РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: інформаційна система станції технічного обслуговування автомобілів.

Мета кваліфікаційної роботи: розробка інформаційної системи для оптимізації і автоматизації роботи станції технічного обслуговування автомобілів, перегляду інформації про постачальників, продажі, а також для перегляду і друку інформації про продані товари.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в можливості завдяки розробленому додатку скоротити витрати часу на пошук певного товару на складі, на редагування інформації про товари, розрахунок і друк замовлення, контроль платежів, здійснення операцій продажу, а також перегляд повного переліку постачальників.

Актуальність теми кваліфікаційної роботи обумовлюється розробкою сучасної інформаційної системи, яка виконує підбір товару і послуг, що надаються СТО, та забезпечує економію коштів фірми за рахунок автоматизації процесу роботи персоналу.

Список ключових слів: БАЗИ ДАНИХ, СКБД, ІНФОРМАЦІЙНА СИСТЕМА, СТАНЦІЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ, ВИТРАТИ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: information system of car maintenance station.

The purpose of the qualification work: development of an information system for optimization and automation of the car service station, viewing information about suppliers, sales, as well as for viewing and printing information about sold goods.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download program, describes the program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of the work is the ability to reduce the time spent searching for a specific product in stock, editing product information, calculating and printing orders, controlling payments, selling transactions, as well as viewing the full list of suppliers.

The relevance of the topic of qualification work is due to the development of a modern information system that performs the selection of goods and services provided by the service station, and saves money by automating the process of staff work.

List of keywords: DATABASES, DBMS, INFORMATION SYSTEM, MAINTENANCE STATION, COSTS.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ООП – об'єктно-орієнтоване програмування;

ОС – операційна система;

ПК – персональний комп'ютер;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

СТО – станція технічного обслуговування;

SQL – структурована мова запитів.

ЗМІСТ

ВСТУП

На сьогоднішній день в Україні існує безліч СТО, робота яких недостатньо оптимізована. Одним з найважливіших умов забезпечення ефективного функціонування будь-якої організації є наявність розвиненої та сучасної інформаційної системи.

Інформаційна система являє собою систему, що реалізовує автоматизований збір, обробку та маніпулювання даними й включає технічні засоби обробки даних, програмне забезпечення та інформацію про обслуговуючий персонал.

База даних забезпечує зберігання інформації і являє собою пойменовану сукупність даних, організованих за певними правилами, які мають загальні принципи опису, зберігання і маніпулювання даними.

Метою кваліфікаційної роботи бакалавра є створення інформаційної системи по наданню послуг СТО для підвищення продуктивності працівників підприємства і для скорочення витрат часу на підбір потрібного товару і оформлення чеків.

Інформаційна система призначена для оптимізації і автоматизації роботи станції технічного обслуговування автомобілів, перегляду інформації про постачальників, продажі, а також для перегляду і друку інформації про продані товари та може бути застосованою в будь-який СТО з подібним родом діяльності.

Розроблена система дозволяє вирішити такі задачі:

– автоматизація процесу роботи СТО;
– дозволяє зв'язати мережу СТО даної фірми в один сервер;
– редагування списку товару;
– контроль кількості товару на складі;
– розрахунок вартості покупки;
– друк чеків і каталогу товарів.

Практичне значення роботи полягає в можливості завдяки розробленому додатку скоротити витрати часу на пошук певного товару на складі, на редагування інформації про товари, розрахунок і друк замовлення, контроль платежів, здійснення операцій продажу, а також облік повного переліку постачальників.

Актуальність теми кваліфікаційної роботи обумовлюється розробкою сучасної інформаційної системи, яка виконує підбір товару і послуг, що надаються СТО, та забезпечує економію коштів фірми за рахунок автоматизації процесу роботи персоналу.

Розроблена інформаційна система призначена для комерційного застосування в будь-якому з підприємств, що займаються технічним обслуговуванням автомобіля.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

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

– прийом замовлень на послуги від населення;

– виконання замовлень;

– реалізація послуг.

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

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

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

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

Замовлення на платні послуги оформляються як спеціальними первинними документами, так і шляхом видачі жетонів, талонів, касових чеків, розписок і т.д.

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

Розрізняють механічний, абразивний, корозійний і втомний знос.

Механічний знос відбувається внаслідок зім'яття або викришування частинок з поверхні деталей, що викликає зміна маси і розміру деталі.

Абразивний знос - це результат дряпаючого або ріжучого впливу більш твердих частинок однієї з сполучених деталей, часток внесених повітрям або тими, що потрапили разом зі змазкою.

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

Втомний знос викликається впливом багаторазових змінних навантажень.

Більшість деталей автомобільного транспорту піддаються одночасному зносу декількох видів.

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

Для забезпечення безперебійної роботи автомобільного транспорту необхідно не тільки повсякденне спостереження за його станом в процесі експлуатації (мастило, огляд і т.п.), але і періодичне проведення ремонту. Нерівномірність зношування окремих вузлів і деталей, що входять до складу того чи іншого об'єкта автомобільного транспорту, зажадала розробки спеціальної системи планово-попереджувального ремонту. Система технічного обслуговування автомобільного транспорту є планово-попереджувальною, і всі роботи, передбачені для кожного обслуговування, є обов'язковими до виконання в повному обсязі. Ця система сприяє постійному підтриманню автомобілів і причепів в працездатному вигляді, зменшенню інтенсивності зносу деталей, попередження відмов і несправностей, зниження витрати палива і мастильних матеріалів, підвищення надійності і безпеки експлуатації і

збільшення пробігу автомобілів до ремонту.

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

Технічне обслуговування є профілактичним заходом, проведених примусово в плановому порядку через певні пробіги або час роботи рухомого складу. Періодичність технічного обслуговування встановлюється за фактично виконаному пробігу в кілометрах з урахуванням умов експлуатації. Для кожної категорії умов експлуатації найбільша періодичність технічного обслуговування прийнята для легкових автомобілів, потім - автобусів і вантажних автомобілів.

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

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

Капітальний ремонт включає контрольно-діагностичні, складальні, регулювальні, слюсарні, механічні, шпалерні, електротехнічні, шиноремонтні, малярні та інші роботи. Ремонтні роботи можуть виконуватися за певними агрегатів вузлів, а також по рухомому складу в цілому. При капітальному ремонті агрегат повністю розбирають, виявляють дефекти, відновлюють або замінюють окремі деталі, потім збирають, регулюють і випробовують. Якщо капітального ремонту підлягає весь автомобіль, то його теж повністю

розбирають, всі деталі дефектують, відновлюють і замінюють, збирають, а вузли та агрегати регулюють і відчувають.

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

Облік ремонту транспортних засобів слід вести по його видам: капітальний і поточний з поділом на середній, малий і міжремонтне обслуговування [13].

Якщо при автосервісі є магазин автозапчастин, то одним з основних питань, яке постає перед автосервісом, - придбання запчастин, а значить, налагодження зв'язків з постачальниками.

Заявку на доставку краще формувати заздалегідь, з огляду на сезонні коливання попиту. Описати їх може будь-який досвідчений працівник цієї сфери (для шиномонтажу, наприклад, піки активності припадають на кінець осені та початок весни, коли настає пора міняти гуму).

Автозапчастини або купують у дилерів, або завозять самостійно, співпрацюючи з постачальниками з країн Азії. Що стосується дрібних автосервісів, то самостійно ввозити запчастини для них досить ризиковано, адже деталі доводиться закуповувати на перспективу. Крім того, коли мова йде про невеликі партії, розраховувати на знижки від виробника не доводиться. Важливо точно враховувати сезонність і своєчасність закупівель: наявність товару на складі не тільки скоротить час обслуговування кожного клієнта, а й прискорить оборотність коштів.

Для спеціалізованих автосервісів, де список необхідних деталей чітко визначено нормативами автовиробників, проблема доставки запчастин вирішується простіше. Існує дві найбільш поширені моделі закупівлі запчастин:

– закупівля деталей «із запасом». Він потребує значних витрат, але зате знімає питання відсутності необхідних запчастин в потрібний момент. Найбільш ефективна така схема для мережевих автосервісів, в роботі яких рано чи пізно виявляється затребуваною практично будь-яка деталь.

– робота зі службами доставки. Всі деталі в міру необхідності замовляють в якому-небудь Інтернет-магазині. Але тоді їх вартість виявляється вищою.

При будь-якому із зазначених вище способах поставок запчастин розроблена база даних повинна містити список контрагентів з можливістю його редагування і доповнення новими постачальниками. Даний список включається в перелік довідників, що містяться в системі. При виборі потрібної запчастини програмний модуль в автоматичному режимі підключить необхідного постачальника по сполучному ключу.

1.2. Призначення розробки та галузь застосування

Інформаційна система призначена для оптимізації і автоматизації роботи станції технічного обслуговування автомобілів, перегляду інформації про постачальників, продажі, а також для перегляду і друку інформації про продані товари та може бути застосованою в будь-який СТО з подібним родом діяльності.

Розроблена інформаційна забезпечує автоматизацію процесу діяльності СТО. Програма дозволяє вирішити такі проблеми: додавання нового товару, редагування і видалення старого, вибір товару або послуги, розрахунок вартості обраного товару, друк чеків і каталогів товарів, зменшення кількості товару при його продажу і повідомлення при відсутності товару на складі.

Розроблена система призначена для комерційного застосування в установах подібних СТО з аналогічним переліком послуг, що надаються.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка інформаційної системи для станції технічного обслуговування автомобілів засобами мови C#» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою кваліфікаційної роботи «Розробка інформаційної системи для станції технічного обслуговування автомобілів засобами мови C#» є створення інформаційної системи, призначеної для автоматизації діяльності та документообігу СТО.

Розроблена система повинна вирішити такі задачі:

– автоматизація процесу роботи СТО;
– зв'язати мережу СТО даної фірми в один сервер;
– редагування списку товару;
– контроль кількості товару на складі;
– розрахунок вартості покупки;
– друк чеків і каталогу товарів.

Для досягнення мети роботи необхідно виконати наступні етапи:

1. На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі
2. Провести обґрунтування, вибір та здійснити реалізацію методів вирішення проблеми.
3. Протестувати розробку та виправити можливі вразливості та помилки.

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

Вхідними даними даної системи є наступна інформація

– мастила (артикул, найменування, кількість, ціна);

– шини (артикул, найменування, кількість, ціна);

– паливо (артикул, найменування, кількість, ціна);

– послуги (артикул, найменування, кількість, ціна);

– охолоджуючі рідини (артикул, найменування, кількість, ціна);

– хімічні речовини або рідини (артикул, найменування, кількість, ціна);

– мийка (артикул, найменування, ціна);

– постачальник (артикул, найменування, артикул нового товару, дата);

– новий товар (артикул, артикул палива, кількість палива, артикул шин. кількість шин, артикул масла, кількість масла, артикул охол. рідини, кількість охол. рідини, артикул миюч. рідини, кількість миюч. рідини);

Вихідними даними є інформація, яку може використовувати користувач даної системи:

– вибір конкретного товару;

– вивід вартості обраних товарів та послуг;

– перегляд і друк звітів: чек для клієнта і прайс-листи товарів.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Програма призначена для забезпечення діяльності СТО та повинна мати наступний функціонал:

– додавання, видалення та редагування інформації про товари і послуги. Необхідно передбачити для забезпечення найбільш ефективної роботи користувача видалення і редагування шляхом безпосереднього вибору запису з таблиці;

– перегляд інформації про товари і послуги;

– перегляд і друк чеку про проданий товар, який містить відомості про найменування товару, кількість і ціну конкретного товару;

– перегляд і друк інформації про товари, що є на складі;

– здійснення пошуку необхідної інформації про товари і послуги.

– здійснення операцій продажу товару, розрахунку ціни проданого товару.

– облік товару на складі і вивід повідомлення при відсутності товару на складі.

1.5.2. Вимоги до інформаційної безпеки

Інформаційна система повинна забезпечувати наступні вимоги до інформаційної безпеки:

– можливість входу в систему з різними рівнями доступу до даних: призначений для користувача (здійснює тільки перегляд інформації, продаж товару і друк чека) і адміністраторський (здійснює всі можливі операції, представлені в системі).

– здійснення контролю введених даних: перевірка на відповідність типів, на введення обов'язкових полів даних, а також, на введення тільки можливих значень, зчитувальних з необхідних таблиць;

– можливість перегляду інформації з таблиць в режимі реального часу.

1.5.3. Вимоги до складу та параметрів технічних засобів

Технічні засоби для роботи програмного продукту повинні відповідати наступним системним вимогам:

– підтримка 64-розрядного процесору та операційної системи;

– операційна система Windows 10 64bit;

– процесор Intel Core i5 3470 3.2 GHz/AMD X8 FX-8350 4 GHz;

– оперативна пам'ять 8GB ОЗУ;

– відеокарта nVidia GTX 650 з 1 Гб відеопам’яті, AMD HD7860 з 1 Гб відеопам’яті;

– DirectX версії 12;
– жорсткий диск мінімум в 72 GB.

1.5.4. Вимоги до інформаційної та програмної сумісності

Робота програми повинна бути розрахована на ОС Windows NT 5/NT 5.1.

Для роботи ІС необхідна також наявність:

– компоненти доступу до даних (MDAC) 2000.085.1117.00;
– Microsoft MSXML 2.6 3.0 4.0 5.0 6.0;
– операційна система Windows NT 5.1.2600.

В якості мови програмування при розробці програми доречно використовувати мову програмування C#.

Програма повинна являти собою самостійний виконуваний модуль і бути структурованою і закоментованою.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Інформаційна система призначена для оптимізації і автоматизації роботи станції технічного обслуговування автомобілів, перегляду інформації про постачальників, продажі, а також для перегляду і друку інформації про продані товари та може бути застосованою в будь-який СТО з подібним родом діяльності.

Розроблена система вирішує наступні задачі:

– автоматизація процесу роботи СТО;

– редагування списку товару;

– контроль кількості товару на складі;

– розрахунок вартості покупки;

– друк чеків і каталогу товарів.

Програма призначена для забезпечення діяльності СТО та має наступний функціонал:

– додавання, видалення та редагування інформації про товари і послуги. Необхідно передбачити для забезпечення найбільш ефективної роботи користувача видалення і редагування шляхом безпосереднього вибору запису з таблиці;

– перегляд інформації про товари і послуги;

– перегляд і друк чеку про проданий товар, який містить відомості про найменування товару, кількість і ціну конкретного товару;

– перегляд і друк інформації про товари, що є на складі;

– здійснення пошуку необхідної інформації про товари і послуги.

– здійснення операцій продажу товару, розрахунку ціни проданого товару.

– облік товару на складі і вивід повідомлення при відсутності товару на

складі.

2.2. Опис застосованих математичних методів

В даній інформаційній системі використовуються прості арифметичні операції для підрахунку кошторису виконаних робіт та сумарної вартості придбаних товарів, які не потребують додаткового опису в даному розділі.

2.3. Опис використаних технологій та мов програмування

Для розробки проекту було вирішено використати мову програмування C#, а середовищем розробки обрана Microsoft Visual Studio.

C# – проста, сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# відноситься до широко відомого сімейства мов C.

C# є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних і самописних пакетів, що реалізують окремі функціональні можливості. Важлива особливість таких компонентів – це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненті, а також вбудовані елементи документації. Таким чином C# відмінно підходить для створення і застосування програмних компонентів.

Розглянемо далі функції мови C#, що забезпечують надійність і стійкість додатків: прибирання сміття автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами; обробка виключень дає структурований і розширюваний спосіб виявляти і обробляти помилки; сувора типізація мови не дозволяє звертатися до неініціалізованих змінних, виходити за межі масиву або виконувати неконтрольоване приведення типів.

У C# існує єдина система типів. Всі типи C#, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Таким

чином, всі типи використовують загальний набір операцій, і значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Крім того, C# підтримує призначені для користувача посилальні типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеці.

Щоб забезпечити сумісність програм і бібліотек C# при подальшому розвитку, при розробці C# багато уваги було приділено управлінню версіями. Багато мов програмування обходять увагою це питання, і в результаті програми на цих мовах виходять з ладу частіше, ніж хотілося б, при виході нових версій залежних бібліотек. Питання управління версіями істотно вплинули на такі аспекти розробки C#, як роздільні модифікатори virtual і override, правила вирішення перевантаження методів і підтримка явного оголошення членів інтерфейсу[2].

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти, як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби, як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як,

наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

У Visual Studio застосовуються різні технології Для наочності вони представлені нижче :

– .NET Framework;
– .NET Framework 3.5;
– .NET Framework 3.0;
– .NET Compact Framework.

.NETFramework – невід'ємне компоненти Windows, які підтримують побудову і розробку додатків нового покоління.

– Windows Presentation Foundation (WPF);
– Windows Communication Foundation (WCF);
– Windows Workflow Foundation;
– Silverlight;
– Windows Forms;
– ASP.NET (AJAX);
– LINQ.

Також у Visual Studio існує можливість програмування наступними мовами програмування:

– Visual Basic;
– Visual C #;
– Visual C ++;
– JScript.

Програмний додаток було виконано за допомогою системи керування базами даних (БД) SQL Server 2012.

Microsoft SQL Server - система керування базами даних (СКБД), розроблена корпорацією Microsoft. Основний використовуваний мову запитів -

Transact-SQL, створений спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO по структурованого мови запитів (SQL) з розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД в цьому сегменті ринку.

Сервер баз даних Microsoft SQL Server в якості мови запитів використовує версію мови SQL, що отримала назву Transact-SQL (скорочено T-SQL). Мова T-SQL є реалізацією SQL-92 (стандарт ISO для мови SQL) з множинними розширеннями. T-SQL дозволяє використовувати додатковий синтаксис для збережених процедур і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим додатком).

При взаємодії з мережею Microsoft SQL Server і Sybase ASE використовують протокол рівня додатка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних). Протокол TDS також був реалізований в проекті FreeTDS з метою забезпечити різним додаткам можливість взаємодії з базами даних Microsoft SQL Server і Sybase.

