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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення для обліку товарів в магазині

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Петрик О.М. (прізвище та ініціали)

Керівник (підпис) Струтинська І.В. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Осухівська Г.М. (прізвище та ініціали)

Тернопіль
2021

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Петрику Олегу Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення для обліку товарів в магазині.

Керівник роботи Струтинська Ірина Володимирівна, д.е.н., професор кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «02» березня 2021 року № 4/7-171

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 25 січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи		*Виконано*
2.	Підбір джерел про розробку програмного забезпечення		*Виконано*
	для обліку товарів		
3.	Переклад та опрацювання джерел про розробку		*Виконано*
	програмного забезпечення для обліку товарів		
4.	Виконання дослідження щодо розробки програмного		*Виконано*
	забезпечення для обліку товарів		
5	Розроблення програмного забезпечення для обліку		*Виконано*
	товарів		
6.	Оформлення розділу «Теоретичні положення розробки		*Виконано*
	програмного забезпечення для обліку товарів»		
7.	Оформлення розділу «Розробка програмного		*Виконано*
	забезпечення для автоматизації обліку товарів»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»		*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони		*Виконано*
	праці»		
10.	Оформлення кваліфікаційної роботи		*Виконано*
11.	Нормоконтроль		*Виконано*
12.	Перевірка на плагіат		*Виконано*
13.	Попередній захист кваліфікаційної роботи		*Виконано*
14.	Захист кваліфікаційної роботи		

Студент _____ Петрик О.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Струтинська І.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення для обліку товарів в магазині // Кваліфікаційна робота освітнього рівня «Бакалавр» // Петрик Олег Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2021 // С. 49, рис. – 13, табл. – 8, кресл. – 0, додат. – 3, бібліогр. – 40.

Ключові слова: база даних, облік, автозапчастини, сутності, магазин, СУБД, C#, SQL.

Кваліфікаційна робота присвячена розробці програмного забезпечення для обліку товарів в магазині автозапчастин.

Метою даної кваліфікаційної роботи є дослідження, проектування та розробка баз даних, шляхом створення програмного забезпечення для обліку товарів в магазині автозапчастин.

В першому розділі кваліфікаційної роботи розглянуто характеристику об'єкта дослідження, необхідність автоматизації складського обліку товарів та аналіз існуючих додатків для ведення обліку в малому і середньому бізнесі. Проводиться огляд сучасних СУБД та технологій, які використовуються при роботі з даними та відбувається обґрунтування вибору СУБД та засобів розробки програмного забезпечення, а також формується постановка завдання кваліфікаційної роботи.

В другому розділі кваліфікаційної роботи розглянуто аналіз предметної області, реалізацію таблиць БД, здійснюється розробка бази даних та тестування розробленого програмного забезпечення.

В розділі «Безпека життєдіяльності, основи охорони праці» розглядається питання маркетингової діяльності магазину та психофізіологічного розвантаження для працівників.

ANNOTATION

Software development for shop accounting // Qualification work of the educational level "Bachelor" // Oleg Petryk // // Ternopil Ivan Pului National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group CH-41 // Ternopil, 2021 // P. 49, fig. – 13, tables – 8, chair. – 0, annexes. – 3, ref. – 40.

Keywords: database, accounting, auto parts, essences, shop, DBMS, C#, SQL.

Qualification work is devoted to the development of software for accounting for goods in the auto parts store.

The purpose of this qualification work is research, design and development of databases by creating software for accounting of goods in the auto parts store.

The first section of the qualification work discusses the characteristics of the object of study, the need to automate inventory of goods and analysis of existing applications for accounting in small and medium-sized businesses. An overview of modern DBMS and technologies used in working with data is carried out and the choice of DBMS and software development tools is substantiated, as well as the task of qualification work is formed.

In the second section of the qualification work the analysis of the subject area, the implementation of database tables, the development of a database and testing of the developed software are considered.

The section "Life safety, basics of labor protection" deals with the marketing activities of the store and psycho-physiological relief for employees.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – інтерфейс прикладного програмування.

CSV (англ. Comma-Separated Values або Character-Separated Values) – значення, розділені комою або значення розділені символом.

ERD (англ. Entity-Relationship Diagram) – діаграма сутність-зв’язок.

IDE (англ. Integrated Development Environment) – інтегроване середовище розробки.

IT (англ. Information technology) – інформаційні технології.

JSON (англ. JavaScript Object Notation) – текстовий формат обміну даних, заснований на JavaScript.

MS (англ. Microsoft або Microsoft Corporation) – корпорація Майкрософт.

SQL (англ. Structured Query Language) – мова структорованих запитів, декларативна мова програмування для взаємодії користувача з базами даних.

SSMS (англ. SQL Server Management Studio) – інтегроване середовище для проектування і управління базами даних.

БД – база даних.

ОС – операційна система.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

ЗМІСТ

ВСТУП

Актуальність теми роботи. Внаслідок розвитку комп'ютерної техніки, область використання інформаційних технологій стрімко розширюється. Зараз вона використовується майже у всіх сферах діяльності. З часом комп'ютерні технології стали доступнішими, за рахунок цього більша кількість підприємств може дозволити собі використання комп'ютерної мережі та програмного забезпечення. Використовуючи сучасні IT-технології можна вирішувати різні проблеми, що пов'язані з організацією діяльності підприємств, включаючи проблеми ведення складського обліку товарів.

Ведення складського обліку на будь-якому підприємстві без автоматизованих систем та спеціалізованих програм, що дозволяють відстежувати всі зміни, які відбуваються з товарними запасами компанії практично неможливо.

Автоматизований комп'ютерний облік полегшує роботу, зменшує час на оформлення документів, виконує аналіз даних про діяльність товарів на складі. Отже, використання автоматизованого обліку послаблює навантаження на працівників, зменшує додаткові витрати на облік за рахунок зменшення людських ресурсів та економить час на оформлення, пошук, ввід інформації.

Мета і завдання дослідження. Метою даної кваліфікаційної роботи є проектування та розробка програмного забезпечення для обліку товарів в магазині автозапчастин «SERVANT AUTO». Програма повинна здійснювати облік продажу товарів за допомогою бази даних. Інтерфейс користувача програми повинен бути простим та інформативним у використанні, для максимальної швидкодії.

Основний перелік **завдань** для досягнення мети дослідження:

- провести аналіз джерел, що стосуються системи обліку товарів;
- аналіз існуючих систем, що використовуються для обліку товарів;
- визначити основні функціональні вимоги до розробки програмного забезпечення для обліку в магазині та визначити сутності і варіанти використання;

- спроектувати та розробити базу даних для обліку товарів в магазині автозапчастин, здійснити їх опис взаємодії та принципів роботи;
- здійснити тестування створеного програмного забезпечення для обліку товарів в магазині автозапчастин «SERVANT AUTO».

Практичне значення одержаних результатів. У ході виконання кваліфікаційної роботи розроблене програмне забезпечення для обліку товарів може успішно використовуватись та бути корисним в магазині автозапчастин «SERVANT AUTO». Дане програмне забезпечення для обліку товарів також може використовуватись для роботи малих підприємств різної сфери діяльності.

1 ТЕОРЕТИЧНІ ПОЛОЖЕННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ ТОВАРІВ В МАГАЗИНІ

1.1 Характеристика об’єкта дослідження

«SERVANT AUTO» – це молода компанія, яка динамічно розвивається у сфері продажів автозапчастин та автомобільних мастил для легкових автомобілів у Львові.

Основними пріоритетами та перевагами компанії є: швидкість та зручність доставки продукції, відповідальність, надійність, якісний та швидкий підбір потрібних запчастин, а також індивідуальний підхід до кожного клієнта.

Мета діяльності «SERVANT AUTO» полягає у досягненні фінансових результатів від продажу запчастин, максимальне задоволення потреб клієнтів, розширення своєї мережі, формування довгострокових партнерських відношень.

Компанія пропонує клієнтам якісні запчастини за помірними цінами для легкових авто під замовлення або зі складу. Основним видом діяльності є продаж запчастин за наступними напрямами:

- автозапчастини для ТО, масла і інші рідини;
- автозапчастини для ходової;
- автозапчастини для двигуна;
- автозапчастини кузовних елементів.

Більшість асортименту автозапчастин для іномарок, замовлення запчастин для вітчизняного виробника робиться за невеликими виключеннями, основні іномарки якими займається магазин: «Volkswagen», «Opel», «BMW», «Audi», «Mercedes-Benz», «Toyota», «Seat», «Renault» та інші. У порівнянні з конкурентами формування цін в магазині має середній рівень, періодично пропонуються акційні пропозиції.

1.2 Необхідність автоматизації складського обліку товарів та аналіз існуючих додатків для ведення обліку в малому та середньому бізнесі

У сучасному світі в умовах глобалізації та інформатизації суспільства на будь-які підприємства, установи чи приватні організації обрушується величезний обсяг інформації.

Електронний документообіг – це об’єднання процесів створення, оброблення, відправлення, передавання, зберігання, використання і знищення електронних документів [1].

Процес контролю за рухом товару називають складським обліком. Розрізняють такі види складського обліку:

- ручний облік товару;
- облік в текстових редактор або в таблицях Excel;
- автоматизований облік товарів на складі.

Автоматизація необхідна для того, щоб організувати роботу складу та мінімізувати участь людини. Можливості системи наступні:

- контроль за залишками на складі;
- відстеження руху товарів;
- фіксовані продажі;
- контроль продажів та формування аналітики;
- відстеження дій працівників;
- працювати з декількома складами чи магазинами [2].

Одним з головних критеріїв ведення бізнесу є автоматизація складу, оскільки вона максимально та ефективно використовує потенціал складських приміщень. Автоматизована система складського обліку виконує всю рутинну працю [3].

Для ведення обліку та автоматизації існує безліч різних бізнес-програм, як платних так і безкоштовних. Серед найпопулярніших виділяють – 1С:Підприємство, BAS, ЕКАМ, МойСклад, LiteBox [4].

Програми 1С:Підприємство є двох видів – типові та галузеві. Типові мають розширений функціонал, який може задовільнити потреби підприємств різних

сфер діяльності, а галузеві спрямовані на конкретну частинку бізнесу, наприклад СТО, ресторани.

BAS ERP – сучасне прикладне рішення для розвитку бізнесу, має широкий функціонал. Підходить для великих підприємств зі складною структурою бізнес-процесів, у випадку необхідності може працювати під конкретну компанію [5].

1.3 Огляд сучасних СУБД та технологій, які використовуються при роботі з даними

1.3.1 Класифікація баз даних

Базою даних називають збір логічно взаємопов'язаних даних, організований відповідно до концептуальної структури, що описує характеристики цих даних і взаємозв'язки між відповідними об'єктами, підтримуючи одну чи декілька сфер застосування [6].

Об'єднавши велику кількість даних в одну єдину базу, отримуємо можливість виконувати безліч різних варіантів групування інформації – дані клієнтів, історію та стан замовлень, каталог товарів. Також, бази даних мають широке застосування для зберігання адміністративної інформації і спеціалізованих даних. Наприклад бази даних використовують: інтернет-сайти з великою кількістю інформації, каталоги та реєстри, автоматизовані системи обліку, лінгвістичні бази.

Основною перевагою баз даних є швидке внесення та використання необхідної інформації. Використовуючи спеціально призначені алгоритми для БД, можемо легко знайти потрібну інформацію за секунди. Завдяки взаємозв'язку інформації в БД при зміні даних в одному рядку, автоматично змінюються дані в інших рядках – це спрощує та пришвидшує роботу з інформацією [7].

За ступенем розподіленості бази даних класифікуються на розподілені та централізовані (зосереджені).

Централізована база даних – це такий тип бази даних, що зберігається, розміщується і обслуговується тільки в одному місці. Модифікація та керування

відбувається з місця розташування. Це місце може бути будь-якою системою бази даних або централізована комп’ютерна система. Централізовану базу даних переважно використовують організації та установи.

Переваги:

- Завдяки зберіганню всіх даних в одному місці, простіше отримати доступ та керувати даними.
- Оскільки, всі дані зберігаються в одному місці, то централізована БД має мінімальну надлишковість даних.
- Дешевша в порівнянні з іншими базами даних.

Недоліки:

- Великий трафік даних
- Якщо в системі виникне будь-який збій, то всі дані можуть бути втрачені, знищені.

Розподіленою базою даних, називають такий тип бази даних, який складається з декількох баз даних, взаємопов’язаних одна з одною і розподіляється за різним фізичним розташуванням. В цьому випадку можна керувати даними, які зберігаються в різних місцях. Зв’язок між базами даних в різних місцях розташуваннях, здійснюється за допомогою комп’ютерної мережі.

Переваги:

- Таку базу даних можна легко розширити;
- Доступ до розподіленої бази даних, можна отримати з різних мереж.
- В порівнянні з централізованою базою даних, розподілена БД є безпечніша.

Недоліки:

- Дуже дорога вартість бази даних
- Важко надати користувачеві одне ціле показання, оскільки вона розподілена за різним місцерозташуванням.

Порівняльну характеристику відмінностей між централізованою та розподіленою базою даних показано у таблиці 1.1.

Таблиця 1.1 – Відмінності між централізованою та розподіленою БД.

№	Централізована база даних	Розподілена база даних
1	Дані зберігаються, розміщуються і обслуговуються тільки в одному місці.	Складається з декількох баз даних, зв’язаних одна з одною і розташовані в різних місцях.
2	Більший час доступу до даних	Менший час доступу до даних
3	Простіше керування, редагування і резервне копіювання, оскільки всі дані знаходяться на одному місці	Керування, редагування та резервне копіювання відбувається дуже складно, оскільки вона розташовується на різних місцях.
4	У випадку будь-якого збою бази даних, користувачі можуть втратити доступ до бази.	Якщо виникне збій в одній базі даних, то користувачі отримують доступ до інших баз даних.
5	Недорога	Дуже дорога

Для вирішення задачі оберемо централізований тип БД, оскільки вона є дешевшою та спрощеною у використанні і підтримці. Завдяки регулярному створенню резервних копій, можна забезпечити надійність бази даних [8].

