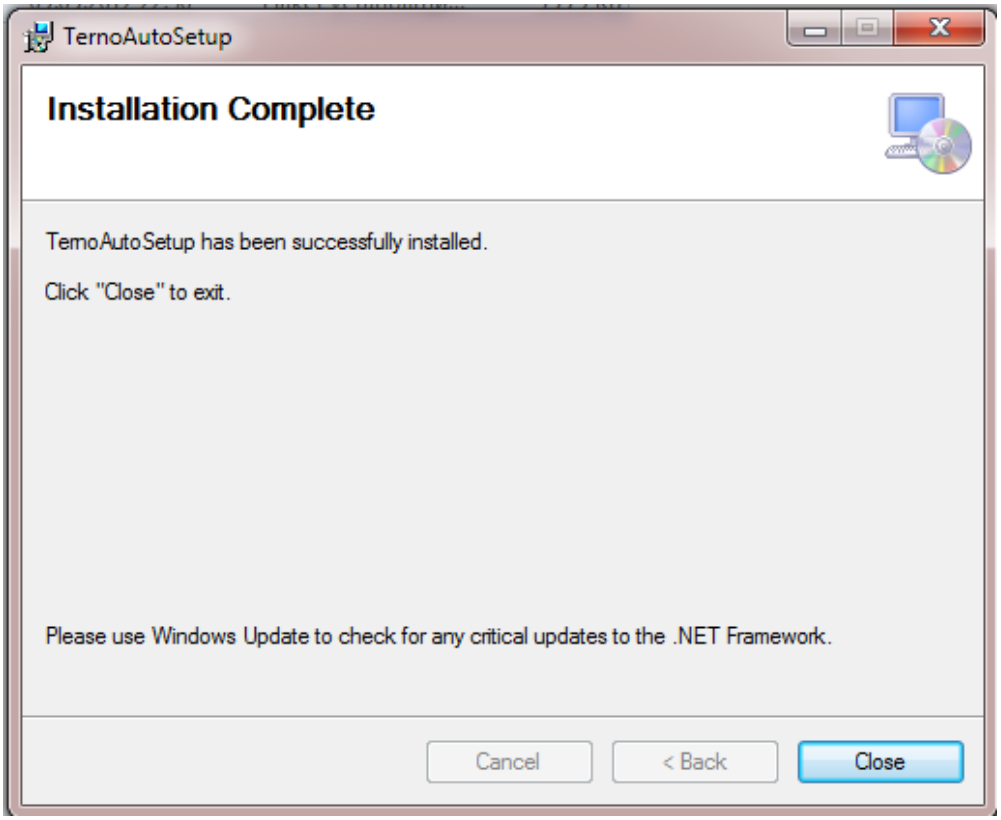

Рисунок 3.16 – Вдале завершення інсталяції програмної системи

Після завершення процесу інсталяції програма обслуговування діяльності станцій технічного обслуговування готова до використання. Запуск програми виконується з допомогою виконуваного файлу програми за розташуванням що було вказане в процесі встановлення програми.

3.4 Інструкції з інсталяції додаткового програмного забезпечення

При інсталяції програм, що використовують Windows Forms виникає залежність від наявної встановленої версії .NET Framework. У випадку використання інсталяційного пакету є можливість включення залежних бібліотек для встановлення в разі відсутності платформи на цільовій машині. Проект інсталяційного пакету в середовищі Visual Studio дозволяє відстежити

залежності для проектів, що потребують розгортання та налаштувати відповідні опції розгортання (рис. 3.17).

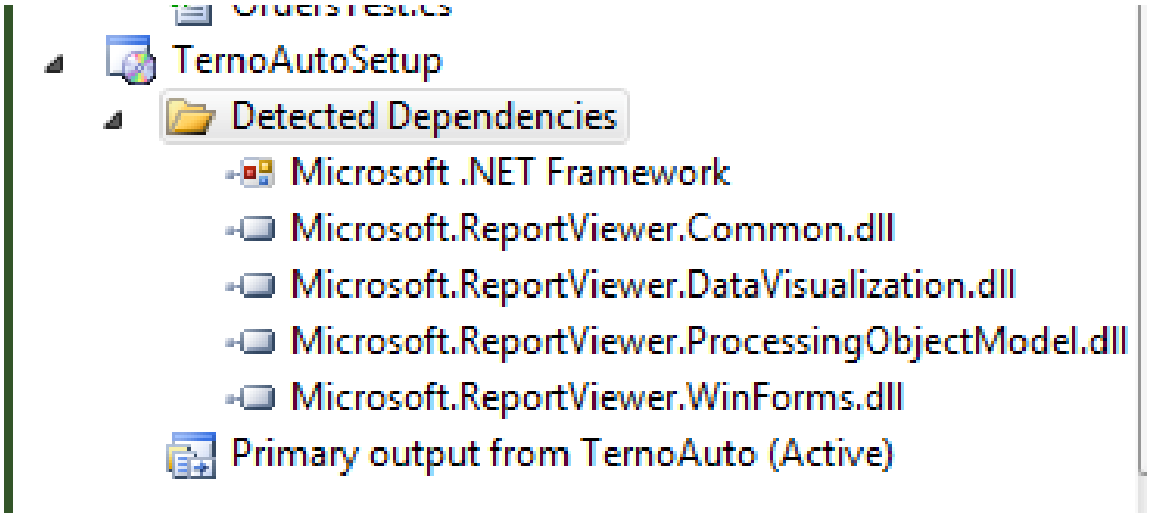

Рисунок 3.17 – Відображення залежностей для інсталяційного пакету

Для компонентів та бібліотек-залежностей є можливість налаштування двох шляхів розгортання – локальне копіювання в пакет інсталяції і перенесення разом із ним, або пропозиція завантаження з мережі WWW. В випадку вибору локального копіювання бібліотеки збільшуватимуть розмір файлу інсталятора, але ви можете бути впевнені, що необхідні компоненти завжди будуть встановлені в цільовому середовищі.

Для розгортання централізованого репозиторію (бази даних) необхідно також встановити MS SQL Server на комп'ютері, що виступатиме в ролі серверної машини.

Розгортання СКБД виконується запуском скрипта генерації та заповнення таблиць з використанням середовища Microsoft SQL Server management. Для цього виконавши вхід в середовище керування СКБД необхідно відкрити скрипт генерації та наповнення бази даних комп'ютеризованої системи обслуговування автомобілів та викликати виконання сценарію натисканням функціональної клавіші F5, як показано на рис. 3.18.

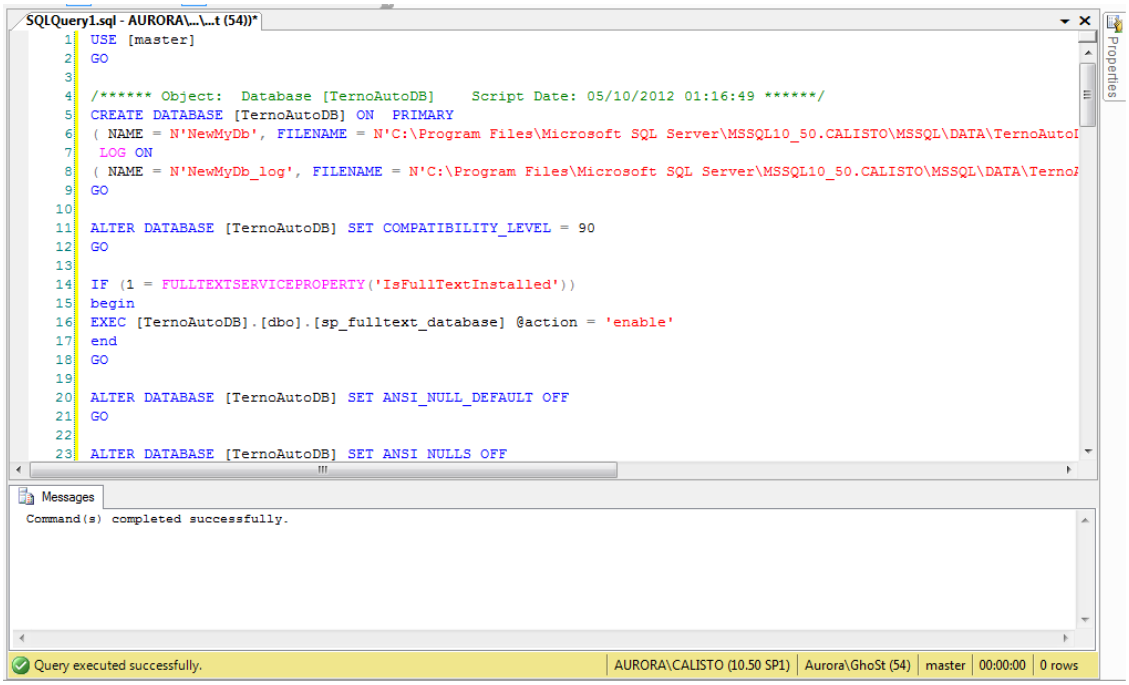

Рисунок 3.18 – Створення бази даних у середовищі MS SQL Management Studio

В разі вдалого створення бази даних та усіх таблиць буде виведено відповідне повідомлення на екран. У випадку виникнення помилок необхідно виконати їх корекцію та повторити процес розгортання бази даних.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Вимоги охорони праці при експлуатації комп'ютеризованої системи технічного обслуговування автомобілів

Під час проектування та експлуатації комп'ютеризованої системи технічного обслуговування автомобілів були врахованими вимоги нормативних документів галузі охорони праці, зокрема, вимоги до приміщень, освітлення, шуму та вібрацій, які викладено у НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» та ДСанПіН 3.3.2-007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Згідно НПАОП 0.00-1.28-10 площа на одне робоче місце, обладнане ПК, повинна становити не менше ніж 6,0 м2, а об'єм - не менше ніж 20 $м^3$.

Приміщення, де передбачається експлуатація ПК, не повинні межувати з будівлями, у яких рівні шуму і вібрації перевищує допустимі значення за нормативними документами СН 3223-85, СН 3044-84, ГР 2411-81, ГОСТ 12.1.003-83. Крім цього, необхідно передбачити звукоізоляцію огороджувальних конструкцій приміщень з ПК від шуму, що задовольняє вимогам СН 3223-85, ГОСТ 12.1.003-83, ГОСТ 12.1.012-90.

У приміщеннях, де планується використання програмного засобу комп'ютеризованої системи технічного обслуговування автомобілів, необхідно обладнати системи опалення, кондиціонування повітря та припливно-витяжної вентиляцією відповідно до СНиП 2.04.05-91. При цьому нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин повинні відповідати вимогам СН 4088-86, СН 2152-80, ГОСТ 12.1.005-88, ГОСТ 12.1.007-76.

Важливим для користувачів комп'ютеризованої системи технічного обслуговування автомобілів, зокрема системних аналітиків та менеджерів, є наявність у приміщеннях віконних прорізів, які обладнані регульованими пристроями – жалюзями.

Для внутрішнього оздоблення приміщень з ПК дозволено використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі – 0,7, для стін – 0,6, а покриття підлоги виконують з матової керамічної плитки, коефіцієнт відбиття якої становить 0,4. При цьому потрібно забезпечити антистатичні властивості підлоги.