Для забезпечення доступу до даних Microsoft SQL Server підтримує Open Database Connectivity (ODBC) - інтерфейс взаємодії додатків з СУБД. Версія SQL Server 2005 забезпечує можливість підключення користувачів через веб-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Компанія Microsoft також випустила сертифікований драйвер JDBC, що дозволяє додаткам під керуванням Java (таким як BEA і IBM WebSphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

Також SQL Server підтримує віддзеркалення і кластеризації баз даних. Кластер сервера SQL - це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Всі сервера мають одне віртуальне ім'я, і дані розподіляються по IP-адресами машин кластеру протягом робочого циклу. Також в разі відмови або збою на одному з серверів кластера доступний автоматичний перенос навантаження на

інший сервер.

SQL Server підтримує надлишкове дублювання даних за трьома сценаріями:

1. Знімок: Проводиться «знімок» бази даних, який сервер відправляє одержувачам.

2. Історія змін: Всі зміни бази даних безперервно передаються користувачам.

3. Синхронізація з іншими серверами: Бази даних декількох серверів синхронізуються між собою. Зміни всіх баз даних відбуваються незалежно один від одного на кожному сервері, а при синхронізації відбувається звірка даних. Даний тип дублювання передбачає можливість вирішення протиріч між БД.

У SQL Server 2012 вбудована підтримка .NET Framework. Завдяки цьому збережені процедури БД можуть бути написані на будь-якій мові платформи .NET, використовуючи повний набір бібліотек, доступних для .NET Framework, включаючи Common Type System (система поводження з типами даних в Microsoft .NET Framework). Однак, на відміну від інших процесів, .NET Framework, будучи базисною системою для SQL Server 2012, виділяє додаткову пам'ять і вибудовує засоби управління SQL Server замість того, щоб використовувати вбудовані засоби Windows. Це підвищує продуктивність в порівнянні з загальними алгоритмами Windows, так як алгоритми розподілу ресурсів спеціально налаштовані для використання в структурах SQL Server.

Microsoft та інші компанії виробляють велику кількість програмних засобів розробки, що дозволяють розробляти бізнес-додатки з використанням баз даних Microsoft SQL Server. Microsoft SQL Server 2012 включає в себе також Common Language Runtime (CLR) Microsoft .NET, що дозволяє реалізовувати збережені процедури і різні функції додаткам, розробленим на мовах платформи .NET (наприклад, VB.NET або C #). Попередні версії засобів розробки Microsoft використовували тільки API для отримання функціонального доступу до Microsoft SQL Server.

Microsoft SQL Server Express є безкоштовно розповсюджуваною версією

SQL Server, розвитком системи MSDE. Дана версія має деякі технічні обмеження. Такі обмеження роблять її непридатною для розгортання великих баз даних, але вона цілком годиться для ведення програмних комплексів у масштабах невеликої компанії. Містить повноцінну підтримку нових типів даних, у тому числі XML-специфікації. Фактично, це повноцінний MS SQL Server, включаючи всі його компоненти програмування, підтримку національних алфавітів і Unicode. Тому використовується в додатках, при проектуванні або для самостійного вивчення.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Опис архітектури системи

При розробці інформаційної системи була використана архітектура «Клієнт-сервер».

В основі клієнт-серверних технологій лежать дві ідеї:

– загальні для всіх користувачів дані на одному або декількох серверах;

– багато користувачів (клієнтів) на різних обчислювальних установках спільно (паралельно і одночасно) оброблюють загальні дані.

Тобто системи, засновані на цих технологіях, розподілені лише щодо користувачів, тому часто їх вважають видом багатокористувацьких систем.

Як і для файл-серверної архітектури, складовими компонентами клієнт-серверної архітектури є сервер, клієнтські місця і мережева інфраструктура. Однак, на відміну від попереднього випадку, сервер тут є вже не сервером файлів, а сервером баз даних або навіть сервером додатків. Таким чином, на сервер лягає не просто зберігання файлів, а підтримання бази даних в цілісному стані або, в разі сервера додатків, навіть виконання тієї чи іншої частини прикладної задачі. Природно, що вимоги до сервера при цьому можуть зростати в рази.

З іншого боку, те, що сервер володіє інформацією про характер збереженої бази даних, дозволяє набагато збільшити ефективність обробки.

Тому для багатьох завдань автоматизації навантаження на сервер за рахунок більш оптимального виконання операцій над даними в порівнянні з аналогічними файл-серверними додатками може навіть зменшитися.

Відповідно, спілкування між клієнтом і сервером відбувається не на рівні файлів, а на рівні обміну запитами. Клієнт передає серверу Високорівневі запити на отримання тієї чи іншої інформації або на її зміну, а сервер повертає клієнту результати виконання запитів. При цьому, на відміну від файл-серверної архітектури, доступ до даних не є прозорим для користувача програми. Тому, технологія розробки таких додатків принципово відмінна від локальних і файл-серверних систем.

На сучасному етапі мережеве забезпечення для архітектури клієнт-сервер аналогічно файл-серверному. Клієнт-серверні системи можуть будуватися з використанням тих же мережевих технологій і на тій же мережевій інфраструктурі. Більш того, як правило, на підприємстві мирно співіснують обидві ці архітектури. Це викликано двома основними причинами. По-перше, що на будь-якому підприємстві багато завдань, пов'язаних із зберіганням та обміном документами, які являють собою окремі файли, а для них архітектура файл-сервер оптимальна. По-друге, файл-серверні завдання в тому чи іншому обсязі майже завжди зберігаються в тому чи іншому обсязі та експлуатуються поряд зі створюваними клієнт-серверними додатками.

На рис. 2.1 представлено зображення клієнт-серверної архітектури:

Рис. 2.2. Клієнт-серверна архітектура

В архітектурі «клієнт-сервер» БД розміщується на комп'ютері-сервері мережі. Додаток, що здійснює роботу з цієї БД, знаходиться на комп'ютері користувача. Додаток користувача є клієнтом. Клієнт і сервер взаємодіють наступним чином: клієнт формує і відсилає запит (SQL-запит) сервера, на якому розміщена БД. Сервер виконує запит і видає клієнтові в якості результатів необхідні дані. До переваг такої архітектури відносяться:

– для роботи з даними використовується реляційний спосіб доступу, що знижує навантаження на мережу;

– додаток безпосередньо не керує базою, управлінням займається тільки сервер; в зв'язку з цим можна забезпечити високу ступінь захисту даних;

Такий підхід забезпечує вирішення трьох важливих завдань:

– зменшення навантаження на мережу;

– зменшення вимог до комп'ютерів-клієнтам;

– підвищення надійності та збереження логічної цілісності бази даних.

Тут додатки також виконуються, в основному, на робочих станціях. Додаток включає модулі для організації діалогу з користувачем і бізнес-правила (транзакції). Ядро СКБД є загальним для всіх робочих станцій і функціонує на сервері. Оператори звернення до СКБД (SQL-оператори), закодовані в транзакції, не виконуються на робочій станції, а пересилаються для обробки на сервер. Ядро СКБД транслює запит і виконує його, звертаючись для цього до індексів та інших проміжних даних. Назад на робочу станцію передаються тільки результати обробки оператора (рис. 2.2).

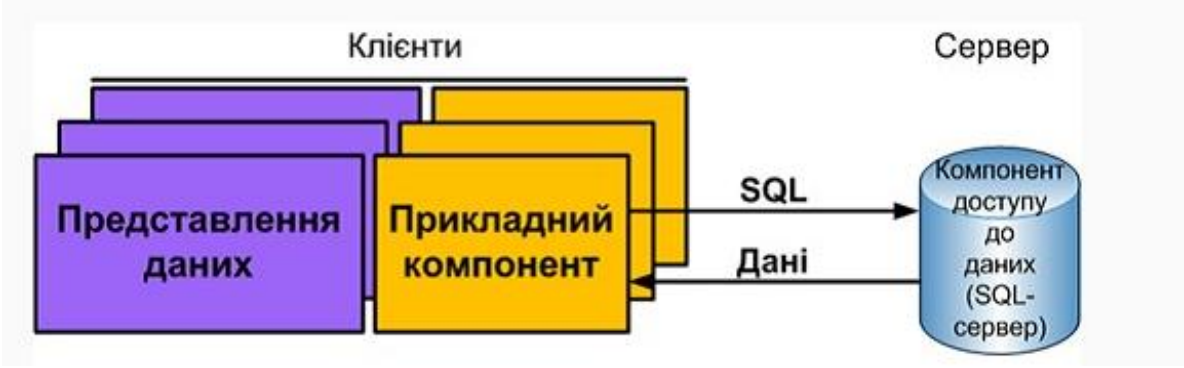


Рис. 2.2. Багатокористувальницька клієнт-серверна архітектура

У файлах БД на сервері знаходиться також і системний каталог БД.

У числі іншого, в каталог БД поміщаються:

– відомості про зареєстрованих користувачів;

– привілеї користувачів;

На клієнтських комп'ютерах встановлюються частини СКБД, які реалізують:

– інтерфейсні функції;

– прикладні функції.

Прикладний компонент включає:

– бібліотеки запитів;

– процедури обробки даних.

Прикладний компонент повністю розміщується і виконується на клієнтській частині - формує SQL-інструкції, що направляються SQL-серверу.

SQL-сервер - спеціальний програмний компонент, який:

– орієнтований на інтерпретацію SQL-інструкцій;

– високошвидкісне виконання низькорівневих операцій з даними;

– приймає і координує SQL-інструкції від різних клієнтів;

– виконує SQL-запити;

– перевіряє та забезпечує виконання обмежень цілісності даних;

– направляє клієнтам результати обробки SQL-інструкцій - набори даних.

2.4.2. Опис функціональних можливостей системи

Основним завданням інформаційної системи є автоматизація бізнес процесів діяльності СТО та документообігу. Розроблена система повинна відповідати ряду вимог.

Інформаційна система дозволяє:

– додавання, видалення та редагування інформації про товари і послуги. Передбачено для забезпечення найбільш ефективної роботи користувача видалення і редагування шляхом безпосереднього вибору записи з таблиці;

– перегляд інформації про товари і послуги;

– перегляд і друк інформації про продані товари, що містить відомості про найменування товару, кількість і ціну конкретного товару;

– перегляд і друк інформації каталогу товару;

– здійснення пошуку необхідної інформації про товари і послуги і сортування товару;

– здійснення операцій продажу і відстеження за кількістю товару, що продається;

– можливість входу в систему з різними рівнями доступу до даних: для користувача (здійснює тільки перегляд інформації, продаж товару і друк чека) і адміністраторський (здійснює всі можливі операції, представлені в системі);

– здійснення контролю введених даних: перевірка на відповідність типів, на введення обов'язкових полів даних, а також на введення тільки можливих значень, зчитувальних з необхідних таблиць;

– можливість перегляду інформації з таблиць в режимі реального часу.

Вхід в БД можливий тільки при введенні імені та пароля.

Схематичне зображення функціонування програми представлено на рис. 2.3.

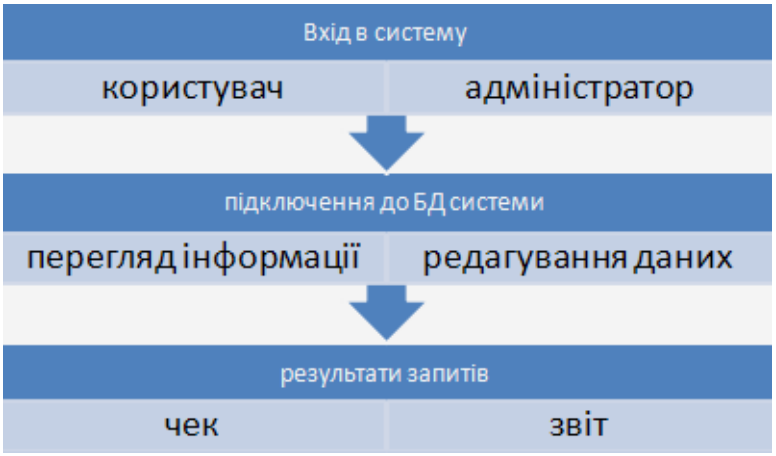


Рис 2.3. Функціонування програми

2.4.3. Опис структури додатку

Структура інформаційної системи являє собою клієнтське додаток, написаний на мові програмування високого рівня C#, що взаємодіє з базою даних «DiplomCTO». База даних розроблена на мові SQL в системі управління базами даних Microsoft SQL Server 2012.

Логічна схема роботи інформаційної системи зображена на рис. 2.4:

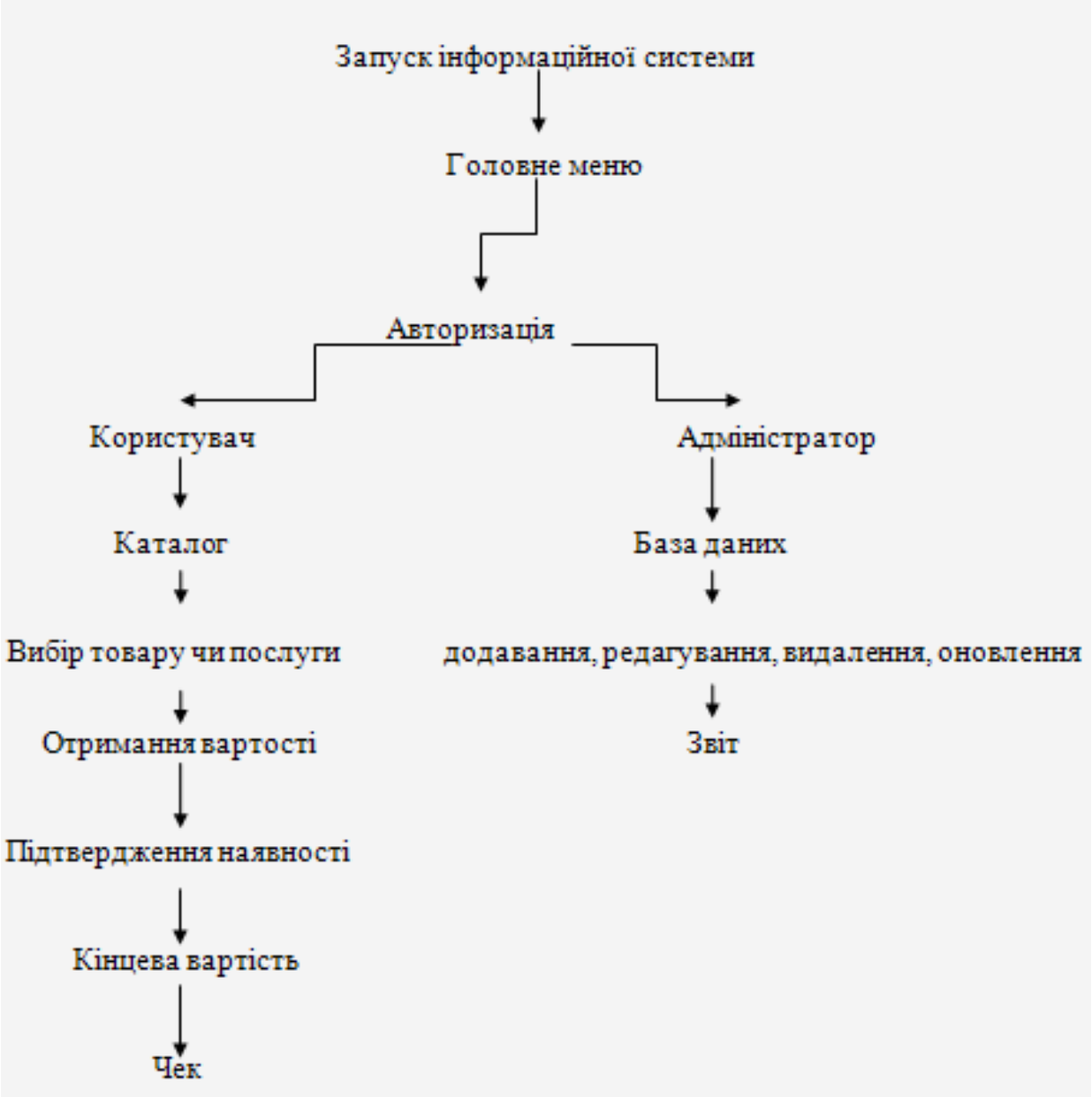


Рис. 2.4. Логічна схема роботи інформаційної системи

2.4.4. Розробка БД системи

Розроблений додаток зберігає дані, які використовуються для роботи в базі даних. В інформаційній системі використовуються наступні структури: список мастил, список шин, список послуг, список охол. рідин, список мийок, список постачальників і список нового товару. База даних складається з 9 таблиць, структура БД зображена на рис. 2.5.

