

Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка інформаційної системи реінжинірінгу бізнес-процесів виробничого підприємства»

Студента 4 курсу, 9 групи,
спеціальності
122 «Комп'ютерні науки»

Каракулов Артем
Дмитрович

підпис студента

Науковий керівник
доктор технічних наук,
професор

Краскевич Валерій
Свєгенович

підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент

Демідов Павло
Георгійович

підпис керівника

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую

Зав. кафедри _____ Пурський О.І.
(підпис, дата)

**Завдання
на випускню кваліфікаційну роботу (проект) студенту**

Каракулова Артема Дмитровича
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка інформаційної системи реінжинірингу бізнес-процесів виробничого підприємства»
Затверджена наказом ректора від «15» грудня 2020 р. № 3780
 2. Строк здачі студентом закінченої роботи 29 травня 2021 року
 3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка та впровадження інформаційної системи ведення супроводжувальних документів виконання на підприємстві вироблення послуг з виконання авторемонтних робіт
Об'єкт дослідження: організація ведення документів підприємства вироблення послуг на виконання ремонту автотранспорту.
Предмет дослідження: створення програмного забезпечення інформаційної системи ведення супроводжувальних документів виконання авторемонтних робіт на підприємстві вироблення послуг з виконання авторемонтних робіт
 4. Перелік графічного матеріалу _____
-

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Краскевич В. Є.	15.12.2020 р.	15.12.2020 р.
2	Краскевич В. Є.	15.12.2020 р.	15.12.2020 р.
3	Краскевич В. Є.	15.12.2020 р.	15.12.2020 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Теоретико-Практичний Підхід Для Огляду Ситуації З Проблематикою Документообігу Підприємства

1.1. Пошук наявних проблем в системі роботи СТО

1.2. Опис сучасних рішень о актуальних проблем в системі роботи СТО

1.3. Опис створення та функціонування баз даних, як засобу для вирішення проблемних питань в системі роботи СТО

1.4 Опис реінжинірингу та РБП

РОЗДІЛ 2. Створення Інформаційної системи документообігу в системі роботи

2.1. Постановка задачі для розробки ІС СТО

2.2. Проектування логічної моделі даних та нормалізація відношень ІС СТО

2.3. Проектування фізичної моделі даних ІС СТО

Розділ 3. Розробка Як Друга Частина Створення Інформаційної Системи Документообігу В Системі Роботи

3.1. Загальна архітектура ІС, яка буде реалізована у вирішенні задачі

3.2. Вхідні та вихідні форми ІС СТО

3.3. Алгоритми роботи ІС СТО

3.4. Характеристика задіяних засобів розробки програмного забезпечення та прикладних програм ІС СТО

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	Вибір теми випускної кваліфікаційної роботи	01.10.2020	
2	Розробка та затвердження завдання на випускну кваліфікаційну роботу	15.12.2020	
3	Вступ	03.02.2021	
4	РОЗДІЛ 1. Теоретико-Практичний Підхід Для Огляду Ситуації з Проблематикою Документообігу Підприємства	26.02.2021	
5	РОЗДІЛ 2. Створення Інформаційної системи документообігу в системі роботи	06.04.2021	
6	РОЗДІЛ 3. Розробка Як Друга Частина Створення Інформаційної Системи Документообігу В Системі Роботи	12.05.2021	
7	Висновки	15.05.2021	
8	Здача випускної кваліфікаційної роботи на кафедрі науковому керівнику	20.05.2021	
9	Попередній захист випускної кваліфікаційної роботи	02.06.2021	
11	Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи	07.06.2021	
12	Представлення готової зшитої випускної кваліфікаційної роботи на кафедрі	11.06.2021	
13	Публічний захист випускної кваліфікаційної роботи	За розкладом роботи ЕК	

8. Дата видачі завдання «15» грудня 2020 р.

9. Керівник випускної кваліфікаційної роботи (проекту)

Краскевич В. Є.

(прізвище, ініціали, підпис)

10. Гарант освітньої програми

Демідов П. Г.

(прізвище, ініціали, підпис)

Каракулов А. Д.

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There is no text or other markings on the paper.

(nidnuc, dama)

Випускна кваліфікаційна робота (проект) студента Каракулов А. Д.

(прізвище, ініціали)

Гарант освітньої програми

(підпис, прізвище, ініціали)

Демідов П. Г.

Завідувач кафедри

(підпис, прізвище, ініціали)

Пурський О.І.

«11» червня 2021 р

Анотація

У випускній кваліфікаційній роботі реалізовано ідею розробки інформаційної системи з реінжинірингу бізнес-процесів на підприємстві, з метою автоматизації та полегшення процесу ведення супроводжувальних документів виконання авторемонтних робіт на підприємстві з вироблення послуг. Відповідно, проаналізовано проблематику автоматизації станцій технічного обслуговування, як об'єктів вироблення послуг з виконання авторемонтних робіт, та запропоновано концепцію створення інформаційної системи ведення документації. Розроблено програму, в якій користувач з правами адміністратора може вести облік клієнтів, деталей, робіт та формувати звіти, а також створено автоматизовану інформаційну систему ведення та зберігання документів для СТО, що є основним прикладним результатом практичної частини роботи, як кульмінації реінжинірингу бізнес-процесів об'єкту дослідження.

Ключові слова: інформаційна система, авторемонт, автоматизація, документообіг, бази даних, логічна модель, фізична модель, інформаційні потоки

Anotation

The final qualifying work implemented the idea of developing an information system for reengineering business processes at the enterprise, in order to automate and facilitate the process of maintaining supporting documents for car repair work at the enterprise to develop services. Accordingly, the problem of automation of service stations as objects of development of services for car repair work is analyzed, and the concept of creating an information system for documentation is proposed. Developed a program in which the user with administrator rights can keep records of customers, parts, works and generate reports, as well as created an automated information system for maintaining and storing documents for service stations, which is the main application of the practical part of work as a culmination of business process reengineering object of study.

Keywords: information system, car repair, automation, document flow, databases, logical model, physical model, information flows.

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1. ТЕОРЕТИКО-ПРАКТИЧНИЙ ПІДХІД ДЛЯ ОГЛЯДУ СИТУАЦІЇ З ПРОБЛЕМАТИКОЮ ДОКУМЕНТООБІГУ ПІДПРИЄМСТВА.....	12
1.1 Пошук наявних проблем в системі роботи СТО.....	12
1.2 Опис сучасних рішень актуальних проблем в системі роботи СТО.....	15
1.3 Опис створення та функціонування баз даних, як засобу для вирішення проблемних питань в системі роботи СТО.....	18
1.4 Опис реінжинірингу та РБП.....	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ ЯК ПЕРША ЧАСТИНА СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДОКУМЕНТООБІГУ В СИСТЕМІ РОБОТИ СТО.....	26
2.1 Постановка задачі для розробки ІС СТО.....	26
2.2 Проектування логічної моделі даних та нормалізація відношень ІС СТО.....	29
2.3 Проектування фізичної моделі даних ІС.....	32
РОЗДІЛ 3. РОЗРОБКА ЯК ДРУГА ЧАСТИНА СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДОКУМЕНТООБІГУ В СИСТЕМІ РОБОТИ СТО.....	37
3.1 Загальна архітектура ІС, яка буде реалізована у вирішенні задачі.....	37
3.2 Вхідні та вихідні форми даної Інформаційної системи.....	39

3.3 Алгоритми роботи та інформаційні потоки даної ІС СТО.....	39
3.4 Характеристика задіяних засобів розробки програмного забезпечення та прикладних програм ІС СТО.....	44
ВИСНОВКИ.....	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	49
ДОДАТОК.....	52

ВСТУП

У наш час електронні обчислювальні машини широко застосовуються в багатьох галузях діяльності людини. Ні одна установа не може обійтись у своїй роботі без використання комп'ютерів, які з успіхом замінюють рутинну роботу, яка виконувалась раніше в ручну, підвищуючи ефективність роботи будь-якого підприємства [23].

Сфера послуг в даний час є однією з найважливіших галузей народного господарства, створеної для задоволення індивідуальних запитів і потреб населення країни в різних її видах. Сфера послуг як галузь економічної діяльності являє собою сукупність організацій, мета яких - надання різноманітних платних послуг за індивідуальними замовленнями населення. Таким чином, сфера послуг вирішує найважливіші соціально-економічні завдання і її значення в житті суспільства стрімко зростає. Одним із видів таких послуг є послуги автосервісу [14].

У сучасному світі, який постійно розвивається, зростає кількість приватних автомобілів. Щоб автомобіль якомога довше зберігав свої технічні якісні показники і гарний зовнішній вигляд необхідно стежити за його станом. А для того щоб бути впевненим у своєму автомобілі, потрібно періодично бувати в автосервісі - щоб раптова несправність зненацька не наздогнала власника автомобіля.

Підприємства, які працюють в сфері діагностики і ремонту автомобілів, знаходяться в умовах високої конкуренції. Щоб підвищити свою конкурентоспроможність більшість підприємств використовують інформаційні системи в процесі своєї роботи, так як обчислювальна техніка здатна в рази прискорити процес обробки інформації і отримання результату. Але в наші дні залишаються і такі підприємства, керівники яких не вирішуються впроваджувати інформаційні системи в силу їх високої ціни або своєї необізнаності в даній сфері, що і зумовило **актуальність** обраної теми дослідження, її мету та завдання.

Мета і завдання дослідження. Метою даного дослідження є розробка інформаційної системи автосервісу, яка дозволить підвищити ефективність управління за рахунок швидкого доступу до інформації про клієнтів, постачальників, співробітників та запасів на складі. Для досягнення поставленої мети необхідно було вирішити наступні **завдання**:

- провести аналітичне дослідження проблематики управління авторемонтними роботами на станціях технічного обслуговування
- дослідити сучасний стан програмних рішень у сфері управління автосервісами
- дослідити методи автоматизації управління автосервісами
- розробити характеристику програмних засобів і засобів створення та ведення баз даних
- розробити логічну та фізичну модель даних інформаційної системи СТО
- розробити метод забезпечення визначення цін та знижок на виконанні автосервісних робіт
- розробити інформаційну систему ведення документів з виконання авторемонтних робіт на СТО
- охарактеризувати задіяні засоби розробки програмного забезпечення

Об'єкт дослідження: діяльність автосервісу «Меридіан».

Предмет дослідження: моделі, методи та інформаційні технології в системі управління авторемонтними роботами на СТО.

Методи дослідження: Теоретичною основою дослідження є загальнонауковий аналітичний метод, а також системний підхід. Для практичного вирішення поставлених задач використовувалися такі методи:

- загальнонауковий аналітичний метод
- методи теорії БД для формування інформаційно-логічної моделі предметної області та БД;
- методи алгоритмічного програмування, для створення автоматизованої системи управління документацією виконання авторемонтних робіт на СТО «Меридіан».

Практичне значення. Запропонована програма реалізує інформаційну довідкову систему, призначену для організації обліку розподілу робіт по відділах і працівникам, а також веде облік вартості виконаних робіт. Додаток дозволяє проводити введення, редагування і перегляд вмісту баз даних, а також відповідати на запити користувача і складати різноманітні звіти.

Отримані результати, можуть бути використані в автосервісах. Дана програма легка в користуванні, дозволяє зберігати велику кількість відомостей в одній базі даних, економить робочий час за рахунок автоматизації деяких процесів таких як, облік вартості робіт, а також облік розподілу робіт по відділах. Тому пропонована програма повинна істотно спростити роботу автосервісів.

РОЗДІЛ 1. ТЕОРЕТИКО-ПРАКТИЧНИЙ ПІДХІД ДЛЯ ОГЛЯДУ СИТУАЦІЇ З ПРОБЛЕМАТИКОЮ ДОКУМЕНТООБІГУ ПІДПРИЄМСТВА

1.1 Пошук наявних проблем в системі роботи СТО

Процес надання автосервісних послуг складається з трьох взаємопов'язаних елементів:

- прийом замовлень на послуги від населення;
- виконання замовлень;
- реалізація послуг.

Прийом замовлень від населення - це початкова стадія процесу надання послуги. Він включає визначення складу послуги. При цьому на даній стадії виконується ряд операцій технологічного характеру, які в значній мірі впливають на весь подальший процес виробництва (наприклад: виявлення дефектів автотранспорту, що підлягає ремонту) [25].

Наступна стадія надання послуг - безпосереднє *виробництво*, організація якого в значній мірі визначається характером виконуваних послуг.

Заключна стадія процесу надання автосервісних послуг - *реалізація замовлень*, тобто доведення послуг до споживача. Однією з особливостей, властивих підприємствам сфери обслуговування, є та обставина, що вони мають безпосередній контакт із споживачем при наданні послуг, тобто в процесі своєї діяльності здійснюють не тільки виробничі, але і торгові функції [17].