Гігієнічні вимоги до параметрів виробничого середовища приміщень з ЕОМ включають вимоги до мікроклімату, освітленості, шуму та вібрації, неіонізуючих та іонізуючих електромагнітних випромінювань та ряду інших.

У приміщеннях з експлуатації комп'ютеризованої системи технічного обслуговування автомобілів на робочих місцях потрібно забезпечити оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря, згідно з ГОСТ 12.1.005-88, СН 4088-86. Рівні позитивних і негативних іонів у повітрі приміщень з ЕОМ мають відповідати санітарно-гігієнічним нормам № 2152-80. Для забезпечення безпеки ПК, периферійні пристрої та устаткування для обслуговування, ремонту і налагодження, електропроводи і кабелі за виконанням і ступенем захисту повинні відповідати класу зони за ПУЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ЕОМ, периферійних пристроїв і устаткування для обслуговування, ремонту та налагодження виконується як окрема групова трьохпровідна мережа, шляхом прокладки фазного, нульового робочого і нульового захисного провідників. Нульовий захисний провідник використовується для занулення (заземлення) електронних пристроїв. Використання нульового робочого провідника як нульового захисного провідника забороняється. Площа перетину нульового робочого і нульового захисного провідника в груповій трьхопровідній мережі повинна бути не менше площі перетину фазного провідника.

У приміщенні, де одночасно експлуатується чи обслуговується понад п'ять персональних ПК, на видному і доступному місці необхідно встановити аварійний резервний вимикач, що може цілком відключити електричне живлення приміщення крім освітлення. Штепсельні з'єднання та електророзетки, крім контактів фазного і нульового робочого провідників, повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх повинна бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазного і нульового робочого провідників.

Особливо неприпустимим є підключення ПК, периферійних пристроїв і устаткування для обслуговування, ремонту та налагодження до звичайної двопровідної електромережі.[19, 20]. Приміщення з ЕОМ повинні бути обладнані системою автоматичної пожежної сигналізації із димовими пожежними сповіщувачами з розрахунку 2 шт. на кожні 20 м2 площі приміщення. В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення з ПК мають бути оснащені переносними вуглекислотними вогнегасниками з розрахунку 1 на 50 м^2, але не менше 2 на приміщення. Підходи до засобів пожежогасіння повинні бути вільними.

Таким чином, визначено основні вимоги охорони праці і техніки безпеки при проектуванні та експлуатації комп’ютеризованої системи технічного обслуговування автомобілів, що дало змогу при їх дотриманні забезпечити збереження здоров’я та мінімізувати негативний вплив ПК на користувачів.

4.2 Психофізіологічне розвантаження для працівників

Успішна профілактика виробничого травматизму можлива лише за умови ретельного вивчення причин їх виникнення, в тому числі психофізіологічних.

Психофізіологічні причини – помилкові дії внаслідок втоми працівника через надмірну важкість і напруженість роботи, монотонність праці, хворобливий стан працівника, необережність, невідповідність

психофізіологічних чи антропометричних даних працівника використовуваній техніці чи виконуваній роботі [17].

До небезпечних та шкідливих психофізіологічних виробничих чинників належать фізичні (статичні, динамічні) і нервово-психічні перевантаження (розумова перенапруга, монотонність праці, перенапруження зорового, слухового, тактильного аналізаторів, емоційні перевантаження).

Так, наприклад, робота працівників у сфері інформаційних технологій характеризується:

– тривалою багатогодинною (8 годин і більше) працею в одноманітному напруженому положенні;

– малою руховою активністю при значних локальних динамічних навантаженнях кістково-м’язового апарату кистей рук.

Трудова діяльність комп’ютерних і програмних інженерів належить до категорії робіт, які пов’язані з використанням великих обсягів інформації, із застосуванням комп’ютеризованих робочих місць, із частим прийняттям відповідальних рішень в умовах дефіциту часу, безпосереднім контактом із людьми різних типів темпераменту тощо.

Тривала робота на комп’ютеризованому робочому місці призводить до значного навантаження на всі елементи зорової системи. Напружена зорова робота викликає біль, печію та різь в очах, почервоніння повік та очей. Все це зумовлює високий рівень нервово-психічного перевантаження, знижує функціональну активність центральної нервової системи, призводить до розвитку втоми, стресу [17].

Праця таких професій як будівельники, сільськогосподарські працівники і механізатори, працівники деревообробної, нафтової, газової промисловостей, металурги та ливарники, вантажники тощо обумовлена надмірним фізичним перевантаженням, що призводить до втоми м’язів через їх напруження. Чим більше навантаження та тривалість напруження м’яза, то швидше він втомлюється.

Надмірні фізичні та нервово-психічні перевантаження зумовлюють зміни у фізіологічному та психічному станах працівника, призводять до розвитку втоми та перевтоми [17].

Як відомо, розвиток втоми та перевтоми веде до порушення координації рухів, зорових розладів, неуважності, втрати пильності та контролю реальної ситуації. При цьому працівник порушує вимоги технологічних інструкцій, припускається помилок та неузгодженості в роботі; у нього знижується відчуття безпеки. Спостерігається погіршення сприйняття подразників, внаслідок чого працівник окремі подразники зовсім не сприймає, а інші сприймає із запізненням; зменшення здатності концентрувати увагу, свідомо її регулювати; посилення мимовільної уваги до побічних подразників, які відволікають працівника від трудового процесу; погіршення запам'ятовування та труднощі пригадування інформації, що знижує ефективність професійних знань; сповільнення процесів мислення; підвищення дратівливості, поява депресивних станів, зміни частоти слуху, зору.

Це призводить до того, що ближче до кінця робочої зміни збільшується кількість випадків виробничого травматизму. Згідно статистичних даних, кожному четвертому випадку передувала явно виражена втома [17].

Основні заходами у запобіганні втомлюваності, а отже і у попередженні виникнення виробничого травматизму є:

– механізація і автоматизація виробничих процесів – вони усувають фізичне напруження і велику кількість рухів руками;

– покращення санітарно-гігієнічних умов виробничого середовища (площа приміщень, мікрокліматичні умови, освітлення, вентиляція, опалення);

– професійний відбір;

– раціональна організація робочого місця, яка має бути спрямована на те, щоб конструкція виробничого устаткування відповідала антропометричним даним і психофізіологічним можливостям людини;

– правильне робоче положення; ритм роботи;

– раціоналізація трудового процесу;

– використання емоційних стимулів, впровадження раціональних режимів праці та відпочинку;

– виконання комплексу вправ для очей, рук та хребта для поліпшення мозкового кровообігу, а також комплексу прийомів психофізіологічного розвантаження.

Безпека праці є основною гарантією стабільності та якості будь якого виробництва. Відсутність випадків виробничого травматизму позначається на професійній активності працюючих, на моральному кліматі в колективі, а отже і на ефективності та продуктивності праці.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи розроблено та проаналізовано технічне завдання на проектування комп’ютеризованої системи обслуговування автомобілів.

На основі аналізу предметної області побудовано діаграму потоків даних та визначено функціональні вимоги до системи, які візуалізовано засобами діаграми варіантів використання. Окрім цього, для зберігання та маніпулювання даними спроектовано схему бази даних, що відображає важливі, з точки зору процесу автоматизації аспекти предметної області.

Базу даних реалізовано із застосуванням реляційного підходу до її проектування та інструментальними засобами СКБД MS SQL Server, зокрема утилітою MS SQL Server Management Studio.

У роботі обгрунтовано доцільність використання архітектурного патерну клієнт-сервер, спроектовано архітектуру програмного забезпечення комп’ютеризованої системи обслуговування автомобілів, враховано вимоги та аспекти до його використання.

Реалізацію програмної складової комп’ютеризованої системи виконано базуючись на підході об’єктно-орієнтованого програмування та за допомогою мови програмування C#, середовищем програмування обрано MS Visual Studio.

У роботі обгрунтовано та обчислено необхідні апаратні ресурси для функціонування сервера баз даних, а також клієнтських станцій. Також висунуто вимоги щодо системного і прикладного програмного забезпечення для ефективного функціонування комп’ютеризованої системи.

Після реалізації програмного забезпечення системи обслуговування автомобілів проведено її тестування шляхом застосування модульного тестування найбільш критичних з точки зору бізнес-процесів аспектів, зокрема щодо коректності запису даних у БД, їх пошуку та маніпулювання.

Програмне забезпечення комп’ютеризованої системи запаковано в окремий інсталяційний пакет, що дозволяє забезпечити простоту його встановлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. М.: Финансы и статистика, 2007 р. 389 с.
2. Дейт К. Введение в системы баз данных. М.: Вильямс, 2005. 1328 с.
3. Титоренко Г.А. Автоматизированные информационные технологии в экономике. М. Компьютер ЮНИТИ. 1998. 336 с.
4. Пасічник В. Організація баз даних та знань / В. Пасічник, В. Резніченко // – К:. Видавнича група BHV, 2006. – 384 с.
5. Документация по C#. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата звернення 18.04.2021 р.)
6. Введение в C#. Язык C# и платформа .NET Core. URL: https://metanit.com/sharp/tutorial/1.1.php (дата звернення 19.04.2021 р.)
7. C# Tutorial. URL: https://www.w3schools.com/cs/ (дата звернення 19.04.2021 р.)
8. SQL Syntax. URL: https://www.w3schools.com/sql/sql_syntax.asp дата звернення 19.04.2021 р.)
9. Building a simple server client application using C#. URL: https://codeabout.wordpress.com/2011/03/06/building-a-simple-server-client-application-using-c. (дата звернення 28.04.2021 р.).
10. Microsoft Visual Studio/ URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio. (дата звернення 01.05.2021 р.)
11. Code Cracker for C# URL: https://visualstudiogallery.msdn.microsoft.com/ab588981-91a5-478c-8e65-74d0ff450862nofollow. (дата звернення 10.05.2021 р.)
12. EntityFramework Core. URL: https://docs.microsoft.com/en-us/ef/core. – (дата звернення 13.05.2021 р.)
13. Автоматизация Code First и EF Power Tools. URL: https://metanit.com/sharp/entityframework/2.3.php (дата звернення 18.05.2021 р.)

14. Simple client-server interactions using C#. URL: https://www.codeproject.com/Articles/12286/Simple-Client-server-Interactions-using-C (дата звернення 22.05.2021 р.)

15. Виейра Р. Программирование баз данных Microsoft SQL Server 2008. Базовый курс. .: Пер. с англ. М.: ООО «И.Д. Вильямс», 2009. 816 с.

16. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство. Питер. 2014. 544 с.

17. Брауде Э. Технология разработки программного обеспечения. СПб: Питер, 2014. 655 с.

18. Кантор М. Управление программными проектами. Практическое руководство по разработке успешного программного обеспечения. М.: Вильямс. 2012. 176 с.

19. Астелс Д., Миллер Г., Новак М. Практическое руководство по экстремальному программированию. М.: «Вильямс». 2012. 320с.

20. Бьорнс Б. Распределенные системы. Паттерны проектирования. Питер. 2019. 224 с.