Основою будь-якої бази даних є модель даних. Модель даних складається зі структур даних та операцій над ними. За допомогою моделі даних можна визначити логічну структуру бази даних та визначити принцип збереження, організації даних та роботу з ними [9].

За моделлю даних розрізняють такі БД:

- Ієрархічні. Дані організовані в деревоподібну структуру. Зберігаються дані у вигляді набору полів, де кожне поле містить тільки одне значення. Записи пов’язані один з одним через зв’язки в відношеннях батько-нащадок. В цій моделі бази даних, кожний запис-нащадок, має одного батька, батько може мати кілька нащадків. Для того, щоб отримати дані поля, необхідно пройти через усе дерево, починаючи від кореня. Дана ієрархічна структура бази даних була

розроблена IBM на початку 1960-х років. Ієрархічні бази даних широко використовуються для створення продуктивних та доступних програм, зазвичай в банківських та телевізійних сферах.

- Мережеві. Мережеві бази даних зазвичай використовуються на великих цифрових комп'ютерах. Вони представляють собою ієрархічні бази даних, але мережевий вузол може мати зв'язок з декількома об'єктами.

Різниця між кожним нащадком в тому, що в них може бути більше одного батька. Дані в мережевій базі даних організовані за принципом «всі до всіх».

- Реляційні. Відношення між даними є реляційними, дані зберігаються в табличній формі стовпців і рядків. Кожен стовбець таблиці є атрибутом, а кожен рядок представляє запис. Мова структурованих запитів (SQL) – це стандартизована мова, яка використовується для запитів до СУБД, включаючи в себе вставлення, оновлення, видалення та пошук записів. Реляційні бази даних працюють з таблицями, у яких є унікальне поле, яке вказує кожен рядок. Ці поля можна використовувати для з'єднання однієї таблиці з іншою.

Реляційна база даних є найпопулярнішою і широко використовуваною базою даних. Серед найпопулярніших – Oracle, SQL Server, MySQL, SQLite.

- Об'єктно-орієнтовані. У цих моделях основним поняттям є об'єкт, інформація передається у вигляді об'єктів, як і в об'єктно-орієнтованих мовах програмування. Завдяки цьому взаємодія між БД та об'єктно-орієнтованою мовою програмування спрощується, і не потрібно прописувати об'єкти для доступу до даних бази [10].

Для виконання задачі у даній роботі було обрано реляційну базу даних, оскільки дана модель є поширеною, реляційні бази даних можна використовувати з мінімальними навиками, під дану модель наявна велика кількість СУБД, можна обрати оптимальний варіант використанням для системи, яка розробляється.

1.3.2 Сучасні системи управління базами даних

Зараз існує дуже багато різних систем управління базами даних. Розглянемо деякі з них:

PostgreSQL – це безкоштовна СУБД з відкритим кодом і необмеженими можливостями масштабування. Підтримує реляційні (SQL) і нереляційні (JSON) запити. Існує вже більше 20-ти років та має певну базу користувачів.

Особливістю PostreSQL є робота як зі структурованими (SQL), так і з неструктурованими (NoSQL) даними. Сама система побудована на ядрі, яке створювали безліч розробників. PostgreSQL має підхід, керований каталогами, який дає йому можливість розширюватися [11]. Він призначений не тільки, щоб зберігати інформацію для ідентифікації таблиць і стовпців, а дозволяє визначити типи даних, типи індексів та функціональні мови. PostgreSQL працює на всіх операційних системах, включаючи Linux, UNIX та Windows. Підтримує значну частину стандартну SQL та містить багато сучасних функцій, а здійснює роботу як з локальними та хмарними серверами.

Використання PostgreSQL підходить для малих та великих компаній, оскільки є повністю безкоштовним та масштабованим [12].

Oracle Database або Oracle RDBMS є найпотужнішою СУБД, яка може створювати програми будь-якої складності. У базі даних зберігається дуже велика кількість інформації. Одночасно з цією інформацією може працювати будь-яка кількість користувачів, при цьому не втрачаючи продуктивності системи. За допомогою засобів та функцій Oracle Database, можна безмежно масштабувати потужність та швидкість роботи сервера Oracle, а також додатків, просто додаючи нові вузли кластеру. Для цього не потрібно припиняти роботу додатків, а в разі виникнення неполадок окремих вузлів кластера програма працюватиме дальше.

СУБД Oracle працює майже на всіх ОС, без виникнення жодних проблем. Запити до даних доступні з будь-якого додатку, розробленого з використанням технологій Microsoft .NET, Visual Studio чи веб-застосунків. Для цього необхідні тільки справні бібліотеки, за допомогою яких можна підключатись до серверу БД Oracle. Найчастіше Oracle використовують у великих організаціях, де необхідно керувати великим обсягом даних.

Microsoft SQL Server є сучасною та потужною системою управління базами даних, розробником якої є компанія Microsoft. Вона включає в себе

великий набір інтегрованих служб по аналізу даних. Отримати доступ до даних, можна за допомогою будь-яких програм, створених за допомогою .Net чи VisualStudio. Основна мова для запитів – Transact-SQL. Використовується для роботи з базами даних для підприємств малого та великого масштабу. Використовуючи великий набір інтегрованих служб, можна: створювати запити, здійснювати пошук даних, синхронізацію та аналіз даних, формування звітів. Отже, за допомогою SQL Server можна створити надійну, просту у використанні, продуктивну платформу, яка задовільняє всі вимоги під час роботи з даними. Підходить для комерційних корпорацій, яка користується продуктами Microsoft та яким необхідна швидкість та надійність [13].

MySQL, є найпопулярнішою системою управління базами даних завдяки своїй надійності, продуктивності та простоті у використанні. Ця СУБД містить відкритий код SQL, розробляється, розповсюджується та підтримується корпорацією Oracle. Програмне забезпечення сумісне більш ніж з 20 платформами (Windows, Linux, MacOS, IRIX та інші), забезпечує гнучкість використання в різних операційних системах. ПЗ забезпечує велику бібліотеку інструментів, послуги підтримки та консультацій для своїх користувачів. Програмним забезпеченням називається СУБД, яка містить великі обсяги даних інформації, використовується для додавання, доступу, та обробки даних, які зберігаються в БД, керує реляційними БД, що зберігаються в таблицях. Все це забезпечує швидкість і простоту в роботі. ПЗ розробляється на основі політики ANSI/ISO SQL, яка є стандартною мовою запитів. Оскільки ПЗ з відкритим вихідним кодом, то воно доступне всім для використання.

Сервер баз даних MySQL простий у встановленні та використанні, одночасно можуть працювати багато користувачів, у таблицях доступна велика кількість рядків, швидке виконання команд, проста та ефективна система захисту [14].

Для виконання завдання обираємо СУБД MS SQL Server, оскільки вона є простою у використанні, безкоштовною і містить достатню кількість теоретичного матеріалу.

1.4 Обґрунтування вибору СУБД та засобів розробки програмного забезпечення

Для створення програмного забезпечення з автоматизації обліку товарів необхідно визначитись з засобами розробки. До засобів розробки належить: інтегроване середовище розробки (англ. IDE), вибір мови програмування, вибір програмного інтерфейсу створення програми (анг. API), вибір системи керування базами даних.

Інтегроване середовище розробки (англ. Integrated development environment, IDE) – це програмне забезпечення для побудови програм, яке поєднує загальні інструменти розробника в єдиний графічний інтерфейс користувача. IDE складається з текстового редактора вихідного коду, автоматизації збірки та налагоджувача [15].

У якості інтегрованого середовища розробки ПЗ для виконання поставленого завдання було обрано середовище Microsoft Visual Studio.

Microsoft Visual Studio – це інтегроване середовище розробки (IDE), для розробки графічного інтерфейсу комп'ютерних програм, консолі, веб-додатків, веб-програм, мобільних додатків, розробником якого є корпорація Microsoft. Використовуючи дану IDE, можна створювати керований код, редагувати та створювати власний код. Для розробки програмного забезпечення Microsoft, Visual Studio використовує різні платформи програмного забезпечення, серед них такі як Windows Forms, Windows store, Windows API. Підтримує популярні мови програмування, такі як C, C++, C#, JavaScript, Python і багато інших мов. Visual Studio доступна для ОС Windows та macOS [16].

.Net Framework – це платформа для розробки програмного забезпечення, для створення та запуску програм Windows, від корпорації Microsoft. Складається з інструментів розробника, мов програмування та бібліотек для побудови додатків, також для веб-сайтів, веб-сервісів та ігор. Розробники можуть вибирати різні мови програмування, доступних на платформі Microsoft .Net Framework. Найпоширенішими серед яких є Visual Basic .Net та C#.

.Net Framework містить набір стандартних бібліотек класів. Бібліотекою класів називається сукупність методів та функцій, які можуть бути використані для задачі. Програми, які побудовані на фреймворку .Net, можуть працювати на будь-якій платформі Windows, також .Net Framework має хороший механізм захисту, який допомагає у перевірці програм [17].

У якості мови програмування обрано C#, що є частиною платформи .Net. На даний час C# та .Net вважається найефективнішою технологією для розробників. Оскільки середовище .Net створене для розробки будь-якого застосунку Windows, а мова програмування C# – використовується у поєднанні з .Net Framework.

C# – це сучасна, об’єктно-орієнтована мова програмування загального призначення, розробником якої є корпорація Microsoft. Мова C# є однією з найпростіших мов для вивчення, заснована на мовах C та C++ [18].

Переваги мови C#:

- Об’єктно-орієнтована мова програмування, яка дозволяє створювати модульні додатки, що підтримуються та коди, що використовуються багаторазово.
- Кросплатформеність. Додаток написаний мовою C#, може використовуватись у будь-якій ОС, як Android, iOS чи Windows або хмарна платформа.
- Автоматичний збір сміття. У мові C# встановлена ефективна система, що збирає та видаляє непотрібні файли, які автоматично присутні в системі. Мова C# є ефективною в управлінні системи, оскільки вона не створює безладу в системі, і система не зависає під час виконання.
- Цілісність та сумісність. C# має потужну резервну копію даних, проблем з втратою даних не виникає.
- Легкість у розробці завдяки великому класу бібліотечних функцій та типів даних, які полегшують реалізацію багатьох функцій.

Завдяки цим перевагам, мова C# стала широко використовуваною мовою [19].

Для розробки графічного інтерфейсу було використано Windows Forms.

Windows Forms – це бібліотека класів графічного інтерфейсу користувача (GUI), яка входить до складу .Net Framework та створює програми для робочих столів Windows. Платформа для розробки Windows Forms підтримує широкий набір функцій розробки програм, включаючи елементи керування, графіку, прив’язку даних та введення користувачем. Щоб легко створювати настільні програми Windows Forms, у Visual Studio наявна можливість перетягувати та розміщувати візуальні елементи керування. За допомогою елемента управління DataGridView можна відобразити дані з бази даних [20].

Вибрано систему управління базами даних MS SQL Server.

MS SQL Server – це популярна реляційна система управління базами даних, розроблена корпорацією Microsoft. Будучи сервером БД, його основна функція – зберігати та отримувати дані відповідно до запитів інших програм.

Microsoft SQL простий у використанні, встановлюється за допомогою майстра налаштувань, пропонує зручний інтерфейс інсталяції. Завдяки автоматичному оновленню зменшуються витрати на обслуговування.

Функції стиснення та шифрування даних, забезпечують SQL-серверу високу продуктивність. База даних SQL Server є захищеною, для цього використовуються складні алгоритми шифрування. SQL Server складається з декількох складних функцій, за допомогою яких можна відновити втрачені чи пошкоджені дані [21].

Середовищем для роботи з SQL скриптами і внесенням змін в БД була використана програма SSMS (SQL Server Managmant Studio).

Microsoft SQL Server Management Studio (SSMS) – це вдосконалене середовище розробки, яке дозволяє налаштовувати, керувати та адмініструвати бази даних SQL Server. SSMS є дуже популярним і широко використовується розробниками та адміністраторами баз даних.

SSMS надає інструменти для налаштування, управління та адміністрування екземплярів Microsoft SQL Server, а також об’єднує ряд інструментів графічного та візуального дизайну та розширені редактори сценаріїв для спрощення роботи з SQL Server.

Функції Microsoft SQL Server Management Studio включають:

- object Exlorer, за допомогою якого можна переглядати та керувати об’єктами в SQL Server;
- template explorer, який створює та керує файлами тексту, які можна використати повторно для пришвидшення розробки запитів та сценаріїв;
- solution explorer, створює проекти, які використовуються для керування елементами адміністрування, такими як запити та сценарії.

Компоненти SSMS здійснюють налаштування комбінацій клавіш та перегляд сторінок властивостей, підключаються до екземплярів модуля баз даних та служб аналізу, засобів візуального дизайну проектування та інтерактивно створюють і налагоджують запити та сценарії [22].

1.5 Постановка завдання кваліфікаційної роботи

Для успішного функціонування сучасного підприємства необхідна автоматизація процесів. Для максимально ефективної роботи слід частково або повністю (при можливості) автоматизувати діяльність підприємства.

Метою кваліфікаційної роботи є розробка програмного забезпечення для обліку товарів в магазині автозапчастин «SERVANT AUTO».

Визначено такі функції програми, які повинні бути реалізовані:

- Облік товарів на складі;
- облік і зберігання даних про клієнтів та виробників;
- облік і ведення замовлень на всіх етапах виконання;
- ведення звітності по замовленням, клієнтах, виробниках та товарах;
- визначення загальної ціни товарів в замовленні.