Взаємовідносини підприємств автосервісу, які надають платні послуги, і замовників в процесі їх обслуговування, регулюються правилами надання послуг, які визначають порядок прийому і оформлення замовлень, виконання замовлень, розрахунків із замовниками, а також майнову відповідальність як підприємства, так і замовника.

Замовлення на платні послуги оформляються як спеціальними первинними документами, так і шляхом видачі жетонів, талонів, касових

чеків, розписок і т.д. На підприємствах автосервісу використовуються такі форми документів суворої звітності [21]:

- **БО-1** - квитанція - оформляється на всі види ремонту, що вимагає витрат матеріалів;
- **БО-9** - касова відомість прийому виручки - оформляється на терміновий і дрібний ремонт, що виконується в присутності замовника.

В процесі експлуатації автомобіля можуть виявитися конструктивні і виробничі недоліки (дефекти), що призводять до зміни технічного стану деталей, вузлів, агрегатів, внаслідок чого відбувається їх природний знос.

Розрізняють механічний, абразивний, корозійний і втомний знос [26].

Механічний знос відбувається внаслідок зминання або викришування частинок з поверхні деталей, що викликає зміну маси і розміру деталі.

Абразивний знос - це результат ріжучого впливу більш твердих частинок деталей, часток внесених повітрям або тих, які потрапили разом зі змазкою.

Корозійний знос є наслідком впливу агресивного середовища (кислот, лугів, кисню) на поверхню деталей.

Втомний знос викликається впливом багаторазових змінних навантажень.

Більшість деталей автомобільного транспорту піддаються одночасному зносу декількох видів.

Відхилення технічного стану автомобіля (причепа) і його агрегатів від встановлених норм називається несправністю. Порушення працездатності автомобіля, що призвело до припинення транспортного процесу, називається відмовою.

Для забезпечення безперебійної роботи автомобільного транспорту необхідно не тільки повсякденне спостереження за його станом в процесі експлуатації (мастило, огляд і т.п.), але і періодичне проведення ремонту. Нерівномірність зношування окремих вузлів і деталей, що входять до складу того чи іншого об'єкта автомобільного транспорту, зажадала розробки спеціальної системи планово-попереджувального ремонту [24]. Система

технічного обслуговування автомобільного транспорту є планово-попереджувального, і всі роботи, передбачені для кожного обслуговування, є обов'язковими до виконання в повному обсязі. Ця система сприяє постійному підтриманню автомобілів і причепів в працездатному вигляді, зменшенню інтенсивності зносу деталей, попередження відмов і несправностей, зниження витрати палива і мастильних матеріалів, підвищення надійності і безпеки експлуатації і збільшення пробігу автомобілів до ремонту.

Кріпильні, мастильні, заправні, регулювальні, електротехнічні та збирально-мийні роботи, що проводяться в передбачені технічним обслуговуванням терміни, дозволяють забезпечити нормальні умови роботи всіх систем і механізмів автомобіля.

Технічне обслуговування є профілактичним заходом, проведеним примусово в плановому порядку через певні проміжки. Періодичність технічного обслуговування встановлюється за фактично виконаний пробіг в кілометрах з урахуванням умов експлуатації. Для кожної категорії умов експлуатації найбільша періодичність технічного обслуговування прийнята для легкових автомобілів, потім - автобусів і вантажних автомобілів [28].

Ремонт рухомого складу автомобільного транспорту призначений для регламентованого відновлення та підтримання працездатності автомобілів і причепів, усунення несправностей, що виникли в роботі або виявлених при технічному обслуговуванні. Ремонтні роботи виконують як за потребою (після появи відповідного відмови або несправності), так і за планом через певний пробіг або час роботи рухомого складу.

Існує два види ремонту: капітальний і поточний [24]. Останній, в свою чергу, поділяється на середнє, мале і поточне (міжремонтне) обслуговування. Капітальний ремонт, як правило, виконують на спеціалізованих ремонтних підприємствах, поточний - на автотранспортних підприємствах або на станціях технічного обслуговування.

Капітальний ремонт включає контрольно-діагностичні, складальні, регулювальні, слюсарні, механічні, шпалерні, електротехнічні, шиноремонтні,

маллярні та інші роботи. Ремонтні роботи можуть виконуватися за певними агрегатами вузлів, а також по рухомому складу в цілому. При капітальному ремонті агрегат повністю розбирають, виявляють дефекти, відновлюють або замінюють окремі деталі, потім збирають, регулюють і випробовують. Якщо капітальному ремонту підлягає весь автомобіль, то його теж повністю розбирають, всі деталі відновлюють і замінюють, збирають, а вузли та агрегати регулюють [27].

Поточним ремонтом вважається такий, при якому агрегат розбирається лише частково, а відновлюються і замінюються тільки ті частини, термін служби яких дорівнює міжремонтному періоду. Поточний ремонт зазвичай здійснюється без зняття агрегату з фундаменту. При цьому середній поточний ремонт відрізняється від малого лише обсягом ремонтних робіт. Поточне (міжремонтне) обслуговування зводиться до повсякденного спостереження за станом обладнання і усунення дрібних несправностей.

Облік ремонту транспортних засобів слід вести по його видам: капітальний і поточний з поділом на середнє, мале і міжремонтне обслуговування.

Розроблена інформаційна система повинна давати можливість ведення бази даних по відділам, видам наданих послуг, співробітникам, а також забезпечувати діалогові засоби введення, редагування, пошуку інформації та виведення звітів.

1.2 Опис сучасних рішень актуальних проблем в системі роботи СТО

В даний час існують автоматизовані системи для підприємств, такі як: 1С, SLS - автосервіс, ZETA СЕРВІС, Універсальна Система Обліку 2.0 і багато інших [3].

До найбільш поширених автоматизованих систем можна віднести наступні:

- "1C";
- "SLS - автосервіс";
- "ZETA СЕРВІС".

Програма "1C" призначена для вирішення широкого спектра завдань автоматизації обліку та управління, що стоять перед сучасними підприємствами, які динамічно розвиваються [31].

До **переваг** даної програми можна віднести наступне:

- за допомогою "1C" можна вести всі існуючі види обліку робіт і матеріалів;
- на сьогоднішній день "1C" є однією з найбільш універсальних програм, яка може використовуватися в самих різних організаціях;
- програма "1C" має високу продуктивність, що дає можливість вирішувати з її допомогою найскладніші завдання;

Також "1C" володіє і рядом **недоліків**, до яких можна віднести наступне:

- в переважній більшості випадків, щоб "1C" вирішувала всі поставлені перед нею завдання, програму доводиться допрацьовувати; Кожне підприємство унікальне, тому для ефективної його роботи, як правило, потрібні індивідуальні рішення по автоматизації бізнес-процесів;
- при переході на "1C" з іншої програми можуть виникнути серйозні труднощі при перенесенні інформації з однієї бази даних в іншу;
- в "1C" утруднений пошук помилок, зроблених під час обробки документів;
- програма "1C" досить складна в освоєнні і вимагає спеціального навчання користувачів.

Програма *SLS-автосервіс* - це комплексне рішення для обліку в компаніях, що спеціалізуються на виконанні робіт по сервісному обслуговуванню різних транспортних засобів [22].

Програма має потужні інструменти, які дають можливість моделювати весь ланцюжок бізнес-процесів автосервісу - починаючи від надходження автомобіля на обслуговування, виконання ремонтних робіт і закінчуючи

збором інформації для повного аналізу діяльності автосервісу і подальшого використання результатів.

SLS-Автосервіс **дозволяє**:

- працювати з замовлення-нарядами;
- роздруковувати необхідні документи;
- вести облік транспортних засобів, що знаходяться в обслуговуванні;
- враховувати запчастини та витратні матеріали;
- враховувати виконані роботи;
- проводити взаєморозрахунки з власниками автотранспорту;
- здійснювати нарахування співробітникам компанії за виконані роботи;
- вести фінансовий облік діяльності автосервісу.

У програму вбудовано докладний опис всіх функцій і можливостей. SLS-Автосервіс розрахований як на одного користувача так і в мережевому варіанті.

Але і у цій програмі є **недоліки**. Багато з них дуже істотні такі як:

- Ціна на одного користувача варіанта близько 15000 грн, на середній СТО автомобілів необхідно як мінімум 3-4 робочих місця.
- Використання даного програмного продукту вимагає використання додаткових платних програм.
- Також програма вимагає від оператора певної комп'ютерної грамотності.
- Чи не занадто зрозумілий і доброзичливий інтерфейс обтяжений зайвими параметрами.

Система *Zeta Servis* - це програма обліку для автосервісу, розроблена на платформі 1С: Підприємство [22]. Вона дозволяє автоматизувати ведення оперативного, управлінського, складського і фінансового обліку. Одною з важливих **переваг** продукту є зручний інтерфейс, який налаштовується для кожного співробітника окремо.

Інтерфейс програми досить продуманий. Великі кнопки, які видно тільки тоді, коли потрібно, але це тільки після спеціалізованого налаштування кожного робочого місця. Що також вимагає або кваліфікованого фахівця

запрошеного з боку підприємства обслуговуючої програми 1С, або мати штатного співробітника, навченого підприємством, що обслуговує програми сімейства 1С, що також коштує не малих грошей.

Решта представників програмного забезпечення в даному сегменті створені приблизно на тих же умовах що і попередні, це або програми сімейства 1С, або програми створені іншими розробниками з тими чи іншими вадами, які виправляти крім програмістів саме цієї організації практично нікому.

Таким чином, рішення, які існують на даний момент, є універсальними. Однак вони платні, складні в освоєнні і досить громіздкі для даного підприємства. Також, в деяких багато зайвих функцій, в інших - нема функцій, необхідних для підприємства автосервісу. Тому було прийнято **рішення** про необхідність власної розробки, яка буде відповідати всім поставленим вимогам і мати інтуїтивно зрозумілий інтерфейс. І не зажадає систематичної покупки ліцензій, що є важливим фактом. Програмний продукт, створений в рамках дипломного проекту, перш за все орієнтований на користувачів, які не мають спеціалізованих комп'ютерних навичок, а саме для працівників СТО.

1.3 Опис створення та функціонування баз даних, як засобу для вирішення проблемних питань в системі роботи СТО

Під час аналізу вимог до системи основна увага приділялася з'ясуванню того, що повинно бути зроблено, незалежно від того, як це зробити. На етапі розробки системи вирішується питання, як реалізувати рішення, прийняті на етапі аналізу [32].

Спочатку розробляється загальна структура (архітектура) системи. Архітектура системи визначає її розбиття на модулі, задає контекст, в рамках якого приймаються проектні рішення на наступних етапах розробки [6].

Для розробки архітектури необхідно вибрати систему управління базами даних (СКБД). За способом доступу до баз даних (БД) розрізняють наступні СУБД [7]:

- клієнт-серверні;
- файл-серверні;
- вбудовані.

Клієнт-серверна система характеризується наявністю двох взаємодіючих самостійних процесів - клієнта і сервера, які можуть виконуватися на різних комп'ютерах, обмінюючись даними по мережі. За такою схемою можуть бути побудовані системи обробки даних на основі СУБД, поштові та інші системи.

У файл-серверної системи дані зберігаються на файловому сервері (наприклад, Novell NetWare або Windows NT Server), а їх обробка здійснюється на робочих станціях, на яких, як правило, функціонує одна з, так званих, "настільних СУБД" - Access, FoxPro, Paradox і тому подібні.

Додаток на робочій станції "відповідає за все" - за формування призначеного для користувача інтерфейсу, логічну обробку даних і за безпосереднє маніпулювання даними. Файловий сервер надає послуги тільки найнижчого рівня - відкриття, закриття та модифікацію **файлів**, не бази даних. База даних існує тільки в "мозку" робочої станції.

Безпосереднім маніпулюванням даними займається кілька незалежних і неузгоджених між собою процесів. Крім того, для здійснення будь-якої обробки всі дані необхідно передати через мережу з сервера на робочу станцію.

У клієнт-серверної системи функціонують (як мінімум) два додатки - клієнт і сервер, що ділять між собою ті функції, які в файл-серверній архітектурі виконує додаток на робочій станції. Зберіганням і безпосереднім маніпулюванням даними займається сервер баз даних, в якості якого може виступати Microsoft SQL Server, Oracle, Sybase, Firebird, Interbase, IBM DB2, Informix, PostgreSQL, MySQL, Caché, Лінтера [9].

Вибір системи управління баз даних (СУБД) є важким завданням і є одним з важливих етапів при розробці додатків баз даних. Обраний програмний продукт повинен задовольняти як поточні, так і майбутні потреби підприємства, при цьому слід враховувати фінансові витрати. Інформаційна система повинна відповідати найбільш загальним технічним вимогам [2].