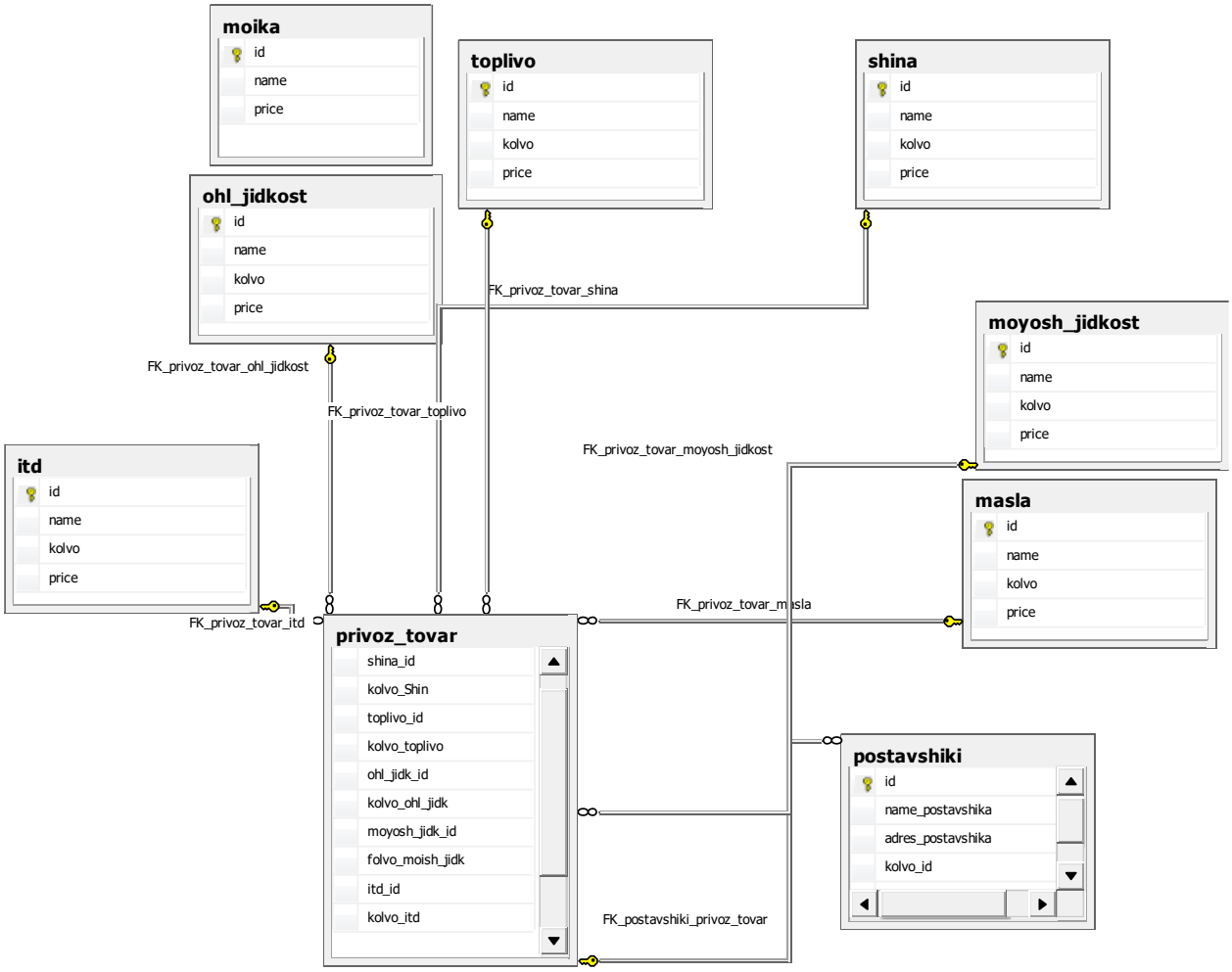


Рис 2.6. Структура БД системи

Таблиця «Мастила» (табл. 2.1.) містить задану інформацію про товар категорії мастил:

– ID_товара, тип ціле, (ПК);

– найменування, тип символьний, містить найменування товару;

– кількість, тип ціле, містить кількість даного товару;
– ціна, тип дійсний, містить ціну про цей товар.

Таблиця 2.1. «Мастила»

Имя столбца	Тип данных	Разрешить значения null
id	int	☐
name	nvarchar(50)	☐
kolvo	int	☐
price	float	☐

Таблиця «Шини» (табл. 2.2.) містить задану інформацію про товар категорії шини:

– ID_товара, тип ціле, (ПК);
– найменування, тип символьний, містить найменування товару;
– кількість, тип ціле, містить кількість даного товару;
– ціна, тип дійсне, містить ціну про цей товар.

Таблиця 2.2. «Шини»

Имя столбца	Тип данных	Разрешить значения null
id	int	☐
name	nvarchar(50)	☐
kolvo	int	☐
price	float	☐

Таблиця «Паливо» (табл. 2.3.) містять задану інформацію про товар категорії паливо:

– ID_товара, тип ціле, (ПК);
– найменування, тип символьний, містить найменування товару;
– кількість, тип ціле, містить кількість даного товару;
– ціна, тип дійсне, містить ціну про даний товар за літр.

Таблиця 2.3. «Паливо»

Имя столбца	Тип данных	Разрешить значения null
id	int	☐
name	nvarchar(50)	☐
kolvo	int	☐
price	float	☐

Таблиця «Мийка» (табл. 2.4.) містить задану інформацію про послуги категорії мийка:

– ID_послуги мийки, тип ціле, (ПК);
– найменування, тип символьний, містить найменування послуги;
– ціна, тип дійсне, містить ціну про послуги в Інтернеті.

Таблиця 2.4. «Мийка»

Имя столбца	Тип данных	Разрешить значения null
id	int	☐
name	nvarchar(50)	☐
price	float	☐

Таблиця «Постачальники» (табл. 2.5.) містить інформацію про постачальників:

– ID_постачальника, тип ціле, (ПК);
– найменування постачальника, тип символьний;
– адреса постачальника, тип символьний;
– ID_нового товару, тип ціле, (ВК);
– дата поставки, тип дата і час.

Таблиця 2.5. «Постачальники»

Имя столбца	Тип данных	Разрешить значения null
id	int	
name_postavshika	nvarchar(50)	
adres_postavshika	nvarchar(50)	
kolvo_id	int	
data_post	datetime	

Таблиця «Охолоджуючі рідини» (табл. 2.6.) містить задану інформацію про товар категорії охолоджуючі рідини:

– ID_товара, тип ціле, (ПК);
– найменування, тип символьний, містить найменування товару;
– кількість, тип ціле, містить кількість даного товару;
– ціна, тип дійсне, містить ціну про цей товар.

Таблиця 2.6. «Охолоджуючі рідини»

Имя столбца	Тип данных	Разрешить значения null
id	int	
name	nvarchar(50)	
kolvo	int	
price	float	

Таблиця «Новий товар» (табл. 2.7.) Містить інформацію про новий товар.

– ID_завозу, тип ціле, (ПК);
– ID_товара масла, тип ціле, (ВК);
– кількість мастила, тип ціле, містить кількість мастила;
– ID_товара шини, тип ціле, (ВК);
– кількість шин, тип ціле, містить кількість шин;
– ID_товара паливо, тип ціле, (ВК);
– кількість палива, тип ціле, містить кількість палива;

– ID_товара охолоджуючі рідини, тип ціле, (ВК);
– кількість охолоджуючою рідини, тип ціле;
– ID_товара миючі рідини, тип ціле, (ВК);
– кількість миючої рідини, тип ціле;
– ID_послуг, тип ціле, (ВК);
– кількість , тип ціле.

Таблиця 2.7. "Новий товар"

Имя столбца	Тип данных	Разрешить значения null
id	int	☐
masla_id	int	☐
kolvo_masla	int	☐
shina_id	int	☐
kolvo_Shin	int	☐
toplivo_id	int	☐
kolvo_toplivo	int	☐
ohl_jidk_id	int	☐
kolvo_ohl_jidk	int	☐
moyosh_jidk_id	int	☐
folvo_moish_jidk	int	☐
itd_id	int	☐
kolvo_itd	int	☐

Таблиця «Миючі рідини» (табл. 2.8.) містить задану інформацію про товар категорії миючі рідини:

– ID_товара, тип ціле, (ПК);
– найменування, тип символьний, містить найменування товару;
– кількість, тип ціле, містить кількість даного товару;
– ціна, тип дійсне, містить ціну про цей товар.

Таблиця 2.8. «Миючі рідини»

Имя столбца	Тип данных	Разрешить значения null
id	int	☐
name	nvarchar(50)	☐
kolvo	int	☐
price	float	☐

Таблиця «Послуги» (табл. 2.9.) містить задану інформацію про товар категорії паливо:

– ID_услугі, тип ціле, (ПК);

– найменування, тип символьний, містить найменування послуги;

– кількість, тип ціле, містить можливу кількість послуг;

– ціна, тип дійсне, містить ціну послуги.

Таблиця 2.9. «Послуги»

Имя столбца	Тип данных	Разрешить значения null
id	int	☐
name	nvarchar(50)	☐
kolvo	int	☐
price	float	☐

2.4.5. Опис основних процесів програми

Структура додатку складається з БД, яка містить в собі необхідну інформацію та інтерфейсу обробки цих даних. Графічний інтерфейс системи надає доступ до таблиць даних у вигляді довідників, робота з якими виконується на відповідних формах програми. Схема роботи з цими даними зображена на рис. 2.6.

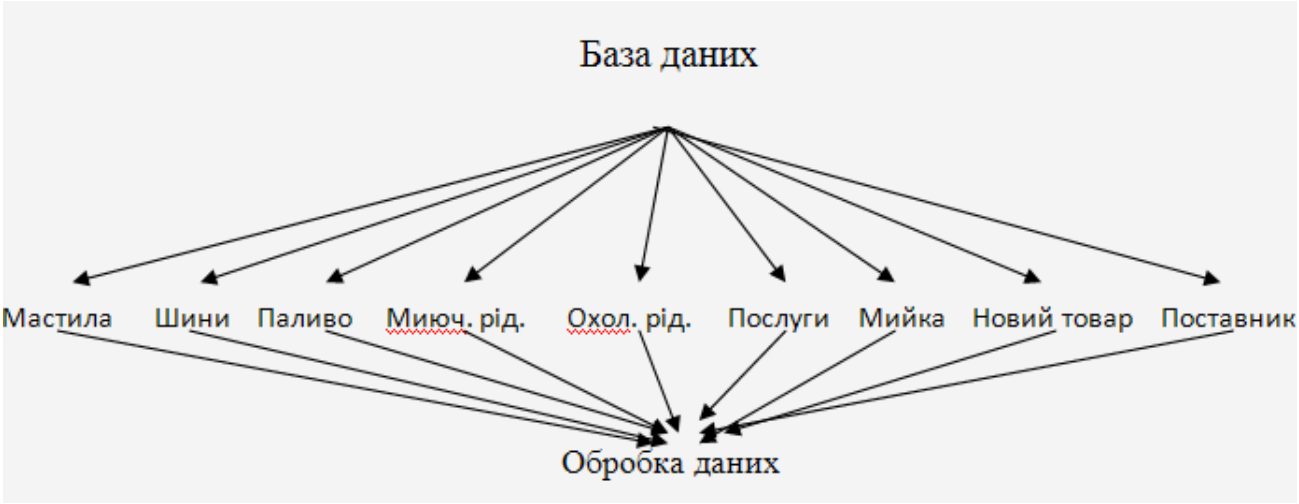


Рис. 2.6. Схема роботи з даними системи

Схема пошуку записів в таблицях зображена на рис. 2.7:

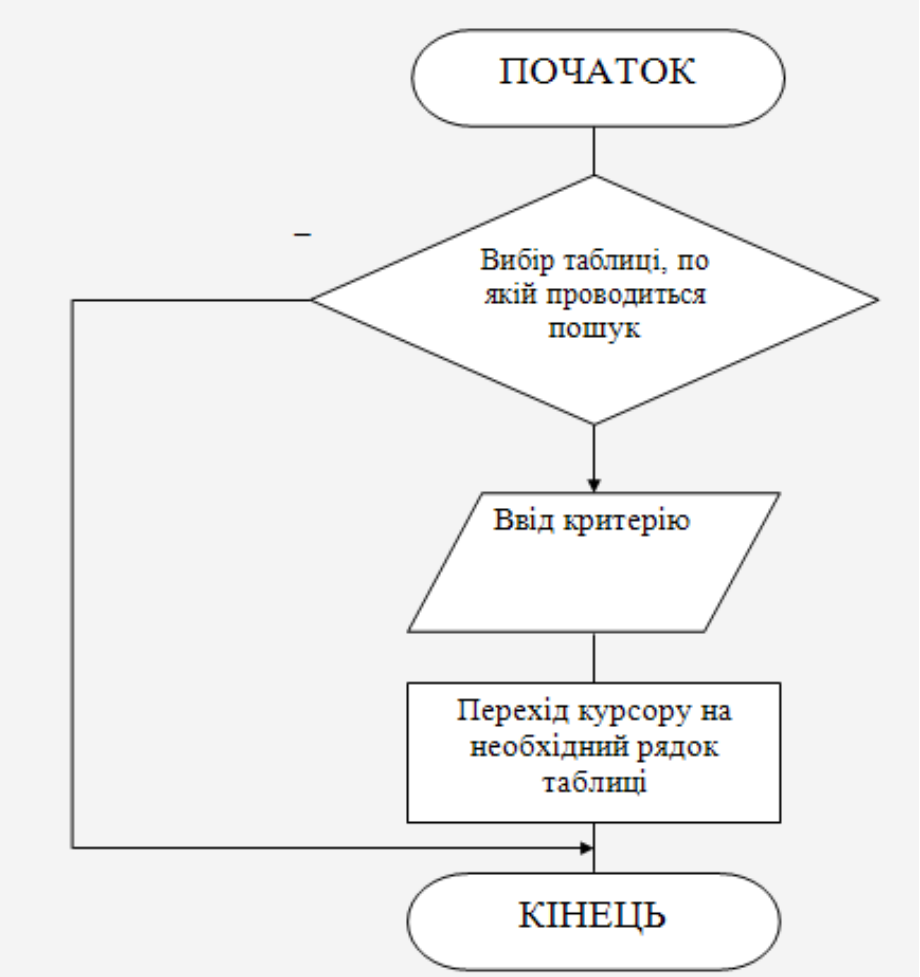


Рис 2.7. Схема пошуку записів в таблицях

На рис. 2.8. зображена схема фільтрації таблиці:

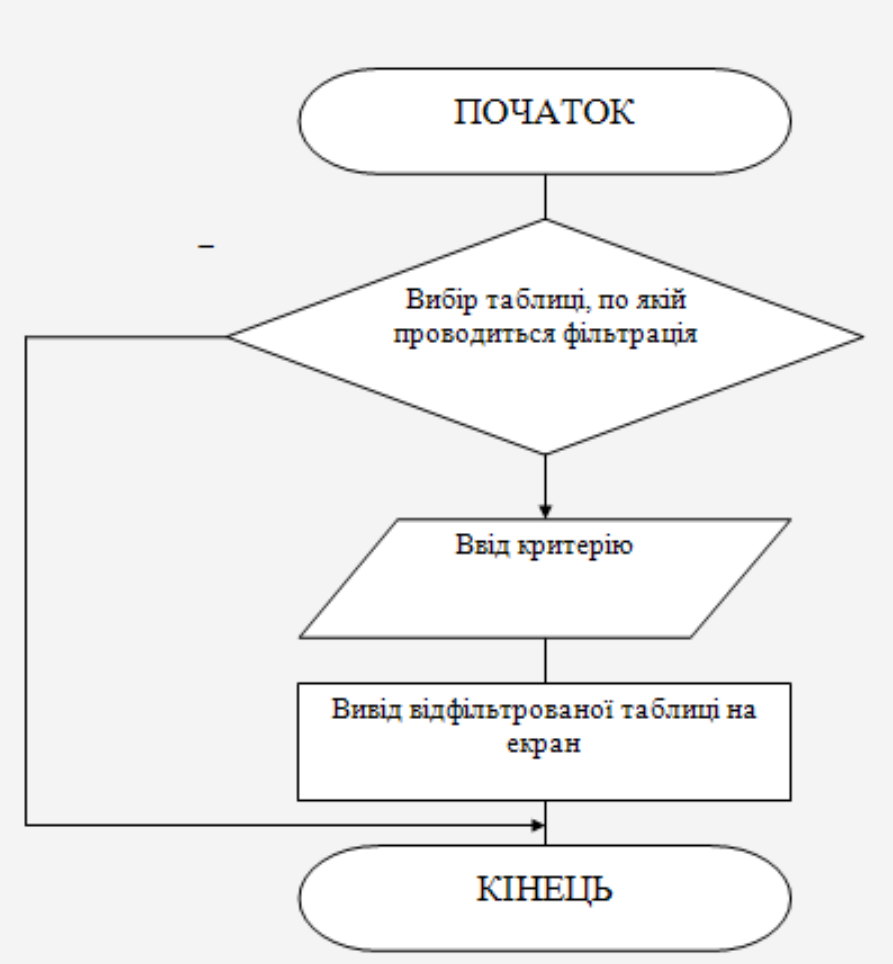


Рис 2.8. Схема процесу фільтрації таблиць

Схема оформлення замовлення представлена на рис. 2.9.

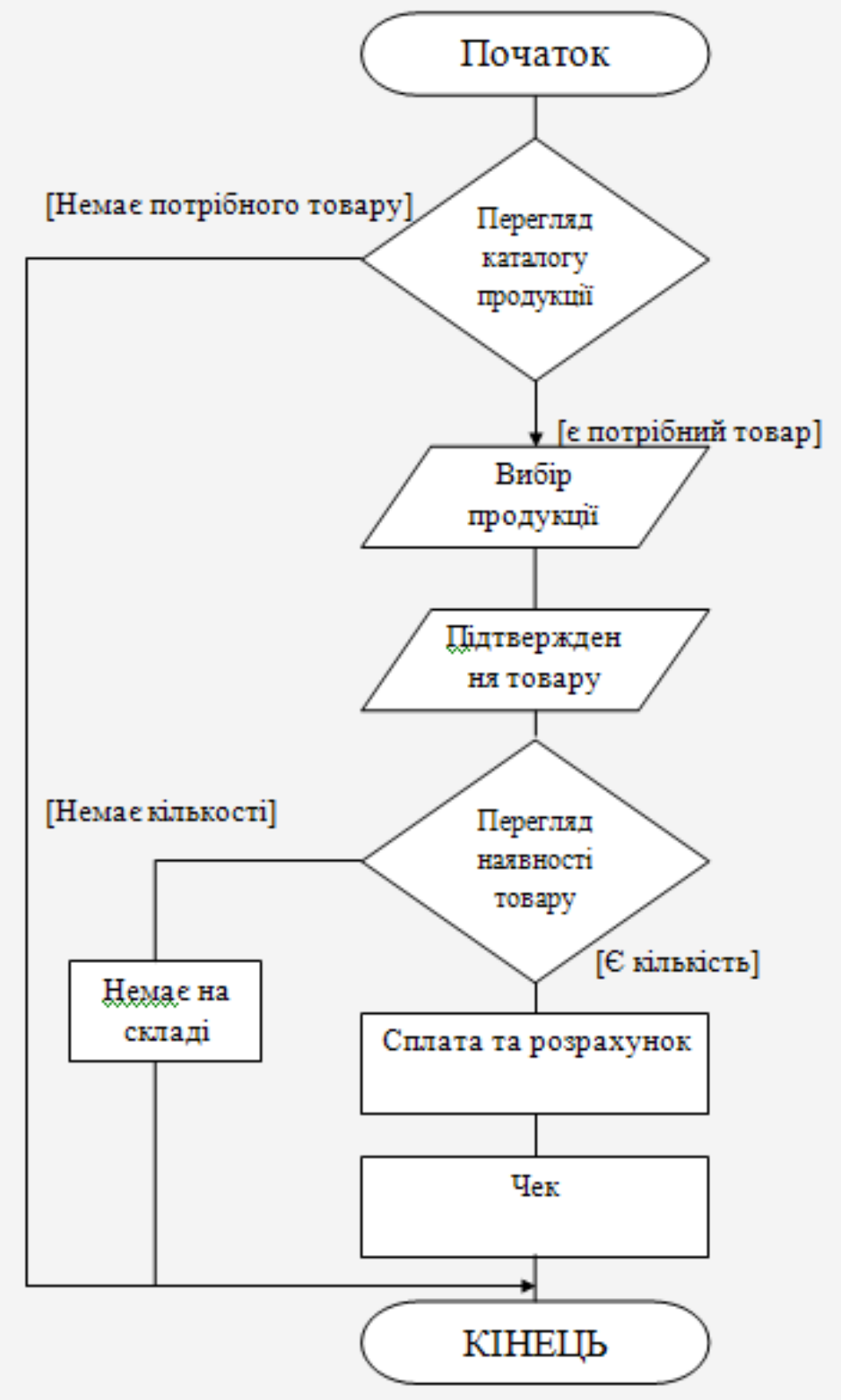


Рис 2.9. Схема оформлення замовлення

2.4.6. Програмна реалізація додатку

Лістинг коду бази даних «DiplomCTO» містить інформацію про таблиці БД, про її поля, первинні і вторинні ключі та про типи зв'язку між таблицями:

```
CREATE TABLE `masla` (
  `id` int,
  `name` nvarchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `kolvo` int,
  `price` float)
ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
CREATE TABLE `shina` (
  `id` int,
  `name` nvarchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `kolvo` int,
  `price` float)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
CREATE TABLE `toplivo`  (
`id` int UNSIGNED NOT NULL,
  `name` nvarchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `kolvo` int,
  `price` float)
ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
CREATE TABLE `moika` (
    (
`id` int UNSIGNED NOT NULL,
  `name` nvarchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
  `kolvo` int,
  `price` float)
```

```
 ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
CREATE TABLE `postavshiki` (
 `id` int,
 `name_postavshika`nvarchar(50),
 `adres_postavshika`nvarchar(50),
 `kolvo-id` int ,
 `data_post` datetime
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
CREATE TABLE `ohl_gidkost` (
  `id` int,
 `name` nvarchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
 `kolvo` int,
 `price` float)
 ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
CREATE TABLE `moyush_gidkost` (
  `id` int,
 `name` nvarchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
 `kolvo` int,
 `price` float)
 ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
CREATE TABLE `std` (
  `id` int,
 `name` nvarchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
 `kolvo` int,
 `price` float)
 ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