21. Гультяев А.К., Машин В.А. Проектирование и дизайн пользовательского интерфейса. С-Пб : "Корона-принт". 2000. 349 с.

22. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Київ. 2018.

23. Бедрій Я. Основи охорони праці користувачів персональних комп’ютерів: навчальний посібник для студентів ВНЗ та інженерів-практиків. Навчальна книга-Богдан. 2014. 144 с.

Додаток А.

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

"Затверджую"

Завідувач кафедри КС

_____ Осухівська Г.М.

"___" _____2021 р

КОМП'ЮТЕРИЗОВАНА СИСТЕМА ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ АВТОМОБІЛІВ

ТЕХНІЧНЕ ЗАВДАННЯ

на _*10*_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІс-44
_____ к.т.н., доц. Луцків А.М.	_____ Ковач О.С.
«____» _____ 2021 р.	«____» _____ 2021 р.

Тернопіль 2021

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютеризована система технічного обслуговування автомобілів».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.171.00.00

1.2 Виконавець

Студент групи СІс-44, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Ковач Олександр Сергійович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№ 4.7-97 від 10.02.2021 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 10.02.2021 р.

Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2021 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ІСО, ГОСТ, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи.

Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Комп’ютеризована система обслуговування автомобілів призначена для автоматизації та контролю виконаних робіт із сервісного та гарантійного обслуговування клієнтів станції технічного обслуговування.

Комп’ютеризована система повинна включати в себе як апаратну складову, так і програмну частину. У кваліфікаційній роботі необхідно проаналізувати апаратне забезпечення підприємства, що займається таким видом діяльності, організацію та канали зв’язку між активними та пасивними компонентами комп’ютерної мережі, організаційну структуру підприємства та наявні операційні системи і прикладне програмне забезпечення користувачів.

При цьому потрібно здійснити їх оцінку, провести вибір альтернативних рішень та розробити програмний комплекс для ведення та управління процесом обслуговування автомобілів. При реалізації програмного комплексу необхідно спроектувати базу даних та налаштувати сервер бази даних, а також створити зручний користувацький інтерфейс для роботи з нею. Комп’ютеризована система обслуговування автомобілів повинна генерувати звіти та візуалізувати результати роботи по кожному авто та автослюсарю за вказаний період часу. Користувачами системи є майстри цехів, бухгалтерія та адміністрація підприємства.

2.2 Мета створення системи

Метою створення комп’ютеризованої системи обслуговування автомобілів є автоматизація процесів обліку та управління щодо надання сервісних послуг власникам автотранспорту. Для досягнення мети потрібно розв’язати наступні завдання:

– автоматизація процесу сервісного та гарантійного обслуговування автомобілів;

– автоматизація процесу контролю використання запчастин;

– автоматизація процесу обліку робочого часу автослюсарів;

– автоматизація процесу нарахування заробітної плати автослюсарам підприємства;

– підвищення ефективності роботи підприємства;

– формування звітів з гарантійного та сервісного обслуговування автомобілів;

– генерування графіків завантаженості автослюсарів та продуктивності їх праці.

2.3 Характеристика об'єкту

2.3.1 Основні задачі та функції об’єкту

Основні задачі, вирішення яких покладається на комп’ютеризовану систему обслуговування автомобілів полягають в наступному:

– забезпечення зв’язку між підрозділами підприємства;

– ведення обліку сервісного обслуговування автомобілів;

– ведення обліку гарантійного обслуговування автомобілів;

– обробка звітної інформації;

– збір та обробка статистичних даних з технічного обслуговування автомобілів.

Функціями комп’ютеризованої системи є автоматизація обліку автомобілів, нарахування заробітної плати, і як наслідок підвищення продуктивності та контролю праці автослюсарів станцій технічного обслуговування.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Вимоги до комп’ютеризованої системи обслуговування автомобілів, які застосовуються станціями технічного обслуговування включають:

– надійність роботи апаратної складової інформаційної системи підприємства;
– відповідність встановленому рівню продуктивності апаратних пристроїв;
– захищеність обладнання;
– захищеність доступу до ресурсів;
– точність обробки даних;
– продуктивність роботи програмного забезпечення;
– паралельний доступ до бази даних;
– розмежування прав доступу;
– часова ефективність;
– ефективність використання ресурсів комп’ютеризованої системи;
– зручність використання програмного забезпечення.

3.1.1 Вимоги до структури та функціонування системи

До структури комп’ютеризованої системи обслуговування автомобілів входить: апаратна частина та програмна складова. Вимоги до апаратної частини:

1. Надійність функціонування апаратної складової комп’ютеризованої системи:

– час безвідмовної роботи – 100000 год.;

– наявність системи резервного живлення;
– захист від стрибків напруги;
– надійність роботи складових апаратного забезпечення;
– надійність каналів зв’язку інформаційної системи;
– захист від неавторизованого доступу;

2. Продуктивність апаратних пристроїв:

– пропускна здатність каналів зв’язку – >=100 Мб/с;
– максимальний розмір опрацьовуваних даних – 100 МБ;

3. Захищеність апаратної складової:

– фізичний захист активного обладнання;
– фізичний захист каналів зв’язку;
– фізичний та логічний захист комутаційного обладнання;

4. Відновлюваність та резервування:

– аварійне відновлення працездатності після збою – 24 год.;
– резервування каналів зв’язку.

Вимоги до програмної частини комп’ютеризованої системи:

5. Функціональність:

– наповнення та робота з довідником послуг та їх вартості;
– наповнення довідників про автослюсарів;
– наповнення та робота з довідниками авто та їхнє гарантійне обслуговування;
– ввід, редагування даних по виконаних роботах;
– нарахування заробітної плати автослюсарям;
– можливість сортування даних за визначеними критеріями;
– можливість запобігання неавторизованому доступу (логічного);
– можливість формування звітів по облікованому робочому часі;
– можливість керування правами доступу до інформаційних ресурсів;
– облік та контроль робочого часу;

6. Продуктивність:

– час реакції – 1 с;

– час відгуку – 2с;
– ефективність використання оперативної пам’яті;
– ефективність використання дискового простору;
– доступність;
– розмежування та визначення прав доступу;
– масштабованість архітектури;

7. Надійність:

– завершеність;
– стійкість до відмов;
– напрацювання на відмови;
– здатність до відновлення;

8. Зручність використання:

– однорідність стильового оформлення;
– наявність довідки;
– зрозумілість виконання операцій;
– зрозумілість одержаних результатів;
– зручність навчання.

Структура апаратної складової комп’ютеризованої системи – робочі станції, сервер баз даних, канали зв’язку, комутаційне обладнання.

Структура програмного забезпечення комп’ютеризованої системи проектується на основі архітектури клієнт-сервер.

В загальному випадку, інфологічна модель бази даних повинна відображати предметну область, а клієнтська частина – відповідати за можливості обліку даних та забезпечення їх захисту.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Зв’язок між компонентами комп’ютеризованої системи: робочі станції – сервер обробки даних – протокол TCP/IP (рівень операційної системи).

3.1.3 Вимоги по діагностуванню системи

Діагностика системи відбувається згідно з планом супроводу та обслуговування комп’ютеризованої системи в цілому та окремих її складових.

3.1.4 Перспективи розвитку, модернізація системи

Модернізація системи можлива у випадку морального старіння апаратного або програмного забезпечення, або ж у випадку радикальної зміни вимог до інформаційної системи. Перспективи розвитку системи можуть включати зміну на більш сучасну апаратну частину і як наслідок модернізація програмної складової.

3.1.5 Вимоги до надійності системи

Комп’ютеризована система обслуговування автомобілів повинна бути захищена на рівні операційної системи та авторизованого доступу до бази даних. Надійність системи повинна забезпечуватись також і у випадку збою роботи апаратного забезпечення.

3.1.6 Вимоги до функцій та задач, які виконує система

Вимоги до функцій та задач, які виконує система, включають:

– забезпечення зв’язку клієнтської частини з базою даних;
– надання точних та адекватних результатів на запит користувачів;
– забезпечення часової ефективності роботи системи;
– забезпечення контролю над доступом до інформації;
– забезпечення зручності використання програмного продукту;
– можливість розгортання бази даних.

3.1.7 Вимоги до апаратного забезпечення

Вимоги до сервера:

– процесор – тактова частота не менше 4,0 ГГц;
– об'єм оперативної пам'яті - не менше 4096 Мб;

– об'єм жорсткого диску - не менше 750 Гб.

Вимоги до робочих станцій:

– процесор - тактова частота не менше 2,2 ГГц;

– об'єм оперативної пам'яті - не менше 2048 Мб;

– об'єм жорсткого диску - не менше 250 Гб.

Периферійні пристрої:

– принтер або мультифункціональний пристрій – 5 шт.

3.1.8 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення сервера визначають платформу операційної системи на базі Windows Server 2012 R2.

Вимоги до програмного забезпечення робочих станцій включають використання платформи Windows 10 та прикладного програмного забезпечення, що використовується для забезпечення необхідного рівня продуктивності праці.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

1. Діаграма потоків даних.
2. Діаграма варіантів використання.
3. Різновиди технології клієнт-сервер.
4. Архітектура комп’ютеризованої системи.
5. ER-діаграма бази даних.
6. Алгоритм модифікації даних.

*Примітка: У комплект документації можуть вноситися міни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка та аналіз вимог технічного завдання	10.02-21.02.2021
2	Аналіз існуючих рішень при проектуванні комп'ютеризованої системи	21.02-07.03.2021
3	Проектування архітектури комп'ютеризованої системи	08.03-20.03.2021
4	Обґрунтування вибору апаратного забезпечення	20.03-26.03.2021
5	Реалізація програмного забезпечення комп'ютерної системи	27.03-10.04.2021
6	Розробка інструкцій з налаштування параметрів комп'ютеризованої системи обслуговування автомобілів	04.05-20.05.2021
7	Безпека життєдіяльності, основи охорони праці	20.05-27.05.2021
8	Оформлення кваліфікаційної роботи	27.05-10.06.2021
9	Попередній захист кваліфікаційної роботи	10.06-20.06.2021
10	Захист кваліфікаційної роботи	21.06-27.06.2021

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б.