Таблиця 1.1 - Характеристики інформаційної системи

Характеристики	Коментарі
Єдина база даних забезпечує розраховану на багато користувачів роботу.	Рекомендується використання централізованої бази даних на основі повноцінних промислових СУБД (MS SQL Server, Oracle, Informix, DB2).
Відсутність обмежень за кількістю об'єктів (максимальна кількість вимірювань, записів, звітів, число одночасно працюючих користувачів і т.д.).	Будь-які обмеження щодо кількості об'єктів, якими оперує система, говорять про недосконалість технічного рішення такого програмного продукту. Навіть якщо ці обмеження складають десятки тисяч одиниць і сьогодні здаються прийнятними, це означає, що в майбутньому виникне нездоланий бар'єр при розвитку бюджетної моделі.
Інтеграція з суміжними автоматизованими системами.	Можливості системи повинні дозволяти здійснювати повноцінний імпорт / експорт, при необхідності попередню обробку даних з різних облікових систем; бажана підтримка двостороннього зв'язку з наявними обліковими системами. Використання стандартних СУБД також полегшує інтеграцію.
Можливості доопрацювання системи на вимогу замовника.	Необхідно з'ясувати чи має постачальник програмного забезпечення можливість його доопрацювання на рівні програмного коду.

Розглянемо більш докладно найбільш поширені СУБД.

1) СУБД *Oracle* одна з найбільш потужних сучасних СУБД, призначених для реалізації баз даних рівня корпорації, що висуває серйозні вимоги до сервера. *Oracle* може працювати в більшості операційних систем: Windows-NT, 2000, Linux, UNIX, AIX, Novell Netware [38].

Використання *Oracle* в якості СУБД дає можливість вибору мови програмування. Традиційно для цього використовується мова PL / SQL, але можна використовувати і набагато потужнішу мову програмування Java.

Oracle має в своєму розпорядженні потужні і зручні засоби адміністрування не тільки одного сервера, але і групи серверів, розташованих в різних частинах планети.

Основними перевагами *Oracle* можна вважати підтримку баз даних дуже великого обсягу (до 64 Гбайт), потужні засоби розробки і адміністрування, підтримку багатопроцесорності і двох мовних середовищ, а також інтеграцію з Web. Разом з цим програма має серйозні апаратні вимоги та високу ціну.

2) СУБД *Borland Interbase* містить все, що потрібно від СУБД, призначеної для потреб малого та середнього бізнесу [10]. До того ж починаючи з версії 6.0 програма стала безкоштовною, що істотно впливає на її використання різними підприємствами. Програма не вимоглива до апаратної частини. *Borland Interbase* підтримується платформами Windows і Linux, а також UNIX, NetBSD, FreeBSD.

Популярні мови програмування від *Borland*, як Delphi, Kylix і C++ Builder, надаються з компонентами, що дозволяють працювати з даною СУБД. Саме це дозволяє досягти дуже високої швидкодії програми.

3) СУБД *MySQL* набула широкого поширення в якості засобу роботи з базами даних в Інтернеті. Програма абсолютно невимоглива до ресурсів сервера, на якому працює, дуже швидка і до того ж абсолютно безкоштовна: вихідні коди для різних платформ доступні на сайті в інтернеті. Спочатку програма була орієнтована на операційну систему Linux, але зараз вже існують версії програми для операційних систем Windows, UNIX, NetBSD, FreeBSD,

AIX. Останнім часом програма завойовує популярність у користувачів Macintosh з використанням операційної системи Mac OSX [4].

5) СУБД *MS Access* використовується для вирішення локальних офісних завдань з обмеженим обсягом даних і формування звітів по результатам роботи, при цьому звіти можуть бути представлені в стандартному для офісних додатків вигляді.

MS Access одночасно є і середовищем розробки на двох мовах програмування (*Visual Basic* і сильно урізаний діалект *SQL*), і *CASE*-засобом, а також потужним і наочним засобом створення звітів за результатами роботи [20].

Програмне забезпечення дозволяє створювати програми, що складаються з одного файлу, що містить як текст програми, так і реляційну базу даних складної структури. *Access* легко інтегрується з іншими рішеннями від *Microsoft*. Це дозволяє використовувати її як клієнтську частину інформаційного комплексу в зв'язці з *MS SQL Server*, яка виступає в якості серверної частини.

6) *MS SQL Server*. Протягом багатьох років продукти *MS SQL Server* характеризуються надійністю, безпекою, високою продуктивністю і зручністю в роботі. Сучасна СУБД *MS SQL Server* це найпотужніший програмний комплекс, що дозволяє створювати додатки будь-якої складності. Ядром цього комплексу є база даних, що зберігає інформацію, кількість якої за рахунок коштів, що надаються, масштабуються практично безмежно. З високою ефективністю працювати з цією інформацією одночасно може будь-яка кількість користувачів, не проявляючи тенденції до зниження продуктивності системи при різкому збільшенні їх числа.

Механізми масштабування в СУБД *MS SQL Server* останніх версій дозволяють збільшувати потужність і швидкість роботи сервера *MS SQL Server* і своїх додатків.

Ще однією складовою успіху СУБД *MS SQL Server* є те, що вона поставляється практично для всіх існуючих на сьогодні операційних систем.

Таким чином, компаніям, які розпочинають роботу з продуктами MS SQL Server не доводиться змінювати вже складене мережеве оточення. Існує лише невелика кількість відмінностей при роботі з СУБД, обумовлених особливостями тієї чи іншої операційної системи. В цілому ж це завжди та ж сама безпечна, надійна і зручна СУБД MS SQL Server.

Потрібно відзначити мудру міграційну політику MS SQL Server. Розуміючи, що перехід з більш старої версії СУБД на нову, досить трудомістка процедура. Пов'язана вона з тестуванням роботи існуючих додатків в новому оточенні. MS SQL Server, при випуску нових продуктів приділяє особливу увагу сумісності знизу-вгору, роблячи цей перехід практично «безболісним». Останні версії СУБД MS SQL Server значно простіші в установці і початковому налаштуванні. Також зросли можливості по спеціалізованому налаштуванні роботи СУБД під конкретну задачу [9].

На вибір серверу СУБД MS SQL Server вплинуло декілька факторів.

По-перше, технічні характеристики, які повністю задовольняли поставлену вимогу, а по-друге, доступність даної СУБД. Серед всіх переваг MS SQL Server слід виділити наступні:

- простота і зручність адміністрування;
- невибагливість і мінімальні системні вимоги;
- ефективність і швидкодія;
- високий ступінь інтеграції в середовище розробки;
- висока надійність;
- можливість розширення бази даних;
- наявність універсальних засобів захисту інформації;
- орієнтованість на Інтернет-технології;
- порівняно низька ціна.

Виходячи з перерахованих вище переваг, MS SQL Server було обрано як оптимальне рішення поставленого завдання, для реалізації СУБД [1].

1.4 Опис реінжинірингу та РБП

Реінжиніринг є фундаментальне переосмислення та радикальне перепроєктування бізнес-процесів з метою досягнення істотних поліпшень у вартісних та кількісних результатах господарської діяльності організації.

Майкл Хаммер оголосив революційні на той час принципи: “Реконструйте роботи, не автоматизовуючи, а спрощуючи або знищуючи”, “Використовуйте комп’ютери не тільки для автоматизації, а й для реконструкції існуючих бізнес-процесів”[41].

Реінжиніринг бізнес-процесів (РБП) передбачає створення цілком нових і більш ефективних бізнес процесів для досягнення вагомих покращень. В сьогоdnішніх ринкових умовах, де постійні лише зміни, не завжди можливо утримувати свої позиції на ринку, процеси можуть перетворитися на “непридатні”, і не буде сенсу налагоджувати або поліпшувати їх; краще створити щось нове із самого початку, відносно поточної ситуації. Отже, суть реінжинірингу побудована на системі докорінних перетворень в організації. Спочатку моделюємо організацію, а потім змінюємо цю модель під рішення конкретних поточних та перспективних завдань, частіше за все шляхом рішучого відрубання нераціональних ланок та функцій. Концепція покращення бізнес-процесів ґрунтується на чотирьох підходах:

1. Методика швидкого аналізу рішення.
2. Бенчмаркінг процесу.
3. Перепроєктування процесу.
4. Реінжиніринг процесу.

Методика швидкого аналізу рішення – проривний підхід, який концентрує увагу групи на певному процесі в ході одно- дводенної наради для визначення способів, якими група може покращити цей процес впродовж наступних 90 днів.

Бенчмаркінг процесу – систематичний метод визначення, усвідомлення та творчого розвитку товарів, проектів, послуг, обладнання, процесів та процедур більш високої якості для покращення поточної діяльності організації через

вивчення того, як різні організації виконують однакові чи схожі операції. Зазвичай знижує витрати, тривалість циклу та кількість помилок на 20-50%. Перепроєктування процесу – концентрує зусилля на вдосконаленні існуючого процесу, як правило використовуються для тих процесів, які вже є досить успішні.

Реінжиніринг бізнес-процесів – найбільш радикальний із усіх чотирьох підходів до покращення бізнес-процесів. Реінжиніринг БП знижує витрати та тривалість циклу на 60-90% і рівень помилок на 40-70%. Часто цей процес стимулює команду, що стає справжнім проривом[29].

РОЗДІЛ 2. СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДОКУМЕНТООБІГУ В СИСТЕМІ РОБОТИ

2.1 Постановка задачі для розробки ІС СТО

Для комплексної автоматизації всієї роботи автосервісу необхідно розібрати основні режими його роботи. Але, загалом, завдання по автоматизації роботи автосервісу повинна дозволяти вирішувати такі питання:

- використовувати весь спектр необхідних первинних документів для оформлення послуг по ремонту (заявка на ремонт, наряд-замовлення, талон на обслуговування);
- вести список автомобілів для будь-якого клієнта фірми, для кожного автомобіля зберігати набір необхідних реквізитів: державний номер, модель, технічні характеристики і т. п.
- отримувати історію обслуговування для кожного автомобіля;
- забезпечувати узгоджену роботу декількох користувачів при оформленні ремонтних документів;
- контролювати незавершений ремонт;
- фіксувати зриви робіт через скарг від клієнтів;
- вести облік запчастин;
- оформляти запчастини, надані для ремонту клієнтом;
- оформляти поставки запчастин;
- враховувати недопоставки, а також проводити інвентаризацію;
- оформляти замовлення від клієнтів на придбання запчастин, а також замовлення постачальникам на поставку товарів [16].

Сучасне життя досить відрізняється від того, яким жили люди ще років 50 тому. Сьогодні автомобіль - наш супутник життя і уявити себе без машини - практично неможливо. День у день ми стикаємося з автомобілями, і, як наслідок, з їх ремонтом.

Ремонту автомобілів піддається практично кожен авто-власник. В умовах сучасного ринку займатися ремонтом автомобіля самостійно стало незручно і невигідно. У авто-власника просто не залишається часу, щоб самостійно розібратися в своєму автомобілі. А з розвитком ринку іноземного автомобіля (ускладненням конструкції самого автомобіля) це стало і неможливо. Сучасний автомобіль вимагає турбот досвідченого, добре підготовленого механіка. З появою авто-сервісів на ринку послуг ремонт автомобілів став здійснюватись набагато простіше - цим стали займатися обізнані й досвідчені фахівці, які мають можливість без праці впоратися з будь-якою з проблемою.

Зараз ремонт автомобілів здійснюється з використанням сучасного діагностичного і технологічного обладнання, тому що воно є засобом для отримання якісного результату.

Віддаючи свій автомобіль в автосервіс, будь-який з авто-власників повинен бути впевнений, що всі роботи по ремонту автомобіля будуть проведені на високому професійному рівні і з гарантією. Більшість сучасних автомобілів, навіть для профілактичних заходів вимагають специфічного обслуговування. Сучасні авто-сервіси повинні справлятися з будь-яким із поставлених завдань. Авто-власнику досить пригнати свій автомобіль автосервіс, і в короткі терміни всі проблеми з ним будуть вирішені.

Сучасний автосервіс має штат висококваліфікованих фахівців, які пройшли відповідне навчання, мають достатній досвід роботи, і мають бажання надійно і якісно виконувати свою роботу. Сьогодні автосервіс - це весь спектр послуг по обслуговуванню автомобіля, який може включати в себе не тільки ремонтні, кузовні та слюсарні роботи, фарбування і полірування автомобіля, але і буксирування, і транспортування вашого авто, і навіть надання юридичних послуг [15]. Наявність інформаційної системи, яка виконує більшість рутинної роботи, дозволить значно знизити витрати на виконання даних робіт. Це дозволить знизити собівартість послуг, що надаються.