```
CREATE TABLE `privoz_tovar` (
  `id` int,
`masla_id`
`kolvo_masla`
`shina_id`
`kolvo_shina` int,
`toplivo_id` int,
`kolvo_toplivo` int,
`ohl_jidk_id` int,
`kolvo_jidk_id` int,
`moyush_jidk_id` int,
`kolvo_moyush_jidk` int,
`std_id` int,
`kolvo_std` int,
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Інформаційна система містить наступні основні файли:

– DiplomCTO.exe: файл запуску програми;

– Main.cs: файл основної форми, на якій знаходяться всі головні елементи для користувача, списки, кнопки, і поля для введення даних. Через цей файл ми можемо перейти до авторизації для входу в редагування БД, перейти на форму роздруківки прайс-листів, роздрукувати вартість обраного товару і роздрукувати чек;

– DataBase.cs: файл форми, на якій розміщені всі необхідні компоненти для зв'язку програми з БД, цей файл пов'язан майже з усіма іншими файлами оскільки на ньому розміщені і компоненти для друку тощо;

– Parol.cs: файл форми, яка реалізує авторизацію для входу на форму редагування БД;

– Katalog.cs: файл форми, яка реалізує вивід на екран інформації про БД та має елементи для додавання, видалення, редагування і пошуку по БД.

– PrintKatalog.cs: файл форми, яка реалізує вивід на екран компоненти для роздруківки прайс-листів обраного виду товару.

На рис. 2.10 зображена схема взаємодії файлів програми:

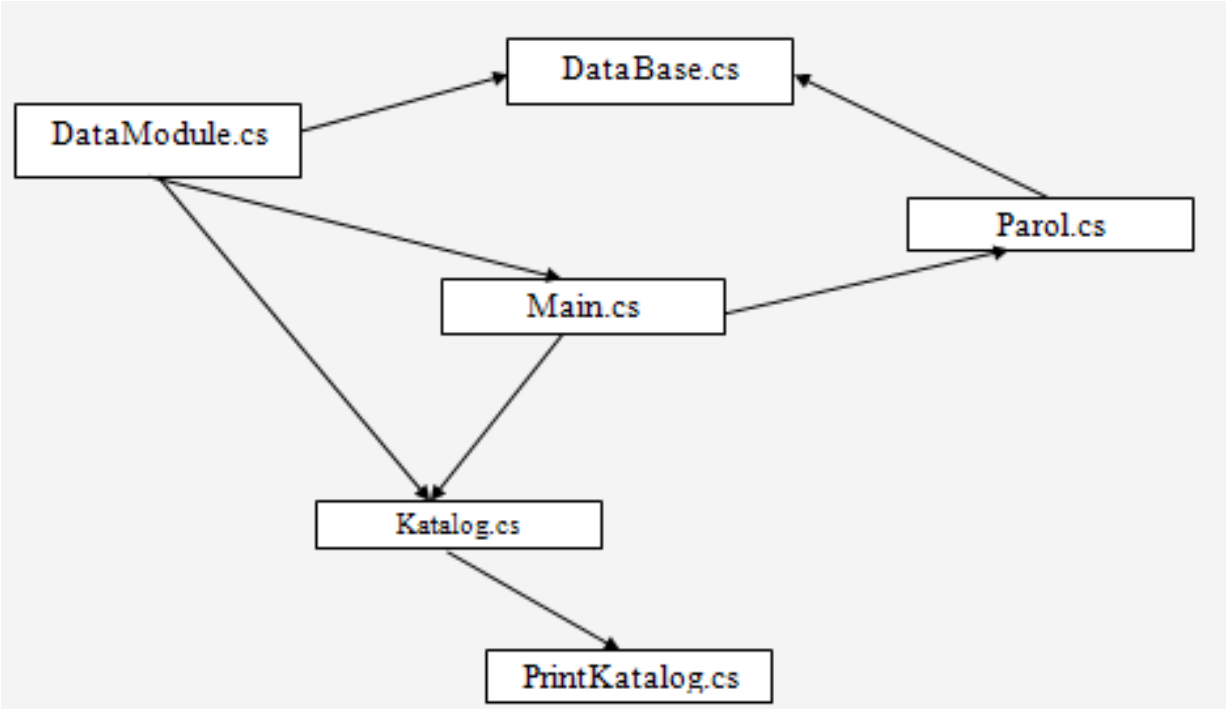


Рис. 2.8. Схема взаємодії файлів програми

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Розроблений додаток зберігає дані, які використовуються для роботи в базі даних.

Вхідними даними даної системи є наступна інформація

– мастила (артикул, найменування, кількість, ціна);
– шини (артикул, найменування, кількість, ціна);
– паливо (артикул, найменування, кількість, ціна);
– послуги (артикул, найменування, кількість, ціна);
– охолоджуючі рідини (артикул, найменування, кількість, ціна);
– хімічні речовини або рідини (артикул, найменування, кількість, ціна);

– мийка (артикул, найменування, ціна);

– постачальник (артикул, найменування, артикул нового товару, дата);

– новий товар (артикул, артикул палива, кількість палива, артикул шин. кількість шин, артикул масла, кількість масла, артикул охол. рідини, кількість охол. рідини, артикул миюч. рідини, кількість миюч. рідини);

Вихідними даними є інформація, яку може використовувати користувач даної системи:

– вибір конкретного товару;

– вивід вартості обраних товарів та послуг;

– перегляд і друк звітів: чек для клієнта і прайс-листи товарів.

У проекті були створені такі звіти:

1. Звіт «Чек».

2. Звіт «Каталог товарів».

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для розробки ІС використовувався комп’ютер з наступними параметрами:

– операційна система Windows XP/7/8/10;

– процесор Dual Core Intel чи AMD з частотою 2.8 GHz;

– оперативна пам'ять 4GB ОЗУ;

– відеокарта nVidia GeForce 8600/9600GT, ATI/AMD

– Radeon HD2600/3600;

– DirectX версії 9.0c;

– жорсткий диск мінімум в 15 GB;

– звукова карта DirectX Compatible.

2.6.2. Використані програмні засоби

Для розробки проекту було використано мову програмування C#, а середовищем розробки обрана Microsoft Visual Studio 19. Програмний додаток було виконано за допомогою системи керування базами даних (БД) SQL Server 2012.

2.6.3. Виклик та завантаження програми

Для запуску інформаційної системи необхідно скопіювати папку проекту DiplomCTO, та за допомогою файлу у кореневій папці DiplomCTO.exe завантажити програму.

2.6.4. Опис інтерфейсу користувача

Після запуску виконуваного файлу програми на екрані з'являється головна форма програми, зображена на рис. 2.9:

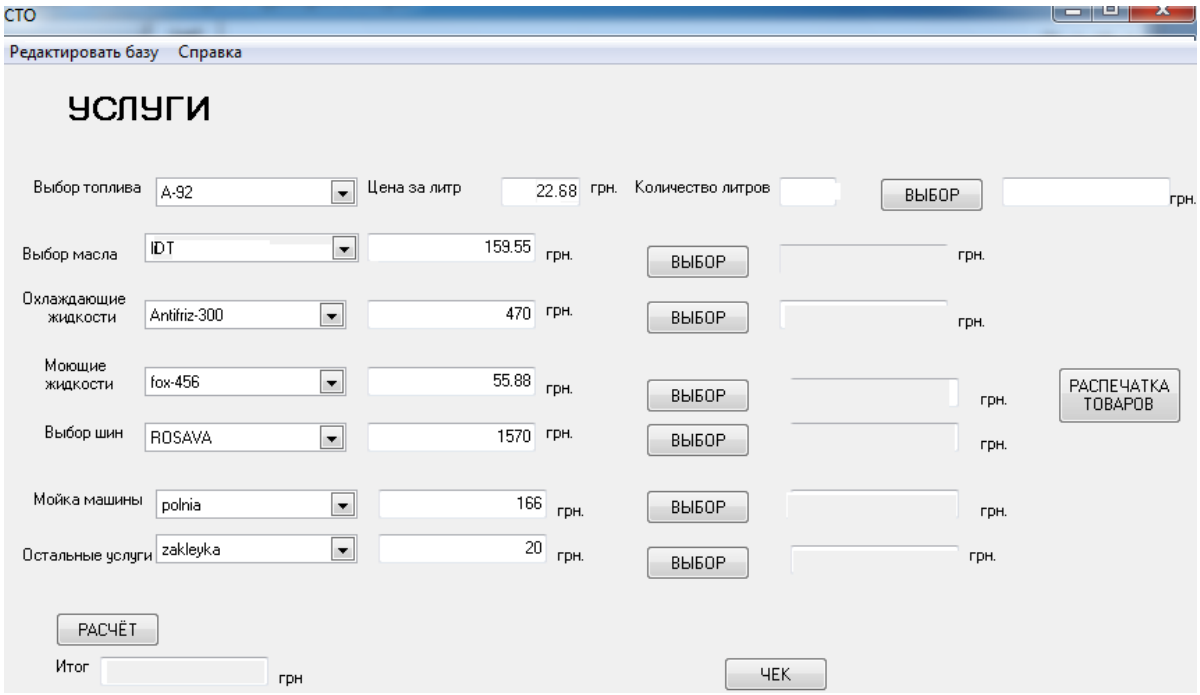


Рис. 2.9. Форма головного вікна програми

На цій формі користувачеві біля кожного спливаючого вікна зазначено вид товару. У спливаючих вікнах він може вибирати відповідний товар з бази даних, при цьому буде виводиться його ціна і при натисканні на кнопку «ВИБІР» ціна товару потрапляє в поле навпроти і при цьому з бази даних віднімається цей товар.

У випадку з вибором палива, з'являється ціна за один літр, тому потрібно вказати кількість літрів у відповідному рядку і також натиснути на кнопку «ВИБІР», від кількості літрів відніметься то, яке ввів користувач, а ціна у відповідному рядку з'явиться за введені кількість літрів (рис. 2.10).

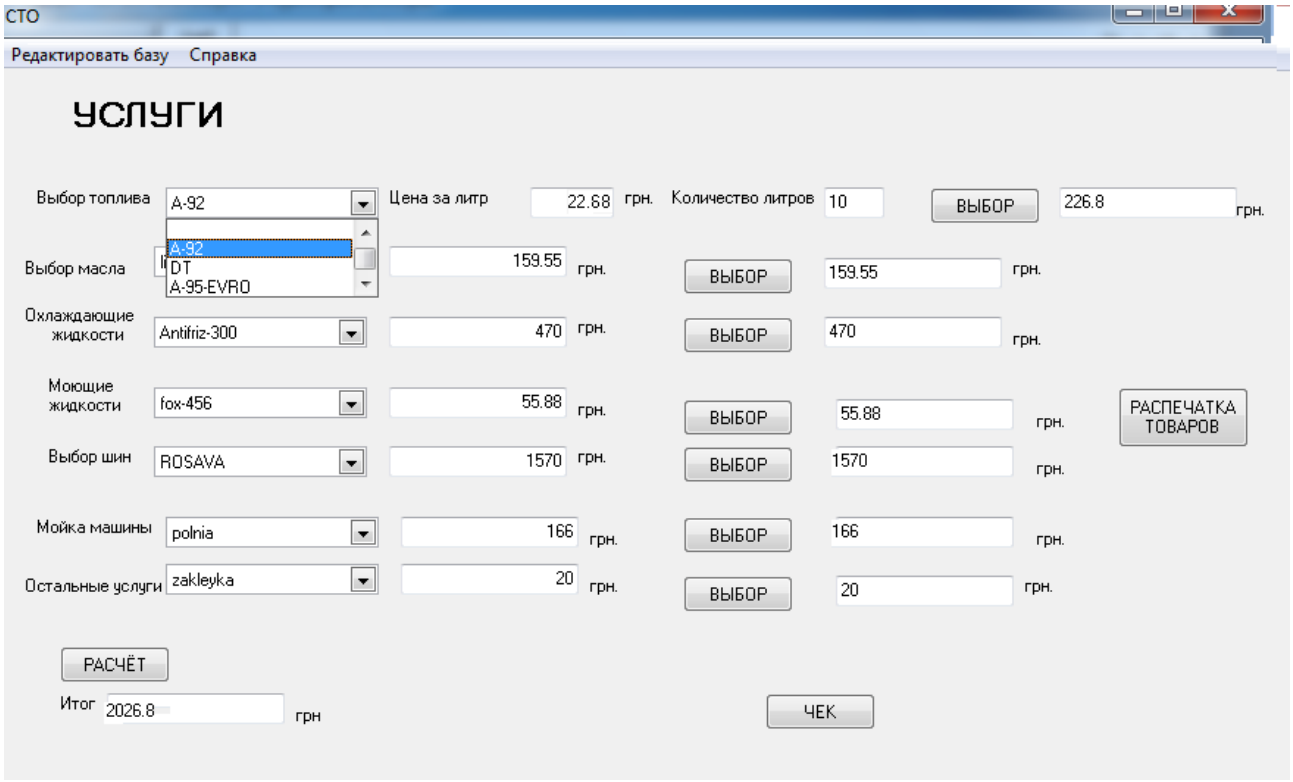


Рис. 2.10. Вибір користувачем товарів з БД системи

Якщо ж необхідної кількості товару на складі немає, то перед користувачем з'явиться попередження, показане на рис. 2.11.

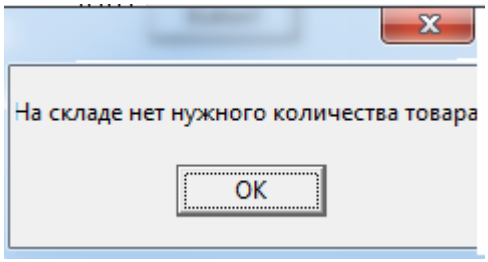


Рис. 2.10. Повідомлення при відсутності необхідної кількості товару на складі

Після вибору всіх товарів, потрібних покупцеві, натискаємо на кнопку «РОЗРАХУНОК» і в рядку «Расчет» програма виводить суму до оплати.

Після того, як користувач вибрав потрібні клієнтові товари, він натискає на кнопку «ЧЕК» щоб роздрукувати клієнтові чек за послуги і проданий товар (рис. 2.11).

Чек.xlsx

	A	B	C	D	E	F	G	H
1								
2				Станция технического обслуживания				
3						Дата	21.04.2021	
4						время	13:40	
5								
6		№ пп	Тип	вид	Цена за ед., грн	кол-во	Всего, грн	
7		1	Топливо	A-92	22,68	10	226,8	
8		2	Масла	litol-45	159.55	1	159,55	
9		3	Охл. Жидкости	Antifriz-300	470	1	470	
10		4	Моющ. Жидкости	fox-456	55.88	1	55,88	
11		5	Шины	ROSAVA	1570	1	1570	
12		6	Мойка авто	polnaja	66	1	66	
13		7	Ост. Услуги	naklejka	20	1	20	
14						Итого	2181,88	грн
15								

Рис. 2.11. Чек для клієнта

Також в головному мене є можливість роздрукувати каталоги певних товарів, натиснувши на кнопку «Распечатка товаров». Після чого відкриється вікно «Прайс-листи» (рис. 2.12.).