Фрагменти лістингу програмного забезпечення комп'ютеризованої системи обслуговування автомобілів

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace TernoAutoGUI
{
  public partial class LogOn : Form
  {
      public LogOn(MainForm parental)
      {
          logged = false;
          InitializeComponent();
          this.parental = parental;
      }

      MainForm parental;

      private bool logged;

        private void btLogin_Click(object sender, EventArgs e)
        {
            string password = this.tbPassword.Text;
            try
            {
                if (password.Equals("test"))
                {
                    parental.Show();
                    logged = true;
                    parental.CurrentUser = new User("", "TESTER",
    "Приемные пункты", "", -1);
                    this.Close();
                }
                else if ((this.cbLogin.SelectedItem as
    DataRowView)["Parol"].ToString().Equals(password))
                {
                    parental.Show();
                    logged = true;
                    DataRowView dr=cbLogin.SelectedItem as
    DataRowView;
                    parental.CurrentUser = new
    User(dr["Login"].ToString(), dr["FIO"].ToString(),
```

```
    dr["RoleName"].ToString(), dr["DepartmentName"].ToString(),
    Int32.Parse(dr["DepartmentID"].ToString()));
                    this.Close();
                }
                else
                {
                    ExceptionListener.EnterError();
                }
            }
            catch
            {
                ExceptionListener.EnterError();
            }

        }

        private void Form6_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в
    таблицу "newMyDbDataSet.Users". При необходимости она может быть
    перемещена или удалена.
            this.usersTableAdapter.Fill(this.newMyDbDataSet.Users);

        }

        private void LogOn_FormClosing(object sender,
    FormClosingEventArgs e)
        {
            if (!logged)
            {
                parental.Close();
            }
        }
    }
}


namespace TernoAutoGUI
{
    partial class DetailsPrice
    {
        /// <summary>
        /// Требуется переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый ресурс
    должен быть удален; иначе ложно.</param>
        protected override void Dispose(bool disposing)
```

```
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Код, автоматически созданный конструктором форм
Windows

    /// <summary>
    /// Обязательный метод для поддержки конструктора - не
изменяйте
    /// содержимое данного метода при помощи редактора кода.
    /// </summary>
    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
        this.dataGridView1 = new
System.Windows.Forms.DataGridView();
        this.priceTableBindingSource = new
System.Windows.Forms.BindingSource(this.components);
        this.newMyDbDataSet = new TernoAutoGUI.newMyDbDataSet();
        this.label1 = new System.Windows.Forms.Label();
        this.comboBox1 = new System.Windows.Forms.ComboBox();
        this.proizvoditelBindingSource = new
System.Windows.Forms.BindingSource(this.components);
        this.priceTableTableAdapter = new
TernoAutoGUI.newMyDbDataSetTableAdapters.PriceTableTableAdapter(
);
        this.proizvoditelTableAdapter = new
TernoAutoGUI.newMyDbDataSetTableAdapters.ProizvoditelTableAdapte
r();
        this.priceIDDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.manufacturerNameDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.nameDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.costDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.partNumverDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.kindIDDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.descriptionIDDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
        this.kindNameDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
```

```
((System.ComponentModel.ISupportInitialize)(this.dataGridView1))
.BeginInit();

((System.ComponentModel.ISupportInitialize)(this.priceTableBindi
ngSource)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.newMyDbDataSet)
).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.proizvoditelBin
dingSource)).BeginInit();
        this.SuspendLayout();
        //
        // dataGridView1
        //
        this.dataGridView1.AutoGenerateColumns = false;
        this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.Aut
oSize;
        this.dataGridView1.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
        this.priceIDDataGridViewTextBoxColumn,
        this.manufacturerNameDataGridViewTextBoxColumn,
        this.nameDataGridViewTextBoxColumn,
        this.costDataGridViewTextBoxColumn,
        this.partNumverDataGridViewTextBoxColumn,
        this.kindIDDataGridViewTextBoxColumn,
        this.descriptionIDDataGridViewTextBoxColumn,
        this.kindNameDataGridViewTextBoxColumn});
        this.dataGridView1.DataSource =
this.priceTableBindingSource;
        this.dataGridView1.Enabled = false;
        this.dataGridView1.Location = new
System.Drawing.Point(36, 76);
        this.dataGridView1.Name = "dataGridView1";
        this.dataGridView1.Size = new System.Drawing.Size(562,
150);
        this.dataGridView1.TabIndex = 0;
        //
        // priceTableBindingSource
        //
        this.priceTableBindingSource.DataMember = "PriceTable";
        this.priceTableBindingSource.DataSource =
this.newMyDbDataSet;
        //
        // newMyDbDataSet
        //
        this.newMyDbDataSet.DataSetName = "newMyDbDataSet";
        this.newMyDbDataSet.SchemaSerializationMode =
System.Data.SchemaSerializationMode.IncludeSchema;
        //
```

```
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(84, 34);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(112, 13);
        this.label1.TabIndex = 1;
        this.label1.Text = "Сортувати по бренду";
        //
        // comboBox1
        //
        this.comboBox1.DataSource =
this.proizvoditelBindingSource;
        this.comboBox1.DisplayMember = "Proizvoditel";
        this.comboBox1.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
        this.comboBox1.FormattingEnabled = true;
        this.comboBox1.Location = new System.Drawing.Point(227,
26);
        this.comboBox1.Name = "comboBox1";
        this.comboBox1.Size = new System.Drawing.Size(150, 21);
        this.comboBox1.TabIndex = 2;
        this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
        //
        // proizvoditelBindingSource
        //
        this.proizvoditelBindingSource.DataMember =
"Proizvoditel";
        this.proizvoditelBindingSource.DataSource =
this.newMyDbDataSet;
        //
        // priceTableTableAdapter
        //
        this.priceTableTableAdapter.ClearBeforeFill = true;
        //
        // proizvoditelTableAdapter
        //
        this.proizvoditelTableAdapter.ClearBeforeFill = true;
        //
        // priceIDDataGridViewTextBoxColumn
        //
        this.priceIDDataGridViewTextBoxColumn.DataPropertyName =
"PriceID";
        this.priceIDDataGridViewTextBoxColumn.HeaderText =
"PriceID";
        this.priceIDDataGridViewTextBoxColumn.Name =
"priceIDDataGridViewTextBoxColumn";
        this.priceIDDataGridViewTextBoxColumn.ReadOnly = true;
        this.priceIDDataGridViewTextBoxColumn.Visible = false;
        //
        // manufacturerNameDataGridViewTextBoxColumn
```

```
        //
this.manufacturerNameDataGridViewTextBoxColumn.DataPropertyName
= "ManufacturerName";
this.manufacturerNameDataGridViewTextBoxColumn.HeaderText =
"Бренд";
        this.manufacturerNameDataGridViewTextBoxColumn.Name =
"manufacturerNameDataGridViewTextBoxColumn";
        //
        // nameDataGridViewTextBoxColumn
        //
        this.nameDataGridViewTextBoxColumn.DataPropertyName =
"Name";
        this.nameDataGridViewTextBoxColumn.HeaderText = "Назва";
        this.nameDataGridViewTextBoxColumn.Name =
"nameDataGridViewTextBoxColumn";
        //
        // costDataGridViewTextBoxColumn
        //
        this.costDataGridViewTextBoxColumn.DataPropertyName =
"Cost";
        this.costDataGridViewTextBoxColumn.HeaderText = "Ціна";
        this.costDataGridViewTextBoxColumn.Name =
"costDataGridViewTextBoxColumn";
        //
        // partNumverDataGridViewTextBoxColumn
        //
this.partNumverDataGridViewTextBoxColumn.DataPropertyName =
"PartNumver";
        this.partNumverDataGridViewTextBoxColumn.HeaderText =
"Номер";
        this.partNumverDataGridViewTextBoxColumn.Name =
"partNumverDataGridViewTextBoxColumn";
        //
        // kindIDDataGridViewTextBoxColumn
        //
        this.kindIDDataGridViewTextBoxColumn.DataPropertyName =
"KindID";
        this.kindIDDataGridViewTextBoxColumn.HeaderText =
"KindID";
        this.kindIDDataGridViewTextBoxColumn.Name =
"kindIDDataGridViewTextBoxColumn";
        this.kindIDDataGridViewTextBoxColumn.Visible = false;
        //
        // descriptionIDDataGridViewTextBoxColumn
        //
this.descriptionIDDataGridViewTextBoxColumn.DataPropertyName =
"DescriptionID";
```

```
        this.descriptionIDDataGridViewTextBoxColumn.HeaderText =
"DescriptionID";
        this.descriptionIDDataGridViewTextBoxColumn.Name =
"descriptionIDDataGridViewTextBoxColumn";
        this.descriptionIDDataGridViewTextBoxColumn.Visible =
false;
        //
        // kindNameDataGridViewTextBoxColumn
        //
        this.kindNameDataGridViewTextBoxColumn.DataPropertyName
= "KindName";
        this.kindNameDataGridViewTextBoxColumn.HeaderText =
"KindName";
        this.kindNameDataGridViewTextBoxColumn.Name =
"kindNameDataGridViewTextBoxColumn";
        this.kindNameDataGridViewTextBoxColumn.Visible = false;
        //
        // DetailsPrice
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F,
13F);
        this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(624, 273);
        this.Controls.Add(this.comboBox1);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.dataGridView1);
        this.Name = "DetailsPrice";
        this.Text = "Прайс запасних частин";
        this.Load += new System.EventHandler(this.Form10_Load);

((System.ComponentModel.ISupportInitialize)(this.dataGridView1))
.EndInit();

((System.ComponentModel.ISupportInitialize)(this.priceTableBindi
ngSource)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.newMyDbDataSet)
).EndInit();

((System.ComponentModel.ISupportInitialize)(this.proizvoditelBin
dingSource)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.DataGridView dataGridView1;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.ComboBox comboBox1;
```

```
        private System.Windows.Forms.DataGridViewTextBoxColumn
    proizvoditelDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    nazvaniezapchastiDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    partNumberDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    cenaDataGridViewTextBoxColumn;
        private newMyDbDataSet newMyDbDataSet;
        private System.Windows.Forms.BindingSource
    priceTableBindingSource;
        private
    TernoAutoGUI.newMyDbDataSetTableAdapters.PriceTableTableAdapter
    priceTableTableAdapter;
        private System.Windows.Forms.BindingSource
    proizvoditelBindingSource;
        private
    TernoAutoGUI.newMyDbDataSetTableAdapters.ProizvoditelTableAdapte
    r proizvoditelTableAdapter;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    priceIDDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    manufacturerNameDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    nameDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    costDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    partNumverDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    kindIDDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    descriptionIDDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    kindNameDataGridViewTextBoxColumn;
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace TernoAutoGUI
{
    public partial class DetailsPrice : Form
    {
        public DetailsPrice()
```

```
        {
            InitializeComponent();
        }

        private void Form10_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в
    таблицу "newMyDbDataSet.Proizvoditel". При необходимости она
    может быть перемещена или удалена.

    this.proizvoditelTableAdapter.Fill(this.newMyDbDataSet.Proizvodi
    tel);
            // TODO: данная строка кода позволяет загрузить данные в
    таблицу "newMyDbDataSet.PriceTable". При необходимости она может
    быть перемещена или удалена.

    this.priceTableTableAdapter.Fill(this.newMyDbDataSet.PriceTable)
    ;

        }

        private void fillByDetailToolStripButton_Click(object
    sender, EventArgs e)
        {
            try
            {

    this.priceTableTableAdapter.FillByDetail(this.newMyDbDataSet.Pri
    ceTable);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }

        }

        private void comboBox1_SelectedIndexChanged(object sender,
    EventArgs e)
        {
            this.priceTableBindingSource.Filter =
    (comboBox1.SelectedItem != null ? "ProizvoditelId=" +
    (comboBox1.SelectedItem as DataRowView)["ProizvoditelId"] : "");
        }
    }
}

namespace TernoAutoGUI
{
    partial class Calculation
    {
        /// <summary>
```