Також автоматизована, злагоджена робота всієї команди співробітників в автосервісі, яка цінує не тільки свій час, але і, в першу чергу, час клієнта, буде гарантовано виробляти всі роботи, піклуватися про свого клієнта і про автомобіль цього клієнта. Завдяки наявності автоматизованої системи в автосервісі співробітники зможуть нагадувати клієнтам про проходження планового огляду технічного стану його автомобіля (ця послуга може надаватися клієнтам з їх згоди).

Користувачами даного програмного продукту будуть менеджери, які укладають договори з клієнтами, робітники, які виконують весь спектр робіт, зазначених в документі, отриманому після закінчення роботи з клієнтом по виявленню несправності, касир, який здійснює розрахунок з клієнтом також за отриманими в результаті розрахунку документами.

Розроблена програма призначена також і для використання її бухгалтером, директором і адміністратором на автосервісі, що займається великою кількістю всіляких операцій, пов'язаних з оглядом автомобіля, веденням та обліком ходу виконання робіт, контролем робітників і рухом грошових мас [19].

Таким чином, в даному сегменті ринку існує потреба в інформаційній системі, яка забезпечувала б автоматизовану обробку і маніпулювання даними, облік взаєморозрахунків та в будь-який момент надавала б оперативну інформацію. Дані в цій системі повинні бути згруповані по клієнтам, за типами договорів, за списками автомобілів і т. д. - з метою надання повної інформації для аналізу і використання в розвитку авто-сервісного бізнесу.

При розробці такої системи необхідно приділити увагу наступним вимогам [33]:

- Простота використання. Системою повинен вміти користуватися оператор без спеціальної комп'ютерної підготовки, фахівець у своїй предметній області.

- Ефективність. Система повинна бути оснащена потужною і добре продуманою системою навігації і пошуку інформації.
- Надійність. В якості платформи потрібно використовувати інструментальні програмні засоби професійного рівня.
- Масштабованість. Розроблена система повинна легко розширюватися і забезпечувати гнучкі можливості для настройки.
- Виконання програми у відповідності зі стандартами побудови сучасних інформаційних систем. Вона повинна мати інтуїтивно-зрозумілий дружній інтерфейс.
- Доступність системи за вартістю для потенційного покупця. По можливості, ціна проекту повинна бути нижче, ніж у існуючих і потенційних аналогів.
- Відповідність системи сучасним вимогам по експлуатаційним параметрам. Система повинна бути простою в обслуговуванні та освоєнні користувачами.
- Високий рівень безпеки і захищеності інформації.
- Можливості розвитку в майбутньому.

2.2 Проектування логічної моделі даних

Метою бази даних, яка розробляється для користувачів, є облік клієнтів, співробітників, постачальників, та робіт, виконаних співробітниками. Користувачами бази даних «автосервіс» є працівники автосервісу.

Реляційна БД - основний тип сучасних баз даних. Складається з таблиць, між якими можуть існувати зв'язки з ключових значень [38].

Таблиця бази даних (table) - регулярна структура, яка складається з однотипних рядків (записів, records), розбитих на стовпці (поля, fields).

В теорії реляційних баз даних синонім таблиці - відношення (relation), в якому рядок називається кортежем, а стовпець називається атрибутом.

У концептуальній моделі реляційної БД аналогом таблиці є сутність (entity), з певним набором властивостей - атрибутів, здатних приймати певні значення (набір допустимих значень - домен).

Ключовий елемент таблиці (ключ, regular key) - таке її поле (простий ключ) або рядковий вираз, утворений із значень декількох полів (складових ключа), за яким можна визначити значення інших полів для однієї або декількох записів таблиці. На практиці для використання ключів створюються індекси - службова інформація, що містить впорядковані відомості про ключові значення. У реляційній теорії і концептуальній моделі поняття "ключ" застосовується для атрибутів відносин або сутності.

Первинний ключ (primary key) - головний ключовий елемент, однозначно ідентифікує рядок в таблиці. Можуть також існувати альтернативний (candidate key) і унікальний (unique key) ключі, слугує також для ідентифікації рядків в таблиці.

У реляційній теорії первинний ключ - мінімальний набір атрибутів, однозначно ідентифікує кортеж у відношенні.

У концептуальній моделі первинний ключ - мінімальний набір атрибутів сутності, однозначно ідентифікує екземпляр сутності [39].

Зв'язок (relation) - функціональна залежність між об'єктами. У реляційних базах даних між таблицями встановлюються зв'язки по ключах, один з яких в головній (parent, батьківської) таблиці - первинний, другий - зовнішній ключ - у зовнішній (child, дочірньої) таблиці, як правило, первинним не є і утворює зв'язок "один до багатьох" (1: N). У разі первинного зовнішнього ключа зв'язок між таблицями має тип "один до одного" (1: 1). Інформація про зв'язки зберігається в базі даних.

Зовнішній ключ (foreign key) - така підмножина атрибутів дочірнього відношення, що для будь-якого його непорожнього значення обов'язково знайдеться рівне значення первинного ключа головного відношення.

Реляційна модель:

- Клієнт (код клієнта, найменування, контакти)

- Автомобіль (код авто, марка, модель, реєстраційний номер)
- Виконавець замовлення (код замовлення, код співробітника, відсоток участі)
- Зовнішні ключі: код співробітника, посилається на таблицю «співробітник».
- Співробітник (код співробітника, ПІБ, код посади, контакти)
- Зовнішні ключі: код посади, посилається на таблицю «посаду».
- Робота (код роботи, найменування, код одиниці виміру, контакти)
- Зовнішні ключі: код одиниці виміру, посилається на таблицю «одиниці виміру».
- Замовлення (код замовлення, дата, код клієнта, код авто, причина, стан)
- Зовнішні ключі: код клієнта, код авто, посилаються на таблиці «клієнт», «автомобіль».
- Посада (код посади, найменування, оклад)
- Повноваження (код посади, об'єкт доступу, читання, зміна, видалення)
- Одиниці виміру (код одиниці виміру, найменування)
- Запаси (код запасів, номер за каталогом, найменування, виробник, код одиниці виміру, ціна відпуску)
- Зовнішні ключі: код одиниці виміру, посилається на таблицю «одиниці виміру»
- Постачальники (код постачальника, найменування, реквізити, контакти)
- Місце зберігання (код місця, найменування)

Модель даних, зроблена в MS SQL Server, представлена на рис. 2.1

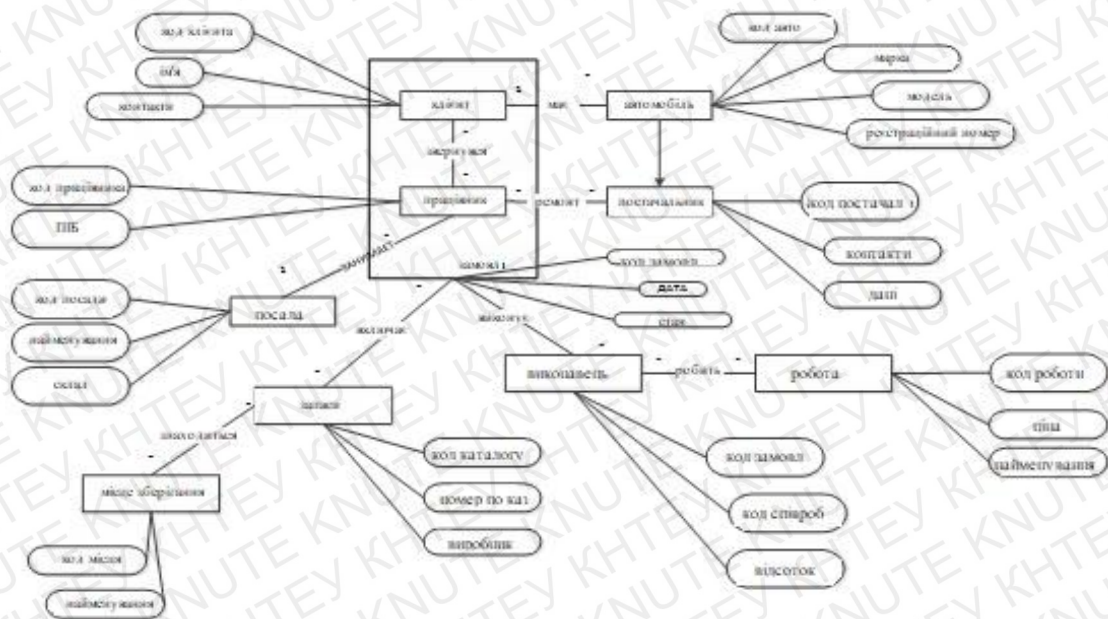


Рис. 2.1. Модель БД

2.3 Проектування фізичної моделі даних ІС

На основі концептуальної і реляційної моделей були спроектовані таблиці БД в MS SQL.

На рис. 2.2 показана таблиця з даними про клієнтів. Для того, щоб оформити замовлення, адміністратору для початку необхідно буде занести контактні дані клієнта в цей довідник, а вже в формі замовлення вибрати зі списку клієнта. Такий спосіб дозволить уникнути втрати контактних даних клієнтів і помилок при введенні прізвища, імені та по батькові.

Посади	Повноваження	Працівники	Одиниці виміру	Місця зберігання	Запаси	Постачальник	Роботи	Клієнти
		КодКлієнта	Найменування	Контакти	Примітка			
		1	Сидоров Сергій Петрович	(063) 555 55 55				
		2	ТОВ Спецперевезення	(095) 125 55 89	знижка 10% на всі види робіт			

Рис. 2.2. Таблиця «клієнти»

На рис. 2.3 представлена таблиця з даними про співробітників, тут зберігаються логіни і паролі для входу в інформаційну систему, паролі зберігаються в зашифрованому вигляді, алгоритм хешування MD5.

Посади	Повноваження	Працівники	Одиниці вимірювання	Місця зберігання	Запаси	Постачальник	Роботи	Клієнти	Авто
	Код	ПІБ	Посада	Контакти		Примітка		Ім'я Користувача	Пароль
▶	3	Белешко Андрій Вікторович	Власник					owner	72 12 2c e9 6...
	4	Овдієнко Володимир Павлович	Адміністратор					admin	21 23 2f 29 7a...

Рис. 2.3. Таблиця «працівники»

На рис. 2.4 представлена таблиця з марками автомобілів, дані марки будуть обиратися адміністратором зі списку при оформленні замовлення.

РегНомер	КодАвто	Марка	Модель	VIN	РегНомер	Примітка
	1	ВАЗ	2110	ХТА211010Y...	а323ва54	
	2	ВАЗ	2109	ХТА210930Y...	у565кк54	
	3	ГАЗ	53	52-04-1601200	а555вв54	

Рис. 2.4. Таблиця «автомобілі»

На рис. 2.5 показана таблиця посад в автосервісі і оклад для кожної посади.

Посади	Повноваження	Працівники	Одиниці виміру	Місця зберігання	Запаси	Постачальник	Роботи	Кл
	КодПосади	Найменування	Оклад	Примітка				
►	12	Власник	0					
	15	Адміністратор	30000					
	16	Майстер-приймщик	20000					
	17	Комірник	20000					
	18	Автослюсар	10000		+ % від робіт			
	19	Автомаляр	10000		+ % від робіт			

Рис. 2.5. Таблиця «посади»

На рис. 2.6 показана таблиця повноважень для кожної посади, ці повноваження визначають доступ до інформаційної системи.

Посади	Повноваження	Працівники	Одиниці Виміру	Місця зберігання	Запаси	Постачальник	Роботи	Клієнт
Обрати посаду		Адміністратор						
Посада	Об'єкт доступу	Читання	Зміна	Видалення				
Адміністратор	Результати	1	0	0				
Адміністратор	ДовідникКлієнтів	1	1	0				
Адміністратор	ДовідникАвто	1	1	0				
Адміністратор	ДовідникПрацівників	1	1	0				
Адміністратор	ДовідникПосад	0	0	0				
Адміністратор	ДовідникМісцьЗберігання	1	1	1				
Адміністратор	ДовідникОдиницьВимірювання	1	1	1				
Адміністратор	ДовідникЗапасів	1	1	0				
Адміністратор	ДовідникРобіт	1	1	0				
Адміністратор	ДовідникПостачальників	1	1	0				
Адміністратор	ЗамовленняОсновнаІнформація	1	1	0				
Адміністратор	ЗамовленняТовари	1	1	1				
Адміністратор	ЗамовленняРоботи	1	1	1				
Адміністратор	ЗамовленняВиконавці	1	1	1				
Адміністратор	ПрихідЗапасів	1	1	0				
Адміністратор	МатрицяПовноважень	0	0	0				

Рис. 2.6. Таблиця «повноваження»

На рис. 2.7 представлена таблиця запасів, в ній перераховані всі запчастини, які використовуються в роботах. Даний перелік буде використовуватися при оформленні замовлення на поставку запчастин майстром-приймальником.