Програму призначено для магазину автозапчастин. Її повинні використовувати продавці, що мають справу з товарами та замовленнями. На даний час, магазин використовує електронні таблиці для ведення інформаційної роботи. Продажі записуються в окремому документі і постійно перевіряються на відповідність в інших документах, такий спосіб забирає багато часу та трудових ресурсів, а також можливе допущення помилок або втрати даних.

Розробка власної системи для обліку товарів забезпечить зручний та швидкий доступ до необхідної інформації, час на оформлення замовлень зменшиться. Появиться можливість переглядати інформацію про замовлення і статус їх виконання, а також перегляд товару та його кількості.

Наявність єдиної бази усуне можливі помилки обліку товарів та дозволить виконувати роботу з будь-якого ПК, на якому буде встановлена дана програма. Для цього підійде програмне забезпечення, з реляційною базою даних MS SQL всередині, що дозволить забезпечити виконання усіх вищеназваних вимог максимально швидко та ефективно.

Отже, створення простої, інтуїтивно зрозумілої та функціональної програми для обліку автомобільних запчастин займе своє місце у даному магазині.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ТОВАРІВ

2.1 Аналіз предметної області

Для максимально точного аналізу предметної області, потрібно чітко розмежувати досліджувані об’єкти між собою. Головним об’єктом в магазині є товар. Замовлення на товар можна додавати, редагувати та видаляти.

Також, для аналізу предметної області було визначено такі сутності: користувач, продукт, виробник, замовлення та клієнти.

Сутність користувача може:

- переглядати наявні товари (запчастини);
- змінювати ціну, кількість запчастин;
- додавати та видаляти запчастини;
- переглядати клієнтів;
- редагувати, додавати чи видаляти дані про клієнта;
- переглядати замовлення;
- створювати нові замовлення;
- редагувати вже створені замовлення;
- змінювати стан замовлень (в обробці/завершені).

Сутність продукту містить інформацію про товари (запчастини):

- ціна;
- виробник;
- назва;
- артикул;
- кількість;

Аналізуючи вимоги до програмного забезпечення для обліку товарів, слід виявити функціональні вимоги. Функціональні вимоги – це вимоги до програмного забезпечення, для опису поведінки системи, її особливості та функції [23].

До функціональних вимог програмного забезпечення належить:

- можливість переглядати виробників;
- можливість додавати і видаляти виробника;
- можливість переглядати клієнтів;
- можливість редагувати, додавати і видаляти клієнтів;
- можливість переглядати товари;
- можливість змінювати ціну та кількість товару;
- можливість додавати і видаляти товар;
- можливість переглядати замовлення;
- можливість додавати та редагувати замовлення;
- можливість змінювати стан замовлення.

Для програмного забезпечення, що розробляється, визначаємо зовнішню сутність – користувач, в нашому випадку користувачем є працівник магазину, тобто продавець.

На основі аналізу функціональних вимог до програмного забезпечення, виділимо варіанти використання системи. Перелік варіантів використання та їх короткий опис показано в таблиці 2.1.

Таблиця 2.1 – Реєстр варіантів використання

Актор	Назва	Опис
Користувач	Перегляд виробників	Дає можливість додавати і видаляти виробника
	Перегляд запчастин	Дає можливість отримати інформацію про товар: кількість, ціну.
	Перегляд клієнтів	Відображає інформацію про клієнта: ПІБ, адрес, номер телефону
	Перегляд замовлень	Можливість переглянути детальну інформацію по замовленні, змінити стан виконання, чи створити нове замовлення

Продовження таблиці 2.1

Користувач	Замовлення	Дозволяє додати або видалити замовлення, а також підтвердити

Для наочного відображення можливостей користувача в системі, на основі таблиці 2.1, було створено діаграму варіантів використання.

Діаграма варіантів використання – це діаграма, що використовується для моделювання поведінки системи. Дана діаграма визначає взаємодію між роботою системи та її учасниками, тобто показує, як буде використовуватись програма [24].

Діаграма використання представлена на рисунку 2.1.

Рисунок 2.1 – Діаграма варіантів використання системи

Аналізуючи побудовану діаграму варіантів використання бачимо, що для сутності користувач доступні чотири варіанти використання: «Перегляд виробників», «Перегляд клієнтів», «Перегляд запчастин», «Перегляд замовлень».

Варіант використання «Перегляд виробників» включає розширення: додати нового виробника, видалити існуючого та здійснити експорт даних в CSV-файл.

Для варіанту використання «Перегляд клієнтів» існують розширення: додати нового клієнта, редагувати інформацію про клієнта, видалити клієнта та експортувати дані про клієнтів у CSV-файл.

Для варіанту використання «Перегляд запчастин» існують розширення: змінити ціну, змінити кількість товару, додати нові запчастини чи видалити існуючі, а також експорт всіх запчастин у CSV.

Варіант використання «Перегляд замовлень» містить такі розширення: створити нове замовлення, редагувати замовлення, скасувати чи завершити замовлення, експортувати наявні замовлення у CSV-файл. Також, включає «Перегляд» вже існуючих замовлень, а саме повну інформацію в замовленні, можливість додати товар до замовлення, підтвердити або видалити замовлення.

2.2 Реалізація таблиць БД

Проаналізувавши предметну область, проектуємо нашу базу даних. База даних, містить набір з різних типів сутностей. Сутністю в базі даних може бути будь-який реальний чи уявний об’єкт, який виділяють з предметної області. У кожному типі сутності міститься свій унікальний набір атрибутів, а сутність має власні значення для кожного атрибуту. Атрибут відображає властивості якими буде володіти об’єкт, що розглядається [25].

Також база даних володіє певним набором відношень. Відношення сутності – це зв’язок полів між двома таблицями, який створюється за допомогою відповідних атрибутів у кожній з двох таблиць.

Зв’язки між таблицями створюються за допомогою відповідних полів, які відображаються в обох таблицях. Ці поля називаються – реляційні ключі. Розрізняють два типи ключів: первинний і зовнішній (вторинний) ключ.

Первинний ключ – це унікальний ідентифікатор поля або набору полів зі значеннями, в межах однієї таблиці. Використовується для зв’язку таблиці з зовнішніми ключами інших таблиць [26].

Зовнішній (вторинний) ключ – це поле або набір полів таблиці, яке містить посилання первинного ключа в іншій таблиці.

Розрізняють такі типи взаємозв’язків між полями двох таблиць:

- один-до-одного – це коли один запис однієї таблиці пов’язаний з одним записом іншої таблиці;
- один-до-багатьох – це коли один запис таблиці пов’язаний з кількома записами в іншій таблиці. Даний тип взаємозв’язку є найпоширенішим;
- багато-до-одного – це коли декілька записів таблиці пов’язані з одним записом іншої таблиці;
- багато-до-багатьох – це коли кілька записів таблиці пов’язані з кількома записами іншої таблиці.

Процес організації даних в БД називають нормалізацією. Вона включає в себе створення таблиць та встановлення зв’язків між таблицями відповідно до певних правил, для того, щоб зробити базу даних більш гнучкою за рахунок видалення в ній надмірності.

Основні принципи нормалізації:

- поля в таблицях БД не повинні повторюватись;
- кожна таблиця повинна містити унікальний ідентифікатор (первинний ключ);
- для первинного ключа повинна бути достатня інформація про об’єкт таблиці;
- редагування чи заміна значень в полях однієї таблиці, не повинна впливати на інформацію у інших полях (окрім змін у ключі) [27].

На основі аналізу було розглянуто такі сутності як: запчастини, виробник, інформація для замовлення, замовлення, статус, клієнти.

Основною сутністю системи є таблиця з інформацією для замовлення в ній зберігається вся інформація про замовлення: посилання на номер замовлення та номер запчастини, кількість та загальну ціну. Структуру таблиці з інформацією для замовлення (OrderDetails) подано в таблиці 2.2.

Таблиця 2.2 – Структура таблиці OrderDetails

Назва	Тип даних
id_order	int
spare_part	int
quantity	int
total_price	money

В кожному замовленні міститься запчастина, інформація про яку зберігається в таблиці SpareParts. Вона складається з: назви запчастини, виробника, кількісті та ціни. Структуру таблиці SpareParts подано в таблиці 2.3. Дана таблиця пов’язана з таблицею OrderDetails за унікальним артикулом запчастин, це зроблено так, тому, що в одному замовленні може бути декілька запчастин, тип зв’язку між ними: «один-до-багатьох».

Таблиця 2.3 – Структура таблиці SpareParts

Назва	Тип даних
article	int
name	text
manufacturer_id	int
quantity	int
price	money

Інформація про виробника запчастини, зберігається в таблиці Manufacturers. Ця таблиця взаємодіє із таблицею SpareParts через зовнішній ключ, і має зв’язок «один-до-багатьох». Структура таблиці Manufacturers, що містить інформацію про виробника показана у таблиці 2.4.

Таблиця 2.4 – Структура таблиці Manufacturers

Назва	Тип даних
id_manufacturer	int
name_manufacturer	text

Інформація про замовлення, зберігається у таблиці Orders. Таблиця містить в собі: номер замовлення (первинний ключ), ідентифікатор клієнта (зовнішній ключ), дату, загальну ціну та статус виконання замовлення (зовнішній ключ). Дана таблиця взаємодіє з таблицею OrderDetails, тип зв’язку між ними «один-до-багатьох». Структуру таблиці з інформацією про замовлення показано в таблиці 2.5.

Таблиця 2.5 – Структура таблиці Orders

Назва	Тип даних
id_order	int
id_client	int
date	datetime
total_price	money
status	int

Замовлення здійснюють клієнти. Отже, нам необхідно отримати інформація про клієнта та зберегти її. У таблиці Clients міститься: ідентифікатор клієнта, ім’я клієнта, адреса, номер телефону та електронна пошта. Взаємодіє із таблицею Orders за допомогою зовнішнього ключа та має зв’язок «один-до-багатьох». Структуру таблиці Clients подано у таблиці 2.6.

Таблиця 2.6 – Структура таблиці Clients

Назва	Тип даних
id	int
fullname	text

Продовження таблиці 2.6

fulladress	text
phone_number	text
e_mail	text

Перевірка статусу замовлення зберігається в таблиці Status. Дана таблиця взаємодіє з таблицею OrderDetails через зовнішній ключ та має тип зв’язку «один-до-багатьох». Структуру таблиці для перевірки статусу замовлень показано в таблиці 2.7.

Таблиця 2.7 – Структура таблиці Status

Назва	Тип даних
status_code	int
status_name	text

Здійснивши опис всіх таблиць та зв’язків між ними, створюємо приклад схеми бази даних. Схема бази даних – це логічне представлення вигляду усієї бази даних. Схема бази даних визначає сутності та взаємозв’язок між ними. Вона не містить конкретних даних, а тільки описує форму даних і відношення до інших таблиць [28].

Схему БД, створено за допомогою ПЗ Microsoft SQL Server Management Studio та показано на рисунку 2.2.

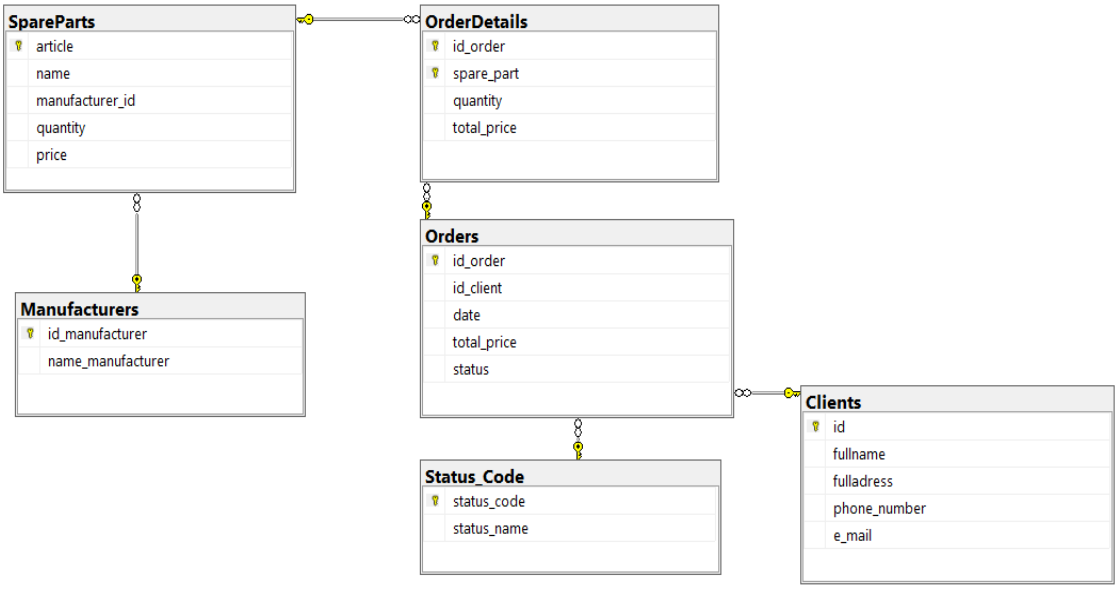

Рисунок 2.2 – Схема бази даних

ER Diagram (Entity Relationship Diagram, ERD) – це діаграма, що показує взаємозв'язок наборів сутностей, які зберігаються в базі даних. ER-діаграми показують повну логічну структуру бази даних. Створюються ці діаграми, на основі трьох основних понять: сутності, атрибутів та відносин.

Вигляд спрощеної ER-діаграми бази даних подано на рисунку 2.3.