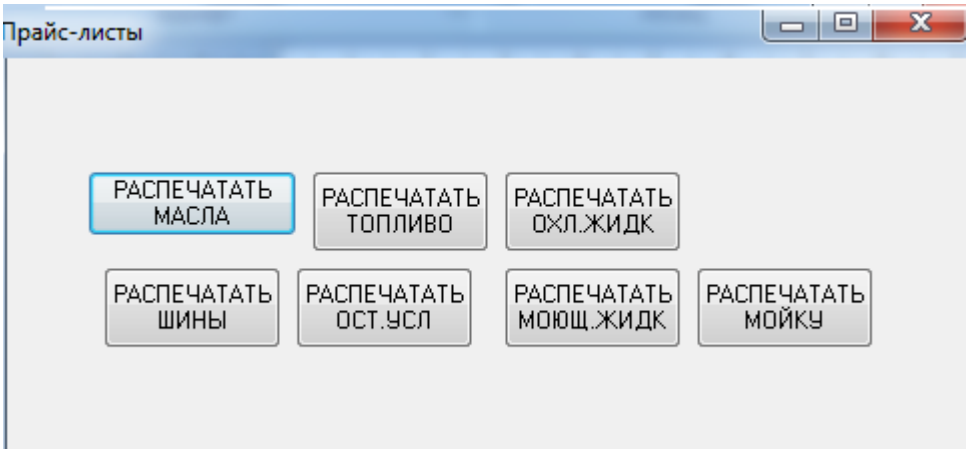


Рис. 2.12. .Прайси

В даному вікні можна роздрукувати прайс-лист обраної категорії товарів. Приклади прайс-листів різних категорій товарів та послуг показані на рис 2.13-2.16:

Каталог_охл_жидк.xlsx

	A	B	C	D	E	F	G	H	I
1									
2				**КАТАЛОГ ТОВАРОВ**					
3									
4		Тип товара:		**охлаждающие жидкости**					
5									
6					вид	kol-vo	price		
7					tipol	76	66		
8					rek-8	89	89		
9					motol	34	123		
10					tok-3	334	120		
11					Antifriz-3	45	470		
12									
13									
14		Дата:	21.04.2021						
15		время:	15:26						
16									
17									

Рис. 2.13. Каталог товарів (охолоджуючі рідини)

Каталог_моющ_жидк.xlsx

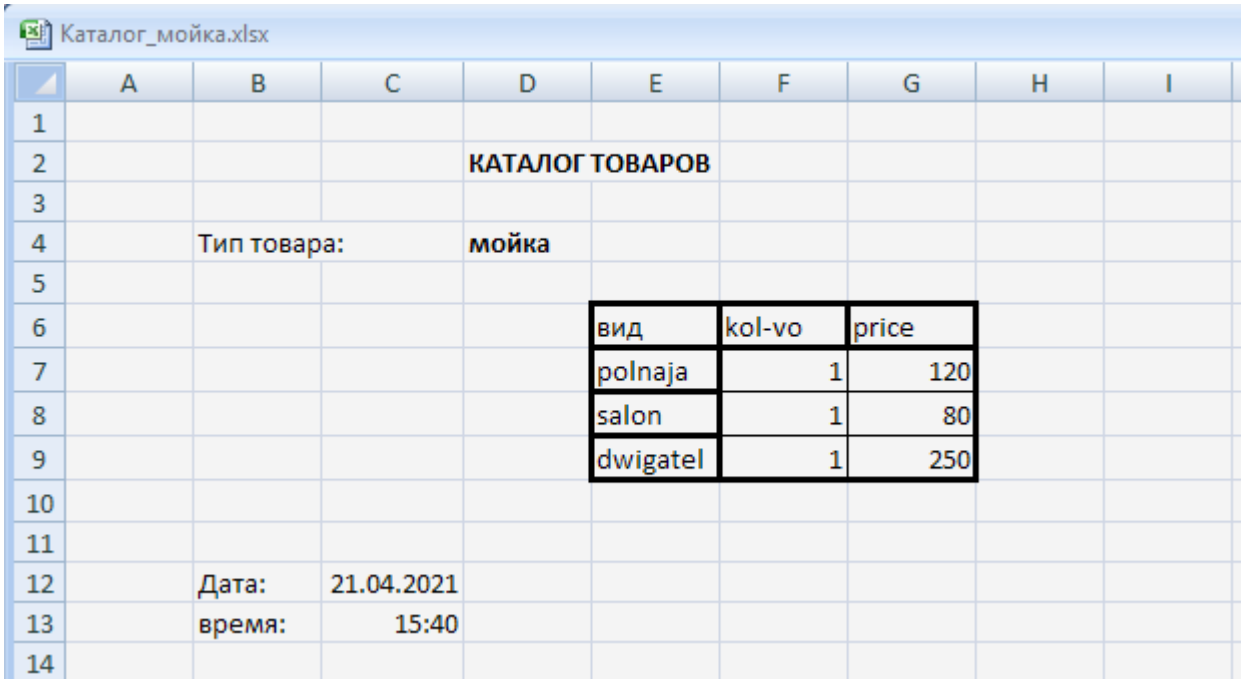
	A	B	C	D	E	F	G	H
1								
2				**КАТАЛОГ ТОВАРОВ**				
3								
4		Тип товара:		**моющие жидкости**				
5								
6					вид	kol-vo	price	
7					Astra	14	20	
8					VC-23	15	45	
9					REX	17	123	
10					Interio	13	45	
11					Combo	15	48	
12					QWEDS-1	14	62	
13					Antifriz-3	12	25	
14								
15								
16		Дата:	21.04.2021					
17		время:	15:28					
18								

Рис. 2.14. Каталог товарів (миючі рідини)

Каталог_шин.xlsx

	A	B	C	D	E	F	G	H
1								
2				**КАТАЛОГ ТОВАРОВ**				
3								
4		Тип товара:		**шины**				
5								
6					вид	kol-vo	price	
7					REX	10	1500	
8					ROSAVA	12	1540	
9					DEMI	22	1200	
10					CORONA	24	2580	
11								
12								
13		Дата:	21.04.2021					
14		время:	15:35					
15								

Рис. 2.15. Каталог товарів (шини)



Каталог_мойка.xlsx

	A	B	C	D	E	F	G	H	I
1									
2				**КАТАЛОГ ТОВАРОВ**					
3									
4		Тип товара:		**мойка**					
5									
6					вид	kol-vo	price		
7					polnaja	1	120		
8					salon	1	80		
9					dwigatel	1	250		
10									
11									
12		Дата:	21.04.2021						
13		время:	15:40						
14									

Рис. 2.16. Каталог товарів (мойка)

Для того, щоб мати можливість змінювати базу даних, необхідно в меню вибрати пункт «Редактировать базу» (рис 2.17).

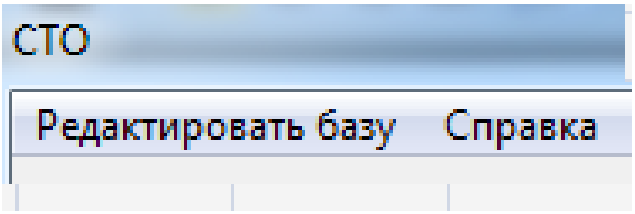


Рис. 2.17. Пункт меню «Редактировать базу»

Після цього відкриється вікно авторизації «Вхід в БД» для входу до бази даних проекту (рис. 2.18):

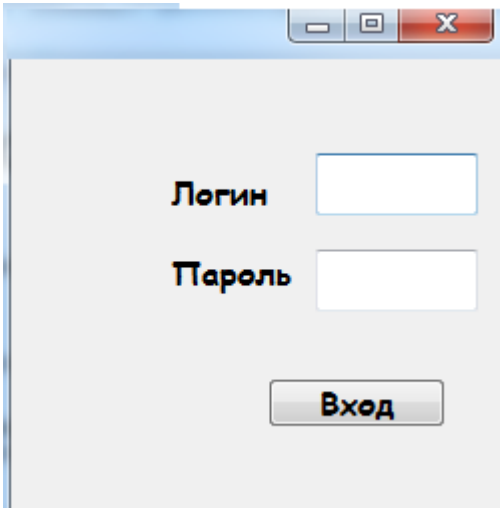


Рис. 2.18. Вікно авторизації

Якщо пароль або логін введений невірно, з'явиться наступне повідомлення (рис. 2.19).

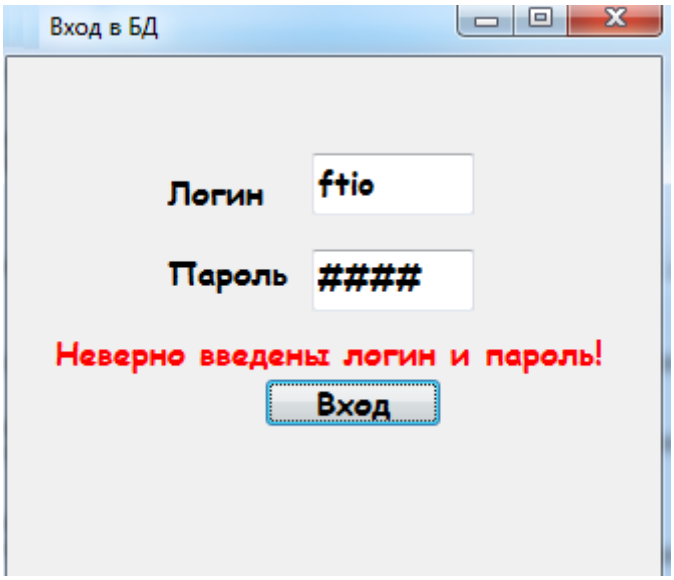


Рис. 2.19. Повідомлення про невірний ввід логіну чи паролю

Якщо ж логін і пароль введені вірно, то відкриється редактор БД «База даних» (рис. 2.20):

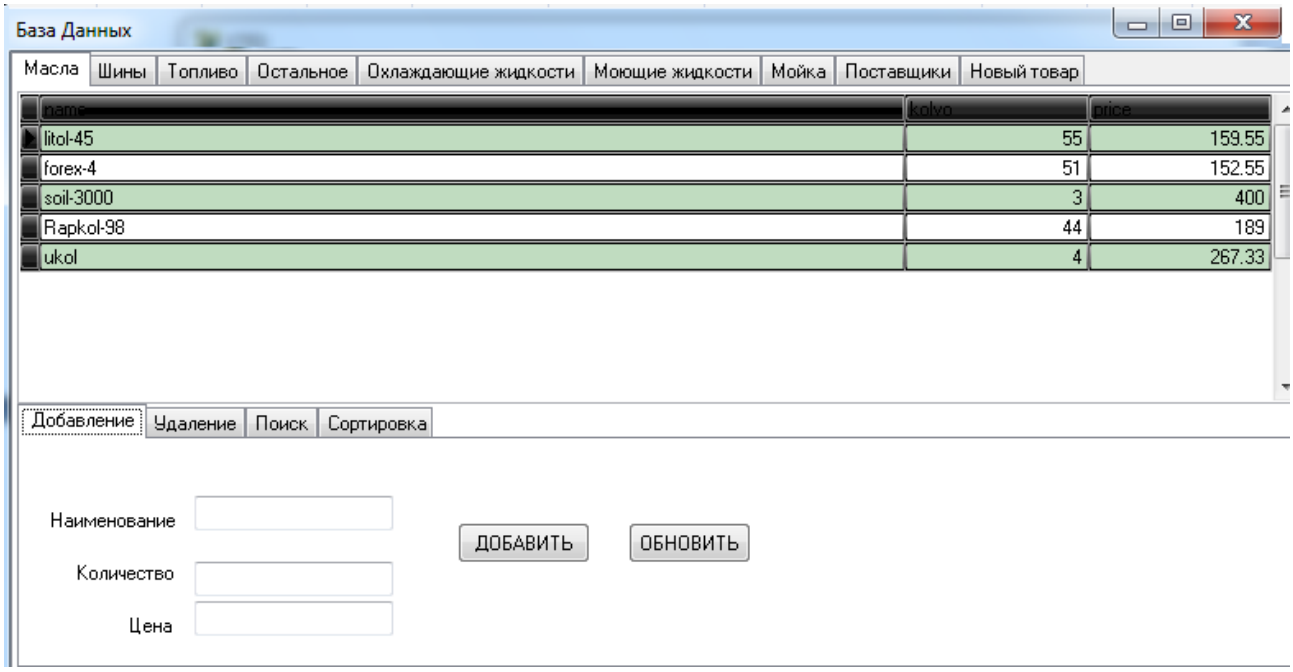


Рис. 2.20. База даних системи на вкладці «Масла»

Після входу в редактор БД можна побачити наступні таблиці: «Масла», «Шини», «Паливо», «Інше», «Охолоджуючі рідини», «Миючі рідини», «Мийка», «Постачальники», «Новий товар», які розташовані на відповідних вкладках.

Щоб додати новий товар або послугу до БД, потрібно ввести дані у відповідні поля для введення і натиснути кнопку «ДОБАВИТЬ», після чого товар з'явиться в БД.

Кнопка «ОБНОВИТЬ» оновлює базу даних на випадок, якщо в ній відбулися якісь зміни.

Для того, щоб видалити будь-який товар, наприклад, шини, необхідно перейти на вкладку «Шини», далі перейти на вкладку «Удалить», виділити потрібний товар в таблиці і натиснути на кнопку «УДАЛИТЬ» (рис. 2.21):

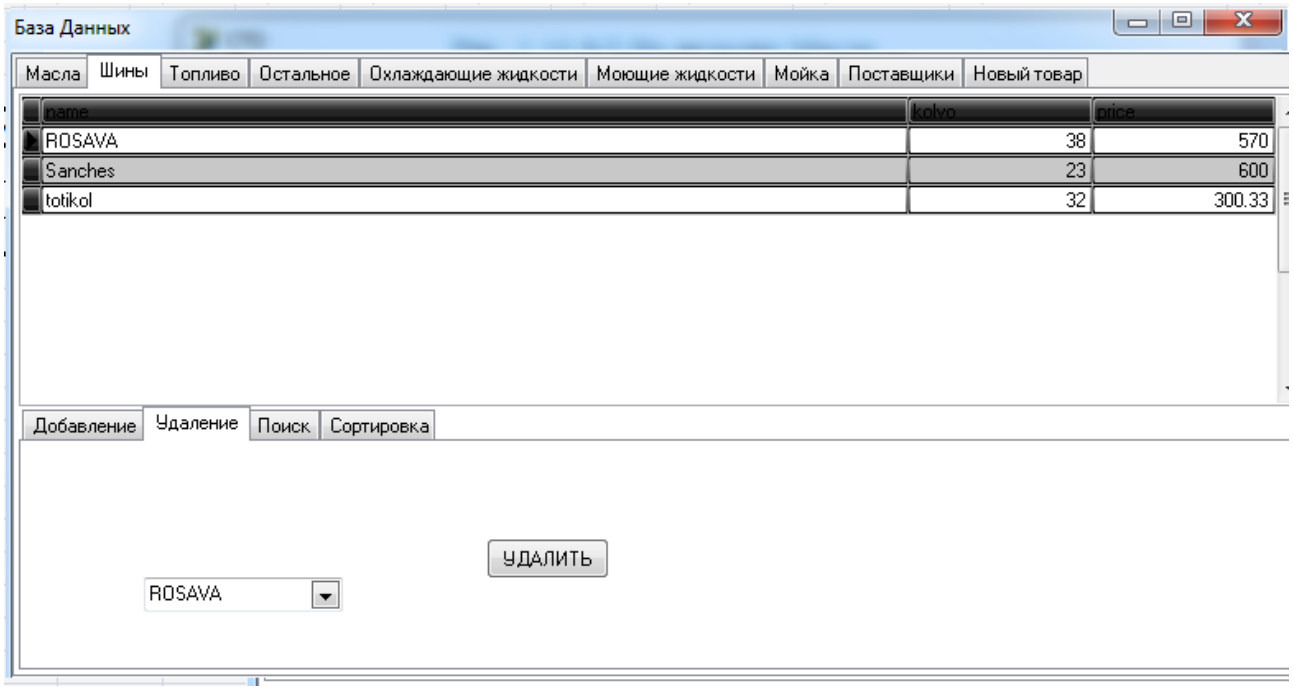


Рис. 2.21. Видалення товару з БД

Для впорядкування товару необхідно перейти на вкладку «Сортировка» і поставити галочку «сортировка по имени» (рис. 2.22):

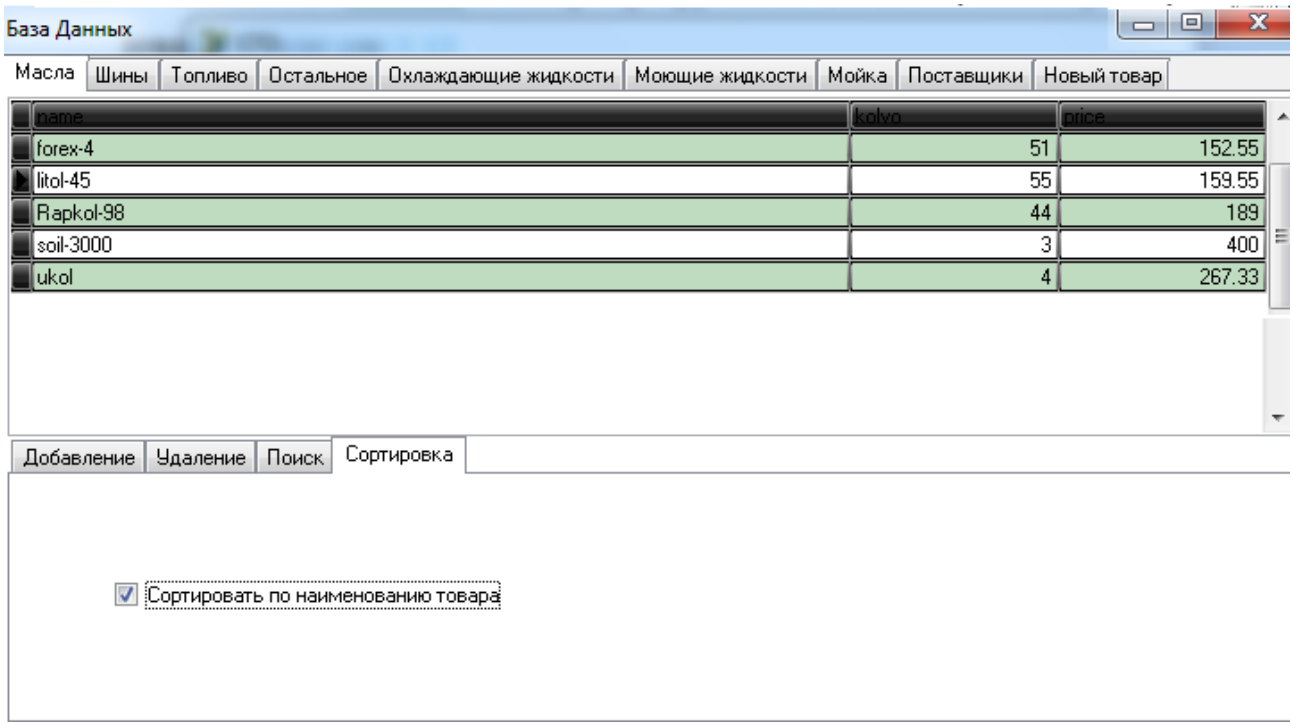


Рис. 2.22. Сортування за іменем

Якщо ж необхідно знайти певний товар, потрібно перейти на вкладку «Поиск» і ввести найменування товару у відповідному полі «Введите название товара» (рис. 2.23):