Посади	Повноваження	Працівники	Одиниці Виміру	Місця зберігання	Запаси	Постачальник	Роботи	Клієнти	Авто
Код запасів	Номер по каталогу	Найменування	Виробник	Од. вим.	Ціна відпускна	Примітка			
9313	21120-1164010-20	АДСОРБЕР ВА3-2110 АВТОВАЗ ЕВ...	АВТОВАЗ	шт	515				
9314	21103-1164010-02	АДСОРБЕР ВА3-2110 АВТОВАЗ ЕВ...	АВТОВАЗ	шт	489				
9315	21112-1164009-00	АДСОРБЕР ВА3-2110 АВТОВАЗ С.О.	АВТОВАЗ	шт	505				
9316	21214-1164009-00	АДСОРБЕР ВА3-21214 АВТОВАЗ	АВТОВАЗ	шт	777				

Рис. 2.7. Таблиця «запаси»

На рис. 2.8 показана таблиця місць зберігання. В автосервісі є два склади, це основний склад і склад паливно мастильних матеріалів. Даний перелік буде використовуватися майстром-приймальником при оформленні поставки.

Посади	Повноваження	Працівники	Одиниці вимірювання	Місця Зберігання	Запаси
	КодМісцяЗберігання	Найменування	Примітка		
►	1	Основний склад	Графік роботи Пн-Пт 8:00-...		
	2	Склад ГЗМ			
*					

Рис. 2.8. Таблиця «місця зберігання»

На рисунку 2.9 показана таблиця одиниць виміру.

	КодОдВим	Найменування
►	1	кг
	2	л
	3	м2
	4	шт
	5	дец2
	6	компл
*		

Рис. 2.9. Таблиця «одиниці виміру»

На рисунку 2.10 представлена таблиця постачальників, в ній зберігаються дані про постачальників. Дані цієї таблиці потрібні для оформлення поставки майстром-приймальником.

Посади	Повноваження	Працівники	Одиниці виміру	Місця зберігання	Запаси	Постачальник	Роботи	Клієнти	Авто
	Найменування	Реквізити	Контакти	Примітка					
►	Авто-Альянс	м. Харків, вул. Євгена Коновальця...	(066) 585 33 88	з 9:00 до 18:00. Сб і Нд - ви...					
	АВТОЛІГА	16658, м. Київ, проспект Юрія Гага...	(055) 555 55 55	Запчастини					
*									

Рис. 2.10. Таблиця «Постачальники»

Під час проектування бази даних так само потрібно подумати про цілісність даних. Правильна структура таблиць дозволяє захистити дані від порушення зв'язків і внесення невірних значень. Необхідно визначити

найкращий шлях забезпечення цілісності даних. Цілісність даних ґрунтується на стійкості і точності даних, які зберігає база даних.

У таблицях БД була використана цілісність полів. У полях, таких як найменування, код замовлення, номер документа, неможливо використовувати нульові значення, так як це призведе до втрати необхідних даних про замовлення.

Для збереження даних, була дотримана цілісність таблиці. Всі рядки в таблицях мають унікальний ідентифікатор, названий *первинним ключем*. При видаленні будь-якої посади з БД, необхідно позначити її на видалення, щоб забезпечити збереження всіх даних, пов'язаних з видаленою посадою.

Так само дотримана цілісність посилань. Відносини між первинним ключем (таблиці, на яку посилаються) і зовнішнім ключем (таблиці, яка посилається на іншу) завжди захищені. Рядок основної таблиці, на яку посилаються, не може бути видалений і первинний ключ не може бути змінений, якщо вторинний ключ посилається на рядок, доки не буде знищений зв'язок. Інакше зв'язок порушується і відновити його потім стає проблематично.

РОЗДІЛ 3. РОЗРОБКА ЯК ДРУГА ЧАСТИНА СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДОКУМЕНТООБІГУ В СИСТЕМІ РОБОТИ

3.1. Загальна архітектура ІС, яка буде реалізована у вирішенні задачі

Розробка будь-якої програми включає в себе кілька етапів. Для початку необхідно визначитися - яка функціональність нам необхідна і, відповідно, яка структура програмного забезпечення зможе це реалізувати. Потім, нам необхідно вибрати спосіб взаємодії з даними (це основна функція нашої програми) і створити необхідні для цього інструменти [37]. На останньому етапі ми повинні забезпечити реалізацію призначеного для користувача інтерфейсу в нашій програмі і реалізувати допоміжні функції.

Згідно поставленого завдання, наша програма повинна:

- Надавати можливість авторизації різних користувачів системи;
- Надавати можливість перегляду таблиць та форм;
- Забезпечувати введення інформації у формах;
- Забезпечувати можливість редагування інформації в таблицях;
- Формувати остаточний звіт по зроблених роботі.

Система забезпечує зберігання даних про клієнтів, замовлення, автомобілі, запчастини, роботи і співробітників. Обробка інформації вручну займає багато часу, як при заповненні бланків, так і при пошуку і виправлення некоректних записів. Забезпечуючи можливість управлінському персоналу коригувати дані безпосередньо в режимі реального часу, багато проблем можуть бути зменшені або зовсім вирішені.

1) Загальні відомості

Повне найменування системи: Інформаційна система «Автосервіс»

Умовне позначення системи: ІС

Призначення і область застосування: програма призначена для створення, управління вмістом бази даних:

- база клієнтів, співробітників, постачальників;
- дані про запаси;
- дані про витрати і доходи;
- дані про роботи.

Програма надає інтерфейс для управління вмістом бази даних.

2) Вимоги до системи

Система забезпечує можливість переглянути опції:

- поділ користувачів, які підключаються через інтерфейс на групи в залежності від займаної посади, доступ до всіх даних повинен бути тільки у генерального директора автосервісу;
- можливість введення і редагування інформації в базі даних;
- наявність зрозумілого інтерфейсу для оформлення замовлення;
- можливість обліку запасів на складі.

3) Вимоги до надійності

- Здійснюється розмежування прав доступу до системи.
- Ведеться журнал подій системи.
- Відмови програми внаслідок некоректних дій користувача при взаємодії з програмою через інтерфейс неприпустимі.
- Кожен співробітник має доступ тільки до тих даних, які необхідні для його роботи. Генеральний директор має доступ до всіх даних інформаційної системи, але тільки для читання.
- Паролі та логіни зберігаються в довіднику співробітників, паролі в зашифрованому вигляді.

4) Вимоги до інформаційної та програмної сумісності та до програмних засобів, які використовуються програмою.

- Системні програмні засоби, що використовуються програмою, повинні бути представлені ліцензійною локалізованою версією операційної системи Windows 8 або Windows 10. Наявність .NET Framework 4 і MS SQL Server CE Client.

3.2. Вхідні та вихідні форми ІС СТО

Вхідними даними повинні бути:

- за заявками - номер, дата заявки, клієнт, вид послуг, що надаються, ;
- по клієнтах – ПІБ клієнта, контакти, урахування знижки;
- по автомобілях – код, марка, модель, VIN, регіональний номер;
- за видами робіт - найменування роботи, працівник, що виконується

даний вид ремонту;

Вихідними даними повинні бути:

- сформований рахунок, за наданий ремонт;
- наряд-замовлення із деталями замовлення, ремонту.

3.3 Алгоритми роботи ІС СТО

В процесі роботи над дипломною роботою була створена інформаційна система, яка забезпечує:

- підвищення ефективності управління;
- оптимізацію процесів збору, обробки, обліку та контролю інформації;
- підвищення якості обслуговування клієнтів, скорочення рутинної роботи;
- оперативність доступу до інформації для всіх підрозділів.

Система працює за принципом 1 автомобіль - 1 замовлення-наряд на рисунку 3.1 схематично показано як формується даний документ на автосервісі. Спочатку збираються дані про замовника та про автомобіль. Далі вибирається класифікація ремонту. Потім кожному класу ремонту підбираються відповідно роботи і деталі. Після всього цього список всіх робіт передається механіку або автослюсарю, а список необхідних деталей комірникові.

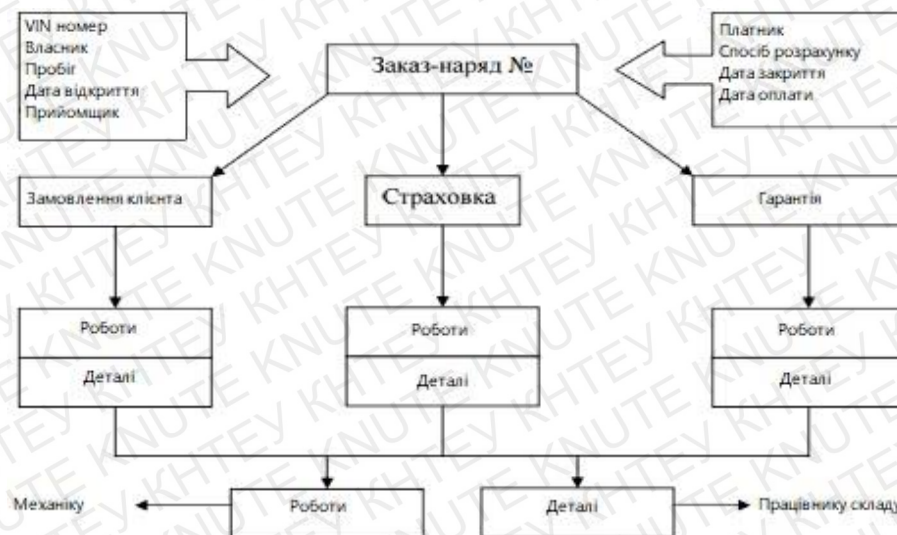


Рис. 3.1. Схема формування замовлення-наряду

Для створення найбільш простого інтерфейсу і підвищення безпеки даних було прийнято рішення про розмежування доступу до бази даних. Логіни і паролі були створені адміністратором баз даних. Так як інформація, яка зберігається в базі даних, має фінансову звітність, було вирішено використовувати алгоритми шифрування для паролів [8]. При запуску програми з'являється вікно з пропозицією вибрати користувача.

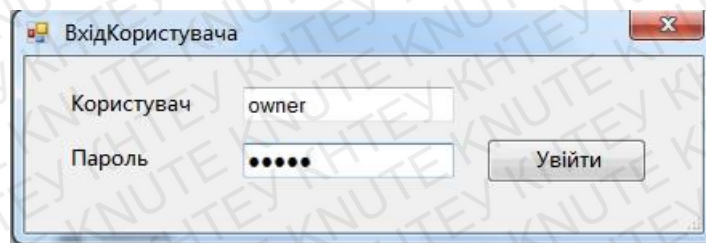


Рис. 3.2. Форма авторизації

Необхідно вибрати співробітника і ввести пароль. Після натискання клавіші «Ок» від пароля обчислюється хеш-функція і отриманий результат порівнюється з даними в базі. Якщо авторизація проходить успішно, завантажуються робоча форма співробітника відповідно до його статусу. Подивитися код програми можна в Додатку.

Інтерфейс генерального директора включає в себе всю інформацію автосервісу, але доступна вона тільки для читання.

На даній формі є можливість для додавання і редагування виконуваних робіт і використовуваних в них запчастин. Все це здійснюється за допомогою додаткових форм.

На вкладці «запаси» можна подивитися всі запчастини, масла, фарбу і інші матеріали, які є в наявності на складі.

Код	Номер на складі	Найменування	Виробник	Ол. м.	Літ. випуску	Примітка
9015	21125-540105	АМОРТИЗАТОР TRALLI SUPERORT BA3-2122 ЗАДНІЙ	АВТОВА3	шт	5/8	
9014	25151-111414	АМОРТИЗАТОР TRALLI SUPERORT BA3-2333 ЗАДНІЙ	АВТОВА3	шт	489	
9015	15155-651656	АМОРТИЗАТОР TRALLI SUPERORT BA3-2589 ЗАДНІЙ	АВТОВА3	шт	598	
9016	35135-466116	АМОРТИЗАТОР TRALLI SUPERORT BA3-4/85 ЗАДНІЙ	АВТОВА3	шт	777	
9017	51515-555555	АМОРТИЗАТОР TRALLI SUPERORT BA3-2589 ЗАДНІЙ	АВТОВА3	шт	432	
9018	55585-4/8952	АМОРТИЗАТОР TRALLI SUPERORT BA3-14/8	АВТОВА3	шт	258	
9019	22585-789654	АМОРТИЗАТОР TRALLI SUPERORT BA3-14/8 ЗАДНІЙ	АВТОВА3	шт	968	
9020	14585-551577	АМОРТИЗАТОР TRALLI SUPERORT BA3-2122 ЗАДНІЙ	АВТОВА3	шт	589	
9021	55854-515616	АМОРТИЗАТОР TRALLI SUPERORT BA3-3698	АВТОВА3	шт	857	
9022	58487-515166	АМОРТИЗАТОР TRALLI SUPERORT BA3-2122 ЗАДНІЙ	АВТОВА3	шт	577	
9023	41616-461651	АМОРТИЗАТОР TRALLI SUPERORT BA3-2122	АВТОВА3	шт	896	

Рис. 3.3. Інтерфейс генерального директора

В інтерфейсі адміністратора можливе додавання і зміна замовлень. Дані клієнта спочатку заносяться в довідник клієнтів, щоб уникнути втрату контактів, а потім вибирається зі списку в формі оформлення замовлення.