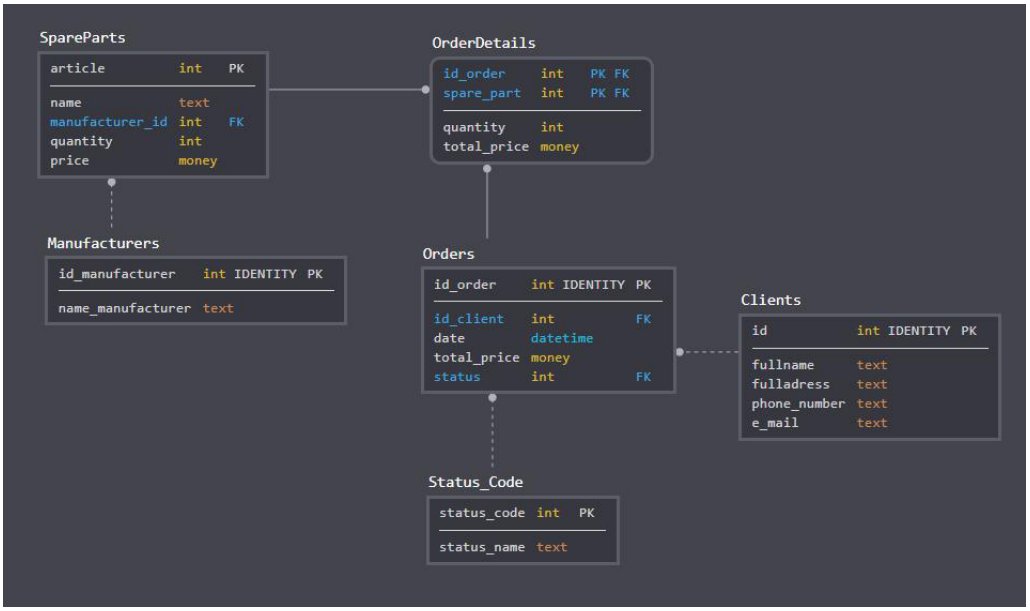

Рисунок 2.3 – Спрощена ER-діаграма бази даних

ER Model (Entity Relationship Model) – це схема моделі даних високого рівня, що допомагає аналізувати вимоги до даних для створення добре

продуманої бази даних. ER-модель представляє реальні сутності та взаємозв'язки між ними. Повне представлення ER-моделі створеної бази даних показано на рисунку 2.4

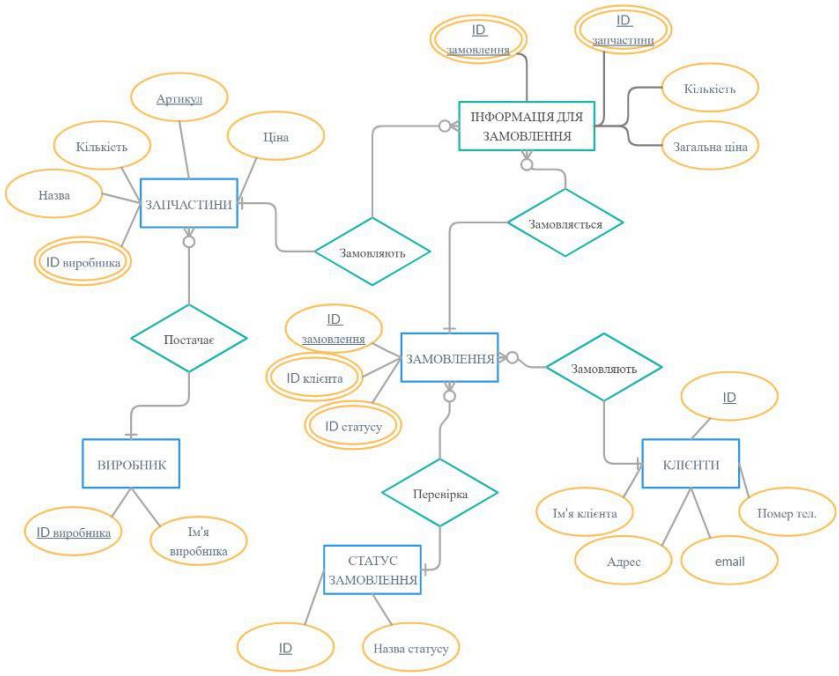

Рисунок 2.4 – Повне представлення ER-моделі даних

Для графічного представлення сутностей на ER-діаграмах використовують такі фігури: прямокутники для представлення сутностей, еліпс для визначення атрибутів, ромб для представлення відносин, подвійні еліпси для представлення багатозначних атрибутів. Для представлення первинного ключа атрибути підкреслюють [29].

2.3 Розробка бази даних

У середовищі Microsoft Visual Studio можна з'єднуватись та взаємодіяти із БД. Виконаємо підключення до бази даних MS SQL Server, ввівши наступні параметри: ім'я користувача PETRYK та назву сервера SQLEXPRESS.

З'єднавшись із базою даних, створюємо таблиці за допомогою запитів CREATE TABLE. Створена такі таблиці: Clients, Manufacturers, OrderDetails,

Orders, SpareParts, StatusCode. SQL-запити для створення таблиць подано у додатку А.

Після створення усіх таблиць, до бази даних пишемо необхідні запити для додавання, редагування, вибірки чи об'єднання даних. Опис деяких запитів до бази даних подано у лістингу 2.1

Лістинг 2.1 – Запити до бази даних

```
INSERT INTO Clients (fullname, fulladress, phone_number, e_mail)
VALUES('Іванов Ваня', 'вул.Карпенка', '+380986500362',
'vania@gmail.com') // Додавання даних про клієнта

UPDATE Clients SET fullname=@fullname, fulladress=@fulladress,
phone_number=@phone_number, e_mail=@e_mail WHERE id=@id //
Редагувати дані клієнта

SELECT * FROM Clients WHERE id=@id // Вибрати клієнта за
ідентифікатором

SELECT * FROM SpareParts LEFT JOIN Manufacturers ON
manufacturer_id = id_manufacturer // Об'єднання даних, що
перетинаються умовою

DELETE FROM SpareParts WHERE article = @article // Видалення
запчастини по артикулу
```

Для бази даних було створено 4 процедури: AddToOrder, RemoveFromOrder, UpdateTotal, CancelOrder. А також 1 тригер: TriggerOrderDetails.

Збережена процедура – це підготовлений SQL код, який можна зберегти і використовувати при необхідності для виконання завдання. Користувач може викликати процедуру, процедура може приймати вхідні дані в якості параметрів, а також збережені процедури можуть повертати значення.

Тригер – це спеціальна збережена процедура, яка запускається автоматично, коли в базі даних відбуваються різні події (update, input, delete). Тригери схожі на обробник подій, який вони запускають під час конкретної події. Тригери не можуть вводити дані, а також не можуть повертати значення [30].

Процедура AddToOrder додає запчастину до замовлення, підраховує ціну запчастини та оновлює загальну ціну в таблиці Orders. Код створення цієї процедури подано у лістингу Б.1.

Процедура RemoveFromOrder видаляє запчастини з замовлення, підраховує ціну та оновлює ціну в таблиці Orders. Код створення даної процедури показано у лістингу Б.2

У процедурі UpdateTotal оновлюється загальна ціна, яка викликається в процедурах AddToOrder та RemoveFromOrder. Код виконання поданий у лістингу Б.3.

Процедура CancelOrder змінює статус замовлення на скасований, під час виклику повертає увесь товар на склад, але з самого замовлення товар не видаляється. Код створення цієї процедури поданий у лістингу Б.4.

Тригер TriggerOrderDetails оновлює кількість товарів на складі, коли товар додається чи видаляється з замовлення. Код створення даного тригера подано у лістингу Б.5.

2.4 Тестування розробленого програмного забезпечення

Система автоматизації обліку товарів в магазині автозапчастин реалізується у вигляді програми, яка створена з використанням Windows Forms. Для взаємодії використовуються форми із елементами керування. Загалом було створено 12 форм для взаємодії з користувачем. Реалізацію цих форм наведено у додатку В.

Дану програму будуть використовувати працівники магазину для роботи з замовленнями, товарами та клієнтами. Отже, інтерфейс користувача програми повинен бути швидким, інтуїтивним та простим у використанні, для максимальної швидкодії.

У програмі реалізовано експорт таблиць баз даних у CSV-файл. Формат CSV використовується для того, щоб зберігати таблиці у текстових файлах. Всі записи таблиць з бази даних, перетворюються в рядки. Стовпці таблиць розділяються комою або крапкою з комою – це спеціальні символи CSV формату. Експорт у даний формат було обрано через ряд переваг: текстові файли досить прості та швидко відкриваються, вони доступні для читання на будь-яких пристроях та в будь-якому середовищі без додаткових інструментів.

Відразу після запуску програми перед користувачем відображається вікно головної форми. На даній формі користувач може обрати необхідну дію, а саме: перегляд виробників, перегляд запчастин, перегляд клієнтів та перегляд замовлень. Вікно головної форми показано на рисунку 2.5.

Рисунок 2.5 – Вікно головної форми програми

Обравши «Перегляд виробників» маємо можливість переглянути усіх виробників, а також можемо додати нового чи видалити вже існуючого виробника. Доступний експорт даних про виробника у CSV-файл. Вікно форми «Перегляд виробників» подано на рисунку 2.6.

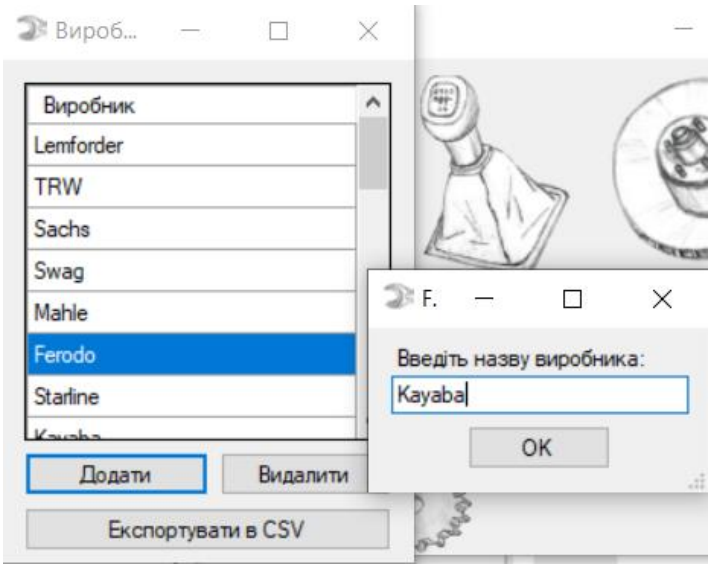

Рисунок 2.6 – Вікно форми виробники

Вибравши «Перегляд запчастин» відкривається віконна форма з запчастинами. На даній формі відображаються всі наявні запчастини, їх кількість на складі та ціна. Вікно даної форми відображено на рисунку 2.7.

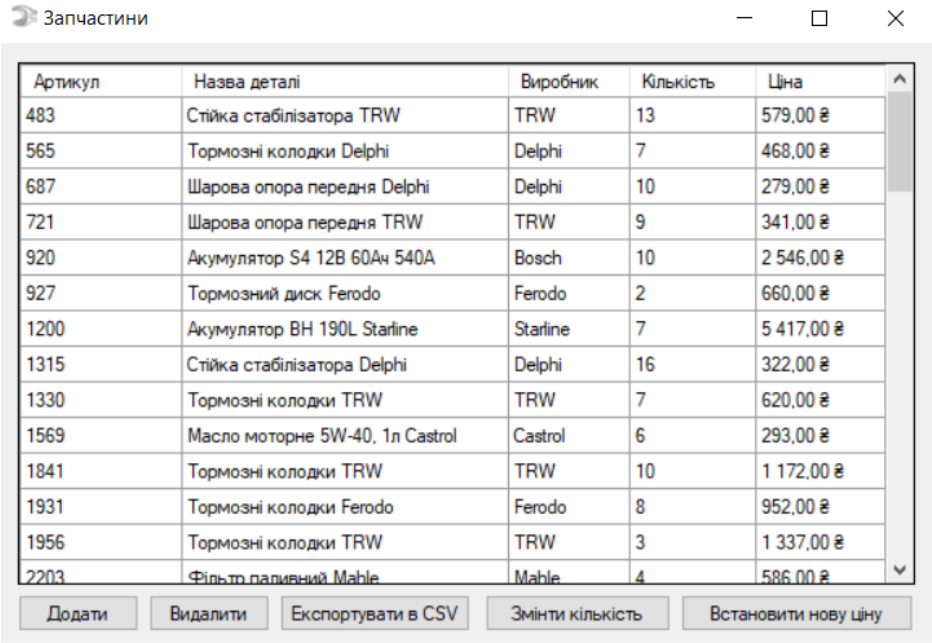

Рисунок 2.7 – Вікно форми запчастини

З рисунку 2.7 бачимо, що є можливість додати нові запчастини на склад або видалити зі складу. Також реалізовано можливість змінювати кількість запчастин і встановити нову ціну, експортувати дані в CSV-файл. Вигляд форми для додавання нової запчастини показаний на рисунку 2.8.

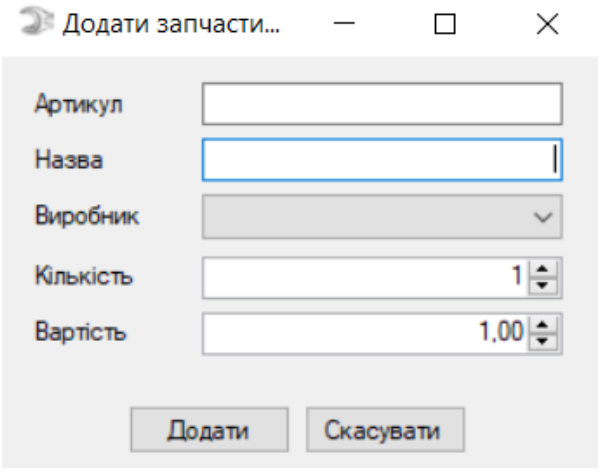

Рисунок 2.8 – Вікно форми для додавання запчастини

Для того, щоб змінити кількість запчастини створено окреме вікно форми. На цій формі реалізовано випадаючий список з необхідною дією: додати до кількості, відняти від кількості чи встановити нову кількість. На рисунку 2.9 показано вікно форми для зміни кількості запчастин.

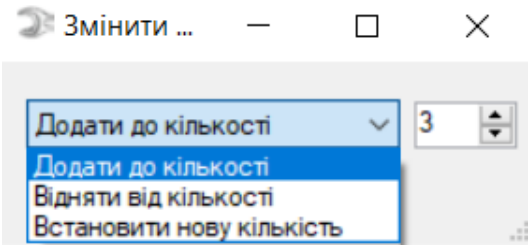

Рисунок 2.9 – Форма для зміни кількості

У вікні «Перегляд клієнтів» доступна можливість переглянути базу клієнтів. Про клієнта зберігається така інформація: ім’я, адреса, номер телефону та електронна адреса. На цій формі можна додати нового клієнта, редагувати або видалити вже існуючого, а також реалізована можливість експорту у формат CSV. Форма для перегляду інформації про клієнтів показана на рисунку 2.10.

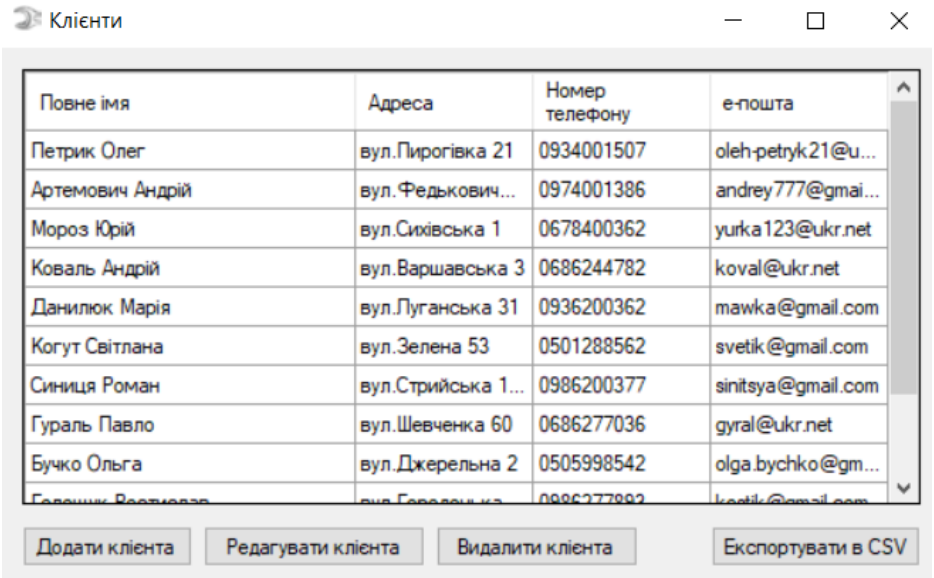

Рисунок 2.10 – Вікно форми з інформацією про клієнтів

Форма «Перегляд замовлень» відображає інформацію про замовлення, а саме: ім’я замовника, дату та час створення замовлення, загальну вартість та

статус виконання замовлення. Вікно форми перегляду замовлень показано на рисунку 2.11.

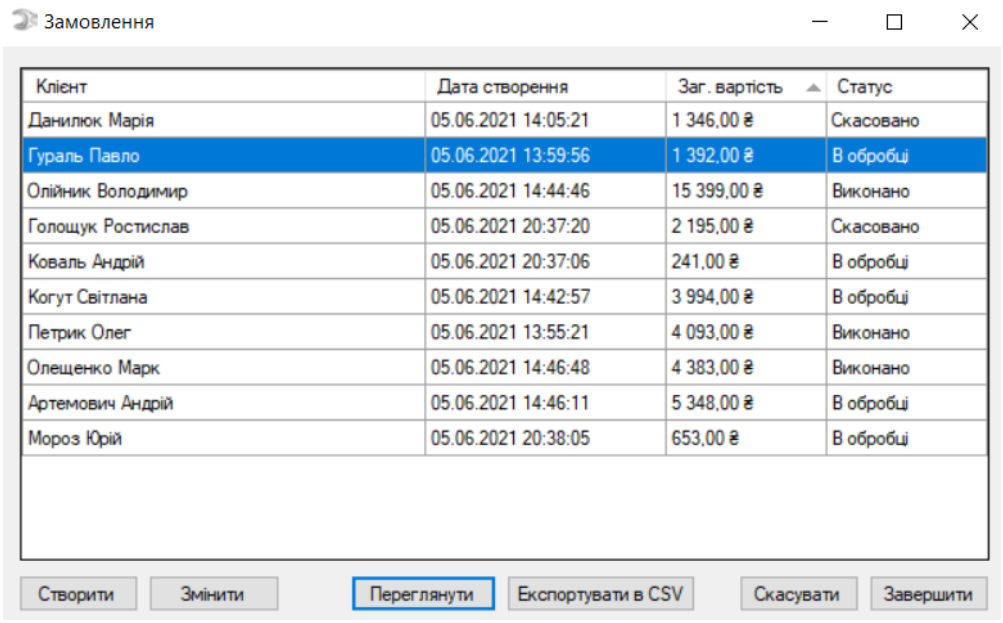

Рисунок 2.11 – Вікно перегляду замовлень

Функціонал форми дозволяє створити нове замовлення або редагувати вже існуюче, а також переглянути детальну інформацію про запчастини у замовленні. Вікно форми для перегляду запчастин у замовленні показано на рисунку 2.12.

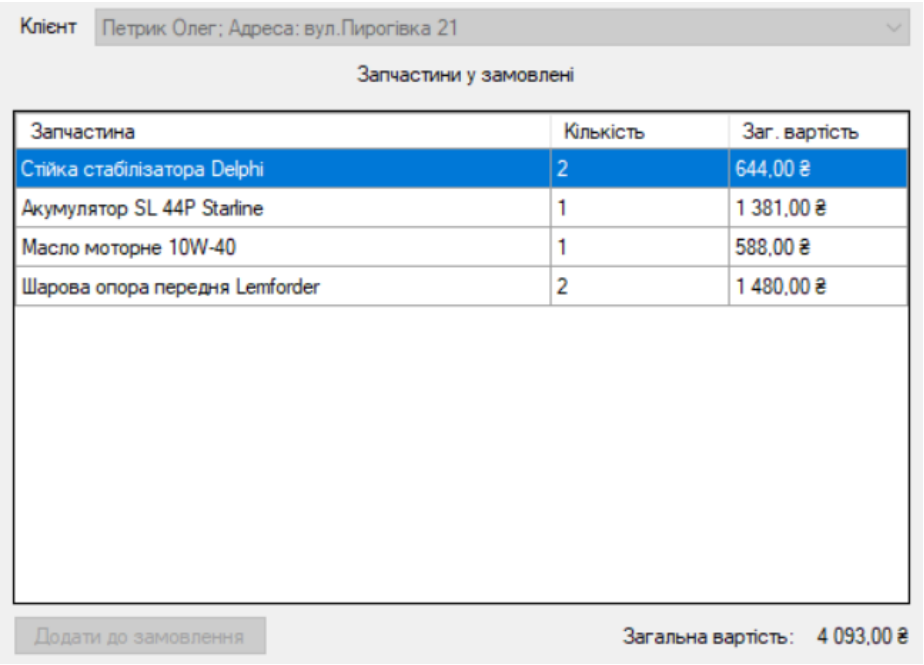

Рисунок 2.12 – Перегляд запчастин у замовленні

Також реалізовано експорт таблиці даних про замовлення у CSV-файл. Замовлення може перебувати у 3 статусах: в обробці, виконано та скасовано. У разі скасування усі запчастини будуть повернуті на склад. Експортований CSV-файл показаний на рисунку 2.13.

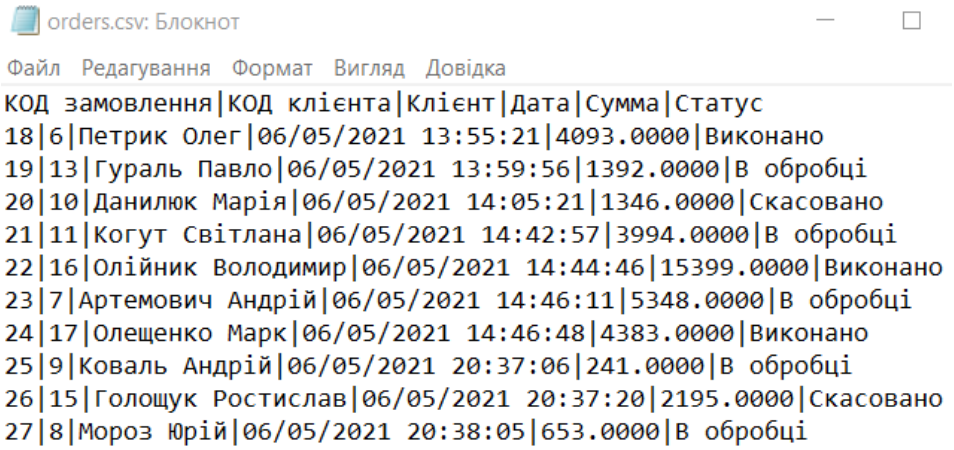

Рисунок 2.13 – Перегляд замовлень у CSV-файлі

Тестування програми успішно завершено. Дана програма має простий та функціональний інтерфейс і повністю відповідає вимогам завдання на розробку. Таким чином дане програмне забезпечення для обліку товарів може успішно використовуватись та бути корисним в магазині автозапчастин «SERVANT AUTO». В подальшому майбутньому програму можна розширити шляхом збільшення бази даних та новими функціональними можливостями програми.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Маркетингова діяльність магазину

Маркетингова діяльність підприємства являє собою творчу управлінську діяльність, завдання якої полягає в розвитку ринку товарів, послуг і робочої сили шляхом оцінки потреб споживачів, а також у проведенні практичних заходів для задоволення цих потреб. За допомогою цієї діяльності координуються можливості виробництва і розподіл товарів і послуг, а також визначається, які кроки необхідно зробити, щоб продати товар або послугу кінцевому споживачеві.

Маркетинг підприємства ґрунтується на таких принципах:

- принцип розуміти споживача, заснований на врахуванні потреб і динаміки ринкової кон'юнктури. Бізнес неможливий, якщо фірма орієнтована лише на прибуток, а не на врахування запитів споживача;
- принцип боротьби за споживача (клієнта). Суть цього принципу – боротьба за споживача, а не збут товарів. Товари і послуги в цьому випадку є лише засобом для досягнення мети, а не самою метою;
- принцип максимального пристосування виробництва до вимог ринку ставить виробництво товарів і надання послуг у функціональну залежність від запитів ринку та потребує виробляти товари в асортименті й обсязі, необхідних для споживача [31].

Для досягнення мети на ринку слід розробити стратегію маркетингу. Стратегія ґрунтується на аналізі стану підприємства, складу та обсягу продукції, яка буде пропонуватися на ринку, розробці нової продукції, форми та методи реклами, доставки і збуту продукції, ціни на продукцію [32]. Під час визначення стратегії маркетингу розрізняють такі види маркетингової діяльності: продуктовий маркетинг, виробничий маркетинг, збутовий маркетинг та маркетинг споживчого типу.

Продуктовий маркетинг – використовує сильні сторони технологій та аналізує потенційний попит споживачів. Для продуктів високої якості не потрібно докладати додаткових зусиль для реклами, збуту, реалізації [33].

Виробничий маркетинг – виготовлення продукції вважають основним способом досягнення комерційного успіху на ринку. Орієнтується на отримання доходу за рахунок збільшення об’ємів виробництва та зниження собівартості продукції.

Збутовий маркетинг – вид маркетингу, який користується системою методів реалізації і послуг. Метою якого є отримання швидкого прибутку за рахунок збільшення обсягу продажу, бажання оновити асортимент товарів та пришвидшити товарообіг.

Маркетинг споживчого попиту – орієнтується на постійний процес задоволення потреб споживачів. Основна увага приділяється на відмінності потреб різних груп споживачів та на зміну уподобань. Головною метою цього виду є підвищення конкурентоспроможності [34].

Маркетингова діяльність спрямована на те, щоб достатньо обґрунтовано, спираючись на запити ринку, встановлювати конкретні стратегічні цілі, шляхи їх досягнення та реальні джерела ресурсів господарської діяльності, визначати асортимент і якість продукції, її пріоритети, оптимальну структуру виробництва і бажаний прибуток.

Одним з найважливіших в плануванні маркетингу є ціноутворення. Ціноутворення – це процес вироблення цінової політики, яка полягає у встановленні і зміні ціни залежно від ситуації на ринку, що дозволяє зберігати певну частку ринку та отримувати прибуток. Ціна товару формується з:

- собівартості продукції;
- ціни конкурентів на аналогічний товар;
- унікальних властивостей товару;
- ціни, яка визначається попитом на певний товар.

На основі собівартості зазвичай встановлюється мінімально можлива ціна товару, відповідна найменшим витратам виробництва. На базі аналізу цін конкурентів визначається середній рівень цін на товар. Максимально можлива

ціна встановлюється для товарів, які відзначаються високою якістю або унікальними властивостями. Ціни, що визначаються попитом або кон'юнктурою ринку даних товарів, можуть коливатися в широкому діапазоні – від мінімальних (які покривають витрати виробництва і навіть нижче за витрати) до максимальних (що забезпечують максимальний прибуток).

Важливим елементом плану маркетингу є схема розподілу товарів, тобто організація каналів збуту. Канал збуту – шлях, яким товари рухаються від виробника до спожива. Рівень каналу збуту – це будь-який посередник, що виконує ту або іншу роботу з наближення товару до кінцевого споживача [35].

До плану маркетингу належить ще один розділ – реклама. Реклама – форма розповсюдження інформації про фірму та її товар, послуги, ідеї, через засоби масової інформації, а також з використанням прямого маркетингу. Головною функцією реклами є індивідуалізація продукту. Розрізняються зовнішню рекламу, рекламу на місці продажу, друковану рекламу та рекламу у інтернеті.

Під час формування маркетингової діяльності для розробки програмного забезпечення обліку товарів в магазині важливим є скласти алгоритм, за яким буде виконуватись маркетингова діяльність. Якщо використовувати ці етапи, то можна досягти ефективного функціонування маркетингової діяльності обліку товарів магазину та поставлених завдань. Правильне врахування принципів щодо управління маркетингом допоможе збільшити кількість клієнтів, за рахунок чого збільшити продажі, тобто дохід.

3.2 Психофізіологічне розвантаження для працівників

Праця – це сукупність фізіологічних та психічних процесів, які спонукають, програмують і регулюють діяльність людини. Психологічні можливості користувача комп’ютера не є постійними, вони залежать від інформаційного перевантаження, високого темпу роботи, емоційного стану людини. На це впливають конфліктні ситуації, виробничі невдачі, образи з боку керівництва чи колег, в результаті чого обсяг уваги різко знижується,

порушується пам’ять. Оператор забуває послідовність дій, неправильно оцінює ситуацію, припускається грубих помилок [36].

Охорона праці забезпечує збереження працівниками високого рівня працездатності. Під працездатністю розуміють потенційні можливості людини для виконання трудової діяльності протягом заданого часу з певною ефективністю. Величиною, зворотною до працездатності, є стомлення. Стомлення – це фізіологічні зміни в організмі працюючого, викликані витратою енергії в процесі трудової діяльності. Процеси, що виникають під час стомлення, впливають на свідомість працівника у вигляді відчуття втоми. Втома – це сукупність тимчасових змін у фізіологічному та психологічному стані людини, які з’являються внаслідок напруженої чи тривалої праці і призводять до погіршення її кількісних і якісних показників, нещасних випадків [37].

Важливою задачею ефективної організації трудового процесу є запобігання професійному стресу. Професійний стрес – це феномен, що виражається у фізіологічних і психологічних реакціях на складну робочу ситуацію. Серед виробничих стрес-факторів виділяють:

- фізичні (вібрація, шум, забруднена атмосфера);
- фізіологічні (змінний графік, відсутність режиму харчування);
- соціально-психологічні (конфлікт ролей, перевантаження або недовантаження працівників, міжособистісні конфлікти, висока відповідальність, дефіцит часу);
- структурно-організаційні («організаційний стрес»).

Стрес – це реакція нервової системи, стан напруження, який виникає внаслідок дії сильного подразника. Найгіршим проявом професійного стресу для працівника є професійне вигорання. Професійне вигорання – це стан фізичного, емоційного або мотиваційного виснаження, що характеризується порушенням продуктивності роботи та втомою [38].

Професія програміста та інших фахівців IT-технологій пов'язана з колосальним розумовим напруженням. Однак мозок не може нескінченно зберігати і переробляти дані практично не втрачаючи продуктивності.

Для психофізіологічного розвантаження працівників ІТ-компанії з обладнують свої офіси кімнатами відпочинку та лаунж-зонами. Наприклад, працівники компанії Inventionland працюють в казкових декораціях гігантського гоночного треку, реалістичного піратського корабля, на палубі якого розміщуються комп'ютерні столи. Робочі місця співробітників компанії Google в Цюріху нагадують гігантські вулики. Роботодавці створюють можливість релаксувати, обладнавши басейни, ігрові кімнати та спортзали. Адже, активний відпочинок відмінно бадьорить, розганяє кров, додає сил. Такий підхід дозволяє зняти надлишкове психофізіологічне перевантаження, підвищити працездатність центральної нервової системи [39].

Важливим є створення у приміщеннях та на територіях підприємств спеціальних візуальних комфортних умов та забезпечення вимог виробничої естетики, дотримання норм рівнів виробничого шуму та акустичної тиші за межами офісу. Для психофізіологічного розвантаження в офісних приміщеннях та кабінетах важлива наявність функціональної музики, яка сприяє попередженню перевтоми і підтриманню необхідного рівня розумової працездатності фахівців комп’ютерної галузі [40].

Психофізіологічне розвантаження є важливим, оскільки впливає на розумову працездатність, для працівників сфери інформаційних технологій, у тому числі для працівників які розробляють програмне забезпечення для обліку товарів в магазині. За допомогою розглянутих методів можна зняти надлишкове психофізіологічне перенавантаження для працівників програмного забезпечення та підвищити працездатність нервової системи. Всі наведені заходи щодо вдосконалення охорони праці фахівців ІТ-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства.

ВИСНОВКИ

У даній кваліфікаційній роботі освітнього рівня «Бакалавр» було досягнуто основної мети дослідження, а саме спроектовано та розроблено програмне забезпечення для обліку товарів в магазині автозапчастин «SERVANT AUTO». Дане програмне забезпечення для обліку товарів в магазині розроблене з використанням технології .NET та мови програмування C# у середовищі MS Visual Studio з використанням СУБД SQL Server.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

- Розглянуто характеристику об’єкта дослідження.
- Проаналізовано необхідність автоматизації складського обліку товарів та існуючі додатки для ведення обліку в малому і середньому бізнесі.
- Проведено огляд сучасних СУБД та технологій, що використовуються при роботі з даними та обґрунтовано вибір СУБД і засобів розробки ПЗ, сформовано постановку завдання кваліфікаційної роботи.

В другому розділі кваліфікаційної роботи:

- Розглянуто аналіз предметної області та реалізовано таблиці баз даних для обліку товарів в магазині.
- Спроектовано та розроблено базу даних для обліку товарів в магазині автозапчастин, здійснено опис їх взаємодії та принципів роботи;
- Протестовано програмне забезпечення для обліку товарів в магазині автозапчастин «SERVANT AUTO.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено питання маркетингової діяльності магазину та досліджено питання охорони праці для психофізіологічного розвантаження працівників.

Результатом кваліфікаційної роботи є розроблене програмне забезпечення для обліку товарів в магазині автозапчастин «SERVANT AUTO», яке має функціональний та простий інтерфейс і повністю відповідає заданим вимогам на розробку. Дане ПЗ може використовуватись та бути корисним у різних сферах, де виконується контроль обліку товарів. Програма може бути розширена шляхом збільшення бази даних та новими функціональними можливостями.

ПЕРЕЛІК ДЖЕРЕЛ

1. Боака Б. В. Електронний документообіг / Б. В. Боака. // Діджиталізація науки як виклик сьогодення: матеріали міжнародної студентської наукової конференції. – Київ, Україна: Молодіжна наукова ліга, 2020. – С. 35–37.

2. Пошаговая инструкция автоматизации складского учета товаров: что такое и как работает [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.ekam.ru/blogs/pos/avtomatizatsiya-skladskogo-ucheta-tovarov.

3. Автоматизація складського обліку [Електронний ресурс] – Режим доступу до ресурсу: http://terkon.com.ua/avtomatuz_sklada.

4. Рейтинг программ складского учета [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://onlinekassa.pro/obzor-sistem-skladskogo-ucheta/.

5. Бізнес-програми для обліку та автоматизації [Електронний ресурс] – Режим доступу до ресурсу: https://a4.com.ua/biznes-programi-obliku-avtomatizatsii/.

6. Information technology — Vocabulary [Електронний ресурс] – Режим доступу до ресурсу: https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en.

7. Герасимик І. Що таке база даних? [Електронний ресурс] / Іван Герасимик – Режим доступу до ресурсу: http://apeps.kpi.ua/shco-take-basa-danykh.

8. Difference between Centralized Database and Distributed Database [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/difference-between-centralized-database-and-distributed-database/.

9. Види моделей даних [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://studfile.net/preview/7075747/page:2/.

10. What is the classification of databases? [Електронний ресурс] – Режим доступу до ресурсу: https://www.quora.com/What-is-the-classification-of-databases.

11. Tobin D. Which Modern Database Is Right for Your Use Case? [Електронний ресурс] / Donal Tobin. – 2020. – Режим доступу до ресурсу: https://www.xplenty.com/blog/which-database/.

12. PostgreSQL - Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/postgresql/postgresql_overview.htm.

13. Бази даних [Електронний ресурс] – Режим доступу до ресурсу: http://lib.mdpu.org.ua/e-book/vstup/L5.htm#L51.

14. MYSQL Database Management System [Електронний ресурс] – Режим доступу до ресурсу: https://diatomenterprises.com/technologies/mysql-database-management-system/.

15. What is an IDE? [Електронний ресурс] – Режим доступу до ресурсу: https://www.redhat.com/en/topics/middleware/what-is-ide.

16. Introduction to Visual Studio [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/introduction-to-visual-studio/.

17. What is .NET Framework? Explain Architecture & Components [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/net-framework.html.

18. Samual S. C# Language advantages and applications [Електронний ресурс] / Sam Samual. – 2018. – Режим доступу до ресурсу: https://www.tutorialspoint.com/Chash-Language-advantages-and-applications.

19. What Is C# Language, Advantages & Features Of C# Language [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.codexoxo.com/advantages-c-sharp-language/.

20. Desktop Guide (Windows Forms .NET) [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-5.0.

21. All About SQL Server: Advantages, Best Practices, and Tools [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.tek-tools.com/database/sql-server-best-practices-and-tools.

22. Sirkin J. Microsoft SQL Server Management Studio (SSMS) [Електронний ресурс] / Jessica Sirkin – Режим доступу до ресурсу: https://searchsqlserver.techtarget.com/definition/Microsoft-SQL-Server-Management-Studio-SSMS.

23. Functional and Nonfunctional Requirements: Specification and Types [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/.

24. Use-case diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case.

25. Поняття ER-моделі. Поняття сутності (entity). Атрибути. Види атрибутів [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.bestprog.net/uk/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ua/.

26. Database Relationships [Електронний ресурс] – Режим доступу до ресурсу: https://www.knowitallninja.com/lessons/database-relationships/.

27. Основні поняття реляційних БД: нормалізація, зв’язок та ключі [Електронний ресурс] – Режим доступу до ресурсу: https://bondarenko.dn.ua/osnovni-ponyattya-relyatsijnih-bd-normalizatsiya-zv-yazok-ta-klyuchi/.

28. Kopecky C. What are database schemas? 5 minute guide with examples [Електронний ресурс] / Christina Kopecky. – 2020. – Режим доступу до ресурсу: https://www.educative.io/blog/what-are-database-schemas-examples.

29. ER Diagram: Entity Relationship Diagram Model | DBMS Example [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/er-diagram-tutorial-dbms.html.

30. Himanshu S. Difference between stored procedure and triggers in SQL [Електронний ресурс] / Shriv Himanshu. – 2020. – Режим доступу до ресурсу: https://www.tutorialspoint.com/difference-between-stored-procedure-and-triggers-in-sql.

31. Маркетингова діяльність та її основні види [Електронний ресурс] – Режим доступу до ресурсу: https://studme.com.ua/13761106/ekonomika/marketingovaya_deyatelnost_osnovnye_vidy.htm

32. Жарська І. О. Практикум з маркетингу / І. О. Жарська. – Одеса: Атлант, 2016. – 263 с.

33. Види сучасного маркетингу [Електронний ресурс] – Режим доступу до ресурсу: https://library.if.ua/book/22/1664.html.

34. Види маркетингу та їх характеристика [Електронний ресурс] – Режим доступу до ресурсу: http://www.info-library.com.ua/books-text-9567.html.

35. Карпенко Н. В. Маркетингова діяльність підприємств / Н. В. Карпенко. – Київ: Центр учбової літератури, 2016. – 252 с.

36. Желібо Є. П. Безпека життєдіяльності / Є. П. Желібо, І. С. Сагайдак. – Ірпінь: Університет державної фіскальної служби України, 2020. – 256 с.

37. Негативний вплив втоми і перевтоми на безпеку праці. [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: http://surl.li/unew.

38. Яскілка В. Я. Охорона праці в галузі / В. Я. Яскілка, М. З. Олійник. – Тернопіль: ТНТУ імені Івана Пулюя, 2016. – 56 с.

39. Почти как у Google: чем удивляют офисы украинских IT-компаний [Електронний ресурс] – Режим доступу до ресурсу: https://lifestyle.segodnya.ua/lifestyle/fun/pochti-kak-u-google-chem-udivlyayut-ofisy-ukrainskih-it-kompaniy--764025.html.

40. Охорона праці в галузі / П. С.Атаманчук, В. В. Мендерецький, О. П. Панчук, Р. М. Білик. – Київ: Центр учбової літератури, 2017. – 324 с.