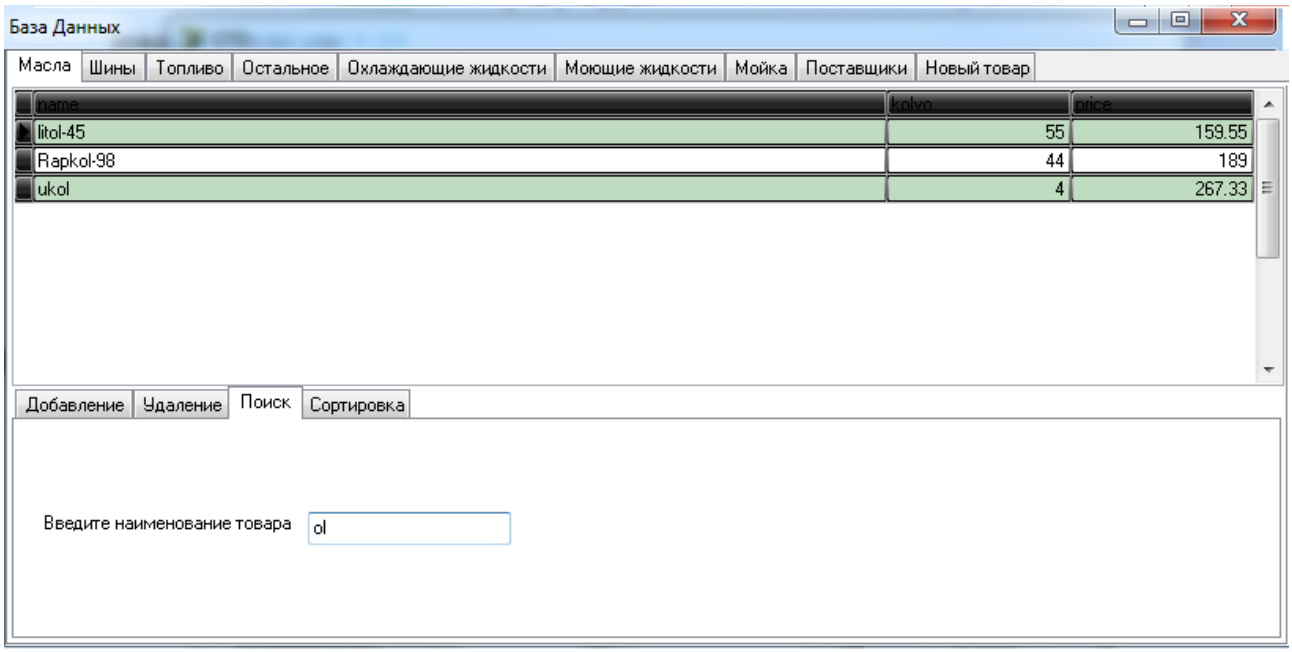


Рис. 2.23. Пошук по таблиці «масла»

Перевага цього пошуку полягає в тому, що варто ввести одну або кілька букв товару в таблиці і програма автоматично залишить товари з комбінацією цих букв в будь-якому місці. Редагування ж можна робити в полях самої таблиці, завдяки пошуку це дуже зручно.

Для виходу з системи необхідно нажати на відповідний значок на головному вікні додатку, при цьому всі дані автоматично зберігаються в БД, закриваються всі файли, закінчується робота всіх запущених функцій та виконується безпечний вихід з програми.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 800;
- коефіцієнт складності програми - 1,5;
- коефіцієнт корекції програми в ході її розробки - 0,05;
- годинна заробітна плата програміста, грн / год - 40.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де:

to - витрати праці на підготовку й опис поставленої задачі (приймається 50);

ti - витрати праці на дослідження алгоритму рішення задачі;

ta - витрати праці на розробку блок-схеми алгоритму;

tп - витрати праці на програмування по готовій блок-схемі;

tотл - витрати праці на налагодження програми на ЕОМ;

tд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \quad (3.2)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 800 \cdot 1{,}5 \cdot (1 + 0{,}05) = 1260 \text{ людино-годин.} \quad (3.3)$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \quad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; B=1.2 ... 1.5;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{1260 \cdot 1{,}2}{80 \cdot 0{,}8} = 23 \text{ , людино-годин.} \quad (3.4)$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \quad (3.5)$$

$$t_a = \frac{1260}{20 \cdot 0{,}8} = 78 \text{ людино-годин.} \quad (3.6)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \quad (3.7)$$

$$t_n = \frac{1260}{25 \cdot 0{,}8} = 63 \text{ людино-годин.} \quad (3.8)$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \quad (3.9)$$

$$t_{отл} = \frac{1260}{4 \cdot 0{,}8} = 393 \text{ людино-годин.} \quad (3.10)$$

за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл}; \quad (3.11)$$

$$t_{отл} = 393 * 1{,}2 = 472{,}5 \quad (3.12)$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.13)$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \quad (3.14)$$

$$t_{др} = \frac{1260}{15 \cdot 0{,}8} = 105 \text{ людино-годин,} \quad (3.15)$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \quad (3.16)$$

$$tdo = 0{,}75 \cdot 105 = 78 \quad (3.17)$$

$$t_д = 105 + 78 = 183 \text{ людино-годин.} \quad (3.18)$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 23 + 78 + 63 + 472 + 183 = 871 \text{ людино-годин.} \quad (3.19)$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.20)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.20)$$

де:

t - загальна трудомісткість, людино-годин;

Спр - середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 871 \cdot 40 = 34865 \text{ грн.} \quad (3.21)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}\text{ , грн,} \quad (3.22)$$

де:

tотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 7 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 472 \times 7 = 3307 \text{ грн.} \quad (3.23)$$

$$Кпо = 34865 + 3307 = 38172 \text{ грн.} \quad (3.24)$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \quad (3.25)$$

де:

Bk - число виконавців;

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{871}{1 \cdot 176} = 4{,}9 \text{ міс.} \quad (3.26)$$

Визначено трудомісткість розробленої інформаційної системи (871 люд-год), проведений підрахунок вартості роботи по створенню програми (38172 грн.) та розраховано час на його створення (4,9 міс).

ВИСНОВКИ

Метою кваліфікаційної роботи бакалавра є створення інформаційної системи по наданню послуг СТО для підвищення продуктивності працівників підприємства і для скорочення витрат часу на підбір потрібного товару і оформлення чеків.

Інформаційна система призначена для оптимізації і автоматизації роботи станції технічного обслуговування автомобілів, перегляду інформації про постачальників, продажі, а також для перегляду і друку інформації про продані товари та може бути застосованою в будь-який СТО з подібним родом діяльності.

Розроблена система дозволяє вирішити такі задачі:

– автоматизація процесу роботи СТО;
– дозволяє зв'язати мережу СТО даної фірми в один сервер;
– редагування списку товару;
– контроль кількості товару на складі;
– розрахунок вартості покупки;
– друк чеків і каталогу товарів.

Для розробки проекту було використано мову програмування C#, а середовищем розробки обрана Microsoft Visual Studio 19. Програмний додаток було виконано за допомогою системи керування базами даних (БД) SQL Server 2012.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (871 люд-год), проведений підрахунок вартості роботи по створенню програми (38172 грн.) та розраховано час на його створення (4,9 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

2. Білл Вагнер. «Найбільш ефективне використання C#», «Вільямс», 2016 рік, 256 с.

3. Грег Харвей. «Microsoft Excel для чайників», «Діалектика», 2013рік, 368 с.

4. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2019.

5. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2021.

6. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

7. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

8. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

9. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп'ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

10. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

11. Офіційний сайт середовища розробки Visual Studio Code URL: https://code.visualstudio.com/docs дата звернення: 15.01.2021.

12. Сайт про предметну область СТО. URL: https://works.doklad.ru/view/u8157hTV55k.html дата звернення: 15.01.2021.

13. Ден Томашевский. «Microsoft Windows 8. Керівництво користувача», «Вільямс», 2013 рік, 352 с.

14. Онлайн довідник Free On-Line Dictionary of Computing, 2006 рік, 140 с.

15. Пол Хенгсен «Підручник з Umbrello UML Modeller», «Umbrello UML Modeller», 2013 рік, 450 с.

16. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. - СПб.: КОРОНА принт, 2008. - 416с. Офіційний сайт середовища розробки Visual Studio Code URL: https://code.visualstudio.com/docs дата звернення: 5.05.2021.

17. Edx: Ускорьте свое будущее. Учитесь в любое время и в любом месте. - URL: https://www.edx.org/. дата звернення: 1.03.2021.

18. Educational Era: СТУДІЯ ОНЛАЙН-ОСВІТИ. - URL: https://www.ed-era.com/. дата звернення: 10.03.2021.

19. IDE MS Visual Studio URL: https://studfile.net/preview/5994722/page:7/ дата звернення: 5.05.2021.

ДОДАТОК А

КОД ПРОГРАМИ

Katalog.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace DProject
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;
username=root;password=root;database=esbd");
      public void openConnection()
      {
         if (connection.State == System.Data.ConnectionState.Closed)
            connection.Open();
      }
      public void closeConnection()
      {
         if (connection.State == System.Data.ConnectionState.Open)
            connection.Close();
```

```
    }
    public MySqlConnection getConnection()
    {
       return connection;
    }
    MySqlConnection connection2 = new MySqlConnection("server=localhost;port=3306;
username=root;password=root;database=lesbd");
    public void openConnection2()
    {
       if (connection2.State == System.Data.ConnectionState.Closed)
          connection2.Open();
    }
    public void closeConnection2()
    {
       if (connection2.State == System.Data.ConnectionState.Open)
          connection2.Close();
    }
    public MySqlConnection getConnection2()
    {
       return connection2;
    }
    public void button1_Click(object sender, EventArgs e)
    {
       string VSLogn = textBox1.Text;
       string VSPass = textBox2.Text;
       string check = null;
       Form2 f2 = new Form2();
       Form1 db = new Form1();
       DataTable table = new DataTable();
       MySqlDataAdapter adapter = new MySqlDataAdapter();
       MySqlCommand command1 = new MySqlCommand("SELECT * FROM workers WHERE
uname = @VSL AND pass = @VSP", db.getConnection());
       command1.Parameters.Add("@VSL",MySqlDbType.VarChar).Value = VSLogn;
       command1.Parameters.Add("@VSP", MySqlDbType.VarChar).Value = VSPass;
       adapter.SelectCommand = command1;
```

```
adapter.Fill(table);
db.openConnection();
MySqlDataReader read1 = command1.ExecuteReader();
while (read1.Read())
{
   check = read1[5].ToString();
}
db.closeConnection();
if ((table.Rows.Count > 0) && (check == "3"))
{
       MessageBoxButtons.OK,
       MessageBoxIcon.Information,
       MessageBoxDefaultButton.Button1,
       MessageBoxOptions.DefaultDesktopOnly);
   this.Hide();
   db.openConnection();
   MySqlDataReader read2 = command1.ExecuteReader();
   while (read2.Read())
   {
      f2.rd1 = read2[3].ToString();
      f2.rd2 = read2[4].ToString();
      f2.ShowDialog();
   }
   db.closeConnection();
}
if ((table.Rows.Count > 0) && (check == "2"))
{
   Form3 f3 = new Form3();
       MessageBoxButtons.OK,
       MessageBoxIcon.Information,
       MessageBoxDefaultButton.Button1,
       MessageBoxOptions.DefaultDesktopOnly);
   this.Hide();
   db.openConnection();
   MySqlDataReader read2 = command1.ExecuteReader();
```

```
            while (read2.Read())
            {
                f3.rd1 = read2[3].ToString();
                f3.rd2 = read2[4].ToString();
                f3.ShowDialog();
            }
            db.closeConnection();
        }
        else
        {
            DataTable table2 = new DataTable();
            MySqlDataAdapter adapter2 = new MySqlDataAdapter();
            MySqlCommand command3 = new MySqlCommand("SELECT * FROM students WHERE uname = @VSL AND pass = @VSP", db.getConnection());
            command3.Parameters.Add("@VSL", MySqlDbType.VarChar).Value = VSLogn;
            command3.Parameters.Add("@VSP", MySqlDbType.VarChar).Value = VSPass;
            adapter2.SelectCommand = command3;
            adapter2.Fill(table2);
            if ((table2.Rows.Count > 0))
            {
                MessageBoxButtons.OK,
                MessageBoxIcon.Information,
                MessageBoxDefaultButton.Button1,
                MessageBoxOptions.DefaultDesktopOnly);
                this.Hide();
                Form4 Form4 = new Form4();
                Form4.Show();
                /*db.openConnection();
                MySqlDataReader read2 = command1.ExecuteReader();
                while (read2.Read())
                {
                    f2.rd1 = read2[3].ToString();
                    f2.rd2 = read2[4].ToString();
                    f2.ShowDialog();
                }
```