Рис. 3.4. Форма оформлення замовлення

Для того, щоб сформувати замовлення-наряд, потрібно просто натиснути кнопку «замовлення-наряд» і після оформлення замовлення з'явиться документ на друк у форматі html. На рисунку 3.5 показаний приклад замовлення-накладної.

Наряд-замовлення №2 від 16.03.2020

Клієнт: Макаренко Анастасій Олександрович

Авто: Toyota

Виконані роботи	Ціна	Кількість	Знижка	Сума
Масло МРПН	400.00	1	0 %	400.00
Заміна фільтрів	55.00	1	0 %	55.00
Всього:				455.00
Запчастини	Ціна	Кількість	Знижка	Сума
Всього:				0.00

В сумі за наряд-замовлення: 455,00

З умовами ремонту, столічки та гарантії ознайомлений і згодний. Замовник: _____

Замовлення виконано _____ Майстер: _____

Претензій по якості і переліку робіт не маю, автомобіль прийняв. Замовник: _____

Рис. 3.5. Приклад замовлення-наряду

З головного меню можна зайти в управління довідниками системи. Натиснувши пункт меню «Довідники» відкривається форма. У довідниках знаходяться дані про співробітників, клієнтів, запаси, постачальників, роботи.

На рисунку 3.7 представлена форма для оформлення поставки, в неї вносить дані майстер-приймальник. Після збереження даних, в вкладку «залишки запасів» автоматично вносяться прийняті майстром-приймальником матеріали.

ІдЗалису	ІдДокументу	КодЗапасів	Кількість	Ціна
1	1	АКБ 6ПР66 СКАТ Чорний	12	500

Рис. 3.7. Форма оформлення поставки

Система надає звіт по заданому періоду. (Рисунок 3.8, рисунок 3.9)

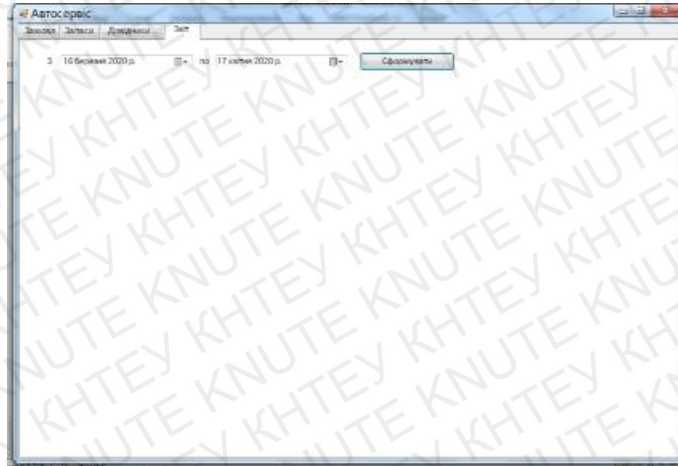


Рис. 3.8. Форма «звіт»

Звіт за період з 16.03.2020 по 17.04.2020

Замовлення

За період з 16.03.2020 по 17.04.2020 оформлено
замовлень: 7
на суму 3990.00 грн в т.ч.
робіт на суму 160 грн;

Поставки

За період з 16.03.2020 по 17.04.2020 оформлено поставок:
11
на суму 49 455.00 грн (кількість: 55).

Рис. 3.9. Звіт

Розроблена інформаційна система дозволяє співробітникам вести облік запасів на складі, облік клієнтів, стан робіт, облік співробітників, а так само проводити розрахунок витрат і доходів від виконаних робіт.

3.4 Характеристика задіяних засобів розробки програмного забезпечення та прикладних програм ІС СТО

Для написання програми була обрано середовище програмування Visual Studio 2012 C#, засноване на мові програмування C#. Дане середовище вигідно відрізняється ефективністю і надійністю [11].

Для стабільного функціонування програми необхідний комп'ютер фірми IBM або сумісний з ним, з об'ємом оперативної пам'яті не менше 128 Мб., Процесор з частотою не менш 600МГц.

Для вирішення поставленого завдання необхідно використовувати функціональну, ефективну і зручну платформу для розробки, що дозволяє застосовувати принципи об'єктно-орієнтованого програмування. В якості такої платформи була обрано середовище .NET.

Середовище розробки Visual Studio, що поставляється разом з .NET, надає необхідний інструментарій для ефективного і швидкого створення додатків з графічним інтерфейсом.

Поява технології .NET спричинило масову реконструкцію деяких мов програмування, які прагнуть використовувати ті чи інші можливості платформи, такі як C++ і Visual Basic. Microsoft вирішили запропонувати розробникам альтернативу - мову, орієнтовану спеціально на .NET і створили C#. Самі розробники мови описують її, як просту, сучасну, об'єктно-орієнтовану і безпечну мову програмування. Синтаксично C # нагадує C ++ і Java, що дозволяє за досить короткий час вивчити тонкощі нової мови.

Незважаючи на те, що C # і .NET призначені в першу чергу для веб-розробки, їх також активно застосовують для створення додатків, які повинні встановлюватися на машині кінцевого користувача, де і буде виконуватися вся обробка даних. Розробку таких додатків забезпечує бібліотека Windows Forms, що дозволяє проектувати графічний інтерфейс. Система, описана в даній роботі, розроблена саме за допомогою бібліотеки Windows Forms [36].

Мова програмування C# претендує на справжню об'єктну орієнтованість.

Мова програмування C# покликана реалізувати компонентно-орієнтований підхід до програмування, який сприяє меншій машинно-архітектурній залежності результуючого програмного коду, більшій гнучкості, переносимості і легкості повторного використання програм.

Принципово важливою відмінністю від попередників є початкова орієнтація на безпеку коду.

Розширена підтримка подієво-орієнтованого програмування.

Мова програмування C# є «рідною» для створення додатків в середовищі Microsoft .NET, оскільки тісно і ефективно інтегрована з ним.

Microsoft Visual Studio - це версія Visual Studio і .NET Framework, яка підтримує нові та вдосконалені об'єкти, включає середовище розробки з оновленим інтерфейсом і відрізняється інтегрованою підтримкою Microsoft SQL Server, дозволяючи створювати і розгортати проекти із застосуванням сервера баз даних. [35].

Інтерфейс Visual Studio традиційно виконаний в одному стилі з MS Office. Є список завдань, в який поміщають інформацію про помилки та про необхідні доробки. Кожному пункту можна призначити пріоритет, а після виконання встановити прапорець, який повідомляє про завершення зазначеного завдання. Task List підтримує сортування записів по тексту, за пріоритетністю статусу. Властивості проекту в Visual Studio можна редагувати за допомогою вбудованого інструменту, який дозволяє змінювати налаштування і підписи збірки, посилання на зовнішні модулі, набір прав, необхідних для її функціонування. Крім того, розробник легко може зберегти налаштування свого призначеного для користувача IDE в файлі налаштувань і застосовувати його в разі переходу на інший комп'ютер. Розробник баз даних може використовувати об'єктно-орієнтовані мови програмування, такі як C# і Visual Basic, спираючись на широкий спектр вбудованих можливостей класів і методів .NET Framework. Крім того, програміст може скористатися компонентами, написаними сторонніми компаніями. З появою SQL Server був вдосконалений механізм доступу до даних.

Класи і структури є двома основними конструкціями системи загальних типів CTS в платформі .NET Framework. Кожна по суті є структурою даних, інкапсулює набір даних і поведінку. Дані і поведінку є членами класу або структури, і в них включені методи, властивості, події і так далі [34].

Оголошення класу або структури подібно кресленням, яке використовується для створення екземплярів або об'єктів під час виконання. При визначенні класу або структури з ім'ям `Person`, `Person` є ім'ям типу. При оголошенні або ініціалізації змінної `p` типу `Person`, `p` вважається об'єктом чи примірником `Person`. Можливе створення декількох екземплярів одного типу `Person`, і кожен екземпляр може мати різні значення в своїх властивостях і полях.

Клас є посилальним типом. При створенні об'єкта класу змінна, до якої призначається об'єкт, зберігає тільки посилання на пам'ять. При призначенні посилання на об'єкт до нової змінної нова змінна посилається на вихідний об'єкт. Зміни, внесені через одну змінну, відображаються в іншій змінній, оскільки обидві вони посилаються на одні дані.

Структура є типом значення. При створенні структури змінна, до якої вона призначається, зберігає фактичні дані структури. При призначенні структури нової змінної виконується її копіювання. Тому нова змінна і вихідна змінна містять дві окремих копії одних даних. Зміни, внесені в одну копію, не впливають на іншу копію.

Приклад класу `Permission` (Лістинг 1.1), використовується для визначення окремого набору прав для окремого користувача:

Лістинг 1.1.

```
public class Permissions: List <Permission>
{
    public Permission this [string name]
    {
        get
```

```

    {
        return this.FirstOrDefault (tTemp => tTemp.name == name);
    }
}

public class Permission
{
    public string name;
    public bool canRead;
    public bool canEdit;
    public bool canDelete;
    public Permission (DataRow row)
    {
        name = row [ "Об'єктДоступу"]. ToString ();
        canRead = (row [ "Читання"]. ToString () == "1");
        canEdit = (row [ "Зміна"]. ToString () == "1");
        canDelete = (row [ "Видалення"]. ToString () == "1");
    }
}

```

ВИСНОВОК

У процесі розробки системи аналізу виконання робіт з системи ведення документів було проаналізовано методика і технологію обробки та зберігання даних на СТО. Це дозволяє зробити певні висновки та пропозиції по вдосконаленню роботи підприємства.

В ході роботи була дана характеристика діючої системи керування та характеристика існуючої системи аналізу виконання робіт з системи ведення документів. Було запропоновано та розроблено нову автоматизовану систему аналізу виконання робіт з системи ведення документів.

З метою обґрунтування доцільності даної розробки були проведені дослідження та аналіз існуючих теоретико-практичних розробок у даному напрямку. Це дало змогу підтвердити, що розробка автоматизованої системи є доцільною і необхідною в умовах ринкової конкуренції. Вона дасть змогу підприємству вийти на новий рівень роботи з замовленнями, більш прогресивний та динамічний.

Сферою застосування спроектованої системи аналізу є різні відділи на підприємстві СТО. За допомогою даної системи можна не тільки відслідкувати кількість виконаних і замовлених послуг, але й повністю проаналізувати роботу підприємства:

- Отримати звіт по тижнях
- Зробити аналіз виконаних послуг
- Проаналізувати темп росту виконаних замовлень

Даний програмний продукт зручний і простий у використанні. Користувачам буде досить легко освоїтись у інтерфейсі програми, а самому підприємству не потрібні нові співробітники.

Надалі можливі поліпшення й доробки програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адміністрування Microsoft SQL Server 2000. Навчальний курс MCSA / MCSE, MCDBA
2. А. П. Сиротинська, І. Д. Лазаришина. Інформаційні системи підприємств малого бізнесу, 2008. – 155 с.
3. Гайдамакин Н. А., Автоматизовані інформаційні системи, бази і банки даних. Вступний курс: Навчальний посібник. - М.: Геліос АРВ, 2002. - 368 с.
4. Автоматизовані інформаційні системи і технології: навчальний посібник / В. Є. Юринець, Р. В. Юринець; М-во освіти і науки, молоді та спорту України, Львів. нац. ун-т ім. І. Франка. Л.: [ЛНУ ім. І. Франка], 2014.
5. Сайт з SQL і клієнт / серверної технології [Електронний ресурс]. - Режим доступу: <http://www.sql.ru/>, вільний.
6. Бусленко Н.П. Моделирование сложных систем / Н.П. Бусленко. К.: [б.в.], 2013. – 165с.
7. Галіцина О.А. Бази даних: навчальний посібник / О. А. Галіцина. - 4-е вид., Перероб. і доп. - Юрайт, - 2014. - 78с.;
8. Денісова О. О. Автоматизоване проектування інформаційних систем: навчальний посібник / О.О. Денісова; Міністерство освіти і науки, молоді та спорту України, Державний вищий навчальний заклад "Київський національний економічний університет імені Вадима Гетьмана". - Київ: КНЕУ, 2014.
9. Бекаревич Ю.Б., Пушкіна Н.В., Смирнова О.Ю. Управління базами даних. [Текст] / Бекаревич Ю.Б., Пушкіна Н.В, 2009.
10. Жерновий Ю. В. Сучасні інформаційні системи та технології / Ю. В. Жерновий: підручник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2013. – 259с.
11. Джон Шарп. Microsoft Visual C#. Детальний опис, 8-е вид., 2017. – 655 с.

12. Мартін Р., Мартін М. Принципи, патерни і методики гнучкої розробки на мові C #, - 2011.
13. Віссер Дж. Розробка обслуговуючих програм на мові C #. Вид. ДМК Пресс, - 2016.
14. Марков О. Д. Організація автосервісу. Львів: Оріяна Нова, 1998. – 332 с.
15. Лудченко О. А. Технічна експлуатація і обслуговування автомобілів. Технологія – К.: Знання, 2009. – 478 с.
16. Канарчук В. Є., Лудченко О. А., Чигринець А. Д. Основи технічного обслуговування і ремонту автомобілів – К.: Вища шк., 2008.
17. Ким С. О., Пушкін П. С., Овчинніков С. І. Організація і планування промислового виробництва, - Мінськ: «Вища школа», - 1980.
18. Кожекін Г. Я., Синиця Л. М. Організація виробництва, - Мінськ: ПП «Екоперспектива», - 2008.
19. Інформаційні системи і технології: / Н.Б. Чорней, Р.К. Чорней ; Міжрегіональна академія управління персоналом. - Київ : [МАУП], 2015. - 187с.
20. Ю. Бекаревич, Н. Пушкіна. Самовчитель MS Office Access 2016, - 2017.
21. Анісімов А. П. Організація і планування автотранспортних підприємств. – М.: Транспорт, - 1982.
22. Будрій А. Г., Коновалов Г. А. Економіка автомобільного транспорту. – Київ: видавничий центр «Академія», - 2012.
23. Семенов Г. А., Станчевський В. К. Організація і планування на підприємстві. – Київ «Центр учбової літератури», - 2006.
24. Польшиков В. І., Сахно Є. Ю. Економіка, організація та управління технічним обслуговуванням і ремонтом машин. – Київ «Центр навчальної літератури», - 2010.
25. НДІАТ Короткий автомобільний довідник.
26. Норми витрат на матеріали і запасні частини.

27. Онищенко В. О., Старовірець А. С., Редкін О. В., Чевганова В. Я. Організація виробництва.
28. Польшаков В. І., Сахно Є. Ю. Економіка, організація та управління технічним обслуговуванням і ремонтом машин.
29. Гончарова О. М., Ефективна економіка, КНУ ім. Тараса Шевченка
30. Ризун Н. О. Сучасні інструменти аналізу складних економічних систем / Н. О. Ризун, Є. Г. Холод // Академічних огляд. – 2008. - № 1.
31. Трубілін І. Т. Автоматизовані інформаційні технології в економіці, 1999. – 416 с.
32. Ендрю Троелсен, Філіпп Джепікс. Мова програмування C # 7 і платформи .NET і .NET Core. Том 2, 2020.
33. Дж. Ріхтер. CLR via C #. Програмування на платформі Microsoft.NET Framework 4.5 мовою C #, 2016.
34. Брайан Нойєс. Прив'язка даних у Windows Forms, 2009.
35. Таха Х. Введення у дослідження операцій. Пер. з англ. [Текст, електронний ресурс] / Хемді Таха. – М.: Вільямс, 2005.
36. Томашевський В. Моделювання систем [Текст] / В. М. Томашевський. – К.: БПУ, 2005.
37. Методи моделювання бізнес-процесів підприємства засобами системного аналізу [Електронний ресурс]. – Режим доступу: <http://vuz-lib.com/content/view/403/84/>.
38. Чекалов А. Бази даних: від проектування до розробки додатків [Текст, електронний ресурс] / А. Чекалов 2003.
39. Янг Б. Об'єктно-орієнтовний аналіз і проектування з прикладами додатків. 3-є видання [Текст, електронний ресурс] / Боббі Янг, Джим Коннален, Граді Буч. – М.: Вільямс, 2008. – 720 с.
40. Підлипний, Ю. В. Розробка моделей, методів, та алгоритмів діагностики технологічних процесів виготовлення ІС : автореферат дисертації на здобуття наукового ступеня кандидата технічних наук / Ю.В. Підлипний. - Львів : 2013. – 166с.
41. М.Я. Гвоздь, Національний університет “Львівська Політехніка” – 106с.

ДОДАТОК

Програмний код реалізації додатку

```
///  
///Represents the strongly named DataTable class.  
///</summary>  
[global::System.Serializable()]  
[global::System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSch  
ema")]  
public partial class КлієнтиDataTable :  
global::System.Data.TypedTableBase<КлієнтиRow> {  
private global::System.Data.DataColumn columnКодКлієнта;  
private global::System.Data.DataColumn columnНайменування;  
private global::System.Data.DataColumn columnКонтакти;  
private global::System.Data.DataColumn columnПримітка;  
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]  
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design  
.TypedDataSetGenerator", "4.0.0.0")]  
public КлієнтиDataTable() {  
this.TableName = "Клієнти";  
this.BeginInit();  
this.InitClass();  
this.EndInit();  
}  
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]  
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design  
.TypedDataSetGenerator", "4.0.0.0")]  
internal КлієнтиDataTable(global::System.Data.DataTable table) {
```

```

this.TableName = table.TableName;
if ((table.CaseSensitive != table.DataSet.CaseSensitive)) {
    this.CaseSensitive = table.CaseSensitive;
}
if ((table.locale.ToString() != table.DataSet.locale.ToString())) {
    this.locale = table.locale;
}
if ((table.Namespace != table.DataSet.Namespace)) {
    this.Namespace = table.Namespace;
}
this.Prefix = table.Prefix;
this.MinimumCapacity = table.MinimumCapacity;
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected КлієнтиDataTable(global::System.Runtime.Serialization.SerializationInfo
info, global::System.Runtime.Serialization.StreamingContext context) :
    base(info, context) {
    this.InitVars();
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public global::System.Data.DataColumn КодКлієнтаColumn {
    get {
        return this.columnКодКлієнта;
    }
}

```

```

}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public global::System.Data.DataColumn НайменуванняColumn {
    get {
        return this.columnНайменування;
    }
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public global::System.Data.DataColumn КонтактиColumn {
    get {
        return this.columnКонтакти;
    }
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public global::System.Data.DataColumn ПриміткаColumn {
    get {
        return this.columnПримітка;
    }
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]

```

```

[global::System.ComponentModel.Browsable(false)]
public int Count {
    get {
        return this.Rows.Count;
    }
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public КлієнтиRow this[int index] {
    get {
        return ((КлієнтиRow)(this.Rows[index]));
    }
}

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public event КлієнтиRowChangeEventHandler КлієнтиRowChanging;
[global::System.CodeDom.Compiler.GeneratedCodeAttribute
"System.Data.Design.TypedDataSetGenerator", "4.0.0.0")]
public event КлієнтиRowChangeEventHandler КлієнтиRowChanged;
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public event КлієнтиRowChangeEventHandler КлієнтиRowDeleting;
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public event КлієнтиRowChangeEventHandler КлієнтиRowDeleted;
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design

```

```

.TypedDataSetGenerator", "4.0.0.0")]

public void AddКлієнтиRow(КлієнтиRow row) {
this.Rows.Add(row);
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]

public КлієнтиRow AddКлієнтиRow(string Найменування, string Контакти, string
Примітка) {
КлієнтиRow rowКлієнтиRow = ((КлієнтиRow)(this.NewRow()));
object[] columnValuesArray = new object[] {
null,
Найменування,
Контакти,
Примітка};
rowКлієнтиRow.ItemArray = columnValuesArray;
this.Rows.Add(rowКлієнтиRow);
return rowКлієнтиRow;
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]

public КлієнтиRow FindByКодКлієнта(int КодКлієнта) {
return ((КлієнтиRow)(this.Rows.Find(new object[] {
КодКлієнта})));
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design

```

```

.TypedDataSetGenerator", "4.0.0.0")]
public override global::System.Data.DataTable Clone() {
    КлієнтиDataTable cln = ((КлієнтиDataTable)(base.Clone()));
    cln.InitVars();
    return cln;
}
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override global::System.Data.DataTable CreateInstance() {
    return new КлієнтиDataTable();
}
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
internal void InitVars() {
    this.columnКодКлієнта = base.Columns["КодКлієнта"];
    this.columnНайменування = base.Columns["Найменування"];
    this.columnКонтакти = base.Columns["Контакти"];
    this.columnПримітка = base.Columns["Примітка"];
}
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
private void InitClass() {
    this.columnКодКлієнта = new global::System.Data.DataColumn "КодКлієнта",
    typeof(int), null, global::System.Data.MappingType.Element);
    base.Columns.Add(this.columnКодКлієнта);
}

```

```

this.columnНайменування = new global::System.Data.DataColumn
"Найменування", typeof(string), null, global::System.Data.MappingType.Element);
base.Columns.Add(this.columnНайменування);
this.columnКонтакти = new global::System.Data.DataColumn "Контакти",
typeof(string), null, global::System.Data.MappingType.Element);
base.Columns.Add(this.columnКонтакти);
this.columnПримітка = new global::System.Data.DataColumn "Примітка",
typeof(string), null, global::System.Data.MappingType.Element);
base.Columns.Add(this.columnПримітка);
this.Constraints.Add(new global::System.Data.UniqueConstraint "Constraint1",
new global::System.Data.DataColumn[] {
    this.columnКодКлієнта}, true));
this.columnКодКлієнта.AutoIncrement = true;
this.columnКодКлієнта.AutoIncrementSeed = -1;
this.columnКодКлієнта.AutoIncrementStep = -1;
this.columnКодКлієнта.AllowDBNull = false;
this.columnКодКлієнта.ReadOnly = true;
this.columnКодКлієнта.Unique = true;
this.columnНайменування.AllowDBNull = false;
this.columnНайменування.MaxLength = 100;
this.columnКонтакти.MaxLength = 200;
this.columnПримітка.MaxLength = 200;
}
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public КлієнтиRow NewКлієнтиRow() {
    return ((КлієнтиRow)(this.NewRow()));
}

```

```

}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override global::System.Data.DataRow
NewRowFromBuilder(global::System.Data.DataRowBuilder builder) {
return new КлієнтиRow(builder);
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override global::System.Type GetRowType() {
return typeof(КлієнтиRow);
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override void
OnRowChanged(global::System.Data.DataRowChangeEventArgs e) {
base.OnRowChanged(e);
if ((this.КлієнтиRowChanged != null)) {
this.КлієнтиRowChanged(this, new
КлієнтиRowChangeEvent(((КлієнтиRow)(e.Row)), e.Action));
}
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]

```

```

protected override void
OnRowChanging(global::System.Data.DataRowChangeEventArgs e) {
base.OnRowChanging(e);
if ((this. КлієнтиRowChanging != null)) {
this. КлієнтиRowChanging(this, new
КлієнтиRowChangeEvent(((КлієнтиRow)(e.Row)), e.Action));
}
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override void
OnRowDeleted(global::System.Data.DataRowChangeEventArgs e) {
base.OnRowDeleted(e);
if ((this. КлієнтиRowDeleted != null)) {
this. КлієнтиRowDeleted(this, new
КлієнтиRowChangeEvent(((КлієнтиRow)(e.Row)), e.Action));
}
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override void
OnRowDeleting(global::System.Data.DataRowChangeEventArgs e) {
base.OnRowDeleting(e);
if ((this. КлієнтиRowDeleting != null)) {
this. КлієнтиRowDeleting(this, new
КлієнтиRowChangeEvent(((КлієнтиRow)(e.Row)), e.Action));
}
}

```

```
}  
}
```

```
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
```

```
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design  
.TypedDataSetGenerator", "4.0.0.0")]
```

```
public static global::System.Xml.Schema.XmlSchemaComplexType
```

```
GetTypedTableSchema(global::System.Xml.Schema.XmlSchemaSet xs) {
```

```
global::System.Xml.Schema.XmlSchemaComplexType type = new
```

```
global::System.Xml.Schema.XmlSchemaComplexType();
```

```
global::System.Xml.Schema.XmlSchemaSequence sequence = new
```

```
global::System.Xml.Schema.XmlSchemaSequence();
```

```
MyDataSet ds = new MyDataSet();
```

```
global::System.Xml.Schema.XmlSchemaAny any1 = new
```

```
global::System.Xml.Schema.XmlSchemaAny();
```

```
any1.Namespace = "http://www.w3.org/2001/XMLSchema";
```

```
any1.MinOccurs = new decimal(0);
```

```
any1.MaxOccurs = decimal.MaxValue;
```

```
any1.ProcessContents =
```

```
global::System.Xml.Schema.XmlSchemaContentProcessing.lax;
```

```
sequence.Items.Add(any1);
```

```
global::System.Xml.Schema.XmlSchemaAny any2 = new
```

```
global::System.Xml.Schema.XmlSchemaAny();
```

```
any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1";
```

```
any2.MinOccurs = new decimal(1);
```

```
any2.ProcessContents =
```

```
global::System.Xml.Schema.XmlSchemaContentProcessing.lax;
```

```
sequence.Items.Add(any2);
```

```
global::System.Xml.Schema.XmlSchemaAttribute attribute1 = new
```

```

global::System.Xml.Schema.XmlSchemaAttribute();
attribute1.Name = "namespace";
attribute1.FixedValue = ds.Namespace;
type.Attributes.Add(attribute1);
global::System.Xml.Schema.XmlSchemaAttribute attribute2 = new
global::System.Xml.Schema.XmlSchemaAttribute();
attribute2.Name = "tableName";
attribute2.FixedValue = "КлієнтиDataTable";
type.Attributes.Add(attribute2);
type.Particle = sequence;
global::System.Xml.Schema.XmlSchema dsSchema = ds.GetSchemaSerializable();
if (xs.Contains(dsSchema.TargetNamespace)) {
    global::System.IO.MemoryStream s1 = new global::System.IO.MemoryStream();
    global::System.IO.MemoryStream s2 = new global::System.IO.MemoryStream();
    try {
        global::System.Xml.Schema.XmlSchema schema = null;
        dsSchema.Write(s1);
        for (global::System.Collections.IEnumerator schemas =
xs.Schemas(dsSchema.TargetNamespace).GetEnumerator(); schemas.MoveNext();
) {
            schema = ((global::System.Xml.Schema.XmlSchema)(schemas.Current));
            s2.Setlength(0);
            schema.Write(s2);
            if ((s1.length == s2.length)) {
                s1.Position = 0;
                s2.Position = 0;
                for (; ((s1.Position != s1.length)
&& (s1.ReadByte() == s2.ReadByte())); ) {

```

```

    ;
}
if ((s1.Position == s1.Length)) {
    return type;
}

///Represents the strongly named DataTable class.
///</summary>
[global::System.Serializable()]
[global::System.Xml.Serialization.XmlSchemaProviderAttribute("GetTypedTableSchema")]
public partial class МісцяЗберіганняDataTable :
    global::System.Data.TypedTableBase<МісцяЗберіганняRow> {
    private global::System.Data.DataColumn columnКодМісцяЗберігання;
    private global::System.Data.DataColumn columnНайменування;
    private global::System.Data.DataColumn columnПримітка;
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")]
    public МісцяЗберіганняDataTable() {
        this.TableName = "МісцяЗберігання";
        this.BeginInit();
        this.InitClass();
        this.EndInit();
    }
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "4.0.0.0")]
    internal МісцяЗберіганняDataTable(global::System.Data.DataTable table) {

```

```

this.TableName = table.TableName;
if ((table.CaseSensitive != table.DataSet.CaseSensitive)) {
    this.CaseSensitive = table.CaseSensitive;
}
if ((table.locale.ToString() != table.DataSet.locale.ToString())) {
    this.locale = table.locale;
}
if ((table.Namespace != table.DataSet.Namespace)) {
    this.Namespace = table.Namespace;
}
this.Prefix = table.Prefix;
this.MinimumCapacity = table.MinimumCapacity;
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected
МісцяЗберіганняDataTable(global::System.Runtime.Serialization.SerializationInfo
info, global::System.Runtime.Serialization.StreamingContext context) :
    base(info, context) {
    this.InitVars();
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public global::System.Data.DataColumn КодМісцяЗберіганняColumn {
    get {
        return this.columnКодМісцяЗберігання;
    }
}

```

```

    }

    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
    public global::System.Data.DataColumn НайменуванняColumn {
        get {
            return this.columnНайменування;
        }
    }

    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
    public global::System.Data.DataColumn ПриміткаColumn {
        get {
            return this.columnПримітка;
        }
    }

    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
    [global::System.ComponentModel.Browsable(false)]
    public int Count {
        get {
            return this.Rows.Count;
        }
    }

    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design

```

```

.TypedDataSetGenerator", "4.0.0.0")]
public МісцяЗберіганняRow this[int index] {
    get {
        return ((МісцяЗберіганняRow)(this.Rows[index]));
    }
}
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public event МісцяЗберіганняRowChangeEventHandler
МісцяЗберіганняRowChanging;
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public event МісцяЗберіганняRowChangeEventHandler
МісцяЗберіганняRowChanged;
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public event МісцяЗберіганняRowChangeEventHandler
МісцяЗберіганняRowDeleting;
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public event МісцяЗберіганняRowChangeEventHandler
МісцяЗберіганняRowDeleted;
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public void AddМісцяЗберіганняRow(МісцяЗберіганняRow row) {
    this.Rows.Add(row);
}

```

```

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public МісцяЗберіганняRow AddМісцяЗберіганняRow(string Найменування,
string Примітка) {
    МісцяЗберіганняRow rowМісцяЗберіганняRow =
    ((МісцяЗберіганняRow)(this.NewRow()));
    object[] columnValuesArray = new object[] {
        null,
        Найменування,
        Примітка};
    rowМісцяЗберіганняRow.ItemArray = columnValuesArray;
    this.Rows.Add(rowМісцяЗберіганняRow);
    return rowМісцяЗберіганняRow;
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public МісцяЗберіганняRow FindByКодМісцяЗберігання (int
КодМісцяЗберігання) {
    return ((МісцяЗберіганняRow)(this.Rows.Find(new object[] {
        КодМісцяЗберігання})));
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public override global::System.Data.DataTable Clone() {
    МісцяЗберіганняDataTable cln = ((МісцяЗберіганняDataTable)(base.Clone()));

```

```

    cln.InitVars();
    return cln;
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override global::System.Data.DataTable CreateInstance() {
    return new МісцяЗберіганняDataTable();
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
internal void InitVars() {
    this.columnКодМісцяЗберігання = base.Columns["КодМісцяЗберігання"];
    this.columnНайменування = base.Columns["Найменування"];
    this.columnПримітка = base.Columns["Примітка"];
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
private void InitClass() {
    this.columnКодМісцяЗберігання = new global::System.Data.DataColumn
"КодМісцяЗберігання", typeof(int), null,
global::System.Data.MappingType.Element);
    base.Columns.Add(this.columnКодМісцяЗберігання);
    this.columnНайменування = new global::System.Data.DataColumn
("Найменування", typeof(string), null,
global::System.Data.MappingType.Element);

```

```

base.Columns.Add(this.columnНайменування);
this.columnПримітка = new global::System.Data.DataColumn("Примітка",
typeof(string), null, global::System.Data.MappingType.Element);
base.Columns.Add(this.columnПримітка);
this.Constraints.Add(new global::System.Data.UniqueConstraint("Constraint1",
new global::System.Data.DataColumn[] {
this.columnКодМісцяЗберігання}, true));
this.columnКодМісцяЗберігання.AutoIncrement = true;
this.columnКодМісцяЗберігання.AutoIncrementSeed = -1;
this.columnКодМісцяЗберігання.AutoIncrementStep = -1;
this.columnКодМісцяЗберігання.AllowDBNull = false;
this.columnКодМісцяЗберігання.ReadOnly = true;
this.columnКодМісцяЗберігання.Unique = true;
this.columnНайменування.AllowDBNull = false;
this.columnНайменування.MaxLength = 100;
this.columnПримітка.MaxLength = 100;
}
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public МісцяЗберіганняRow NewМісцяЗберіганняRow() {
return ((МісцяЗберіганняRow)(this.NewRow()));
}
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override global::System.Data.DataRow
NewRowFromBuilder(global::System.Data.DataRowBuilder builder) {

```

```

return new МісцяЗберіганняRow(builder);
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override global::System.Type GetRowType() {
return typeof(МісцяЗберіганняRow);
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override void
OnRowChanged(global::System.Data.DataRowChangeEventArgs e) {
if ((this. МісцяЗберіганняRowChanged != null)) {
this. МісцяЗберіганняRowChanged(this, new
МісцяЗберіганняRowChangeEvent(((МісцяЗберіганняRow)(e.Row)), e.Action));
}
}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
protected override void
OnRowChanging(global::System.Data.DataRowChangeEventArgs e) {
base.OnRowChanging(e);
if ((this. МісцяЗберіганняRowChanging != null)) {
this. МісцяЗберіганняRowChanging(this, new
МісцяЗберіганняRowChangeEvent(((МісцяЗберіганняRow)(e.Row)), e.Action));
}
}

```

```

    }

    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
    protected override void
    OnRowDeleted(global::System.Data.DataRowChangeEventArgs e) {
        base.OnRowDeleted(e);
        if ((this. МісцяЗберіганняRowDeleted != null)) {
            this. МісцяЗберіганняRowDeleted(this, new
            МісцяЗберіганняRowChangeEvent(((МісцяЗберіганняRow)(e.Row)), e.Action));
        }
    }

    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
    protected override void
    OnRowDeleting(global::System.Data.DataRowChangeEventArgs e) {
        base.OnRowDeleting(e);
        if ((this. МісцяЗберіганняRowDeleting != null)) {
            this. МісцяЗберіганняRowDeleting(this, new
            МісцяЗберіганняRowChangeEvent(((МісцяЗберіганняRow)(e.Row)), e.Action));
        }
    }

    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
    public void RemoveМісцяЗберіганняRow(МісцяЗберіганняRow row) {
        this.Rows.Remove(row);
    }

```

```

}

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design
.TypedDataSetGenerator", "4.0.0.0")]
public static global::System.Xml.Schema.XmlSchemaComplexType
GetTypedTableSchema(global::System.Xml.Schema.XmlSchemaSet xs) {
    global::System.Xml.Schema.XmlSchemaComplexType type = new
    global::System.Xml.Schema.XmlSchemaComplexType();
    global::System.Xml.Schema.XmlSchemaSequence sequence = new
    global::System.Xml.Schema.XmlSchemaSequence();
    MyDataSet ds = new MyDataSet();
    global::System.Xml.Schema.XmlSchemaAny any1 = new
    global::System.Xml.Schema.XmlSchemaAny();
    any1.Namespace = "http://www.w3.org/2001/XMLSchema";
    any1.MinOccurs = new decimal(0);
    any1.MaxOccurs = decimal.MaxValue;
    any1.ProcessContents =
    global::System.Xml.Schema.XmlSchemaContentProcessing.lax;
    sequence.Items.Add(any1);
    global::System.Xml.Schema.XmlSchemaAny any2 = new
    global::System.Xml.Schema.XmlSchemaAny();
    any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1";
    any2.MinOccurs = new decimal(1);
    any2.ProcessContents =
    global::System.Xml.Schema.XmlSchemaContentProcessing.lax;
    sequence.Items.Add(any2);
    global::System.Xml.Schema.XmlSchemaAttribute attribute1 = new
    global::System.Xml.Schema.XmlSchemaAttribute();

```

```

attribute1.Name = "namespace";
attribute1.FixedValue = ds.Namespace;
type.Attributes.Add(attribute1);
global::System.Xml.Schema.XmlSchemaAttribute attribute2 = new
global::System.Xml.Schema.XmlSchemaAttribute();
attribute2.Name = "tableName";
attribute2.FixedValue = "МісцяЗберіганняDataTable";
type.Attributes.Add(attribute2);
type.Particle = sequence;
global::System.Xml.Schema.XmlSchema dsSchema = ds.GetSchemaSerializable();
if (xs.Contains(dsSchema.TargetNamespace)) {
    global::System.IO.MemoryStream s1 = new global::System.IO.MemoryStream();
    global::System.IO.MemoryStream s2 = new global::System.IO.MemoryStream();
    try {
        global::System.Xml.Schema.XmlSchema schema = null;
        dsSchema.Write(s1);
        for (global::System.Collections.IEnumerator schemas =
xs.Schemas(dsSchema.TargetNamespace).GetEnumerator(); schemas.MoveNext();
) {
            schema = ((global::System.Xml.Schema.XmlSchema)(schemas.Current));
            s2.Setlength(0);
            schema.Write(s2);
            if ((s1.length == s2.length)) {
                s1.Position = 0;
                s2.Position = 0;
                for (; ((s1.Position != s1.length)
&& (s1.ReadByte() == s2.ReadByte())); ) {
                    ;
                }
            }
        }
    }
}

```

```
}  
if ((s1.Position == s1.length)) {  
    return type;  
}  
}  
}  
}  
finally {  
    if ((s1 != null)) {  
        s1.Close();  
    }  
    if ((s2 != null)) {  
        s2.Close();  
    }  
}
```