```
    /// Требуется переменная конструктора.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Освободить все используемые ресурсы.
    /// </summary>
    /// <param name="disposing">истинно, если управляемый ресурс
должен быть удален; иначе ложно.</param>
    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #region Код, автоматически созданный конструктором форм
Windows

    /// <summary>
    /// Обязательный метод для поддержки конструктора - не
изменяйте
    /// содержимое данного метода при помощи редактора кода.
    /// </summary>
    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
        this.textBox1 = new System.Windows.Forms.TextBox();
        this.label1 = new System.Windows.Forms.Label();
        this.cbUsluga = new System.Windows.Forms.ComboBox();
        this.bindingSource1 = new
System.Windows.Forms.BindingSource(this.components);
        this.newMyDbDataSet = new TernoAutoGUI.newMyDbDataSet();
        this.button1 = new System.Windows.Forms.Button();
        this.dataGridView1 = new
System.Windows.Forms.DataGridView();
        this.bindingSource2 = new
System.Windows.Forms.BindingSource(this.components);
        this.dataGridView2 = new
System.Windows.Forms.DataGridView();
        this.calculationDetailsBindingSource = new
System.Windows.Forms.BindingSource(this.components);
        this.label5 = new System.Windows.Forms.Label();
        this.tbSumm = new System.Windows.Forms.TextBox();
        this.label2 = new System.Windows.Forms.Label();
        this.label3 = new System.Windows.Forms.Label();
        this.comboBox2 = new System.Windows.Forms.ComboBox();
        this.priceTableBindingSource = new
System.Windows.Forms.BindingSource(this.components);
        this.cbPart = new System.Windows.Forms.ComboBox();
```

```
            this.bindingSource3 = new
System.Windows.Forms.BindingSource(this.components);
            this.label4 = new System.Windows.Forms.Label();
            this.button2 = new System.Windows.Forms.Button();
            this.label6 = new System.Windows.Forms.Label();
            this.button3 = new System.Windows.Forms.Button();
            this.calculationDetailsTableAdapter = new
TernoAutoGUI.newMyDbDataSetTableAdapters.CalculationDetailsTable
Adapter();
            this.priceTableTableAdapter = new
TernoAutoGUI.newMyDbDataSetTableAdapters.PriceTableTableAdapter(
);
            this.detalioplatuIdDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.priceIdDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.remontIdDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.costDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.nameDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.kindNameDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.proizvoditelDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.partNumverDataGridViewTextBoxColumn = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.detalioplatuIdDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.priceIdDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.remontIdDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.proizvoditelDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.costDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.nameDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.partNumverDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();
            this.kindNameDataGridViewTextBoxColumn1 = new
System.Windows.Forms.DataGridViewTextBoxColumn();

((System.ComponentModel.ISupportInitialize)(this.bindingSource1)
).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.newMyDbDataSet)
).BeginInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.dataGridView1))
.BeginInit();

((System.ComponentModel.ISupportInitialize)(this.bindingSource2)
).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.dataGridView2))
.BeginInit();

((System.ComponentModel.ISupportInitialize)(this.calculationDeta
ilsBindingSource)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.priceTableBindi
ngSource)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.bindingSource3)
).BeginInit();
        this.SuspendLayout();
        //
        // textBox1
        //
        this.textBox1.Enabled = false;
        this.textBox1.Location = new System.Drawing.Point(79,
5);
        this.textBox1.Name = "textBox1";
        this.textBox1.Size = new System.Drawing.Size(120, 20);
        this.textBox1.TabIndex = 0;
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(12, 12);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(57, 13);
        this.label1.TabIndex = 1;
        this.label1.Text = "№ заявки";
        //
        // cbUsluga
        //
        this.cbUsluga.DataSource = this.bindingSource1;
        this.cbUsluga.FormattingEnabled = true;
        this.cbUsluga.Location = new System.Drawing.Point(359,
35);
        this.cbUsluga.Name = "cbUsluga";
        this.cbUsluga.Size = new System.Drawing.Size(172, 21);
        this.cbUsluga.TabIndex = 2;
        //
        // newMyDbDataSet
        //
        this.newMyDbDataSet.DataSetName = "newMyDbDataSet";
```

```
        this.newMyDbDataSet.SchemaSerializationMode =
System.Data.SchemaSerializationMode.IncludeSchema;
        //
        // button1
        //
        this.button1.Location = new System.Drawing.Point(588,
33);
        this.button1.Name = "button1";
        this.button1.Size = new System.Drawing.Size(107, 23);
        this.button1.TabIndex = 6;
        this.button1.Text = "Додати";
        this.button1.UseVisualStyleBackColor = true;
        this.button1.Click += new
System.EventHandler(this.button1_Click);
        //
        // dataGridView1
        //
        this.dataGridView1.AutoGenerateColumns = false;
        this.dataGridView1.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.Aut
oSize;
        this.dataGridView1.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
        this.detalioplatuIdDataGridViewTextBoxColumn,
        this.priceIdDataGridViewTextBoxColumn,
        this.remontIdDataGridViewTextBoxColumn,
        this.costDataGridViewTextBoxColumn,
        this.nameDataGridViewTextBoxColumn,
        this.kindNameDataGridViewTextBoxColumn,
        this.proizvoditelDataGridViewTextBoxColumn,
        this.partNumverDataGridViewTextBoxColumn});
        this.dataGridView1.DataSource = this.bindingSource2;
        this.dataGridView1.Enabled = false;
        this.dataGridView1.Location = new
System.Drawing.Point(12, 72);
        this.dataGridView1.Name = "dataGridView1";
        this.dataGridView1.Size = new System.Drawing.Size(692,
159);
        this.dataGridView1.TabIndex = 10;
        //
        // bindingSource2
        //
        this.bindingSource2.DataMember = "CalculationDetails";
        this.bindingSource2.DataSource = this.newMyDbDataSet;
        this.bindingSource2.Filter = "KindName=\'Услуга\'";
        //
        // dataGridView2
        //
        this.dataGridView2.AutoGenerateColumns = false;
        this.dataGridView2.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.Aut
oSize;
```

```
        this.dataGridView2.Columns.AddRange(new
System.Windows.Forms.DataGridViewColumn[] {
        this.detalioplatuIdDataGridViewTextBoxColumn1,
        this.priceIdDataGridViewTextBoxColumn1,
        this.remontIdDataGridViewTextBoxColumn1,
        this.proizvoditelDataGridViewTextBoxColumn1,
        this.costDataGridViewTextBoxColumn1,
        this.nameDataGridViewTextBoxColumn1,
        this.partNumverDataGridViewTextBoxColumn1,
        this.kindNameDataGridViewTextBoxColumn1});
        this.dataGridView2.DataSource =
this.calculationDetailsBindingSource;
        this.dataGridView2.Enabled = false;
        this.dataGridView2.Location = new
System.Drawing.Point(12, 280);
        this.dataGridView2.Name = "dataGridView2";
        this.dataGridView2.Size = new System.Drawing.Size(692,
147);
        this.dataGridView2.TabIndex = 15;
        //
        // calculationDetailsBindingSource
        //
        this.calculationDetailsBindingSource.DataMember =
"CalculationDetails";
        this.calculationDetailsBindingSource.DataSource =
this.newMyDbDataSet;
        this.calculationDetailsBindingSource.Filter =
"KindName=\'Деталь\'";
        //
        // label5
        //
        this.label5.AutoSize = true;
        this.label5.BackColor =
System.Drawing.SystemColors.Control;
        this.label5.Font = new System.Drawing.Font("Microsoft
Sans Serif", 9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
        this.label5.ForeColor =
System.Drawing.SystemColors.InactiveCaptionText;
        this.label5.ImageKey = "(отсутствует)";
        this.label5.Location = new System.Drawing.Point(58,
454);
        this.label5.Name = "label5";
        this.label5.Size = new System.Drawing.Size(200, 16);
        this.label5.TabIndex = 18;
        this.label5.Text = "Кінцева вартість ремонту";
        this.label5.TextAlign =
System.Drawing.ContentAlignment.TopCenter;
        //
        // tbSumm
        //
        this.tbSumm.Enabled = false;
```

```
            this.tbSumm.Location = new System.Drawing.Point(303,
450);
            this.tbSumm.Name = "tbSumm";
            this.tbSumm.Size = new System.Drawing.Size(228, 20);
            this.tbSumm.TabIndex = 19;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Arial",
9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label2.Location = new System.Drawing.Point(281,
40);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(60, 16);
            this.label2.TabIndex = 20;
            this.label2.Text = "Послуга";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Arial",
9.75F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label3.Location = new System.Drawing.Point(58,
245);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(69, 16);
            this.label3.TabIndex = 21;
            this.label3.Text = "Запчасть";
            //
            // comboBox2
            //
            this.comboBox2.DataSource =
this.priceTableBindingSource;
            this.comboBox2.DisplayMember = "ManufacturerName";
            this.comboBox2.FormattingEnabled = true;
            this.comboBox2.Location = new System.Drawing.Point(205,
244);
            this.comboBox2.Name = "comboBox2";
            this.comboBox2.Size = new System.Drawing.Size(140, 21);
            this.comboBox2.TabIndex = 22;
            this.comboBox2.SelectedIndexChanged += new
System.EventHandler(this.comboBox2_SelectedIndexChanged);
            //
            // priceTableBindingSource
            //
            this.priceTableBindingSource.DataMember = "PriceTable";
            this.priceTableBindingSource.DataSource =
this.newMyDbDataSet;
            //
```

```
        // cbPart
        //
        this.cbPart.DataSource = this.bindingSource3;
        this.cbPart.DisplayMember = "PartNumver";
        this.cbPart.FormattingEnabled = true;
        this.cbPart.Location = new System.Drawing.Point(438,
244);
        this.cbPart.Name = "cbPart";
        this.cbPart.Size = new System.Drawing.Size(121, 21);
        this.cbPart.TabIndex = 23;
        //
        // bindingSource3
        //
        this.bindingSource3.DataMember = "PriceTable";
        this.bindingSource3.DataSource = this.newMyDbDataSet;
        //
        // label4
        //
        this.label4.AutoSize = true;
        this.label4.Location = new System.Drawing.Point(148,
248);
        this.label4.Name = "label4";
        this.label4.Size = new System.Drawing.Size(38, 13);
        this.label4.TabIndex = 24;
        this.label4.Text = "Бренд";
        //
        // button2
        //
        this.button2.Location = new System.Drawing.Point(597,
242);
        this.button2.Name = "button2";
        this.button2.Size = new System.Drawing.Size(107, 23);
        this.button2.TabIndex = 25;
        this.button2.Text = "Додати";
        this.button2.UseVisualStyleBackColor = true;
        this.button2.Click += new
System.EventHandler(this.button2_Click);
        //
        // label6
        //
        this.label6.AutoSize = true;
        this.label6.Location = new System.Drawing.Point(356,
248);
        this.label6.Name = "label6";
        this.label6.Size = new System.Drawing.Size(75, 13);
        this.label6.TabIndex = 26;
        this.label6.Text = "Номер деталі";
        //
        // button3
        //
        this.button3.Location = new System.Drawing.Point(577,
446);
```

```
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(94, 23);
            this.button3.TabIndex = 27;
            this.button3.Text = "Зберегти";
            this.button3.UseVisualStyleBackColor = true;
            //
            // calculationDetailsTableAdapter
            //
            this.calculationDetailsTableAdapter.ClearBeforeFill =
true;
            //
            // priceTableTableAdapter
            //
            this.priceTableTableAdapter.ClearBeforeFill = true;
            //
            // detalioplatuIdDataGridViewTextBoxColumn
            //

this.detalioplatuIdDataGridViewTextBoxColumn.DataPropertyName =
"Detali_oplatuId";
            this.detalioplatuIdDataGridViewTextBoxColumn.HeaderText
= "Detali_oplatuId";
            this.detalioplatuIdDataGridViewTextBoxColumn.Name =
"detalioplatuIdDataGridViewTextBoxColumn";
            this.detalioplatuIdDataGridViewTextBoxColumn.ReadOnly =
true;
            this.detalioplatuIdDataGridViewTextBoxColumn.Visible =
false;
            //
            // priceIdDataGridViewTextBoxColumn
            //
            this.priceIdDataGridViewTextBoxColumn.DataPropertyName =
"PriceId";
            this.priceIdDataGridViewTextBoxColumn.HeaderText =
"PriceId";
            this.priceIdDataGridViewTextBoxColumn.Name =
"priceIdDataGridViewTextBoxColumn";
            this.priceIdDataGridViewTextBoxColumn.Visible = false;
            //
            // remontIdDataGridViewTextBoxColumn
            //
            this.remontIdDataGridViewTextBoxColumn.DataPropertyName
= "RemontId";
            this.remontIdDataGridViewTextBoxColumn.HeaderText =
"RemontId";
            this.remontIdDataGridViewTextBoxColumn.Name =
"remontIdDataGridViewTextBoxColumn";
            this.remontIdDataGridViewTextBoxColumn.Visible = false;
            //
            // costDataGridViewTextBoxColumn
            //
```

```
        this.costDataGridViewTextBoxColumn.DataPropertyName =
"Cost";
        this.costDataGridViewTextBoxColumn.HeaderText = "Ціна";
        this.costDataGridViewTextBoxColumn.Name =
"costDataGridViewTextBoxColumn";
        this.costDataGridViewTextBoxColumn.Width = 150;
        //
        // nameDataGridViewTextBoxColumn
        //
        this.nameDataGridViewTextBoxColumn.DataPropertyName =
"Name";
        this.nameDataGridViewTextBoxColumn.HeaderText = "Назва
послуги";
        this.nameDataGridViewTextBoxColumn.Name =
"nameDataGridViewTextBoxColumn";
        this.nameDataGridViewTextBoxColumn.Width = 450;
        //
        // kindNameDataGridViewTextBoxColumn
        //
        this.kindNameDataGridViewTextBoxColumn.DataPropertyName
= "KindName";
        this.kindNameDataGridViewTextBoxColumn.HeaderText =
"KindName";
        this.kindNameDataGridViewTextBoxColumn.Name =
"kindNameDataGridViewTextBoxColumn";
        this.kindNameDataGridViewTextBoxColumn.Visible = false;
        //
        // proizvoditelDataGridViewTextBoxColumn
        //

this.proizvoditelDataGridViewTextBoxColumn.DataPropertyName =
"Proizvoditel";
        this.proizvoditelDataGridViewTextBoxColumn.HeaderText =
"Proizvoditel";
        this.proizvoditelDataGridViewTextBoxColumn.Name =
"proizvoditelDataGridViewTextBoxColumn";
        this.proizvoditelDataGridViewTextBoxColumn.Visible =
false;
        //
        // partNumverDataGridViewTextBoxColumn
        //

this.partNumverDataGridViewTextBoxColumn.DataPropertyName =
"PartNumver";
        this.partNumverDataGridViewTextBoxColumn.HeaderText =
"PartNumver";
        this.partNumverDataGridViewTextBoxColumn.Name =
"partNumverDataGridViewTextBoxColumn";
        this.partNumverDataGridViewTextBoxColumn.Visible =
false;
        //
        // detalioplatuIdDataGridViewTextBoxColumn1
```

```
        //

this.detalioplatuIdDataGridViewTextBoxColumn1.DataPropertyName =
"Detali_oplatuId";
        this.detalioplatuIdDataGridViewTextBoxColumn1.HeaderText
= "Detali_oplatuId";
        this.detalioplatuIdDataGridViewTextBoxColumn1.Name =
"detalioplatuIdDataGridViewTextBoxColumn1";
        this.detalioplatuIdDataGridViewTextBoxColumn1.ReadOnly =
true;
        this.detalioplatuIdDataGridViewTextBoxColumn1.Visible =
false;
        //
        // priceIdDataGridViewTextBoxColumn1
        //
        this.priceIdDataGridViewTextBoxColumn1.DataPropertyName
= "PriceId";
        this.priceIdDataGridViewTextBoxColumn1.HeaderText =
"PriceId";
        this.priceIdDataGridViewTextBoxColumn1.Name =
"priceIdDataGridViewTextBoxColumn1";
        this.priceIdDataGridViewTextBoxColumn1.Visible = false;
        //
        // remontIdDataGridViewTextBoxColumn1
        //
        this.remontIdDataGridViewTextBoxColumn1.DataPropertyName
= "RemontId";
        this.remontIdDataGridViewTextBoxColumn1.HeaderText =
"RemontId";
        this.remontIdDataGridViewTextBoxColumn1.Name =
"remontIdDataGridViewTextBoxColumn1";
        this.remontIdDataGridViewTextBoxColumn1.Visible = false;
        //
        // proizvoditelDataGridViewTextBoxColumn1
        //

this.proizvoditelDataGridViewTextBoxColumn1.DataPropertyName =
"Proizvoditel";
        this.proizvoditelDataGridViewTextBoxColumn1.HeaderText =
"Бренд";
        this.proizvoditelDataGridViewTextBoxColumn1.Name =
"proizvoditelDataGridViewTextBoxColumn1";
        this.proizvoditelDataGridViewTextBoxColumn1.Width = 150;
        //
        // costDataGridViewTextBoxColumn1
        //
        this.costDataGridViewTextBoxColumn1.DataPropertyName =
"Cost";
        this.costDataGridViewTextBoxColumn1.HeaderText = "Ціна";
        this.costDataGridViewTextBoxColumn1.Name =
"costDataGridViewTextBoxColumn1";
        this.costDataGridViewTextBoxColumn1.Width = 150;
```

```
            //
            // nameDataGridViewTextBoxColumn1
            //
            this.nameDataGridViewTextBoxColumn1.DataPropertyName =
"Name";
            this.nameDataGridViewTextBoxColumn1.HeaderText =
"Назва";
            this.nameDataGridViewTextBoxColumn1.Name =
"nameDataGridViewTextBoxColumn1";
            this.nameDataGridViewTextBoxColumn1.Width = 250;
            //
            // partNumverDataGridViewTextBoxColumn1
            //

this.partNumverDataGridViewTextBoxColumn1.DataPropertyName =
"PartNumver";
            this.partNumverDataGridViewTextBoxColumn1.HeaderText =
"Номер деталі";
            this.partNumverDataGridViewTextBoxColumn1.Name =
"partNumverDataGridViewTextBoxColumn1";
            //
            // kindNameDataGridViewTextBoxColumn1
            //
            this.kindNameDataGridViewTextBoxColumn1.DataPropertyName
= "KindName";
            this.kindNameDataGridViewTextBoxColumn1.HeaderText =
"KindName";
            this.kindNameDataGridViewTextBoxColumn1.Name =
"kindNameDataGridViewTextBoxColumn1";
            this.kindNameDataGridViewTextBoxColumn1.Visible = false;
            //
            // Calculation
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F,
13F);
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(716, 492);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.cbPart);
            this.Controls.Add(this.comboBox2);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.tbSumm);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.dataGridView2);
            this.Controls.Add(this.dataGridView1);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.cbUsluga);
```

```
        this.Controls.Add(this.label1);
        this.Controls.Add(this.textBox1);
        this.Name = "Calculation";
        this.Text = "Калькуляція";
        this.Load += new System.EventHandler(this.Form8_Load);

((System.ComponentModel.ISupportInitialize)(this.bindingSource1)
).EndInit();

((System.ComponentModel.ISupportInitialize)(this.newMyDbDataSet)
).EndInit();

((System.ComponentModel.ISupportInitialize)(this.dataGridView1))
.EndInit();

((System.ComponentModel.ISupportInitialize)(this.bindingSource2)
).EndInit();

((System.ComponentModel.ISupportInitialize)(this.dataGridView2))
.EndInit();

((System.ComponentModel.ISupportInitialize)(this.calculationDeta
ilsBindingSource)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.priceTableBindi
ngSource)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.bindingSource3)
).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();
    }
    #endregion
    private System.Windows.Forms.TextBox textBox1;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.ComboBox cbUsluga;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.DataGridView dataGridView1;
    private System.Windows.Forms.DataGridView dataGridView2;
    private System.Windows.Forms.TextBox tbSumm;
    private System.Windows.Forms.Label label5;
   private System.Windows.Forms.DataGridViewTextBoxColumn
praisUIdDataGridViewTextBoxColumn;
    private System.Windows.Forms.DataGridViewTextBoxColumn
cenaDataGridViewTextBoxColumn;
    private System.Windows.Forms.DataGridViewTextBoxColumn
nazvanieuslugiDataGridViewTextBoxColumn;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.ComboBox comboBox2;
    private System.Windows.Forms.ComboBox cbPart;
    private System.Windows.Forms.Label label4;
```

```
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Button button3;
        private newMyDbDataSet newMyDbDataSet;
        private System.Windows.Forms.BindingSource bindingSource2;
        private
    TernoAutoGUI.newMyDbDataSetTableAdapters.CalculationDetailsTable
    Adapter calculationDetailsTableAdapter;
        private System.Windows.Forms.BindingSource
    calculationDetailsBindingSource;
        private System.Windows.Forms.BindingSource
    priceTableBindingSource;
        private
    TernoAutoGUI.newMyDbDataSetTableAdapters.PriceTableTableAdapter
    priceTableTableAdapter;
        private System.Windows.Forms.BindingSource bindingSource1;
        private System.Windows.Forms.BindingSource bindingSource3;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    detalioplatuIdDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    priceIdDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    remontIdDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    costDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    nameDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    kindNameDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    proizvoditelDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    partNumverDataGridViewTextBoxColumn;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    detalioplatuIdDataGridViewTextBoxColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    priceIdDataGridViewTextBoxColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    remontIdDataGridViewTextBoxColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    proizvoditelDataGridViewTextBoxColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    costDataGridViewTextBoxColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    nameDataGridViewTextBoxColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    partNumverDataGridViewTextBoxColumn1;
        private System.Windows.Forms.DataGridViewTextBoxColumn
    kindNameDataGridViewTextBoxColumn1;
    }
}
```

Додаток В.

Скрипт створення бази даних комп’ютеризованої системи обслуговування автомобілів

Додаток В

Скрипт генерування бази даних

```
USE [TernoAutoDB]
GO
/****** Object:  Table [dbo].[PunktSklad]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[ProductsKinds](
    [KindID] [int] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](75) NULL,
PRIMARY KEY CLUSTERED
(
    [KindID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Client]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Client](
    [ClientId] [bigint] IDENTITY(1,1) NOT NULL,
    [FIO] [varchar](50) NOT NULL,
    [Adress] [varchar](50) NOT NULL,
    [Telefon] [varchar](50) NOT NULL,
    [Email] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Client] PRIMARY KEY CLUSTERED
(
    [ClientId] ASC
```

```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[MasterSalaryRates]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[MasterSalaryRates](
    [MasterSalaryId] [int] NOT NULL,
    [MasterId] [int] NOT NULL,
    [SalaryRate] [money] NOT NULL,
    [StartDate] [date] NOT NULL,
    [ToDate] [date] NOT NULL,
    [IsHourRate] [bit] NOT NULL
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Roles]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Roles](
    [RoleID] [int] IDENTITY(1,1) NOT NULL,
    [Description] [varchar](50) NULL,
PRIMARY KEY CLUSTERED
(
    [RoleID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Tip_apparata]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[TransportKind](
```

```
    [TransportKindId] [bigint] IDENTITY(1,1) NOT NULL,
    [TransportKind] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Tip_apparata] PRIMARY KEY CLUSTERED
(
    [TransportKindId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Status_remonta]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Status_remonta](
    [Status_remontaId] [bigint] NOT NULL,
    [Vid_remonta] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Status_remonta] PRIMARY KEY CLUSTERED
(
    [Status_remontaId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Otdel_Remonta]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Otdel_Remonta](
    [OtdelID] [bigint] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](100) NULL,
PRIMARY KEY CLUSTERED
(
    [OtdelID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
/****** Object:  StoredProcedure [dbo].[SelectTipById]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectTipById]
(
    @Tip_apparataId bigint
)
AS
    SET NOCOUNT ON;
SELECT Tip_apparataId, Tip_apparata FROM dbo.Tip_apparata Where
    Tip_apparataId=@Tip_apparataId
GO
/****** Object:  StoredProcedure [dbo].[SelectStatusById]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectStatusById]
(
    @Status_remontaId bigint
)
AS
    SET NOCOUNT ON;
SELECT Status_remontaId, Vid_remonta FROM dbo.Status_remonta
WHERE Status_remontaId=@Status_remontaId
GO
/****** Object:  StoredProcedure [dbo].[UpdateClient]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[UpdateClient]
(
    @FIO varchar(50),
    @Adress varchar(50),
    @Telefon varchar(50),
    @Email varchar(50),
    @ClientId bigint
)
AS
    SET NOCOUNT OFF;
UPDATE [dbo].[Client] SET [FIO] = @FIO, [Adress] = @Adress,
    [Telefon] = @Telefon, [Email] = @Email  WHERE (ClientId =
    @ClientId)
GO
```

```
/****** Object:  StoredProcedure [dbo].[SelectClientById]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectClientById]
(
    @ClientId bigint
)
AS
    SET NOCOUNT ON;
SELECT ClientId, FIO, Adress, Telefon, Email FROM dbo.Client WHERE
    clientId=@ClientId
GO
/****** Object:  StoredProcedure [dbo].[SelectProizvoditelById]
    Script Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectProizvoditelById]
(
    @ProizvoditelId bigint
)
AS
    SET NOCOUNT ON;
SELECT ProizvoditelId, Proizvoditel FROM dbo.Proizvoditel
WHERE ProizvoditelId=@ProizvoditelId
GO
/****** Object:  Table [dbo].[Oborudovanie]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Oborudovanie](
    [OborudovanieId] [bigint] IDENTITY(1,1) NOT NULL,
    [Tip_apparataId] [bigint] NOT NULL,
    [ProizvoditelId] [bigint] NOT NULL,
    [Model] [varchar](50) NULL,
    [Nomer_garantii] [varchar](max) NULL,
 CONSTRAINT [PK_Oborudovanie] PRIMARY KEY CLUSTERED
(
    [OborudovanieId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Master]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Master](
    [MasterId] [bigint] IDENTITY(1,1) NOT NULL,
    [FIO_mastera] [varchar](50) NOT NULL,
    [Otdel_remontaId] [bigint] NOT NULL,
 CONSTRAINT [PK_Master] PRIMARY KEY CLUSTERED
(
    [MasterId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[ProductDescription]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ProductDescription](
    [DescriptionID] [int] IDENTITY(1,1) NOT NULL,
    [ManufacturerID] [bigint] NOT NULL,
    [PartNumver] [varchar](100) NULL,
PRIMARY KEY CLUSTERED
(
    [DescriptionID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[PriceTable]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[PriceTable](
    [PriceID] [int] IDENTITY(1,1) NOT NULL,
    [Cost] [money] NOT NULL,
    [Name] [varchar](100) NULL,
    [KindID] [int] NOT NULL,
    [DescriptionID] [int] NULL,
PRIMARY KEY CLUSTERED
(
    [PriceID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[Polzovatel]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Polzovatel](
    [PolzovatelID] [bigint] IDENTITY(1,1) NOT NULL,
    [Login] [varchar](50) NOT NULL,
    [Parol] [char](10) NOT NULL,
    [RoleID] [int] NOT NULL,
    [MaterID] [bigint] NOT NULL,
 CONSTRAINT [PK_Polzovatel] PRIMARY KEY CLUSTERED
(
    [PolzovatelID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  StoredProcedure [dbo].[InsertOborydovanije]
    Script Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[InsertOborydovanije]
(
    @Tip_apparataId bigint,
    @ProizvoditelId bigint,
```

```
    @Model varchar(50)
)
AS
    SET NOCOUNT OFF;
INSERT INTO [dbo].[Oborudovanie] ([Tip_apparataId],
    [ProizvoditelId], [Model]) VALUES (@Tip_apparataId,
    @ProizvoditelId, @Model)
GO
/****** Object:  Table [dbo].[Remont]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Remont](
    [RemontId] [bigint] IDENTITY(1,1) NOT NULL,
    [MasterId] [bigint] NOT NULL,
    [OborudovanieId] [bigint] NOT NULL,
    [ClientId] [bigint] NOT NULL,
    [Data_priema] [datetime] NOT NULL,
    [Data_gotovnosti] [datetime] NULL,
    [Data_vidachi] [datetime] NULL,
    [Status_remontaId] [bigint] NOT NULL,
    [Seriinii_nomer] [varchar](50) NOT NULL,
    [Zayavlennii_defect] [varchar](50) NOT NULL,
    [Diagnostika] [varchar](50) NOT NULL,
    [Zakluchenie] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Remont] PRIMARY KEY CLUSTERED
(
    [RemontId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  StoredProcedure [dbo].[SelectOborudovanieById]
    Script Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectOborudovanieById]
(
    @OborudovanieId bigint
)
AS
    SET NOCOUNT ON;
```

```
SELECT OborudovanieId, Tip_apparataId, ProizvoditelId, Model,
    Nomer_garantii FROM dbo.Oborudovanie WHERE
    OborudovanieId=@OborudovanieId
GO
/****** Object:  StoredProcedure [dbo].[SelectMasterById]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectMasterById]
(
    @MasterId bigint
)
AS
    SET NOCOUNT ON;
SELECT MasterId, FIO_mastera, Otdel_remontaId FROM dbo.Master
WHERE MasterId=@MasterId
GO
/****** Object:  StoredProcedure [dbo].[UpdateOborydovanije]
    Script Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[UpdateOborydovanije]
(
    @Tip_apparataId bigint,
    @ProizvoditelId bigint,
    @Model varchar(50),
    @OborudovanieId bigint
)
AS
    SET NOCOUNT OFF;
UPDATE [dbo].[Oborudovanie] SET [Tip_apparataId] = @Tip_apparataId,
    [ProizvoditelId] = @ProizvoditelId, [Model] = @Model WHERE
    (OborudovanieId = @OborudovanieId)
GO
/****** Object:  StoredProcedure [dbo].[UpdateRemont]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[UpdateRemont]
(
    @MasterId bigint,
    @OborudovanieId bigint,
    @ClientId bigint,
    @Data_priema datetime,
    @Data_gotovnosti datetime,
    @Data_vidachi datetime,
```

```
    @Status_remontaId bigint,
    @Seriinii_nomer varchar(50),
    @Zayavlennii_defect varchar(50),
    @Diagnostika varchar(50),
    @Zakluchenie varchar(50),
    @RemontId bigint
)
AS
    SET NOCOUNT OFF;
UPDATE       Remont
SET               MasterId = @MasterId, OborudovanieId =
    @OborudovanieId, ClientId = @ClientId, Data_priema =
    @Data_priema, Data_gotovnosti = @Data_gotovnosti,
                          Data_vidachi = @Data_vidachi,
    Status_remontaId = @Status_remontaId, Seriinii_nomer =
    @Seriinii_nomer, Zayavlennii_defect = @Zayavlennii_defect,
                          Diagnostika = @Diagnostika, Zakluchenie =
    @Zakluchenie
WHERE        (RemontId = @RemontId)
GO
/****** Object:  StoredProcedure [dbo].[SelectSearch]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectSearch]
(
    @RemontID bigint,
    @FIO varchar(50),
    @ProizvoditelId bigint,
    @OtdelId bigint,
    @TipApparata bigint
)
AS
    SET NOCOUNT ON;
SELECT        r.RemontId, r.MasterId, r.OborudovanieId, r.ClientId,
    r.Data_priema, r.Data_gotovnosti, r.Data_vidachi,
    r.Status_remontaId, r.Seriinii_nomer, r.Zayavlennii_defect,
                          r.Diagnostika, r.Zakluchenie,
    ta.Tip_apparata as AppType, m.FIO_mastera as MasterName
FROM            Remont AS r LEFT OUTER JOIN
                          Client AS c ON c.ClientId = r.ClientId LEFT
    OUTER JOIN
                          Oborudovanie AS o ON o.OborudovanieId =
    r.OborudovanieId LEFT OUTER JOIN
                          Proizvoditel AS p ON p.ProizvoditelId =
    o.ProizvoditelId LEFT OUTER JOIN
                          Master AS m ON m.MasterId = r.MasterId LEFT
    OUTER JOIN
                          Otdel_Remonta AS orm ON orm.OtdelID =
    m.Otdel_remontaId LEFT OUTER JOIN
```

```
                        Tip_apparata AS ta ON ta.Tip_apparataId =
    o.Tip_apparataId
WHERE       (r.RemontId = @RemontID) OR (c.FIO LIKE @FIO) OR
    (p.ProizvoditelId = @ProizvoditelId) OR (orm.OtdelID = @OtdelId)
    OR (ta.Tip_apparataId = @TipApparata)
GO
/****** Object:  StoredProcedure [dbo].[SelectRemontDetails]
    Script Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROC [dbo].[SelectRemontDetails]
AS
SELECT * FROM Remont as r
LEFT JOIN [Master] as m
ON r.MasterId=m.MasterId
LEFT JOIN Oborudovanie as o
ON o.OborudovanieId=r.OborudovanieId
LEFT JOIN Client as c
ON c.ClientId=r.ClientId
LEFT JOIN Status_remonta as sr
ON sr.Status_remontaId=r.Status_remontaId
GO
/****** Object:  StoredProcedure [dbo].[SelectRemontById]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectRemontById]
(
    @RemontId bigint
)
AS
    SET NOCOUNT ON;
SELECT       RemontId, MasterId, OborudovanieId, ClientId,
    Data_priema, Data_gotovnosti, Data_vidachi, Status_remontaId,
    Seriinii_nomer, Zayavlennii_defect, Diagnostika,
                        Zakluchenie
FROM           Remont
WHERE RemontId=@RemontId
GO
/****** Object:  StoredProcedure [dbo].[SelectUslugi]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectUslugi]
AS
    SET NOCOUNT ON;
```

```
SELECT        pt.PriceID, pt.Cost, pt.Name, pt.KindID,
    pt.DescriptionID, pk.Name AS KindName
FROM            PriceTable AS pt LEFT OUTER JOIN
                         ProductsKinds AS pk ON pt.KindID =
    pk.KindID
WHERE        (pk.Name = 'Послуга')
GO
/****** Object:  StoredProcedure [dbo].[SelectDetails]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SelectDetails]
AS
    SET NOCOUNT ON;
SELECT PriceID, Cost, pt.Name,pt.KindID, pt.DescriptionID, pk.Name
    as KindName,pd.PartNumver, pr.Proizvoditel as ManufacturerName
    FROM dbo.PriceTable pt
LEFT JOIN ProductsKinds pk
ON pt.KindID=pk.KindID
LEFT JOIN ProductDescription pd
ON pd.DescriptionID=pt.DescriptionID
LEFT JOIN Proizvoditel pr
ON pr.ProizvoditelId=pd.ManufacturerID
WHERE pk.Name='Деталь'
GO
/****** Object:  Table [dbo].[Oplata]    Script Date: 05/10/2021
    01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Oplata](
    [OplataId] [bigint] IDENTITY(1,1) NOT NULL,
    [Data_oplati] [datetime] NOT NULL,
    [Dokument] [varchar](50) NOT NULL,
    [RemontId] [bigint] NOT NULL,
    [Summ] [money] NOT NULL,
 CONSTRAINT [PK_Oplata] PRIMARY KEY CLUSTERED
(
    [OplataId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
```

```
/****** Object:  Table [dbo].[Mestonahozhdenie]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Mestonahozhdenie](
    [MestonahozhdenieId] [bigint] IDENTITY(1,1) NOT NULL,
    [PunktSkladId] [bigint] NOT NULL,
    [RemontId] [bigint] NOT NULL,
    [Data_peremesheniya] [datetime] NOT NULL,
 CONSTRAINT [PK_Mestonahozhdenie] PRIMARY KEY CLUSTERED
(
    [MestonahozhdenieId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[CalculationDetails]    Script Date:
    05/10/2021 01:27:25 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[CalculationDetails](
    [Detali_oplatuId] [bigint] IDENTITY(1,1) NOT NULL,
    [PriceId] [int] NOT NULL,
    [RemontId] [bigint] NOT NULL,
PRIMARY KEY CLUSTERED
(
    [Detali_oplatuId] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
    IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
    ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  StoredProcedure [dbo].[InsertMesto]    Script Date:
    05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[InsertMesto]
(
    @SkladId bigint,
    @Data datetime,
    @RemontId bigint
)
AS
    SET NOCOUNT OFF;
```

```
INSERT INTO
    Mestonahozhdenie(PunktSkladId,Data_peremesheniya,RemontId)
VALUES (@SkladId,@Data,@RemontId)
GO
/****** Object:  StoredProcedure [dbo].[InsertCalculation]    Script
    Date: 05/10/2021 01:27:27 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[InsertCalculation]
(
    @PriceId int,
    @RemontId bigint
)
AS
    SET NOCOUNT OFF;
INSERT INTO CalculationDetails(PriceId, RemontId)
VALUES (@PriceId, @RemontId)
GO
/****** Object:  StoredProcedure [dbo].[GetSumm]    Script Date:
    05/10/2021 01:27:26 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[GetSumm]
(
    @RemontId bigint
)
AS
    SET NOCOUNT ON;
SELECT        SUM(pt.Cost) AS CostSumm
FROM            CalculationDetails AS cd LEFT OUTER JOIN
                         PriceTable AS pt ON cd.PriceId = pt.PriceID
WHERE        (cd.RemontId = @RemontId)
GO
/****** Object:  Default [DF_MasterSalaryRates_IsHourRate]    Script
    Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[MasterSalaryRates] ADD  CONSTRAINT
    [DF_MasterSalaryRates_IsHourRate]  DEFAULT ((0)) FOR
    [IsHourRate]
GO
/****** Object:  Default [DF__Oplata__Summ__22751F6C]    Script
    Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Oplata] ADD  DEFAULT ((0)) FOR [Summ]
GO
/****** Object:  ForeignKey [FK__Calculati__Price__19DFD96B]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[CalculationDetails]  WITH CHECK ADD FOREIGN
    KEY([PriceId])
REFERENCES [dbo].[PriceTable] ([PriceID])
```

```
GO
/****** Object:  ForeignKey [FK__Calculati__Remon__1AD3FDA4]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[CalculationDetails]  WITH CHECK ADD FOREIGN
    KEY([RemontId])
REFERENCES [dbo].[Remont] ([RemontId])
ON DELETE CASCADE
GO
/****** Object:  ForeignKey [FK_Otdel]    Script Date: 05/10/2021
    01:27:25 ******/
ALTER TABLE [dbo].[Master]  WITH CHECK ADD  CONSTRAINT [FK_Otdel]
    FOREIGN KEY([Otdel_remontaId])
REFERENCES [dbo].[Otdel_Remonta] ([OtdelID])
GO
ALTER TABLE [dbo].[Master] CHECK CONSTRAINT [FK_Otdel]
GO
/****** Object:  ForeignKey [FK_Mestonahozhdenie_PunktSklad]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Mestonahozhdenie]  WITH CHECK ADD  CONSTRAINT
    [FK_Mestonahozhdenie_PunktSklad] FOREIGN
    KEY([MestonahozhdenieId])
REFERENCES [dbo].[PunktSklad] ([PunktSkaldId])
GO
ALTER TABLE [dbo].[Mestonahozhdenie] CHECK CONSTRAINT
    [FK_Mestonahozhdenie_PunktSklad]
GO
/****** Object:  ForeignKey [FK_RemontMesto]    Script Date:
    05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Mestonahozhdenie]  WITH CHECK ADD  CONSTRAINT
    [FK_RemontMesto] FOREIGN KEY([RemontId])
REFERENCES [dbo].[Remont] ([RemontId])
GO
ALTER TABLE [dbo].[Mestonahozhdenie] CHECK CONSTRAINT
    [FK_RemontMesto]
GO
/****** Object:  ForeignKey [FK_ProizvObor]    Script Date:
    05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Oborudovanie]  WITH CHECK ADD  CONSTRAINT
    [FK_ProizvObor] FOREIGN KEY([ProizvoditelId])
REFERENCES [dbo].[Proizvoditel] ([ProizvoditelId])
GO
ALTER TABLE [dbo].[Oborudovanie] CHECK CONSTRAINT [FK_ProizvObor]
GO
/****** Object:  ForeignKey [FK_TipObor]    Script Date: 05/10/2021
    01:27:25 ******/
ALTER TABLE [dbo].[Oborudovanie]  WITH CHECK ADD  CONSTRAINT
    [FK_TipObor] FOREIGN KEY([Tip_apparataId])
REFERENCES [dbo].[Tip_apparata] ([Tip_apparataId])
GO
ALTER TABLE [dbo].[Oborudovanie] CHECK CONSTRAINT [FK_TipObor]
GO
```

```
/****** Object:  ForeignKey [FK__Oplata__RemontId__2180FB33]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Oplata]  WITH CHECK ADD FOREIGN KEY([RemontId])
REFERENCES [dbo].[Remont] ([RemontId])
GO
/****** Object:  ForeignKey [FK__Polzovate__Mater__7B5B524B]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Polzovatel]  WITH CHECK ADD FOREIGN
    KEY([MaterID])
REFERENCES [dbo].[Master] ([MasterId])
GO
/****** Object:  ForeignKey [FK__Polzovate__RoleI__25869641]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Polzovatel]  WITH CHECK ADD FOREIGN KEY([RoleID])
REFERENCES [dbo].[Roles] ([RoleID])
GO
/****** Object:  ForeignKey [FK__PriceTabl__Descr__6FE99F9F]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[PriceTable]  WITH CHECK ADD FOREIGN
    KEY([DescriptionID])
REFERENCES [dbo].[ProductDescription] ([DescriptionID])
GO
/****** Object:  ForeignKey [FK__PriceTabl__KindI__2B3F6F97]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[PriceTable]  WITH CHECK ADD FOREIGN KEY([KindID])
REFERENCES [dbo].[ProductsKinds] ([KindID])
GO
/****** Object:  ForeignKey [FK__ProductDe__Manuf__6EF57B66]
    Script Date: 05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[ProductDescription]  WITH CHECK ADD FOREIGN
    KEY([ManufacturerID])
REFERENCES [dbo].[Proizvoditel] ([ProizvoditelId])
GO
/****** Object:  ForeignKey [FK_remontclient]    Script Date:
    05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Remont]  WITH CHECK ADD  CONSTRAINT
    [FK_remontclient] FOREIGN KEY([ClientId])
REFERENCES [dbo].[Client] ([ClientId])
GO
ALTER TABLE [dbo].[Remont] CHECK CONSTRAINT [FK_remontclient]
GO
/****** Object:  ForeignKey [FK_remontmaster]    Script Date:
    05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Remont]  WITH CHECK ADD  CONSTRAINT
    [FK_remontmaster] FOREIGN KEY([MasterId])
REFERENCES [dbo].[Master] ([MasterId])
GO
ALTER TABLE [dbo].[Remont] CHECK CONSTRAINT [FK_remontmaster]
GO
/****** Object:  ForeignKey [fk_remontobor]    Script Date:
    05/10/2021 01:27:25 ******/
```

```
ALTER TABLE [dbo].[Remont]  WITH CHECK ADD  CONSTRAINT
    [fk_remontobor] FOREIGN KEY([OborudovanieId])
REFERENCES [dbo].[Oborudovanie] ([OborudovanieId])
GO
ALTER TABLE [dbo].[Remont] CHECK CONSTRAINT [fk_remontobor]
GO
/****** Object:  ForeignKey [FK_remontstate]    Script Date:
    05/10/2021 01:27:25 ******/
ALTER TABLE [dbo].[Remont]  WITH CHECK ADD  CONSTRAINT
    [FK_remontstate] FOREIGN KEY([Status_remontaId])
REFERENCES [dbo].[Status_remonta] ([Status_remontaId])
GO
ALTER TABLE [dbo].[Remont] CHECK CONSTRAINT [FK_remontstate]
GO
```