```
            db.closeConnection();*/
        }
        if ((table.Rows.Count == 0) && (table2.Rows.Count == 0))
        {
            MessageBox.Show("Вхід не виконано", " ",
                MessageBoxButtons.OK,
                MessageBoxIcon.Warning,
                MessageBoxDefaultButton.Button1,
                MessageBoxOptions.DefaultDesktopOnly);
        }
      }
    }
    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
      Application.Exit();
    }
    private void Form1_Load(object sender, EventArgs e)
    {
    }
  }
}
```

BD.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace DProject
{
   public partial class Form2 : Form
   {
      public string rd1
      {
         get { return label1.Text; }
         set { label1.Text = value; }
      }
      public string rd2
      {
         get { return label2.Text; }
         set { label2.Text = value; }
      }
      public Form2()
      {
         InitializeComponent();
         Form1 db = new Form1();
         DataTable tablead = new DataTable();
         MySqlDataAdapter adapter = new MySqlDataAdapter();
         MySqlCommand comad = new MySqlCommand("SELECT * FROM workers WHERE idposition = 2", db.getConnection());
         adapter.SelectCommand = comad;
         adapter.Fill(tablead);
         db.openConnection();
         MySqlDataReader read2 = comad.ExecuteReader();
         comboBox1.Items.Clear();
         while (read2.Read())
         {
            comboBox1.Items.Add(read2[3].ToString());
         }
         read2.Close();
         db.closeConnection();
      }
      MySqlConnection connection = new
```

```
MySqlConnection("server=localhost;port=3306;username=root;password=root;database=esbd");
    public void openConnection()
    {
       if (connection.State == System.Data.ConnectionState.Closed)
          connection.Open();
    }
    public void closeConnection()
    {
       if (connection.State == System.Data.ConnectionState.Open)
          connection.Close();
    }
    public MySqlConnection getConnection()
    {
       return connection;
    }
    private void Form2_Load(object sender, EventArgs e)
    {
    }
    private void Form2_FormClosing(object sender, FormClosingEventArgs e)
    {
       Application.Exit();
    }
    Form1 db = new Form1();
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    private void sndloaddata()
    {
       tabControl1.SelectedTab = tabControl1.TabPages["tabPage1"];
       label5.Text = "";
       comboBox1.Text = "";
       textBox5.Text = "";
    }
    private void button1_Click(object sender, EventArgs e)
    {
       sndloaddata();
```

```
    }
    private void reploaddata()
    {
        tabControl1.SelectedTab = tabControl1.TabPages["tabPage2"];
        MySqlCommand command5 = new MySqlCommand("SELECT * FROM reports",
db.getConnection());
        adapter.SelectCommand = command5;
        adapter.Fill(table);
        db.openConnection();
        MySqlDataReader read5 = command5.ExecuteReader();
        dataGridView5.Rows.Clear();
        List<string[]> data = new List<string[]>();
        while (read5.Read())
        {
            data.Add(new string[4]);
            data[data.Count - 1][0] = read5[4].ToString();
            data[data.Count - 1][1] = read5[1].ToString();
            data[data.Count - 1][2] = read5[2].ToString();
            if (read5[3].ToString() == "") data[data.Count - 1][3] = "Прикріпленого файлу немає";
            else data[data.Count - 1][3] = "Є прикріплений файл";
        }
        read5.Close();
        db.closeConnection();
        foreach (string[] s in data)
            dataGridView5.Rows.Add(s);
    }
    private void button2_Click(object sender, EventArgs e)
    {
        reploaddata();
    }
   private void dataGridView5_CellDoubleClick_1(object sender, DataGridViewCellEventArgs e)
    {
        DataTable reptable = new DataTable();
        MySqlDataAdapter repadapter = new MySqlDataAdapter();
        db.openConnection();
```

```
        if (dataGridView5.CurrentRow != null)
        {
           string no0 = dataGridView5[0, dataGridView5.CurrentRow.Index].Value.ToString();
           string no1 = dataGridView5[1, dataGridView5.CurrentRow.Index].Value.ToString();
           string no2 = dataGridView5[2, dataGridView5.CurrentRow.Index].Value.ToString();
           string no3 = dataGridView5[3, dataGridView5.CurrentRow.Index].Value.ToString();
           var no0dt = DateTime.Parse(no0);
           MySqlCommand command6 = new MySqlCommand("SELECT adder FROM reports
WHERE sender = @no1 AND mesge = @no2 AND date = @no0", db.getConnection());
           command6.Parameters.Add("@no1", MySqlDbType.VarChar).Value = no1;
           command6.Parameters.Add("@no2", MySqlDbType.VarChar).Value = no2;
           command6.Parameters.Add("@no0", MySqlDbType.DateTime).Value = no0dt;
           db.openConnection();
           string adder = "";
           adder = command6.ExecuteScalar().ToString();
           db.closeConnection();
           string chekadd = "";
           chekadd = "" + "C:/Users/nikbe/Desktop" + "/" + Path.GetFileName(adder);
           if (File.Exists(chekadd) == true)
           {
               MessageBoxButtons.OK,
               MessageBoxIcon.Warning,
               MessageBoxDefaultButton.Button1,
               MessageBoxOptions.DefaultDesktopOnly);
                 return;
           }
           if (adder != "")
           {
              File.Copy(adder, "C:/Users/nikbe/Desktop" + "/" + Path.GetFileName(adder));
               MessageBoxButtons.OK,
               MessageBoxIcon.Information,
               MessageBoxDefaultButton.Button1,
               MessageBoxOptions.DefaultDesktopOnly);
           }
           else
```

```
            {
                MessageBoxButtons.OK,
                MessageBoxIcon.Information,
                MessageBoxDefaultButton.Button1,
                MessageBoxOptions.DefaultDesktopOnly);
            }
        }
        db.closeConnection();
    }
    private void sreloaddata()
    {
        tabControl1.SelectedTab = tabControl1.TabPages["tabPage3"];
        MySqlCommand command1 = new MySqlCommand("SELECT * FROM ",
db.getConnection());
        adapter.SelectCommand = command1;
        adapter.Fill(table);
        db.openConnection();
        MySqlDataReader read1 = command1.ExecuteReader();
        dataGridView1.Rows.Clear();
        List<string[]> data = new List<string[]>();
                while (read1.Read())
        {
            data.Add(new string[6]);
            data[data.Count - 1][0] = read1[3].ToString();
            data[data.Count - 1][1] = read1[4].ToString();
            data[data.Count - 1][2] = read1[6].ToString();
            data[data.Count - 1][3] = read1[7].ToString();
            data[data.Count - 1][4] = read1[8].ToString();
            data[data.Count - 1][5] = read1[9].ToString();
        }
                read1.Close();
        db.closeConnection();
        foreach (string[] s in data)
            dataGridView1.Rows.Add(s);
              }
```

```
private void button3_Click(object sender, EventArgs e)
{
  sreloaddata();
}
private void button7_Click(object sender, EventArgs e)
{
   sadd sadd = new sadd();
   sadd.Show();
}
private void button9_Click(object sender, EventArgs e)
{
      Form1 db = new Form1();
      db.openConnection();
      if (dataGridView1.CurrentRow != null)
      {
         string no = dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString();
         dataGridView1.Rows.Remove(dataGridView1.CurrentRow);
         MySqlCommand com2 = new MySqlCommand("DELETE FROM students WHERE
fname = @fn", db.getConnection());
         com2.Parameters.Add("@fn", MySqlDbType.VarChar).Value = no;
         com2.ExecuteNonQuery();
      }
      db.closeConnection();
}
private void button10_Click(object sender, EventArgs e)
{
   sreloaddata();
}
private void button6_Click(object sender, EventArgs e)
{
   for (int i = 0; i < dataGridView1.RowCount; i++)
   {
      dataGridView1.Rows[i].Selected = false;
      for (int j = 0; j < dataGridView1.ColumnCount; j++)
         if (dataGridView1.Rows[i].Cells[j].Value != null)
```

```
                if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(textBox1.Text))
                {
                    dataGridView1.Rows[i].Selected = true;
                    break;
                }
            }
            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                if (dataGridView1.Rows[i].Cells[0].Value != null &&
dataGridView1.Rows[i].Cells[0].Value.ToString().Contains(textBox1.Text))
                {
                    dataGridView1.FirstDisplayedScrollingRowIndex = i;
                    dataGridView1.Update();
                }
            }
        }
        private void greloaddata()
        {
            tabControl1.SelectedTab = tabControl1.TabPages["tabPage4"];
            MySqlCommand command2 = new MySqlCommand("SELECT * FROM ",
db.getConnection());
            adapter.SelectCommand = command2;
            adapter.Fill(table);
            db.openConnection();
            MySqlDataReader read2 = command2.ExecuteReader();
            dataGridView2.Rows.Clear();
            List<string[]> data = new List<string[]>();
            while (read2.Read())
            {
                data.Add(new string[3]);
                data[data.Count - 1][0] = read2[1].ToString();
                data[data.Count - 1][1] = read2[2].ToString();
                data[data.Count - 1][2] = read2[3].ToString();
            }
            read2.Close();
```

```
        db.closeConnection();
        foreach (string[] s in data)
            dataGridView2.Rows.Add(s);
    }
    private void button5_Click(object sender, EventArgs e)
    {
        greloaddata();
    }
    private void button8_Click(object sender, EventArgs e)
    {
        greloaddata();
    }
    private void button13_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < dataGridView2.RowCount; i++)
        {
            dataGridView2.Rows[i].Selected = false;
            for (int j = 0; j < dataGridView2.ColumnCount; j++)
                if (dataGridView2.Rows[i].Cells[j].Value != null)
                    if (dataGridView2.Rows[i].Cells[j].Value.ToString().Contains(textBox2.Text))
                    {
                        dataGridView2.Rows[i].Selected = true;
                        break;
                    }
        }
        for (int i = 0; i < dataGridView2.RowCount; i++)
        {
            if (dataGridView2.Rows[i].Cells[0].Value != null &&
dataGridView2.Rows[i].Cells[0].Value.ToString().Contains(textBox2.Text))
            {
                dataGridView2.FirstDisplayedScrollingRowIndex = i;
                dataGridView2.Update();
            }
        }
    }
```

```
        private void button12_Click(object sender, EventArgs e)
        {
            gadd gadd = new gadd();
            gadd.Show();
        }
        private void button14_Click(object sender, EventArgs e)
        {
            Form1 db = new Form1();
            db.openConnection();
            if (dataGridView2.CurrentRow != null)
            {
                string no = dataGridView2[0, dataGridView2.CurrentRow.Index].Value.ToString();
                dataGridView2.Rows.Remove(dataGridView2.CurrentRow);
                MySqlCommand com2 = new MySqlCommand("DELETE FROM groups WHERE group_fname = @fn", db.getConnection());
                com2.Parameters.Add("@fn", MySqlDbType.VarChar).Value = no;
                com2.ExecuteNonQuery();
            }
            db.closeConnection();
        }
        private void wreloaddata()
        {
            tabControl1.SelectedTab = tabControl1.TabPages["tabPage5"];
            MySqlCommand command3 = new MySqlCommand("SELECT * FROM workers", db.getConnection());
            adapter.SelectCommand = command3;
            adapter.Fill(table);
            db.openConnection();
            MySqlDataReader read3 = command3.ExecuteReader();
            dataGridView3.Rows.Clear();
            List<string[]> data = new List<string[]>();
            while (read3.Read())
            {
                data.Add(new string[5]);
                data[data.Count - 1][0] = read3[3].ToString();
```

```
            data[data.Count - 1][1] = read3[4].ToString();
            data[data.Count - 1][2] = read3[6].ToString();
            data[data.Count - 1][3] = read3[7].ToString();
            data[data.Count - 1][4] = read3[8].ToString();
        }
        read3.Close();
        db.closeConnection();
        foreach (string[] s in data)
            dataGridView3.Rows.Add(s);
    }
    private void button4_Click(object sender, EventArgs e)
    {
        wreloaddata();
    }
    private void button11_Click(object sender, EventArgs e)
    {
        wreloaddata();
    }
    private void button17_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < dataGridView3.RowCount; i++)
        {
            dataGridView3.Rows[i].Selected = false;
            for (int j = 0; j < dataGridView3.ColumnCount; j++)
                if (dataGridView3.Rows[i].Cells[j].Value != null)
                    if (dataGridView3.Rows[i].Cells[j].Value.ToString().Contains(textBox3.Text))
                    {
                        dataGridView3.Rows[i].Selected = true;
                        break;
                    }
        }
        for (int i = 0; i < dataGridView3.RowCount; i++)
        {
            if (dataGridView3.Rows[i].Cells[0].Value != null &&
dataGridView3.Rows[i].Cells[0].Value.ToString().Contains(textBox3.Text))
```

```
            {
                dataGridView3.FirstDisplayedScrollingRowIndex = i;
                dataGridView3.Update();
            }
        }
    }
    private void button15_Click(object sender, EventArgs e)
    {
        Form1 db = new Form1();
        db.openConnection();
        if (dataGridView3.CurrentRow != null)
        {
            string no = dataGridView3[0, dataGridView3.CurrentRow.Index].Value.ToString();
            dataGridView3.Rows.Remove(dataGridView3.CurrentRow);
            MySqlCommand com2 = new MySqlCommand("DELETE FROM workers WHERE
fname = @fn", db.getConnection());
            com2.Parameters.Add("@fn", MySqlDbType.VarChar).Value = no;
            com2.ExecuteNonQuery();
        }
        db.closeConnection();
    }
    private void button16_Click(object sender, EventArgs e)
    {
        wadd wadd = new wadd();
        wadd.Show();
    }
    private void lreloaddata()
    {
        tabControl1.SelectedTab = tabControl1.TabPages["tabPage6"];
        MySqlCommand command4 = new MySqlCommand("SELECT * FROM lessons",
db.getConnection());
        adapter.SelectCommand = command4;
        adapter.Fill(table);
        db.openConnection();
        MySqlDataReader read4 = command4.ExecuteReader();
```

```
    dataGridView4.Rows.Clear();
    List<string[]> data = new List<string[]>();
    while (read4.Read())
    {
        data.Add(new string[3]);
        data[data.Count - 1][0] = read4[1].ToString();
        data[data.Count - 1][1] = read4[2].ToString();
        data[data.Count - 1][2] = read4[3].ToString();
    }
    read4.Close();
    db.closeConnection();
    foreach (string[] s in data)
        dataGridView4.Rows.Add(s);
}
private void button18_Click(object sender, EventArgs e)
{
    lreloaddata();
}
private void button19_Click(object sender, EventArgs e)
{
    lreloaddata();
}
private void dataGridView4_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
{
    linfo linfo = new linfo();
    Form1 db = new Form1();
    db.openConnection();
    if (dataGridView4.CurrentRow != null)
    {
        string no1 = dataGridView4[1, dataGridView4.CurrentRow.Index].Value.ToString();
        string no2 = dataGridView4[2, dataGridView4.CurrentRow.Index].Value.ToString();
        string no3 = dataGridView4[0, dataGridView4.CurrentRow.Index].Value.ToString();
        linfo.rd1 = no1.ToString();
        linfo.rd2 = no2.ToString();
        MySqlCommand command4 = new MySqlCommand("SELECT valuation FROM lessons
```

```
WHERE title = @no3 AND teacher = @no1 AND `group`= @no2", db.getConnection());
            command4.Parameters.Add("@no1", MySqlDbType.VarChar).Value = no1;
            command4.Parameters.Add("@no2", MySqlDbType.VarChar).Value = no2;
            command4.Parameters.Add("@no3", MySqlDbType.VarChar).Value = no3;
            db.openConnection();
            string valuation = command4.ExecuteScalar().ToString();
            linfo.rd3 = valuation.ToString();
            db.closeConnection();
            linfo.ShowDialog();
        }
        db.closeConnection();
    }
    private void button22_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < dataGridView4.RowCount; i++)
        {
            dataGridView4.Rows[i].Selected = false;
            for (int j = 0; j < dataGridView4.ColumnCount; j++)
                if (dataGridView4.Rows[i].Cells[j].Value != null)
                    if (dataGridView4.Rows[i].Cells[j].Value.ToString().Contains(textBox4.Text))
                    {
                        dataGridView4.Rows[i].Selected = true;
                        break;
                    }
        }
        for (int i = 0; i < dataGridView4.RowCount; i++)
        {
            if (dataGridView4.Rows[i].Cells[0].Value != null &&
dataGridView4.Rows[i].Cells[0].Value.ToString().Contains(textBox4.Text))
            {
                dataGridView4.FirstDisplayedScrollingRowIndex = i;
                dataGridView4.Update();
            }
        }
    }
```

```
    private void button20_Click(object sender, EventArgs e)
    {
       Form1 db = new Form1();
       db.openConnection();
       if (dataGridView4.CurrentRow != null)
       {
          string no1 = dataGridView4[0, dataGridView4.CurrentRow.Index].Value.ToString();
          string no2 = dataGridView4[1, dataGridView4.CurrentRow.Index].Value.ToString();
          string no3 = dataGridView4[2, dataGridView4.CurrentRow.Index].Value.ToString();
          dataGridView4.Rows.Remove(dataGridView4.CurrentRow);
          MySqlCommand com2 = new MySqlCommand("DELETE FROM lessons WHERE title
= @no1 AND teacher = @no2", db.getConnection());
          com2.Parameters.Add("@no1", MySqlDbType.VarChar).Value = no1;
          com2.Parameters.Add("@no2", MySqlDbType.VarChar).Value = no2;
          com2.Parameters.Add("@no3", MySqlDbType.VarChar).Value = no3;
          com2.ExecuteNonQuery();
       }
       db.closeConnection();
    }
    private void button21_Click(object sender, EventArgs e)
    {
       ladd ladd = new ladd();
       ladd.Show();
    }
    private void panel1_DragEnter(object sender, DragEventArgs e)
    {
       if (e.Data.GetDataPresent(DataFormats.FileDrop))
       { e.Effect = DragDropEffects.Copy;
         }
    }
    private void panel1_DragLeave(object sender, EventArgs e)
    {
    }
     private void panel1_DragDrop(object sender, DragEventArgs e)
    {
```

```
        string[] add = (string[])e.Data.GetData(DataFormats.FileDrop);
        label5.Text = add[0];
    }
    private void button23_Click(object sender, EventArgs e)
    {
        Form1 db = new Form1();
        if ((comboBox1.Text == "") || (textBox5.Text == ""))
                {
                    MessageBoxButtons.OK,
                        MessageBoxIcon.Warning,
                        MessageBoxDefaultButton.Button1,
                        MessageBoxOptions.DefaultDesktopOnly);
                   return;
                }
            string clink = ""+("C:/Users/nikbe/source/repos/DProject/Files" + "/" +
Path.GetFileName(label5.Text));
            MySqlCommand com1 = new MySqlCommand("INSERT INTO `posts` (`sender`,
`recpnt`, `mesege`, `adder`) " +
                                    "VALUES (@se,@re,@ms,@ad)", db.getConnection());
        com1.Parameters.Add("@se", MySqlDbType.VarChar).Value = label1.Text;
        com1.Parameters.Add("@re", MySqlDbType.VarChar).Value = comboBox1.Text;
        com1.Parameters.Add("@ms", MySqlDbType.VarChar).Value = textBox5.Text;
        com1.Parameters.Add("@ad", MySqlDbType.VarChar).Value = clink;
            db.openConnection();
        if (com1.ExecuteNonQuery() == 1)
        {
           File.Copy(label5.Text,      "C:/Users/nikbe/source/repos/DProject/Files"      +      "/"      +
Path.GetFileName(label5.Text));
            sndloaddata();
        }
        else
                MessageBoxButtons.OK,
                MessageBoxIcon.Warning,
                MessageBoxDefaultButton.Button1,
                MessageBoxOptions.DefaultDesktopOnly);
```

```
            db.closeConnection();
        }
        private void dataGridView2_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
        {
          /* ginfo ginfo = new ginfo();
            Form1 db = new Form1();
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            db.openConnection();
            if (dataGridView4.CurrentRow != null)
            {
                string no1 = dataGridView4[1, dataGridView4.CurrentRow.Index].Value.ToString();
                string no2 = dataGridView4[2, dataGridView4.CurrentRow.Index].Value.ToString();
                linfo.rd1 = no1.ToString();
                linfo.rd2 = no2.ToString();
                MySqlCommand command4 = new MySqlCommand("SELECT valuation FROM lessons
WHERE teacher = @no1", db.getConnection());
                command4.Parameters.Add("@no1", MySqlDbType.VarChar).Value = no1;
                db.openConnection();
               string valuation = command4.ExecuteScalar().ToString();
                linfo.rd3 = valuation.ToString();
                db.closeConnection();
                linfo.ShowDialog();
            }
            db.closeConnection();*/
        }
    }
}
Main_frm
namespace CTO
{    public partial class Main_frm : Form
    {       Request req = new Request();
        private DataTable _dataTable_Kassa_Money = null;
        private DataTable _dataTable_Client_Dogovor_Operations = null;
        private DataTable _dataTable_Client_Dogovor = null;
```

```
        private DataTable _dataTable_ _ _Podpus = null;
        private DataTable _dataTable_ _ _Podr = null;
        private DataTable _dataTable_ = null;
        public DataGridViewCellStyle boldStyle = new
System.Windows.Forms.DataGridViewCellStyle();
        public long ID_Client = 0;
        public long Card_Number_ID_Selected = 0;
       public double Balance_Card_Number_Selected = 0;
        public string Valuta_Card_Selected = string.Empty;
        public string username;
        public double Balance_Paypal = 0;
        public long Role_User = 0;
        public static Main_frm SelfRefMain_frm
        {  get;
           set;
       }
        public Main_frm()
        {
           InitializeComponent();
           SelfRefMain_frm = this;
           bindingSource_ _ _Podpus_Podr.DataSource = _dataTable__Podr;
           _dataTable_C = new DataTable();
           bindingSource_C.DataSource = _dataTable_Cassa;
           DGV_C.DataSource = bindingSource_Cassa;
           bindingSource_C.DataMember = _dataTable_Cassa.TableName;
           boldStyle.Font = new System.Drawing.Font("Calibri", 11.25F,
System.Drawing.FontStyle.Bold);
           boldStyle.BackColor = Color.FromArgb(220, 240, 210);
        }
        private void _Click(object sender, EventArgs e)
        {
           Registration regi_ = new Registration();
           regi_.ID_Client = this.ID_Client;
           regi_.ShowDialog();
        }
```

```
private void ToolStripMenuItem_Click(object sender, EventArgs e)
{
   Login login_ = new Login();
   login_.Show();
   this.Hide();
}

private void _Click(object sender, EventArgs e)
{
   Environment.Exit(0);
}
public void Fill_Combo_Client_Card(long ID_Cl)
{
   comboBox_Client_Schet.Items.Clear();
   Request req1 = new Request();
   req1.con.Open();
   SqlCommand exeSql = new SqlCommand(sql_command, req1.con);
   SqlDataReader dr = exeSql.ExecuteReader();
   while (dr.Read())
   {
      row_count++;
   }
   dr.Close();
   if (row_count > 0)
   {
      dr = exeSql.ExecuteReader(CommandBehavior.CloseConnection);
      while (dr.Read())
      {
         comboBox_Client_Schet.Items.Add(dr[0].ToString());
      }
      dr.Close();
      req1.con.Close();
   }
   dr.Close();
   req1.con.Close();
```

```
            if (comboBox_Client_Schet.Items.Count > 0)
            {
                comboBox_Client_Schet.SelectedIndex = 0;
                label_Balance.Visible = true;
                label_Valuta.Visible = true;
            }
            else
            {
                label_Balance.Visible = false;
                label_Valuta.Visible = false;
            }
        }
        public void Fill_DGV_Operatoins(string Card_Number, long ID_Cl)
        {
            _dataTable_.Clear();
            string sql_command = "SELECT BCU.ID, BCU.Balance, VAL.Code FROM
[dbo].[Bank_Card_User] BCU inner join VAL on BCU.ID_ = VAL.ID " +
                        int row_count = 0;
            Request req1 = new Request();
            req1.con.Open();
            SqlCommand exeSql = new SqlCommand(sql_command, req1.con);
            SqlDataReader dr = exeSql.ExecuteReader();
            while (dr.Read())
            {
                row_count++;
            }
            dr.Close();
            if (row_count > 0)
            {
                dr = exeSql.ExecuteReader(CommandBehavior.CloseConnection);
                while (dr.Read())
                {
                    if (dr[0] != DBNull.Value)
                    {
                        Card_Number_ID_Selected = Convert.ToInt64(dr[0].ToString());
```

```
            }
            if (dr[1] != DBNull.Value)
            {
                Balance_Card_Number_Selected = Convert.ToDouble(dr[1].ToString());
            }
            if (dr[2] != DBNull.Value)
            {
                Valuta_Card_Selected = dr[2].ToString();
            }
        }
        dr.Close();
        req1.con.Close();
    }
    dr.Close();
    req1.con.Close();
    string sql_command_Operations = "SELECT " +
        "[ID] as 'ID', " +
        "[Type_Payment] as, " + "[Summa] as "+ "[Information] as, " +as 'Дата " +
        "FROM [dbo].[ _Operations] " + "Where " +
    SqlDataAdapter dataAdapter = new SqlDataAdapter(sql_command_Operations, req.con);
    dataAdapter.Fill(_dataTable_ _Money);
    DGV_Cassa_Money.ClearSelection();
    DGV_Cassa_Money.Columns[0].Visible = false;
    label_Balance.Text = ": " + Math.Round(__Number_Selected, 2).ToString() + "";
    label_Valuta.Text = ": " + _ _Selected.ToString();
}
public void Fill_DGV_ _Dogovor(long ID_Cl)
{
    _dataTable_Client_.Clear();
    string sql_command_Dogovor = "SELECT " +
        "DOGOV.[ID] as 'Номер, " +
        "DOGOV.[Date] as 'Дата', " +
        "BN_CRD_US_SELL.[Number_Card] as, " +"BN_CRD_US_BUY.[Number_]  as, " +
        "DOGOV.[Valuta_SELL] as, " +  "DOGOV.[Valuta_BUY] as, " +
        "DOGOV.[Summa] as 'Сума, " +"DOGOV.[Status] as " +    "FROM [dbo].[Dogovor] " +
```

```
            "(DOGOV.ID_ =" + ID_Cl + ")";
        SqlDataAdapter dataAdapter = new SqlDataAdapter(sql_command_Dogovor, req.con);
        dataAdapter.Fill(_dataTable_Client_Dogovor);
        DGV_Dogovor.ClearSelection();
    }
    public void Fill_DGV_Client_ _Operations(long ID_Dogovor)
    {
        _dataTable_Client_Dogovor_Operations.Clear();
        string sql_command_Dogovor_Podr = "SELECT " +
            "BN_OP.[ID] as 'Номер договору', " +
            "Convert(datetime,BN_OP.[Date],101) as, " +   "BN_OP.[Type_Payment] as 'Тип
платежу', " +
            "BN_OP.[Summa] as, " +      "BN_OP.[, " +         "BN_OP. as, " +
            "(CL_BN.[FirstName] + ' ' + CL_BN.[LastName] + ' ' + CL_BN.[Patronymic]) as 'ФІО
клієнта' " +       "FROM [dbo].[ _] BN_OP " +           " join _ BN_CRD on BN_OP.ID_ _
=BN_CRD.ID " +                " on BN_OP.ID_Bank_Card_User=BN_CRD_US.ID " +                "
        SqlDataAdapter dataAdapter = new SqlDataAdapter(sql_command_Dogovor_Podr,
req.con);
        dataAdapter.Fill(_dataTable_Client_Dogovor_Operations);
        DGV_Client_Dogovor_Operations.ClearSelection();
    }
    public void Fill_DGV_Client_Dogovor_Podpus()
    {
        _dataTable_ _Dogovor_Podpus.Clear();
        string sql_command_Dogovor_Podpus = "SELECT " +
            "(CL_BN.[FirstName] + ' ' + CL_BN.[LastName] + ' ' + CL_BN.[Patronymic]) as 'ПІБ', "
+             "DOGOV.[ID] as 'Номер, " +
            "DOGOV.[Date] as 'Дата', " +
        SqlDataAdapter dataAdapter = new SqlDataAdapter(sql_command_Cassa, req.con);
        dataAdapter.Fill(_dataTable_Cassa);
        DGV_Cassa.ClearSelection();
    }
    private void Main_frm_Load(object sender, EventArgs e)
    {
        string FIO = string.Empty;
```

```
string sql_command = "SELECT ([FirstName] + ' ' + [LastName] + ' ' + [Patronymic]) as 'FIO',
US.ID_Role FROM [dbo].[Client_Bank] CLB inner join Users US on CLB.ID=US.ID_Client
where US.ID_Client=" + ID_Client + "";
        int row_count = 0;
        req.con.Open();
        SqlCommand exeSql = new SqlCommand(sql_command, req.con);
        SqlDataReader dr = exeSql.ExecuteReader();
        while (dr.Read())
        {
           row_count++;}
        dr.Close();
        if (row_count > 0)
        {
           dr = exeSql.ExecuteReader(CommandBehavior.CloseConnection);
           while (dr.Read())
           {
              FIO = dr[0].ToString();
              Role_User = Convert.ToInt64(dr[1].ToString());
           }
           dr.Close();
           req.con.Close();
        }
        dr.Close();
        req.con.Close();
        switch (Role_User.ToString())
        {
          case "1":
              break;
           case "2":
              TAB1.TabPages.Remove(TAB1_Client);
              TAB1.TabPages.Remove(TAB1_Client_Orders);
              break;
           case "3":
              TAB1.TabPages.Remove(TAB1_Orders);
              TAB1.TabPages.Remove(TAB1_Cassa);
```

```
            break;
        }
        ToolTip tolltip_Add_ _ = new ToolTip();
          Privat_data[] parsed =
    JsonConvert.DeserializeObject<Privat_data[]>(
          new WebClient { Encoding = System.Text.Encoding.UTF8 }.
           for (int i = 0; i < Valutas_Code.Length; i++)
           {
              foreach (var item in parsed)
              {
                 if (item.ccy.ToString() == _Code[i].ToString())
                 {
                    string sale = item.sale.ToString();
                    sale = sale.Replace(",", ".");
                    string buy = item.buy.ToString();
                    buy = buy.Replace(",", ".");
                    string sql_command_Update_Course = "UPDATE [dbo].[] SET " +
                       "[Course_SELL] = " + sale + ", " +
                       "[Course_BUY] = " + buy + " " +
                       "WHERE (Code='" + Valutas_Code[i] + "')";
                    long rez_Update_Course = req.insert_del_update(sql_command_Update_Course);
                 }
              }
              foreach (var item in parsed)
              {
                 switch (item.ccy.ToString())
                 {
                    case "U":
                       _ _BUY = Convert.ToDouble(item.buy.ToString());
                       U_Course_SELL = Convert.ToDouble(item.sale.ToString());
                       break;

        catch (Exception ex)
        {
```

```
            MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            return;
        }
    }
    private void comboBox_Valuta_Course_SelectedIndexChanged(object sender, EventArgs e)
    {
        try
        {
            Privat_data[] parsed =
            JsonConvert.DeserializeObject<Privat_data[]>(
                new WebClient { Encoding = System.Text.Encoding.UTF8
            {
                if (item.ccy.ToString() == comboBox_Valuta_Course.SelectedItem.ToString())
                {
                    string sale = item.sale.ToString();
                    sale = sale.Replace(",", ".");
                    string buy = item.buy.ToString();
                   buy = buy.Replace(",", ".");
                    Label_Course_BUY.Text = ": " + buy + " грн";
                    Label_Course_SELL.Text = ": " + sale + " грн";
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            return;
        }
    }
    private void Main_frm_FormClosed(object sender, FormClosedEventArgs e)
    {
        Environment.Exit(0);
    }
```

```
        private void button_Add_New_Client_Card_Click(object sender, EventArgs e)
        {
            Add_Client_Card add_Client_Card_ = new Add_Client_Card();
            add_Client_Card_.ID_Client = this.ID_Client;
            add_Client_Card_.ShowDialog();
        }
        private void comboBox_Client_Schet_SelectedIndexChanged(object sender, EventArgs e)
        {
            Fill_DGV_Operatoins(comboBox_Client_Schet.SelectedItem.ToString(), ID_Client);
        }
        private void button_Clear_Filter_Click(object sender, EventArgs e)
        {
            bindingSource_ .Sort = string.Empty;
            bindingSource_ .Filter = string.Empty;
            DGV_ .ClearFilter();
            DGV_ .ClearSort();
            DGV__.ClearSelection();
        }
        private void DGV_ _ _FilterStringChanged(object sender, EventArgs e)
        {
            bindingSource_ _.Filter = DGV_ _.FilterString;
            DGV_Cassa_Money.ClearSelection();
        }
        private void DGV_ _ _SortStringChanged(object sender, EventArgs e)
        {
            bindingSource_ _.Sort = DGV_ _.SortString;
            DGV_ _.ClearSelection();
        }
        private void button_Add_Operation_Click(object sender, EventArgs e)
        {
            if (comboBox_Client_Schet.Items.Count > 0)
            {
;
                Edit_ _.ShowDialog();
            }
```

```
        else
        {
        }
    }
    private void TAB1_SelectedIndexChanged(object sender, EventArgs e)
    {
        switch (TAB1.SelectedTab.Name)
        {
            case "TAB1_Client":
                Fill_Combo_Client_Card(ID_Client);
                break;
            case "TAB1_Client_Orders":
                Fill_DGV_Client_ (ID_Client);
                break;
            case "TAB1_Orders":
                Fill_DGV_Client_ _Podpus();
                break;
            case "TAB1_ ":
                Fill_DGV_ ();
                break;
        }
    }
          string sql_command_select_balance_ _ _BUY = "SELECT TOP 1 " +  "BN_CRD.[ID],
" +  "BN_.[] " +   "FROM [dbo].[ _] BN_CRD "
            int row_count1 = 0;
            req.con.Open();
            SqlCommand exeSql1 = new
SqlCommand(sql_command_select_balance_BANK_CARD_BUY,   q.con);
            SqlDataReader dr1 = exeSql1.ExecuteReader();
            while (dr1.Read())
            {
               row_count1++;
            }
            dr1.Close();
            if (row_count1 > 0)
```

```
{
   dr1 = exeSql1.ExecuteReader(CommandBehavior.CloseConnection);
   while (dr1.Read())
   {
      ID_ _ _BUY = Convert.ToInt64(dr1[0].ToString());
      BALANCE_ _ = Convert.ToDouble(dr1[1].ToString());
   }
   dr1.Close();
   req.con.Close();
}
dr1.Close();
req.con.Close();
int row_count2 = 0;
req.con.Open();
SqlCommand exeSql2 = new
SqlDataReader dr2 = exeSql2.ExecuteReader();
while (dr2.Read())
{
   row_count2++;
}
dr2.Close();
if (row_count2 > 0)
{
   dr2 = exeSql2.ExecuteReader(CommandBehavior.CloseConnection);
   while (dr2.Read())
   {
      ID_ __SELL = Convert.ToInt64(dr2[0].ToString());
      _ _SELL = Convert.ToDouble(dr2[1].ToString());
   }
   dr2.Close();
   req.con.Close();
}
dr2.Close();
req.con.Close();
double _tmp_ _Sell = 0;
```

```
            double _tmp_ _BUY = 0;
            string _SELL = string.Empty;
            string _BUY = string.Empty;
            switch (Dogovor_Valuta_SELL)
            {
                Course_SELL = USD_Course_BUY.ToString().Replace(",", ".");
                break;
                Course_SELL = EUR_Course_BUY.ToString().Replace(",", ".");
                break;
              case "RUR":
                Balance_tmp_Obmen_Sell = Math.Round((_Summa * RUR_, 2);
                Course_SELL = RUR_Course_BUY.ToString().Replace(",", ".");
                break;
            }
                  string Date = DateTime.Now.ToString();
          string tmp_ _User_Update_SELL = Convert.ToString(__SELL -
ogovor_Summa).Replace(",", ".");
            string tmp_ _User_Update_BUY = Convert.ToString(__BUY + alance_tmp_
_BUY).Replace(",", ".");
            string tmp_ _ _Update_BUY = Convert.ToString(__BUY + Replace(",", ".");
            string tmp_balance_ _Update_SELL = Convert.ToString(_ _SELL - _tmp_
_BUY).Replace(",", ".");
    private void button_Print_Click(object sender, EventArgs e)
    {
      DataGridView DGV1 = new DataGridView();
      CopyDataGridView(DGV_ _Podpus, DGV1);
      DataTable dt_Print = new DataTable();
      DataColumn[] dcs = new DataColumn[] { };
      foreach (DataGridViewColumn c in DGV1.Columns)
      {
        DataColumn dc = new DataColumn();
        dc.ColumnName = c.Name;
        dt_Print.Columns.Add(dc);

      }
```

```
            foreach (DataGridViewRow r in DGV1.Rows)
            {
                DataRow drow = dt_Print.NewRow();
                foreach (DataGridViewCell cell in r.Cells)
                {
                    drow[cell.OwningColumn.Name] = cell.Value;
                }
                dt_Print.Rows.Add(drow);
            }
            Obmen_.FormReport print_Report_ = new Obmen_Valut.FormReport();
            print_Report_.dsOrders = dt_Print;
            print_Report_.Check_Orders = true;
            print_Report_.ShowDialog();
        }
        private void _Click(object sender, EventArgs e)
        {
            Help a = new Help();
            this.Hide();
            a.Show();
        }
        }
    }
```