ДОДАТКИ

Додаток А

Лістинги скриптів для створення таблиць

Лістинг А.1 – Створення таблиці Clients

```
CREATE TABLE [dbo].[Clients] (
    [id]           INT  IDENTITY (1, 1) NOT NULL,
    [fullname]     TEXT NOT NULL,
    [fulladress]   TEXT NOT NULL,
    [phone_number] TEXT NOT NULL,
    [e_mail]       TEXT NULL,
    PRIMARY KEY CLUSTERED ([id] ASC)
);
```

Лістинг А.2 – Створення таблиці Manufacturers

```
CREATE TABLE [dbo].[Manufacturers] (
    [id_manufacturer]   INT  IDENTITY (1, 1) NOT NULL,
    [name_manufacturer] TEXT NOT NULL,
    PRIMARY KEY CLUSTERED ([id_manufacturer] ASC)
);
```

Лістинг А.3 – Створення таблиці OrderDetails

```
CREATE TABLE [dbo].[OrderDetails] (
    [id_order]    INT   NOT NULL,
    [spare_part]  INT   NOT NULL,
    [quantity]    INT   NOT NULL,
    [total_price] MONEY NOT NULL,
    PRIMARY KEY CLUSTERED ([id_order] ASC, [spare_part] ASC),
    FOREIGN   KEY   ([spare_part])   REFERENCES   [dbo].[SpareParts]
([article]),
    FOREIGN KEY ([id_order]) REFERENCES [dbo].[Orders] ([id_order])
);
```

Лістинг А.4 – Створення таблиці Orders

```
CREATE TABLE [dbo].[Orders] (
    [id_order]    INT      IDENTITY (1, 1) NOT NULL,
    [id_client]   INT      NOT NULL,
    [date]        DATETIME NOT NULL,
    [total_price] MONEY    NOT NULL,
    [status]      INT      NOT NULL,
    PRIMARY KEY CLUSTERED ([id_order] ASC),
    CONSTRAINT   [FK_status]   FOREIGN   KEY   ([status])   REFERENCES
[dbo].[Status_Code] ([status_code]),
    CONSTRAINT  [FK_client]  FOREIGN  KEY  ([id_client])  REFERENCES
[dbo].[Clients] ([id])
);
```

Лістинг А.5 – Створення таблиці SpareParts

```
CREATE TABLE [dbo].[SpareParts] (
```

```
    [article]         INT   NOT NULL,
    [name]            TEXT  NOT NULL,
    [manufacturer_id] INT   NOT NULL,
    [quantity]        INT   NOT NULL,
    [price]           MONEY NOT NULL,
    PRIMARY KEY CLUSTERED ([article] ASC),
    CONSTRAINT [FK_Manufacturer_Id] FOREIGN KEY ([manufacturer_id])
REFERENCES [dbo].[Manufacturers] ([id_manufacturer])
);
```

Лістинг А.6 – Створення таблиці Status_Code

```
CREATE TABLE [dbo].[Status_Code] (
    [status_code] INT  NOT NULL,
    [status_name] TEXT NOT NULL,
    PRIMARY KEY CLUSTERED ([status_code] ASC)
);
```

Додаток Б

Лістинги створення збережуваних процедур та тригерів

Лістинг Б.1 – Створення процедури AddToOrder

```
CREATE PROCEDURE AddToOrder
     @orderId int,
     @article int,
     @quantity int
AS
BEGIN
     DECLARE @old_quantity int;
     DECLARE @price decimal;
     SELECT @price=price FROM SpareParts WHERE article = @article;
     SELECT @old_quantity=quantity FROM OrderDetails WHERE id_order
= @orderId AND spare_part = @article;
     IF @old_quantity is NULL
          BEGIN
               INSERT INTO OrderDetails VALUES (@orderId, @article,
@quantity, @quantity * @price);
          END
     ELSE
          BEGIN
               UPDATE OrderDetails SET quantity = @old_quantity +
@quantity,
               total_price = (@old_quantity + @quantity) * @price;
          END
     EXEC UpdateTotal @orderId;
END
```

Лістинг Б.2 – Створення процедури RemoveFromOrder

```
CREATE PROCEDURE RemoveFromOrder
     @orderId int,
     @article int
AS
BEGIN
     DELETE FROM OrderDetails WHERE id_order = @orderId AND
spare_part = @article;
     EXEC UpdateTotal @orderId;
END
```

Лістинг Б.3 – Створення процедури UpdateTotal

```
CREATE PROCEDURE UpdateTotal
     @orderId int
AS
     DECLARE @total decimal;
     SELECT @total=SUM(total_price) FROM OrderDetails WHERE id_order
= @orderId;
     SET @total = ISNULL(@total, 0);
```

```
     UPDATE Orders SET total_price = @total WHERE id_order =
@orderId;
```

Лістинг Б.4 – Створення процедури cancelOrder

```
CREATE PROCEDURE cancelOrder
     @orderId int
AS
     DECLARE @article int
     DECLARE @quantity int
     DECLARE my_cur CURSOR FOR
          SELECT spare_part, quantity FROM OrderDetails WHERE
id_order=@orderId;
     OPEN my_cur
   FETCH NEXT FROM my_cur INTO @article, @quantity
   WHILE @@FETCH_STATUS = 0
   BEGIN
        UPDATE SpareParts SET quantity += @quantity WHERE article =
@article;
        FETCH NEXT FROM my_cur INTO @article, @quantity
   END
   CLOSE my_cur
   DEALLOCATE my_cur
```

Лістинг Б.5 – Створення тригера TriggerOrderDetails

```
CREATE TRIGGER TriggerOrderDetails
     ON [dbo].[OrderDetails]
     FOR DELETE, INSERT, UPDATE
     AS
     BEGIN
          SET NOCOUNT ON
          DECLARE @article int;
          DECLArE @article2 int;
          DECLARE @new_quantity int;
          DECLARE @old_quantity int;
          SELECT @article=spare_part, @new_quantity=quantity FROM
inserted;
          SELECT @article2=spare_part, @old_quantity=quantity FROM
deleted;
          IF @article is NULL
               BEGIN
                    SET @article = @article2
               END
          IF @new_quantity is NULL
               SET @new_quantity = 0
          IF @old_quantity is NULL
               SET @old_quantity = 0
          UPDATE SpareParts SET quantity += @old_quantity -
@new_quantity WHERE article = @article;
     END
```

Додаток В

Лістинги створення форм для взаємодії з користувачем

Лістинг В.1 – Код файлу «FormMain.cs»

```
using System;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormMain : Form
    {
        public FormMain()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            FormManufacturers fm = new FormManufacturers();
            fm.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            FormSpareParts fsp = new FormSpareParts();
            fsp.Show();
        }

        private void FormMain_Load(object sender, EventArgs e)
        {
            DB.GetConnection();
        }

        private void btn_clients_Click(object sender, EventArgs e)
        {
            FormClients fc = new FormClients();
            fc.Show();
        }

        private void btn_orders_Click(object sender, EventArgs e)
        {
            FormOrders fo = new FormOrders();
            fo.Show();
        }

    }
}
```

Лістинг В.2 – Код файлу «FormAddClient.cs»

```
using System;
using System.Windows.Forms;
```

```
namespace AutoService
{
    public partial class FormAddClient : Form
    {
        public EntityClient client;
        public FormAddClient(EntityClient client = null)
        {
            InitializeComponent();
            this.client = client;
        }

        private void AddClient_Load(object sender, EventArgs e)
        {
            if (client != null)
            {
                btn_add_edit.Text = "Змінити дані клієнта";
                tb_fullname.Text = client.fullname;
                tb_fulladress.Text = client.fulladress;
                tb_phone_number.Text = client.phone_number;
                tb_email.Text = client.email;

                Text = "Змінити дані клієнта";
            }
        }

        private void btn_add_edit_Click(object sender, EventArgs e)
        {
            String fullname = tb_fullname.Text.Trim();
            String fulladress = tb_fulladress.Text.Trim();
            String phone = tb_phone_number.Text.Trim();
            String email = tb_email.Text.Trim();

            if (fullname.Length == 0)
                return;
            if (fulladress.Length == 0)
                return;
            if (phone.Length == 0)
                return;

            EntityClient clientForInsert = new EntityClient();


            clientForInsert.fullname = fullname;
            clientForInsert.fulladress = fulladress;
            clientForInsert.phone_number = phone;
            clientForInsert.email = email;

            if (client != null)
            {
                clientForInsert.id = client.id;
                DB.updateClient(clientForInsert);
            }
            else
            {
```

```
                DB.addClient(clientForInsert);
            }

            DialogResult = DialogResult.OK;
            Close();
        }
    }
     }
```

Лістинг В.3 – Код файлу «FormAddManufacturer.cs»

```
using System;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormAddManufacturer : Form
    {
        public String newManufacturer;
        public FormAddManufacturer()
        {
            InitializeComponent();
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            String manufacturer_name = tb_manufacturer.Text.Trim();
            if (manufacturer_name.Length != 0)
            {
                DB.addManufacturer(manufacturer_name);
                newManufacturer = manufacturer_name;
                DialogResult = DialogResult.OK;
                Close();
            }
        }
    }
}
```

Лістинг В.4 – Код файлу «FormAddPartToOrder.cs»

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormAddPartToOrder : Form
    {
        private SqlTransaction tran;
        public FormAddPartToOrder(SqlTransaction t)
        {
            InitializeComponent();
            tran = t;
```

```
        }

        public EntitySparePart selectedSparePart = null;
        public int quantity = 1;

        private void FormAddPartToOrder_Load(object sender,
EventArgs e)
        {
            List<EntitySparePart> spareParts =
DB.getSparePartsInStock(tran);
            for (int i = 0; i < spareParts.Count; i++)
            {
                combo_spare_part.Items.Add(spareParts[i]);
            }
        }

        private void btn_add_Click(object sender, EventArgs e)
        {
            if (combo_spare_part.SelectedItem == null)
                return;

            quantity = (int)numeric_count.Value;

            if (quantity > selectedSparePart.quantity)
            {
                MessageBox.Show("Залишилось " +
selectedSparePart.quantity + " шт.\n" +
                    "Ви вибрали " + quantity + " шт. Виберіть
менше.");
                return;
            }

            DialogResult = DialogResult.OK;
            Close();
        }

        private void combo_spare_part_SelectedIndexChanged(object
sender, EventArgs e)
        {
            selectedSparePart =
(EntitySparePart)combo_spare_part.SelectedItem;
            numeric_count.Enabled = true;
            btn_add.Enabled = true;
        }
    }
}
```

Лістинг В.5 – Код файлу «FormAddSparePart.cs»

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace AutoService
{
```

```
    public partial class FormAddSparePart : Form
    {
        public FormAddSparePart()
        {
            InitializeComponent();
            List<EntityManufacturer>        manufacturers        =
DB.getManufacturers();
            for (int i = 0; i < manufacturers.Count; i++)
            {
                combo_manufacturer.Items.Add(manufacturers[i]);
            }
        }

        private void Add_Click(object sender, EventArgs e)
        {
            object              selectedManufacturer              =
combo_manufacturer.SelectedItem;
            if (selectedManufacturer == null)
                return;
            EntityManufacturer            manufacturer            =
(EntityManufacturer)selectedManufacturer;

            string article = text_article.Text.Trim();
            if (article.Length == 0)
                return;
            string name = text_name.Text.Trim();
            if (name.Length == 0)
                return;


            EntitySparePart sparePart = new EntitySparePart();
            sparePart.article = Convert.ToInt32(article);
            sparePart.name = name;
            sparePart.quantity                                    =
Convert.ToInt32(numeric_quantity.Value);
            sparePart.price = numeric_price.Value;
            sparePart.manufacturer = manufacturer;

            DB.addSparePart(sparePart);
            DialogResult = DialogResult.OK;
            Close();
        }

        private void Cancel_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.Cancel;
            Close();
        }
    }
}
```

Лістинг В.6 – Код файлу «FormChangePrice.cs»

```
using System;
using System.Windows.Forms;
```

```
namespace AutoService
{
    public partial class FormChangePrice : Form
    {
        public decimal price;
        public FormChangePrice()
        {
            InitializeComponent();
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            price = numericPrice.Value;
            DialogResult = DialogResult.OK;
            Close();
        }
    }
}
```

Лістинг В.7 – Код файлу «FormClients.cs»

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormClients : Form
    {
        public FormClients()
        {
            InitializeComponent();
        }

        public void updateTable()
        {
            tb_clients.Rows.Clear();

            List <EntityClient> clients = DB.getClients();
            for (int i = 0; i < clients.Count; i++)
            {
                tb_clients.Rows.Add(clients[i].id,
clients[i].fullname,
                    clients[i].fulladress, clients[i].phone_number,
                    clients[i].email);
            }

            buttonsUpdate();
        }

        private void btn_add_Click(object sender, EventArgs e)
        {
            FormAddClient fac = new FormAddClient();
            if (fac.ShowDialog() == DialogResult.OK)
```

```
            {
                updateTable();
                tb_clients.Rows[tb_clients.Rows.Count - 1].Selected
= true;
            }
        }

        private void btn_edit_Click(object sender, EventArgs e)
        {
            int                          id                          =
(int)tb_clients.SelectedRows[0].Cells[0].Value;
            EntityClient client = DB.getClient(id);

            FormAddClient fac = new FormAddClient(client);
            if (fac.ShowDialog() == DialogResult.OK)
            {
                updateTable();
            }
        }

        private void btn_remove_Click(object sender, EventArgs e)
        {
            int                          id                          =
(int)tb_clients.SelectedRows[0].Cells[0].Value;
            if (DB.haveClientReference(id))
            {
                MessageBox.Show("Видалення неможливо, оскільки на
це значення " +
                    "посилаються елементи таблиці Orders");
            }
            else
            {
                DB.removeClient(id);
                updateTable();
            }
        }

        private void buttonsUpdate()
        {
            if (tb_clients.RowCount == 0)
            {
                btn_edit.Enabled = false;
                btn_remove.Enabled = false;
            }
            else
            {
                btn_edit.Enabled = true;
                btn_remove.Enabled = true;
            }
        }

        private void FormClients_Load(object sender, EventArgs e)
        {
            updateTable();
        }
```

```
        private int getSelectedRowIdx()
        {
            if (tb_clients.Rows.Count == 0)
                return -1;
            return tb_clients.SelectedRows[0].Index;
        }

        private void btn_export_Click(object sender, EventArgs e)
        {
            CSV.exportClients();
        }
    }
}
```

Лістинг В.8 – Код файлу «FormCreateOrder.cs»

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormCreateOrder : Form
    {
        private SqlTransaction tran;
        private String mode = null;
        private int idOrder = -1;
        private EntityClient selectedClient = null;

        public FormCreateOrder(int idOrder = -1, String mode = null)
        {
            InitializeComponent();
            tran = DB.GetConnection().BeginTransaction();
            this.mode = mode;
            this.idOrder = idOrder;
        }

        private void FormCreateOrder_Load(object sender, EventArgs
e)
        {
            if (idOrder != -1) {
                selectedClient     =     DB.getClientOfOrder(tran,
idOrder);
            }

            List<EntityClient> clients = DB.getClients(tran);
            for (int i = 0; i < clients.Count; i++)
            {
                combo_client.Items.Add(clients[i]);
                if (selectedClient != null)
                    if (clients[i].id == selectedClient.id)
                        combo_client.SelectedIndex = i;
            }
```

```
            updateTable();

            FormClosing += FormCreateOrder_FormClosing;
        }

        private void updateTable()
        {

            table_parts.Rows.Clear();

            decimal totalPrice = 0;
            List<EntityOrderDetails>        ordersDetails        =
DB.getOrdersDetailsForOrder(tran, idOrder);
            for (int i = 0; i < ordersDetails.Count; i++)
            {

table_parts.Rows.Add(ordersDetails[i].sparePart_id,
                    ordersDetails[i].sparePartName,
ordersDetails[i].quantity,
                    ordersDetails[i].total_price.ToString("C2"));
                totalPrice += ordersDetails[i].total_price;
            }

            updateButtons();

            lbl_total_price.Text = totalPrice.ToString("C2");
        }

        private void updateButtons()
        {
            if (mode == "view")
            {
                btn_AddToOrder.Enabled = false;
                btn_RemoveFromOrder.Enabled = false;
                btn_ok.Enabled = false;
                combo_client.Enabled = false;
                btn_ok.Enabled = false;
                btn_export.Enabled = true;
            }

            else
            {
                btn_export.Enabled = false;

                if (table_parts.Rows.Count == 0)
                {
                    btn_RemoveFromOrder.Enabled = false;
                }
                else
                {
                    btn_RemoveFromOrder.Enabled = true;
                }

                if (idOrder != -1)
```

```
                {
                    btn_AddToOrder.Enabled = true;
                    btn_ok.Enabled = true;
                }
                else
                {
                    btn_AddToOrder.Enabled = false;
                    btn_ok.Enabled = false;
                }
            }
        }

        private void btn_AddToOrder_Click(object sender, EventArgs
e)
        {
            FormAddPartToOrder          fapto          =          new
FormAddPartToOrder(tran);
            if (fapto.ShowDialog() == DialogResult.OK)
            {
                int article = fapto.selectedSparePart.article;
                int quantity = fapto.quantity;
                DB.addToOrder(tran, idOrder, article, quantity);
                updateTable();
                table_parts.Rows[table_parts.Rows.Count           -
1].Selected = true;
            }
        }

        private   void   btn_RemoveFromOrder_Click(object   sender,
EventArgs e)
        {
            int                        article                        =
(int)table_parts.SelectedRows[0].Cells[0].Value;
            DB.removeFromOrder(tran, idOrder, article);
            updateTable();
        }

        private void btn_ok_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
            Close();
        }

        private  void  FormCreateOrder_FormClosing(Object  sender,
FormClosingEventArgs e)
        {
            if (DialogResult == DialogResult.OK)
            {
                tran.Commit();
            }
            else
            {
                tran.Rollback();
            }
        }
```

```
        private    void    combo_client_SelectedIndexChanged(object
sender, EventArgs e)
        {
            EntityClient oldClient = selectedClient;
            selectedClient                                          =
(EntityClient)combo_client.SelectedItem;
            if (oldClient == null)
            {
                idOrder = DB.createOrder(tran, selectedClient.id);
            }
            else if (oldClient.id != selectedClient.id)
            {
                DB.changeClientForOrder(tran,              idOrder,
selectedClient.id);
            }

            updateButtons();
        }

        private void btn_export_Click(object sender, EventArgs e)
        {
            CSV.exportOrderDetails(this.tran, this.idOrder);
        }
    }
}
```

Лістинг В.9 – Код файлу «FormManufacturers.cs»

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormManufacturers : Form
    {
        public FormManufacturers()
        {
            InitializeComponent();
        }

        private    void    FormManufacturers_Load(object    sender,
EventArgs e)
        {
            updateTable();
        }

        private void updateTable() {
            List<EntityManufacturer>        manufacturers        =
DB.getManufacturers();
            for (int i = 0; i < manufacturers.Count; i++)
            {
                tableManufacturers.Rows.Add(manufacturers[i].id,
manufacturers[i].name);
```

```
            }

            removeButtonUpdate();
        }

        private void removeButtonUpdate()
        {
            if (tableManufacturers.RowCount == 0)
                btnRemove.Enabled = false;
            else
                btnRemove.Enabled = true;
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            FormAddManufacturer fam = new FormAddManufacturer();
            if (fam.ShowDialog() == DialogResult.OK)
            {
                tableManufacturers.Rows.Clear();
                updateTable();
            }
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            int                          id                          =
(int)tableManufacturers.SelectedRows[0].Cells[0].Value;
            if (DB.haveManufacturerReference(id))
            {
                MessageBox.Show("Видалення неможливо, оскільки на
це значення " +
                    "посилаються елементи таблиці SpareParts");
            }
            else
            {

tableManufacturers.Rows.Remove(tableManufacturers.SelectedRows[0])
;
                DB.removeManufacturer(id);
                removeButtonUpdate();
            }
        }

        private void btn_export_Click(object sender, EventArgs e)
        {
            CSV.exportManufacturers();
        }
    }
}
```

Лістинг В.10 – Код файлу «FormOrders.cs»

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
```

```
namespace AutoService
{
    public partial class FormOrders : Form
    {
        public FormOrders()
        {
            InitializeComponent();
        }

        private void FormOrders_Load(object sender, EventArgs e)
        {
            updateTable();
        }

        private void updateTable()
        {
            table_orders.Rows.Clear();
            List <EntityOrder> orders = DB.getOrders();
            for (int i = 0; i < orders.Count; i++)
            {
                table_orders.Rows.Add(orders[i].order_id,
orders[i].client_id,
                    orders[i].client, orders[i].date,
                    orders[i].total_price.ToString("C2"),
orders[i].status);
            }
            updateButtons();
        }

        private void updateButtons()
        {
            if (table_orders.Rows.Count == 0)
            {
                btn_cancel.Enabled = false;
                btn_change.Enabled = false;
                btn_finish.Enabled = false;
                btn_view.Enabled = false;
            }
            else if (table_orders.SelectedRows.Count != 0)
            {
                btn_view.Enabled = true;

                String                    status                    =
(String)table_orders.SelectedRows[0].Cells["status"].Value;
                if (status == "В обробці")
                {
                    btn_cancel.Enabled = true;
                    btn_change.Enabled = true;
                    btn_finish.Enabled = true;
                }
                else
                {
                    btn_cancel.Enabled = false;
                    btn_change.Enabled = false;
```

```
                btn_finish.Enabled = false;
            }
        }

    }

    private void btn_create_Click(object sender, EventArgs e)
    {
        FormCreateOrder fco = new FormCreateOrder();
        if (fco.ShowDialog() == DialogResult.OK)
        {
            updateTable();
            table_orders.Rows[table_orders.Rows.Count - 
1].Selected = true;
        }
    }

    private void btn_change_Click(object sender, EventArgs e)
    {
        FormCreateOrder fco = new 
FormCreateOrder(getSelectedOrderId(), "edit");
        if (fco.ShowDialog() == DialogResult.OK)
        {
            int lastIdx = getSelectedRowIdx();
            updateTable();
            if (lastIdx != -1)
                table_orders.Rows[lastIdx].Selected = true;
        }
    }

    private void btn_cancel_Click(object sender, EventArgs e)
    {
        String text = "Ви дійсно бажаєте скасувати 
замовлення?\n" +
            "Усі запчастини буде повернуто на склад";
        var result = MessageBox.Show(text, "Скасування",
                             MessageBoxButtons.YesNo,
                             MessageBoxIcon.Question);
        if (result == DialogResult.No)
            return;

        int orderId = 
(int)table_orders.SelectedRows[0].Cells[0].Value;
        DB.changeStatus(orderId, 0);
        int idx = getSelectedRowIdx();
        updateTable();
        table_orders.Rows[idx].Selected = true;
    }

    private void btn_finish_Click(object sender, EventArgs e)
    {
        String text = "Ви дійсно бажаєте завершити замовлення?";
        var result = MessageBox.Show(text, "Завершення",
                             MessageBoxButtons.YesNo,
                             MessageBoxIcon.Question);
```

```
            if (result == DialogResult.No)
                return;

            int                         orderId                         =
(int)table_orders.SelectedRows[0].Cells[0].Value;
            DB.changeStatus(orderId, 2);
            int idx = getSelectedRowIdx();
            updateTable();
            table_orders.Rows[idx].Selected = true;
        }

        private void btn_view_Click(object sender, EventArgs e)
        {
            FormCreateOrder           fco           =           new
FormCreateOrder(getSelectedOrderId(), "view");
            fco.ShowDialog();
        }

        private int getSelectedOrderId()
        {
            int                         orderId                         =
(int)table_orders.SelectedRows[0].Cells[0].Value;
            return orderId;
        }

        private  void  table_orders_SelectionChanged(object  sender,
EventArgs e)
        {
            updateButtons();
        }

        private int getSelectedRowIdx()
        {
            if (table_orders.Rows.Count == 0)
                return -1;
            return table_orders.SelectedRows[0].Index;
        }

        private void btn_export_Click(object sender, EventArgs e)
        {
            CSV.exportOrders();
        }
    }
}
```

Лістинг В.11 – Код файлу «FormSpareParts.cs»

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormSpareParts : Form
    {
```

```
        public FormSpareParts()
        {
            InitializeComponent();
        }

        private void FormSpareParts_Load(object sender, EventArgs e)
        {
            updateTable();
        }

        private void updateTable()
        {
            tableSpareParts.Rows.Clear();
            List<EntitySparePart> spareParts = DB.getSpareParts();
            for (int i = 0; i < spareParts.Count; i++)
            {
                tableSpareParts.Rows.Add(spareParts[i].article,
                    spareParts[i].name,
spareParts[i].manufacturer.name,
                    spareParts[i].quantity,
spareParts[i].price.ToString("C2"));
            }

            buttonsUpdate();
        }

        private void buttonsUpdate()
        {
            if (tableSpareParts.RowCount == 0)
            {
                removeSparePart.Enabled = false;
                btn_change_quantity.Enabled = false;
                btn_set_price.Enabled = false;
            }
            else
            {
                removeSparePart.Enabled = true;
                btn_change_quantity.Enabled = true;
                btn_set_price.Enabled = true;
            }
        }

        private void addSparePart_Click(object sender, EventArgs e)
        {
            FormAddSparePart fam = new FormAddSparePart();
            if (fam.ShowDialog() == DialogResult.OK)
            {
                updateTable();
                tableSpareParts.Rows[tableSpareParts.Rows.Count   -
1].Selected = true;
            }
        }

        private void removeSparePart_Click(object sender, EventArgs
e)
```

```
        {
            int                         article                         =
(int)tableSpareParts.SelectedRows[0].Cells[0].Value;
            if (DB.haveSparePartReference(article))
            {
                MessageBox.Show("Видалення неможливо, оскільки на
це значення " +
                    "посилаються елементи таблиці SpareParts");
            }
            else
            {

tableSpareParts.Rows.Remove(tableSpareParts.SelectedRows[0]);
                DB.removeSparePart(article);
                buttonsUpdate();
            }
        }

        private   void   btn_change_quantity_Click(object   sender,
EventArgs e)
        {
            FormSparePartsChangeQuantity      fspcq      =      new
FormSparePartsChangeQuantity();
            if (fspcq.ShowDialog() == DialogResult.OK)
            {
                int quantity = fspcq.quantity;
                int                     article                     =
(int)tableSpareParts.SelectedRows[0].Cells[0].Value;
                int                   oldQuantity                   =
(int)tableSpareParts.SelectedRows[0].Cells[3].Value;
                if (fspcq.type == 0)
                {
                    DB.updateQuantitySparePart(article, oldQuantity
+ quantity);
                }
                else if (fspcq.type == 1)
                    DB.updateQuantitySparePart(article, oldQuantity
- quantity);
                else
                    DB.updateQuantitySparePart(article, quantity);

                int lastIdx = getSelectedRowIdx();
                updateTable();
                tableSpareParts.Rows[lastIdx].Selected = true;
            }
        }

        private void btn_set_price_Click(object sender, EventArgs e)
        {
            FormChangePrice fcp = new FormChangePrice();
            if (fcp.ShowDialog() == DialogResult.OK)
            {
                int                     article                     =
(int)tableSpareParts.SelectedRows[0].Cells[0].Value;
                DB.updatePriceSparePart(article, fcp.price);
```

```
                int lastIdx = getSelectedRowIdx();
                updateTable();
                tableSpareParts.Rows[lastIdx].Selected = true;
            }
        }

        private int getSelectedRowIdx()
        {
            if (tableSpareParts.Rows.Count == 0)
                return -1;
            return tableSpareParts.SelectedRows[0].Index;
        }

        private void btn_export_Click(object sender, EventArgs e)
        {
            CSV.exportSpareParts();
        }
    }
}
```

Лістинг В.12 – Код файлу «FormSparePartsChangeQuantity.cs»

```
using System;
using System.Windows.Forms;

namespace AutoService
{
    public partial class FormSparePartsChangeQuantity : Form
    {
        public int quantity;
        public int type;
        public FormSparePartsChangeQuantity()
        {
            InitializeComponent();
        }

        private void comboType_SelectedIndexChanged(object sender,
EventArgs e)
        {
            btn_confirm.Enabled = true;
            numericQuantity.Enabled = true;
        }

        private void btn_confirm_Click(object sender, EventArgs e)
        {
            quantity = (int)numericQuantity.Value;
            type = comboType.SelectedIndex;
            DialogResult = DialogResult.OK;
            Close();
        }
    }
}
```