Client.cs

```
namespace Obmen_
{
    public partial class Add_Client_Card : Form
    {
        Request req = new Request();
        public long ID_Client = 0;
        public Add_ _ ()
        {
            InitializeComponent();
        }
        private void textBox_Summa_KeyPress(object sender, KeyPressEventArgs e)
        {
```

```
    if (!(Char.IsDigit(e.KeyChar)) && !(((e.KeyChar == '.') || (e.KeyChar == ',')) &&
  ((((TextBox)sender).Text.IndexOf(".") == -1) && (((TextBox)sender).Text.IndexOf(",") == -1))
&&
  (((TextBox)sender).Text.Length != 0)))
        {
            if (e.KeyChar != (char)Keys.Back)
            {
                e.Handled = true;
            }
        }
    }
    private void Add__Load(object sender, EventArgs e)
    {
        Fill_Combo_ ();
    }
    public void Fill_Combo_ ()
    {
        comboBox_.Items.Clear();
        string sql_command = "SELECT [Code] FROM [dbo].[]";
        int row_count = 0;
        Request req1 = new Request();
        req1.con.Open();
        SqlCommand exeSql = new SqlCommand(sql_command, req1.con);
        SqlDataReader dr = exeSql.ExecuteReader();
        while (dr.Read())
        {
            row_count++;
        }
        dr.Close();
        if (row_count > 0)
        {
            dr = exeSql.ExecuteReader(CommandBehavior.CloseConnection);
            while (dr.Read())
            {
                comboBox_.Items.Add(dr[0].ToString());
```

```
        }
        dr.Close();
        req1.con.Close();
    }
    dr.Close();
    req1.con.Close();

    if (comboBox_.Items.Count > 0)
    {
        comboBox_.SelectedIndex = 0;
    }
}
private void butto_Cancel_Click(object sender, EventArgs e)
{
    this.Close();
}
private void textBox_Summa_TextChanged(object sender, EventArgs e)
{
    if ((textBox_Summa.Text == ",") || (textBox_Summa.Text == "."))
    {
        textBox_Summa.Text = "0,";
        return;
    }
    if (textBox_Summa.Text.ToString() == "")
    {
        textBox_Summa.Text = "0";
        return;
    }
}
private void button_Save_Click(object sender, EventArgs e)
{
    if (Convert.ToDouble(textBox_Summa.Text.ToString()) <= 0)
    {
            return;
    }
```

```
        if (String.IsNullOrEmpty(maskedTextBox_Card_Number.Text.ToString()) ||
((maskedTextBox_Card_Number.MaskFull != true)))
        {
                return;
        }
        string _Number = maskedTextBox_Card_Number.Text.ToString();
        string _Crd = textBox_Summa.Text.ToString();
        _Crd = Balance_Crd.Replace(",", ".");
        string Combo_Selected_Valuta = comboBox_Valuta.SelectedItem.ToString();
        long ID_Valuta = 0;
       string sql_command_select_ID_Valuta = "SELECT ID FROM [dbo].[Valuta] where
(Code='" + Combo_Selected_Valuta + "')";
.        int row_count0 = 0;
        Request req1 = new Request();
        req1.con.Open();
        SqlCommand exeSql0 = new SqlCommand(sql_command_select_ID_Valuta, req1.con);
        SqlDataReader dr0 = exeSql0.ExecuteReader();
        while (dr0.Read())
        {
           row_count0++;
        }
        dr0.Close();
        if (row_count0 > 0)
        {
           dr0 = exeSql0.ExecuteReader(CommandBehavior.CloseConnection);
           while (dr0.Read())
           {
              ID_Val = Convert.ToInt64(dr0[0].ToString());
           }
           dr0.Close();
           req1.con.Close();
        }
        dr0.Close();
        req1.con.Close();
```

```
        string sql_command = "SELECT [ID] FROM [dbo].[ __User] where (Number_C='" +
C_Number + "')";
        int row_count = 0;
        req.con.Open();
        SqlCommand exeSql = new SqlCommand(sql_command, req.con);
        SqlDataReader dr = exeSql.ExecuteReader();
        while (dr.Read())
        {
           row_count++;
        }
        dr.Close()
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи