Міністерство освіти і науки україни
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
_____д. е. н., проф. П. М. Грицюк
« _____»_____ 20____ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «бакалавр»
за освітньо-професійною програмою «Інформаційні системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Проєктування та розробка інформаційної системи для автоматизації обліку діяльності АЗС»

Виконав: здобувач вищої освіти 4 курсу, групи ІСТ-41
Жила Вадим Сергійович
Керівник: к. т. н., доц. Барановський С. В.
Рецензент: ст. викладач Шевченко І.М.

Рівне – 2025

АНОТАЦІЯ

Дипломна кваліфікаційна бакалаврська робота «Інформаційна система обліку автозаправної станції» / Кваліфікаційна бакалаврська робота / м. Рівне: НУВГП, 2025. 142 ст. Українською мовою.

Бакалаврська робота складається зі вступу, 3 розділів, висновків, списку використаних джерел, додатку, 21 ілюстрації, 2 таблиць.

Метою бакалаврської дипломної роботи є створення автоматизованої інформаційної системи обліку діяльності заправної станції, яка забезпечує зручну взаємодію персоналу з різними аспектами обліку: клієнтів, транспорту, заправок, бонусних карток тощо.

Для досягнення поставленої мети були виділені й виконані наступні завдання:

- вибір інструментів та середовища розробки;
- проєктування користувацького інтерфейсу;
- реалізація функціоналу системи за допомогою відповідних програмних засобів.

Об’єкт дослідження – заправна станція.

Предмет дослідження – процеси інформатизації обліку роботи заправної станції.

У рамках дипломної роботи **розроблено інформаційну систему**, що вирішує поставлені задачі за допомогою інструментів СУБД. Створена **автоматизована система** забезпечує ефективне зберігання й цілісність даних, використовуючи форми, кожна з яких відповідає певній таблиці або сукупності таблиць. Система дозволяє зручно вводити, редагувати, шукати дані в базі, а також формувати вибірки за критеріями за допомогою SQL-запитів. Отримані результати можна експортувати у звіти для подальшого друку.

Створення такої автоматизованої інформаційної системи спрямоване на підвищення ефективності обліку, контролю та аналізу діяльності заправної станції.

ЗМІСТ

ВСТУП

Актуальність теми дослідження: У сучасних умовах розвитку економіки та технологій автоматизація бізнес-процесів стає невід'ємною частиною ефективного управління будь-яким підприємством. Особливо це стосується галузі автозаправних станцій (АЗС), які є важливим елементом транспортної інфраструктури та енергетичного сектору. Автозаправні станції щодня обслуговують велику кількість клієнтів, що потребує чіткого управління ресурсами, запасами палива, фінансовими операціями та персоналом. Розробка та впровадження інформаційної системи для автоматизації обліку діяльності АЗС є ключовим кроком до підвищення ефективності роботи станцій, покращення якості обслуговування клієнтів та забезпечення конкурентоспроможності на ринку [1].

Актуальність дослідження підкріплюється такими факторами:

1. **Зростання кількості АЗС та складність управління:** З кожним роком кількість автозаправних станцій зростає, що ускладнює управління ними без використання автоматизованих систем. Ручне ведення обліку стає неефективним і може призводити до помилок, які впливають на роботу станцій та їх прибутковість [2].
2. **Потреба у швидкому доступі до інформації:** Сучасні АЗС потребують оперативного доступу до даних про запаси палива, фінансові операції, постачальників, клієнтів та персонал. Це дозволяє швидко приймати управлінські рішення та підвищує конкурентоспроможність станцій [3].
3. **Оптимізація логістичних процесів:** Ефективна база даних допомагає оптимізувати логістику поставок палива, зменшуючи витрати та підвищуючи рентабельність бізнесу. Це особливо важливо для великих мереж АЗС, де невчасне оновлення даних може призвести до зупинки роботи станцій [4].

4. **Поліпшення якості обслуговування клієнтів:** Використання інформаційної системи дозволяє автоматизувати облік клієнтів, нарахування бонусів та знижок, що сприяє підвищенню лояльності клієнтів та покращенню якості обслуговування [5].
5. **Безпека даних:** У контексті зростаючої кіберзагрози важливо забезпечити захист даних, які стосуються фінансових операцій, особистих даних клієнтів та інформації про запаси палива. Сучасні бази даних надають засоби для забезпечення безпеки та захисту інформації [6].
6. **Відповідність нормативним вимогам:** Зміни в законодавстві та посилення вимог до звітності змушують АЗС модернізувати свої системи обліку. Використання сучасної бази даних допомагає відповідати новим нормативним вимогам та прискорює процеси звітності [7].

Таким чином, розробка інформаційної системи для автоматизації обліку діяльності автозаправних станцій є надзвичайно актуальним завданням, які відповідають вимогам сучасного бізнесу та сприяє підвищенню ефективності, безпеки та якості в галузі АЗС.

Ступінь розробленості теми: Сьогодні існує велика кількість програмних рішень для автоматизації роботи АЗС, проте більшість із них орієнтовані на великі мережі станцій і потребують значних фінансових вкладень [8]. У той же час, для малих та середніх АЗС часто відсутні доступні та ефективні рішення, які б враховували їх специфіку. Це створює потребу у розробці спеціалізованої інформаційної системи, яка б відповідала потребам таких станцій та була доступною для впровадження.

Метою дослідження є проектування та розробка інформаційної системи для автоматизації обліку діяльності автозаправних станцій, яка забезпечить ефективне управління ресурсами, покращить якість обслуговування клієнтів та забезпечить збереження та захист інформації. Система повинна автоматизувати основні процеси обліку, такі як управління запасами палива, облік

продажів, фінансові операції та управління персоналом.

Завдання дослідження:

1. Провести аналіз предметної області та визначити основні вимоги до інформаційної системи АЗС.
2. Розробити нормалізовану базу даних для АЗС з використанням сучасних методів проектування баз даних.
3. Створити інформаційну систему на основі розробленої бази даних, яка забезпечує автоматизацію основних процесів обліку діяльності АЗС.
4. Реалізувати функціональність, яка дозволяє автоматизувати процеси управління запасами палива, генерації звітів про продажі та управління клієнтськими даними.
5. Провести тестування інформаційної системи для перевірки її функціональності, надійності та відповідності вимогам.
6. Оцінити ефективність розробленої системи та її вплив на діяльність АЗС.
7. Підготувати документацію для інформаційної системи, включаючи опис структури бази даних, функціональних можливостей та інструкцій з використання.

Об'єкт дослідження: Процеси обліку та управління діяльністю автозаправних станцій, включаючи облік палива, продажів, фінансових операцій, управління персоналом та клієнтською базою.

Предмет дослідження: Інформаційна система для автоматизації обліку діяльності автозаправних станцій, яка включає базу даних, програмне забезпечення та інтерфейси для взаємодії з користувачами.

Методи дослідження: У роботі використовувалися такі методи дослідження:

- Аналіз літературних джерел та нормативних документів, що регулюють діяльність АЗС.
- Моделювання бізнес-процесів для виявлення ключової оптимізації.
- Проектування бази даних з використанням принципів нормалізації.
- Розробка програмного забезпечення з використанням сучасних інструментів та

технологій.

- Тестування системи для перевірки її функціональності та надійності.

Наукова новизна дослідження: Розроблена інформаційна система враховує специфіку роботи малих та середніх АЗС, що дозволяє їй бути більш доступною та ефективною для таких підприємств. Система включає нові підходи до автоматизації обліку палива, управління запасами та генерації звітів, що відрізняє її від існуючих аналогів.

Практична цінність дослідження: Розроблена інформаційна система може бути впроваджена на автозаправних станціях для покращення ефективності їх роботи. Вона забезпечує автоматизацію рутинних процесів, знижує ризик помилок та підвищує якість обслуговування клієнтів. Система також може бути використана як основа для подальшого розширення функціоналу та інтеграція з іншими системами управління.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 **Загальна характеристика предметної області**

Автозаправні станції (АЗС) відіграють ключову роль у транспортній інфраструктурі, забезпечуючи паливом широкий спектр транспортних засобів, а також надаючи супутні послуги та товари. У сучасних умовах функціонування АЗС включає в себе не лише реалізацію різних видів палива (бензин, дизельне паливо, газ), а й облік операцій, управління запасами, роботу з постачальниками, фінансовий контроль та кадрове адміністрування. З огляду на великий обсяг даних, що обробляється щоденно, використання автоматизованих систем обліку є необхідністю. Ефективне програмне забезпечення дозволяє значно підвищити продуктивність АЗС, зменшити витрати часу на рутинні операції та мінімізувати ризики людського фактора.

1.2 **Облік палива та супутніх товарів**

Ведення обліку пального та інших товарів є важливою складовою управління АЗС. Без автоматизації цього процесу виникають проблеми, пов’язані з неточностями підрахунків, нестачами та надлишками, що можуть призвести до фінансових втрат.

Впровадження програмного забезпечення дозволяє:

- Автоматизувати облік наявних запасів пального та товарів у режимі реального часу;
- Оптимізувати процеси постачання та поповнення складів;
- Формувати звіти щодо руху товарів, продажів та залишків;
- Прогнозувати потреби в паливі на основі статистичних даних.

1.3 **Управління клієнтською базою та облік продажів**

Для покращення якості обслуговування необхідно мати ефективну систему ведення клієнтської бази. Це дозволяє не лише здійснювати облік

продажів, а й реалізовувати програми лояльності, що підвищують рівень задоволеності клієнтів та сприяють їхньому поверненню.

Програмне забезпечення для АЗС повинно включати:

- Облік клієнтів, включаючи їхні контактні дані та історію покупок;
- Автоматизоване нарахування знижок та бонусних балів;
- Генерацію аналітичних звітів щодо продажів та активності клієнтів.

1.4 **Контроль постачальників та логістика поставок**

Автоматизація процесу постачання дозволяє АЗС ефективно керувати взаємодією з постачальниками пального та інших товарів. Програмна система повинна забезпечувати:

- Збереження інформації про постачальників, умови співпраці та контактні дані;
- Автоматичний контроль виконання договорів та постачань;
- Формування звітності щодо отриманих товарів та аналіз постачань.

1.5 **Автоматизація фінансового обліку та звітності**

Ефективна робота АЗС неможлива без системи фінансового контролю, яка дозволяє керівництву отримувати точні дані про прибутки та витрати підприємства.

Функціональність програмного забезпечення має включати:

- Облік фінансових надходжень та витрат;
- Автоматичне формування бухгалтерських звітів;
- Контроль грошових потоків та розрахунок прибутковості;
- Інтеграцію з банківськими системами та електронними платіжними сервісами.

1.6 **Управління персоналом**

АЗС зазвичай має змінний графік роботи персоналу, що потребує гнучкої системи управління кадрами. Програмне забезпечення повинно включати:

- Облік працівників та їхніх посадових обов’язків;
- Планування змін і графіків роботи;
- Контроль заробітної плати, премій та інших виплат;
- Формування звітності щодо продуктивності працівників.

Автоматизація облікових процесів на АЗС є важливим напрямком розвитку, який дозволяє підвищити ефективність роботи, мінімізувати ризики та забезпечити точний контроль усіх бізнес-процесів. Впровадження сучасного програмного забезпечення оптимізує роботу станції, покращить якость обслуговування клієнтів та підвищенню рівня прибутковості підприємства.

РОЗДІЛ 2. РОЗРОБКА ПРОЄКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 **Аналіз предметної області**

Автозаправна станція (АЗС) є об’єктом торгівлі, що спеціалізується на реалізації пального, а також супутніх товарів (мастила, автохімія, товари повсякденного вжитку). Діяльність АЗС супроводжується рядом адміністративних, технічних та облікових процесів, які вимагають точного й оперативного контролю.

Ключовими об’єктами предметної області є:

- **Клієнти** — особи, які здійснюють купівлю пального або товарів. Деякі з них мають бонусні картки.
- **Працівники** — персонал, який здійснює обслуговування клієнтів, оформлення продажу, ведення обліку.
- **Пальне** — основна категорія товару, що зберігається в резервуарах (цистернах) і реалізується за літражем.
- **Товари магазину** — додаткові одиниці товарів, які продаються поряд із пальним (кава, омивач, їжа тощо).
- **Бонусні картки** — система лояльності, що дає можливість нарахування бонусів, обліку вподобань.
- **Постачальники** — юридичні або фізичні особи, що здійснюють постачання пального до АЗС.

Функціонування АЗС вимагає злагодженої взаємодії між цими об’єктами. Раніше ці процеси здійснювались вручну або за допомогою несистематизованих інструментів (наприклад, Excel), що призводило до:

- дублювання даних;
- людських помилок під час продажу;
- труднощів з інвентаризацією пального;
- відсутності механізму для контролю дій персоналу.

Інформаційна система, розроблена у межах проєкту, покликана усунути ці проблеми, автоматизуючи основні процеси управління та обліку.

2.2 **Логічна модель даних**

Для відображення предметної області в електронному вигляді було побудовано логічну модель (рис. 2.1), яка описує взаємозв’язки між сутностями, необхідними для функціонування системи.

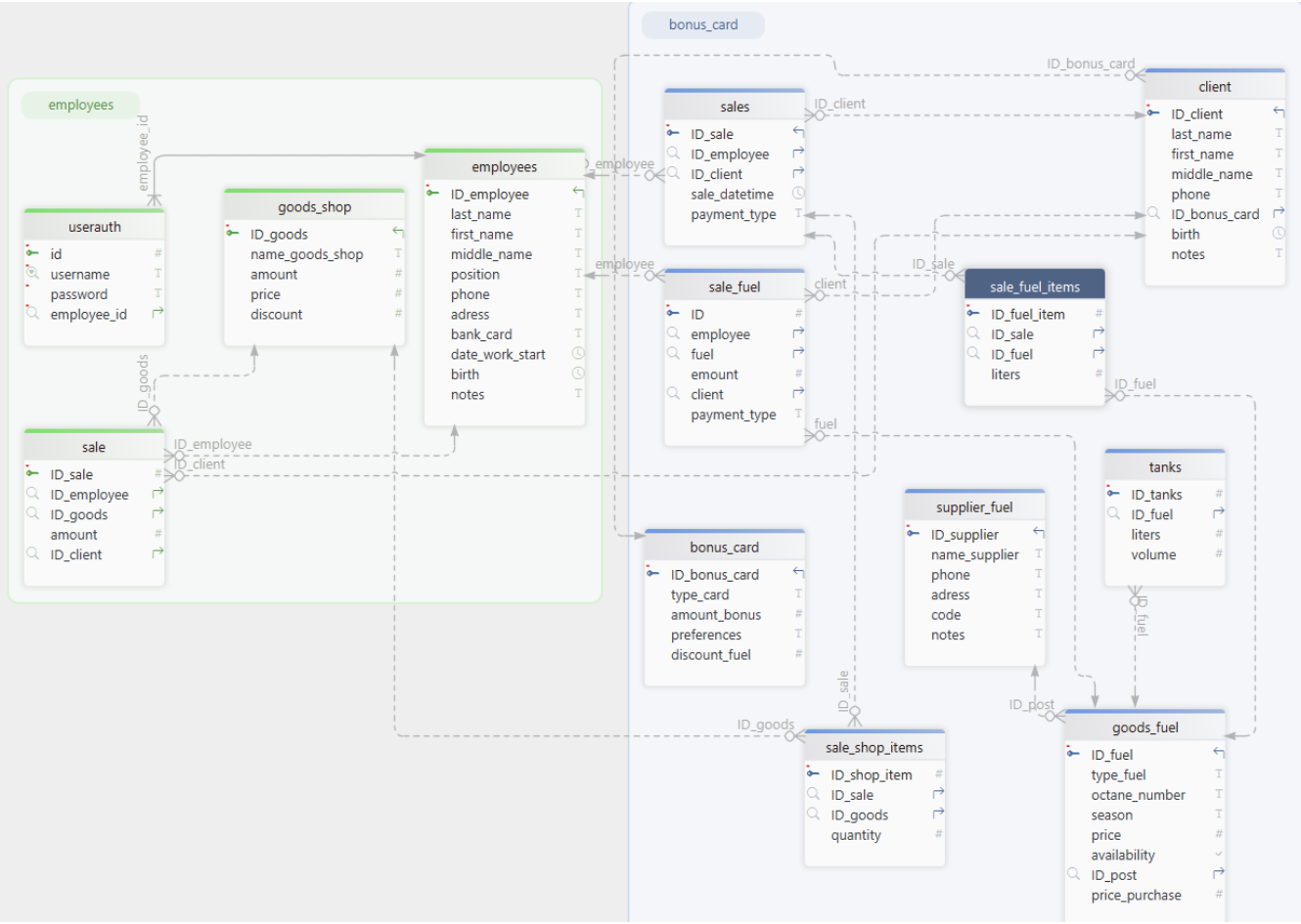


Рис. 2.1. Вигляд структури бази даних даного проекту

Основні сутності та їх атрибути:

- **Client** (Клієнт): ID, прізвище, ім’я, по батькові, телефон, дата народження, примітки, ID бонусної картки.

- **Bonus_card** (Бонусна картка): ID, тип, сума бонусів, вподобання клієнта, знижка.
- **Employees** (Працівники): ID, ПІБ, посада, телефон, адреса, банківська картка, дата початку роботи, дата народження, нотатки.
- **Userauth** (Авторизація): логін, пароль, ID працівника.
- **Goods_fuel** (Пальне): ID, тип пального, октанове число, сезон, ціна, наявність, постачальник.
- **Goods_shop** (Товари): ID, назва товару, кількість, ціна, знижка.
- **Sale_fuel** (Продаж пального): ID, працівник, клієнт, кількість, тип пального, тип оплати.
- **Sale** (Продаж товарів): ID, працівник, клієнт, товар, кількість, дата, тип оплати.
- **Sale_fuel_items** / **Sale_shop_items** — деталізація продажів.
- **Tanks** (Цистерни): ID, ID пального, поточний об’єм, максимальний об’єм.
- **Supplier_fuel** (Постачальник): ID, назва, телефон, адреса, код.

Між сутностями реалізовано численні зв’язки:

- клієнт — бонусна картка (1:1);
- клієнт — продаж (1:N);
- працівник — продаж (1:N);
- пальне — цистерна (1:N);
- продаж — деталізація (1:N).

Для створення зв’язків необхідно, щоб у головній таблиці були визначенні ключі. Встановлення первинного ключа для зв’язаної (підпорядкованої) таблиці не є обов’язково. Для підпорядкованої таблиці треба визначити поле вторинного ключа, тип даних і розмір якого повинні збігатись з полем первинного ключа головної таблиці. Імена полів первинного та вторинного ключів, між якими встановлюється зв’язок можуть не збігтися. Вторинні ключі відрізняються від первинних, для них допускається дублювання значення [9].

Ця модель (табл. 2.1) дозволяє ефективно представляти як довготривалі дані (довідники), так і транзакційні (продажі).

Таблиця 2.1. Структура таблиць БД проекту

Назва таблиці	**Поля**	**Типи даних / атрибути**
bonus_card	ID_bonus_card, type_card, amount_bonus, preferences, discount_fuel	INT, VARCHAR, DECIMAL(10,2), TEXT, DECIMAL(5,2)
client	ID_client, last_name, first_name, middle_name, phone, ID_bonus_card, birth, notes	INT, VARCHAR, VARCHAR, VARCHAR, VARCHAR, INT (FK), DATE, TEXT
employees	ID_employee, last_name, first_name, middle_name, position, phone, adress, bank_card, date_work_start, birth, notes	INT, VARCHAR, VARCHAR, VARCHAR, VARCHAR, VARCHAR, TEXT, VARCHAR, DATE, DATE, TEXT
goods_fuel	ID_fuel, type_fuel, octane_number, season, price, availability, ID_post, price_purchase	INT, VARCHAR, VARCHAR, VARCHAR, DECIMAL(10,2), BOOLEAN/INT, INT (FK), DECIMAL(10,2)
goods_shop	ID_goods, name_goods_shop, amount, price, discount	INT, VARCHAR, INT, DECIMAL(10,2), DECIMAL(5,2)
sale	ID_sale, ID_employee, ID_goods, amount, ID_client	INT, INT (FK), INT (FK), INT, INT (FK)
sale_fuel	ID, employee, fuel, amount, client, payment_type	INT, INT (FK), INT (FK), DECIMAL(10,2), INT (FK),

		VARCHAR
sale_fuel_items	ID_fuel_item, ID_sale, ID_fuel, liters	INT, INT (FK), INT (FK), DECIMAL(10,2)
sale_shop_items	ID_shop_item, ID_sale, ID_goods, quantity	INT, INT (FK), INT (FK), INT
sales	ID_sale, ID_employee, ID_client, sale_datetime, payment_type	INT, INT (FK), INT (FK), DATETIME, VARCHAR
supplier_fuel	ID_supplier, name_supplier, phone, adress, code, notes	INT, VARCHAR, VARCHAR, VARCHAR, VARCHAR, TEXT
tanks	ID_tanks, ID_fuel, liters, volume	INT, INT (FK), DECIMAL(10,2), DECIMAL(10,2)
userauth	id, username, password, employee_id	INT, VARCHAR, VARCHAR, INT (FK)

2.3 **База даних інформаційної системи**

Для збереження, обробки та доступу до даних у системі застосовано реляційну базу даних, побудовану на СУБД **MySQL та з залученням Navicat**. Її структура відображає логічну модель даних, що була розглянута вище, й оптимізована для реальної роботи системи.

Серед ключових характеристик БД:

- **Нормалізація до 3NF**, що усуває надлишковість даних;
- **Зовнішні ключі**, що забезпечують зв’язність таблиць;
- **Індекси**, які прискорюють пошук за основними полями (ID клієнтів, пального, продажів);
- **Автоінкрементовані первинні ключі** для усіх основних таблиць.

База даних підтримує наступні функції:

- збереження всієї історії продажів;

- оновлення залишків на складах та в резервуарах;
- реєстрація працівників і контроль їхніх дій;
- підтримка пошуку, сортування та фільтрації записів;
- гнучке керування клієнтськими бонусами та вподобаннями.

Фізична реалізація БД підтримує масштабування та можливість інтеграції з іншими модулями. Розширення функціоналу не потребує зміни існуючих таблиць, що забезпечує гнучкість системи в майбутньому.

За допомогою програмного засобу для формування і керування базами даних Navicat було розроблену базу даних яка задовільняє завдання поставлене перед додатком. Розглянемо найважливіші з них:

bonus_card — Бонусна картка клієнта

Таблиця зберігає інформацію про картки лояльності, які надають бонуси або знижки клієнтам.

```
CREATE TABLE bonus_card (
    ID_bonus_card INT PRIMARY KEY AUTO_INCREMENT,
    type_card VARCHAR(50),
    amount_bonus DECIMAL(10,2),
    preferences TEXT,
    discount_fuel DECIMAL(5,2)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_bonus_card	int	11		☑	☐	🔑 1
type_card	varchar	50		☐	☐	
amount_bonus	decimal	10	2	☐	☐	
preferences	text			☐	☐	
discount_fuel	decimal	5	2	☐	☐	

Рис. 2.2. Таблиця bonus_card

client — Інформація про клієнтів

Зберігає контактні дані клієнтів, зв’язок із бонусною карткою та додаткові відомості.

```
CREATE TABLE client (
    ID_client INT PRIMARY KEY AUTO_INCREMENT,
    last_name VARCHAR(50),
    first_name VARCHAR(50),
    middle_name VARCHAR(50),
    phone VARCHAR(20),
    ID_bonus_card INT,
    birth DATE,
    notes TEXT,
    FOREIGN KEY (ID_bonus_card) REFERENCES bonus_card(ID_bonus_card)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_client	int	11		☑	☐	🔑 1
last_name	varchar	50		☐	☐	
first_name	varchar	50		☐	☐	
middle_name	varchar	50		☐	☐	
phone	varchar	20		☐	☐	
ID_bonus_card	int	11		☐	☐	
birth	date			☐	☐	
notes	text			☐	☐	

Рис. 2.3. Таблиця Client

employees – працівники АЗС

Містить дані про працівників: ПІБ, посаду, реквізити, дату народження, стаж тощо.

```
CREATE TABLE employees (
    ID_employee INT PRIMARY KEY AUTO_INCREMENT,
    last_name VARCHAR(50),
    first_name VARCHAR(50),
```

```
    middle_name VARCHAR(50),
    position VARCHAR(50),
    phone VARCHAR(20),
    adress TEXT,
    bank_card VARCHAR(30),
    date_work_start DATE,
    birth DATE,
    notes TEXT
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_employee	int	11		☑	☐	🔑 1
last_name	varchar	50		☐	☐	
first_name	varchar	50		☐	☐	
middle_name	varchar	50		☐	☐	
position	varchar	50		☐	☐	
phone	varchar	20		☐	☐	
adress	varchar	100		☐	☐	
bank_card	varchar	30		☐	☐	
date_work_start	date			☐	☐	
birth	date			☐	☐	
notes	text			☐	☐	

Рис. 2.4. Таблиця employees

goods_fuel — Товари (пальне)

Таблиця містить відомості про види пального, його характеристики та постачальника.

```
CREATE TABLE goods_fuel (
    ID_fuel INT PRIMARY KEY AUTO_INCREMENT,
    type_fuel VARCHAR(50),
    octane_number VARCHAR(10),
    season VARCHAR(20),
    price DECIMAL(10,2),
```

```
    availability BOOLEAN,
    ID_post INT,
    price_purchase DECIMAL(10,2),
    FOREIGN KEY (ID_post) REFERENCES supplier_fuel(ID_supplier)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_fuel	int	11		☑	☐	🔑 1
type_fuel	varchar	50		☐	☐	
season	varchar	20		☐	☐	
price	decimal	10	2	☐	☐	
availability	tinyint	1		☐	☐	
ID_post	int	11		☐	☐	
price_purchase	decimal	10	2	☐	☐	

Рис. 2.5. Таблиця goods_fuel

goods_shop — Магазинні товари

Облік супутніх товарів (омивачі, їжа, мастила тощо).

```
CREATE TABLE goods_shop (
    ID_goods INT PRIMARY KEY AUTO_INCREMENT,
    name_goods_shop VARCHAR(100),
    amount INT,
    price DECIMAL(10,2),
    discount DECIMAL(5,2)
);
```

Name	Type	Length	Decimals	Not null	Virtual
ID_goods	int	11		☑	☐
name_goods_shop	varchar	100		☐	☐
amount	int	11		☐	☐
price	decimal	10	2	☐	☐
discount	decimal	5	2	☐	☐

Рис. 2.6. Таблиця goods_shop

sale — Продаж товарів

Фіксує операції продажу магазинних товарів клієнтам працівниками.

```
CREATE TABLE sale (
    ID_sale INT PRIMARY KEY AUTO_INCREMENT,
    ID_employee INT,
    ID_goods INT,
    amount INT,
    ID_client INT,
    FOREIGN KEY (ID_employee) REFERENCES employees(ID_employee),
    FOREIGN KEY (ID_goods) REFERENCES goods_shop(ID_goods),
    FOREIGN KEY (ID_client) REFERENCES client(ID_client)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_sale	int	11		☑	☐	🔑 1
ID_employee	int	11		☐	☐	
ID_goods	int	11		☐	☐	
amount	int	11		☐	☐	
ID_client	int	11		☐	☐	

Рис. 2.7. Таблиця sale

sale_fuel — Продаж пального

Містить інформацію про продаж пального: обсяг, тип пального, клієнт і спосіб оплати.

```
CREATE TABLE sale_fuel (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    employee INT,
    fuel INT,
    amount DECIMAL(10,2),
```

```
    client INT,
    payment_type VARCHAR(50),
    FOREIGN KEY (employee) REFERENCES employees(ID_employee),
    FOREIGN KEY (fuel) REFERENCES goods_fuel(ID_fuel),
    FOREIGN KEY (client) REFERENCES client(ID_client)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID	int	11		☑	☐	🔑 1
employee	int	11		☐	☐	
fuel	int	11		☐	☐	
emount	decimal	10	2	☐	☐	
client	int	11		☐	☐	
payment_type	varchar	255		☐	☐	

Рис. 2.8. Таблиця sale_fuel

sale_fuel_items — Деталізація продажу пального

Пов’язує продаж з конкретним типом пального та літражем у кожному чеку.

```
CREATE TABLE sale_fuel_items (
    ID_fuel_item INT PRIMARY KEY AUTO_INCREMENT,
    ID_sale INT,
    ID_fuel INT,
    liters DECIMAL(10,2),
    FOREIGN KEY (ID_sale) REFERENCES sales(ID_sale),
    FOREIGN KEY (ID_fuel) REFERENCES goods_fuel(ID_fuel)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_fuel_item	int	11		☑	☐	🔑 1
ID_sale	int	11		☐	☐	
ID_fuel	int	11		☐	☐	
liters	decimal	10	2	☐	☐	

Рис. 2.9. Таблиця sale_fuel_items

sale_shop_items — Деталізація продажу товарів

Містить інформацію про кількість одиниць кожного товару в межах одного продажу.

```
CREATE TABLE sale_shop_items (
    ID_shop_item INT PRIMARY KEY AUTO_INCREMENT,
    ID_sale INT,
    ID_goods INT,
    quantity INT,
    FOREIGN KEY (ID_sale) REFERENCES sales(ID_sale),
    FOREIGN KEY (ID_goods) REFERENCES goods_shop(ID_goods)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_shop_item	int	11		☑	☐	🔑 1
ID_sale	int	11		☐	☐	
ID_goods	int	11		☐	☐	
quantity	int	11		☐	☐	

Рис. 2.10. Таблиця sale_shop_items

sales — Журнал продажів

Універсальна таблиця для зберігання інформації про всі завершені продажі.

```
CREATE TABLE sales (
    ID_sale INT PRIMARY KEY AUTO_INCREMENT,
    ID_employee INT,
    ID_client INT,
    sale_datetime DATETIME,
    payment_type VARCHAR(50),
    FOREIGN KEY (ID_employee) REFERENCES employees(ID_employee),
    FOREIGN KEY (ID_client) REFERENCES client(ID_client)
```

);

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_sale	int	11		☑	☐	🔑 1
ID_employee	int	11		☐	☐	
ID_client	int	11		☐	☐	
sale_datetime	datetime			☐	☐	
payment_type	enum			☐	☐	

Рис. 2.11. Таблиця sales

supplier_fuel — Постачальники пального

Зберігає інформацію про організації або фізичних осіб, що постачають паливо.

```
CREATE TABLE supplier_fuel (
    ID_supplier INT PRIMARY KEY AUTO_INCREMENT,
    name_supplier VARCHAR(100),
    phone VARCHAR(20),
    adress TEXT,
    code VARCHAR(20),
    notes TEXT );
```

Name	Type	Length	Decimals	Not null	Virtual	Key
ID_supplier	int	11		☑	☐	🔑 1
name_supplier	varchar	100		☐	☐	
phone	varchar	20		☐	☐	
adress	varchar	100		☐	☐	
code	varchar	20		☐	☐	
notes	text			☐	☐	

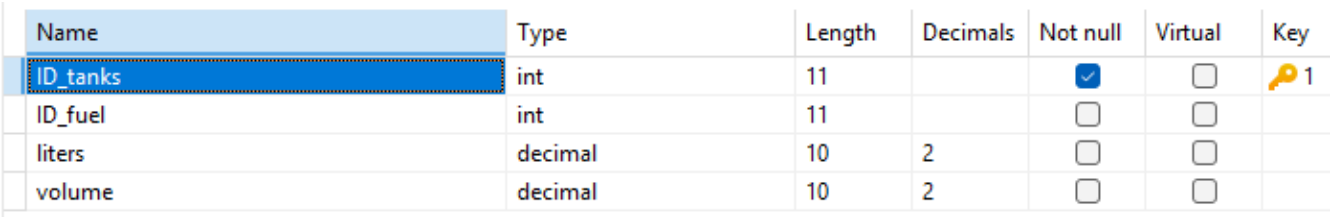
Рис. 2.12. Таблиця supplier_fuel

tanks — Цистерни для зберігання пального

Фіксує залишки та максимальний об’єм резервуарів для кожного типу пального.

```
CREATE TABLE tanks (
    ID_tanks INT PRIMARY KEY AUTO_INCREMENT,
    ID_fuel INT,
    liters DECIMAL(10,2),
```

```
    volume DECIMAL(10,2),
    FOREIGN KEY (ID_fuel) REFERENCES goods_fuel(ID_fuel)
);
```



Name	Type	Length	Decimals	Not null	Virtual	Key
ID_tanks	int	11		☑	☐	🔑 1
ID_fuel	int	11		☐	☐	
liters	decimal	10	2	☐	☐	
volume	decimal	10	2	☐	☐	

Рис. 2.13. Таблиця tanks

userauth — Авторизація працівників

Зберігає логіни, паролі та відповідність до запису працівника.

```
CREATE TABLE userauth (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50),
    password VARCHAR(255),
    employee_id INT,
    FOREIGN KEY (employee_id) REFERENCES employees(ID_employee)
);
```

Name	Type	Length	Decimals	Not null	Virtual	Key
id	int	11		☑	☐	🔑 1
username	varchar	50		☑	☐	
password	varchar	256		☑	☐	
employee_id	int	11		☑	☐	

Рис. 2.14. Таблиця userauth

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

Повний вихідний код додатку наведено в додатку А.

3.1 **Авторизація.**

Однією з перших реалізованих функціональних складових програмного забезпечення є модуль **авторизації користувача**, що дозволяє забезпечити контроль доступу до функціоналу системи відповідно до ролей персоналу. Крім того, саме на цьому етапі відбувається **ініціалізація з'єднання з сервером бази даних**.

У зв’язку з тим, що програма використовується кількома працівниками з різним рівнем відповідальності (оператор, адміністратор), виникає потреба контролювати доступ до частин функціоналу. Зокрема, лише деякі працівники мають право:

- видаляти записи;
- редагувати облікові дані інших співробітників;
- формувати звіти про продажі тощо.

Для цього реалізовано **таблицю userauth**, у якій містяться пари логін/пароль та відповідність до запису з таблиці employees.

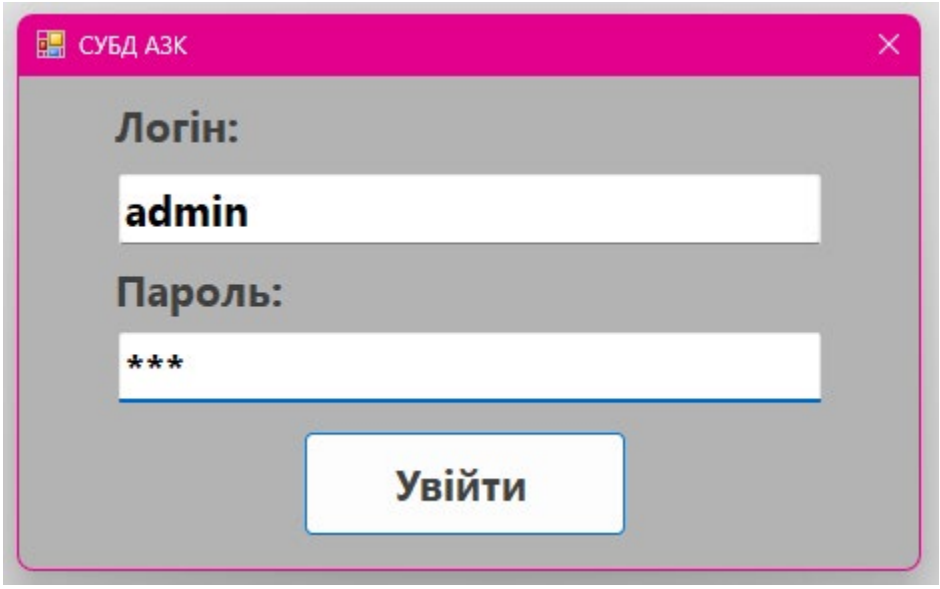


Рис. 3.1. Форма авторизації

Після запуску програми користувачу необхідно ввести логін і пароль (рис. 3.1). Після натискання кнопки "Увійти" виконується запит до бази даних:

string query = "SELECT * FROM userauth WHERE username = @login AND password = @pass";

Пароль може бути збережений у відкритому вигляді або в хешованому (у разі вдосконалення системи безпеки). Якщо запис знайдено — завантажується відповідна інформація про користувача, а також права доступу на основі його посади (поле position з таблиці employees). У разі помилкових даних — користувач отримує повідомлення з проханням повторити спробу:

MessageBox.Show("Неправильний логін або пароль");

Підключення до MySQL-серверу здійснюється через стандартну бібліотеку MySql.Data.MySqlClient. Для спрощення та уніфікації роботи з базою реалізовано допоміжний клас DBHelper, який інкапсулює логіку підключення та виконання запитів.

```
public static class DBHelper
{
  private static readonly string connectionString = "server=localhost;user=root;password=1234;database=fuelstation;";

  public static DataTable ExecuteQuery(string query)
  {
    using (MySqlConnection conn = new MySqlConnection(connectionString))
    {
      conn.Open();
      MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
      DataTable table = new DataTable();
      adapter.Fill(table);
      return table;
```

```
        }
    }
    // Інші методи: ExecuteNonQuery, ExecuteScalar...
}
```

Особливості реалізації

- Рядок підключення захований у класі й не змінюється під час виконання;
- Для кращої безпеки можливо винести логін/пароль у конфігураційний файл з шифруванням;
- Усі SQL-запити виконуються через параметризовані запити (у наступних розділах), щоб уникнути SQL-ін’єкцій.

Таким чином, підсистема авторизації виконує як контроль доступу до системи, так і забезпечує ініціалізацію сеансу роботи з базою даних, що є критично важливою умовою для подальшого функціонування інформаційної системи.

3.2 **Головна форма**

Після проходження авторизації користувач переходить до **головної форми додатку** (рис. 3.2), яка виступає центральним навігаційним елементом. У ній розміщено кнопки або пункти меню для переходу до ключових розділів системи: керування клієнтами, працівниками, товарами, паливом, продажами тощо.

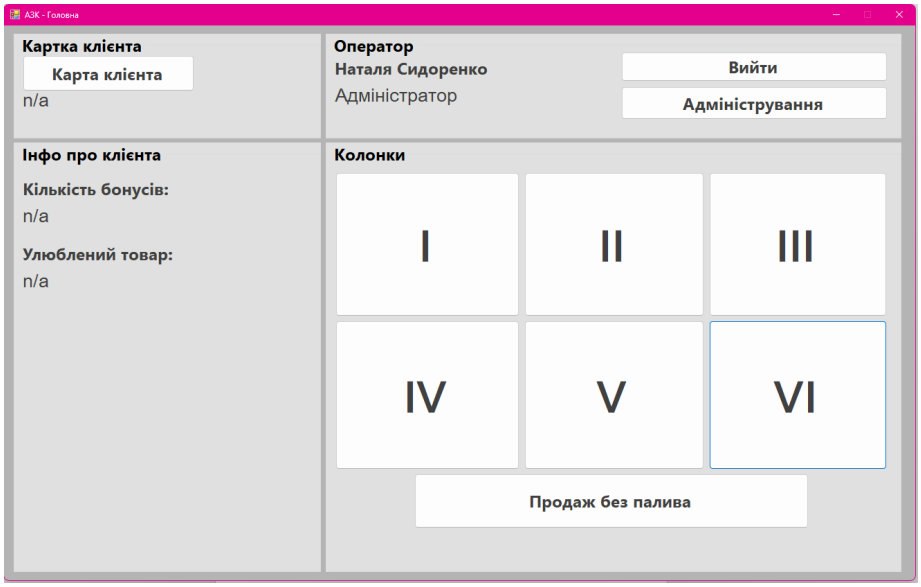


Рис. 3.2. Головна форма

Основні функції головної форми:

- Відображення поточного користувача (ПІБ, посада);
- Перехід до основних форм додатку (через кнопки або меню);
- Вихід із системи;
- Можливе відображення статистики (опційно).

Приклад виклику підформи з головної форми:

```
private void btnClients_Click(object sender, EventArgs e)
{
   ClientsForm clientsForm = new ClientsForm();
   clientsForm.ShowDialog();
}
```

Централізована структура дозволяє зробити інтерфейс зрозумілим навіть для малодосвідчених користувачів. Залежно від ролі, деякі кнопки можуть бути приховані або заблоковані.

3.3 **Формування чеку (продаж пального)**

Модуль продажу пального реалізований у формі **FuelSaleForm** (рис. 3.3), яка дозволяє оператору здійснити оформлення покупки пального клієнтом. Це одна з основних бізнес-операцій АЗС.

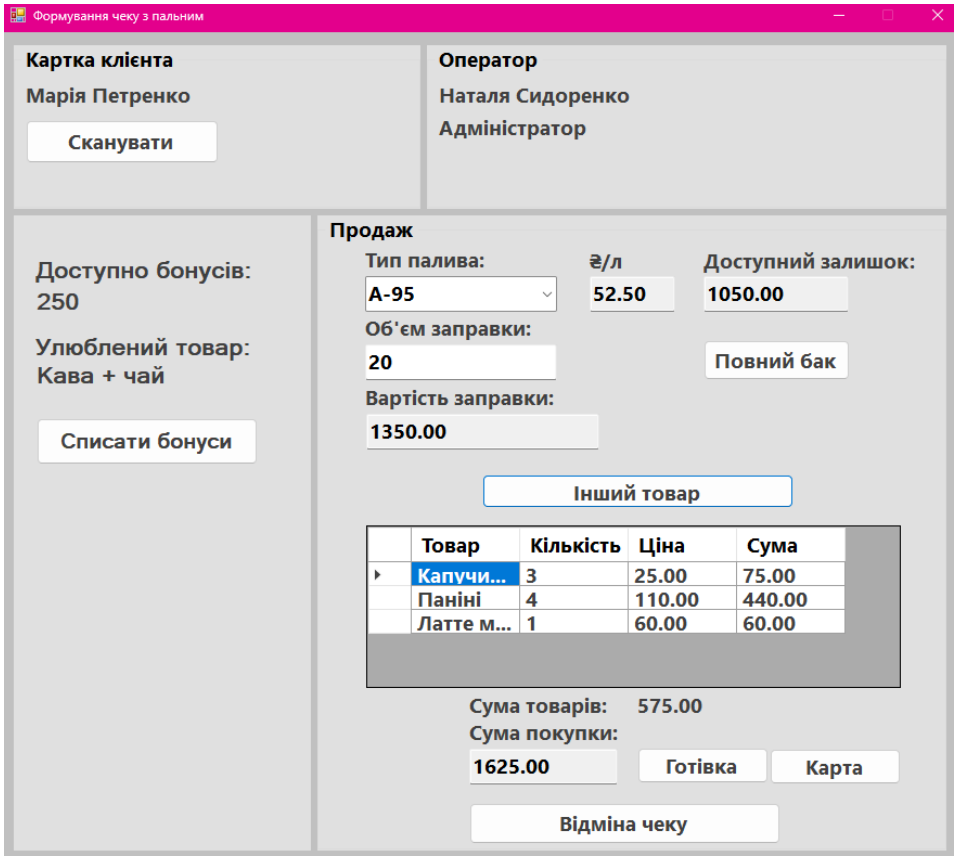


Рис. 3.3. Форма продажу пального разом з іншими товарами

Функціональні можливості:

- Вибір клієнта (із бази даних);
- Вибір пального (тип, октанове число);
- Введення кількості літрів;
- Вибір типу оплати (готівка, карта, бонуси);
- Формування та збереження чеку;
- Зменшення залишків пального в таблиці tanks.

Запит на створення запису про продаж:

```
string query = $@"
    INSERT INTO sale_fuel (employee, fuel, amount, client, payment_type)
    VALUES ({empId}, {fuelId}, {liters}, {clientId}, '{payType}');
";
```

Збереження інформації:

Таблиця sale_fuel містить основні поля продажу.

- Залишок у tanks оновлюється через UPDATE.
- За потреби, формується друкований чек або вивід на екран.

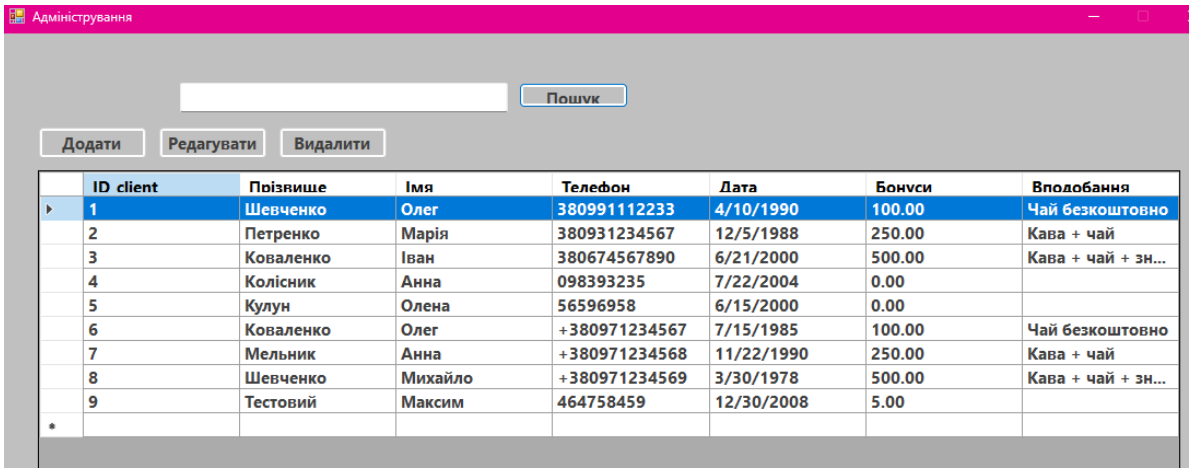
Переваги реалізації:

- Простота введення інформації;
- Автоматичне оновлення залишків;
- Прив'язка до клієнта та працівника для аналітики.

Таким чином, модуль продажу пального дозволяє швидко та безпечно оформити транзакцію із збереженням усіх необхідних даних для контролю й обліку.

3.4 **Робота з клієнтами**

Модуль керування клієнтами реалізовано у формі **ClientsForm** (рис. 3.4). Вона дозволяє переглядати, додавати, редагувати та видаляти дані клієнтів. Зокрема, це ПІБ, телефон, дата народження, нотатки, а також зв’язок із бонусною карткою.



ID client	Прізвище	Ім'я	Телефон	Дата	Бонуси	Вподобання
1	Шевченко	Олег	380991112233	4/10/1990	100.00	Чай безкоштовно
2	Петренко	Марія	380931234567	12/5/1988	250.00	Кава + чай
3	Коваленко	Іван	380674567890	6/21/2000	500.00	Кава + чай + зн...
4	Колісник	Анна	098393235	7/22/2004	0.00	
5	Кулун	Олена	56596958	6/15/2000	0.00	
6	Коваленко	Олег	+380971234567	7/15/1985	100.00	Чай безкоштовно
7	Мельник	Анна	+380971234568	11/22/1990	250.00	Кава + чай
8	Шевченко	Михайло	+380971234569	3/30/1978	500.00	Кава + чай + зн...
9	Тестовий	Максим	464758459	12/30/2008	5.00	

Рис. 3.4. Форма роботи з клієнтами

Основні функції:

- Виведення таблиці з усіма зареєстрованими клієнтами;

- Фільтрація та пошук за ПІБ, номером телефону або бонусною карткою;
- Додавання нового клієнта через окрему форму з полями введення;
- Автоматичне підв’язування існуючої або нової бонусної картки.

Приклад додавання нового клієнта:

```
string query = $@"
INSERT INTO client (last_name, first_name, middle_name, phone, ID_bonus_card, birth, notes)
VALUES ('{lname}', '{fname}', '{mname}', '{phone}', {bonusId}, '{birth}', '{notes}')";
```

Валідація введених даних:

Перед збереженням перевіряється коректність введених телефонів, дати народження та чи всі обов’язкові поля заповнено.

Робота з бонусними картками:

Форма дає змогу переглядати й редагувати тип картки, кількість бонусів, знижки, вподобання. Це реалізовано через зв’язану таблицю bonus_card.

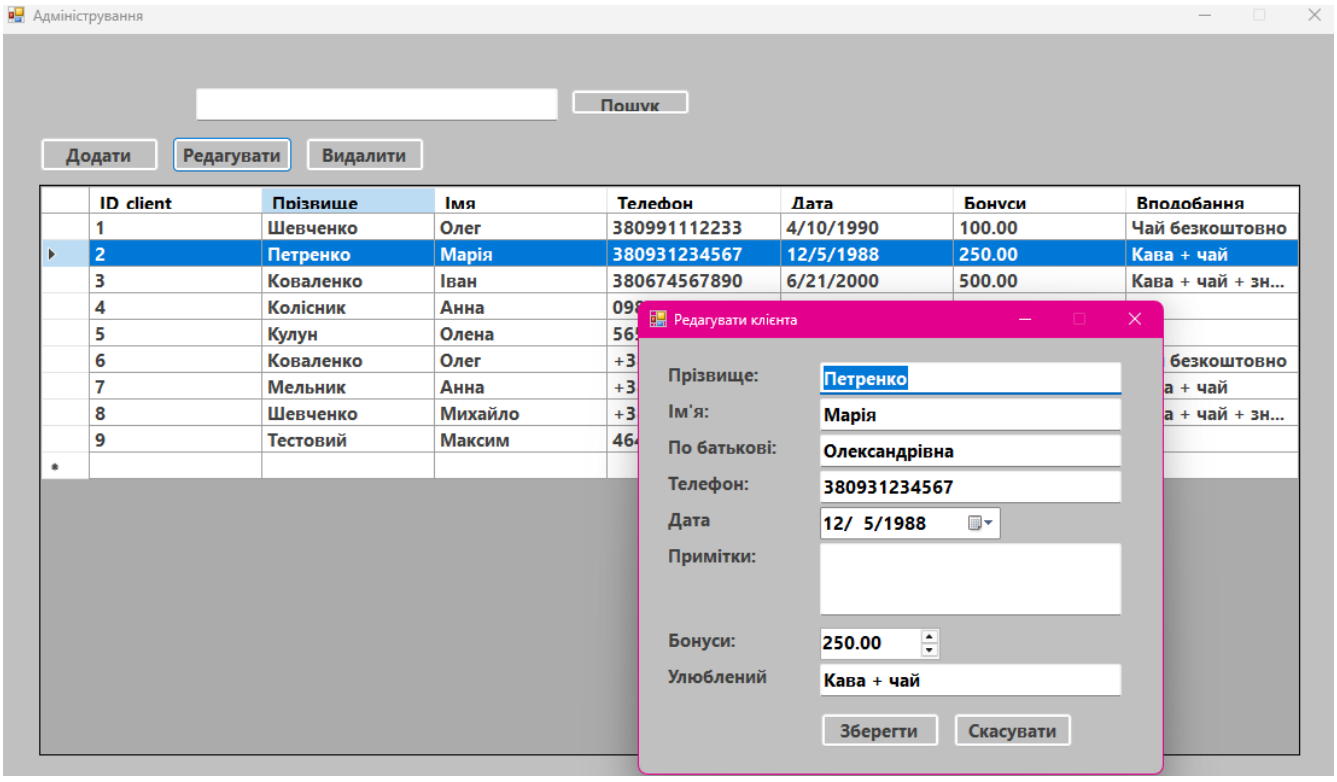


Рис. 3.5. Форми адміністрування клієнтів

Таким чином, модуль клієнтів забезпечує повний облік користувачів та їх взаємодію із системою лояльності АЗС.

3.5 **Облік пального**

Для контролю за видами пального, їхніми залишками, цінами та характеристиками реалізовано окрему форму — **FuelForm**. Вона відображає всю наявну продукцію (бензин, дизель тощо) та резервуари, в яких вона зберігається.

ID tanks	Назва	Тип палива	Залишок	Макс.	%
1	Цистерна 1	А-95	1330.00	2000.00	66.50
2	Цистерна 2	А-92	725.00	1500.00	48.33
3	Цистерна 3	ДП	13.00	1200.00	1.08
4	Цистерна 4	А-102	0.00	1000.00	0.00

Рис. 3.6. Форма обліку пального

Функціональні можливості:

- Виведення таблиці з усіма видами пального;
- Додавання, редагування та видалення записів;
- Вказування октанового числа, сезону (зимове/літнє), ціни реалізації та закупівлі.

SQL-запит додавання нового пального:

```
string query = $@"
INSERT INTO goods_fuel (type_fuel, season, price, availability, ID_post, price_purchase)
VALUES ('{type}', '{season}', {price}, 1, {supplierId}, {pricePurchase})";
```

Облік залишків у цистернах

Форма також відображає поточний рівень пального в кожній цистерні. Ці дані зберігаються в таблиці tanks.

Під час продажу пального залишки автоматично оновлюються, що дозволяє відстежувати рівень ресурсу й вчасно поповнювати запаси.

Таким чином, модуль пального забезпечує повний контроль за основним ресурсом АЗС — від технічних характеристик до логістики постачання та запасів.

3.6 **Облік товарів магазину**

Окрім пального, автозаправні станції реалізують супутні товари (їжу, напої, автохімію, аксесуари тощо). Для їх обліку створено форму **ShopForm**, яка працює з таблицею goods_shop.

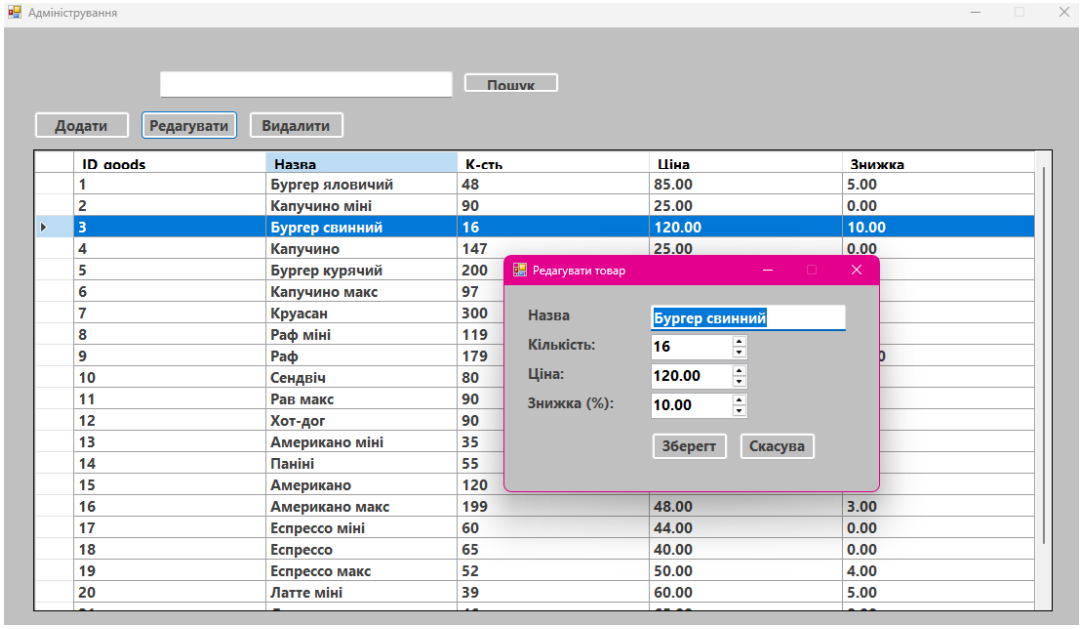


Рис. 3.7. Форма обліку товарів

Функції форми:

- Перегляд асортименту товарів із вказанням назви, кількості, ціни та знижки;
- Додавання нових товарів до асортименту;
- Редагування цін і наявності товару;
- Видалення позицій, які більше не реалізуються.

SQL-запит на додавання товару:

```
string query = $@"
INSERT INTO goods_shop (name_goods_shop, amount, price, discount)
VALUES ('{name}', {amount}, {price}, {discount})";
```

Особливості реалізації:

- Поля мають валідацію (кількість має бути додатною, знижка — в межах 0–100);
- Пошук за назвою або фільтрація по наявності;
- Автоматичне оновлення залишків після продажу (взаємодія з таблицею sale або sale_shop_items).

Таким чином, форма дозволяє тримати облік товарів у магазині в актуальному стані й ефективно керувати асортиментом.

3.7 **Тестування системи**

Для впевненості у стабільності функціонування програмного забезпечення було проведено ручне тестування всіх ключових модулів.

Об'єкти тестування:

- Авторизація та підключення до бази даних;
- Перегляд, додавання, редагування, видалення даних у формах:
  - клієнтів,
  - пального,
  - магазинних товарів;
- Проведення операцій продажу (пальне та товари);
- Валідація полів вводу.

Результати (табл. 3.1)**:**

Таблиця 3.1. Результати тестування

Модуль	Стан	Коментар
Авторизація	Успішно	Невірні логіни обробляються коректно
Робота з клієнтами	Успішно	Всі CRUD-функції працюють стабільно
Облік пального	Успішно	Залишки оновлюються після продажу
Продаж пального	Успішно	Дані зберігаються в базі, форма інформативна

Типові помилки:

- Помилки при незаповнених обов’язкових полях;
- Невалідні дані (наприклад, текст у полі "кількість").

Методи усунення:

- Додано перевірку через if-блоки до кожної критичної кнопки;
- Повідомлення користувачу через MessageBox.Show(...);
- Обмеження доступу до кнопок, якщо поля порожні або некоректні.

Таким чином, тестування показало, що система є стабільною, а користувач отримує зворотний зв’язок у разі помилкових дій, що відповідає вимогам надійності інформаційних систем.

ВИСНОВКИ

У результаті виконання дипломного проєкту було розроблено та впроваджено інформаційну систему управління операційною діяльністю автозаправної станції. Система реалізована у вигляді десктопного застосунку на платформі Windows Forms з використанням мови програмування C# та СУБД MySQL.

У процесі проєктування були досліджені потреби предметної області, сформована структура бази даних, реалізований інтерфейс користувача та впроваджено функціонал для обліку пального, товарів, клієнтів, працівників і проведення торгових операцій. Особливу увагу приділено механізмам авторизації, валідації даних, обліку залишків та системі лояльності на основі бонусних карт.

Після розробки проведено тестування, яке показало стабільну роботу всіх основних функцій. Реалізовано механізми контролю помилок і захисту від некоректних дій користувача.

Запропонована система дозволяє автоматизувати рутинні процеси, зменшити ймовірність людських помилок, підвищити точність обліку ресурсів і товарів, а також покращити рівень обслуговування клієнтів. Вона може бути ефективно застосована як у межах однієї АЗС, так і масштабована на більші підприємства із мережевою структурою.

Таким чином, поставлені в дипломному завданні цілі досягнуто повністю, а створене програмне забезпечення відповідає вимогам сучасної автоматизації бізнес-процесів у сфері роздрібної торгівлі пально-мастильними матеріалами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Advantages of Database Management Systems" – URL: https://pipeline.zoominfo.com/operations/6-benefits-of-using-database-management-systems-dbms
2. "Types of human error" – URL: https://psychsafety.com/psychological-safety-human-error/
3. "How automation can help companies remain competetive" – URL: https://sidgs.com/how-automation-can-help-companies-remain-competitive/#:~:text=Automation%20driving%20Competition&text=Automation%20can%20also%20reduce%20labor,to%20invest%20in%20other%20areas.
4. “The impact of bad data hygiene on business revenue" – URL: https://www.lakeb2b.com/blog/impact-of-bad-data-hygiene/
5. "9 effective ways to build and improve customer loyalty" – URL: https://www.indeed.com/career-advice/career-development/improve-customer-loyalty
6. "What is data security?" – URL: https://www.ibm.com/think/topics/data-security
7. "Best practices for creating effective business reports" – URL: https://www.clearpointstrategy.com/blog/business-reporting-best-practices#:~:text=An%20effective%20business%20report%20must,the%20necessary%20information%20without%20overcomplication.
8. "Why software development costs seem high and why it’s worth every penny" – URL: https://medium.com/@treveshan.naidoo/why-software-development-costs-seem-high-and-why-its-worth-every-penny-afed7fb0b4d2
9. Бен Форта “MySQL Crash Course” – друковане видання «Addison-Wesley», с.121
10. Інформаційні системи (шпаргалка) – URL: https://ukrreferat.com/chapters/shpory/informatsijni-sistemi-shpargalka.html

11. 13 Tier Architecture in C# .NET – URL: https://ravisatyadarshi.wordpress.com/2012/06/12/three-tier-architecture-in-c-net/

ДОДАТОК А

Вихідний код проекту

```
using MySql.Data.MySqlClient;
using System;
using System.Windows.Forms;

namespace FuelStation
{
   public partial class AddClientForm : Form
   {
      public int NewClientId { get; private set; } = 0;

      public AddClientForm()
      {
         InitializeComponent();
      }

      private void btnSave_Click(object sender, EventArgs e)
      {
         if (string.IsNullOrWhiteSpace(txtLastName.Text) ||
         string.IsNullOrWhiteSpace(txtFirstName.Text))
         {
            MessageBox.Show("Будь ласка, заповніть обов'язкові поля (Прізвище та Ім'я)");
            return;
         }
```

```
        try
        {
            using (MySqlConnection conn = new MySqlConnection(DBHelper.connectionString))
            {
                conn.Open();

                // Створюємо бонусну картку для клієнта
                string bonusQuery = "INSERT INTO bonus_card (amount_bonus, preferences) VALUES (0, NULL); SELECT LAST_INSERT_ID();";
                MySqlCommand bonusCmd = new MySqlCommand(bonusQuery, conn);
                int bonusCardId = Convert.ToInt32(bonusCmd.ExecuteScalar());

                // Створюємо клієнта
                string clientQuery = @"
                INSERT INTO client
                (last_name, first_name, middle_name, phone, ID_bonus_card, birth, notes)
                VALUES
                (@lastName, @firstName, @middleName, @phone, @bonusCardId, @birth, @notes);
                SELECT LAST_INSERT_ID();";

                MySqlCommand clientCmd = new MySqlCommand(clientQuery, conn);
                clientCmd.Parameters.AddWithValue("@lastName", txtLastName.Text);
                clientCmd.Parameters.AddWithValue("@firstName", txtFirstName.Text);
                clientCmd.Parameters.AddWithValue("@middleName", txtMiddleName.Text);
                clientCmd.Parameters.AddWithValue("@phone", txtPhone.Text);
```

```
                clientCmd.Parameters.AddWithValue("@bonusCardId", bonusCardId);
                clientCmd.Parameters.AddWithValue("@birth", dtpBirth.Value);
                clientCmd.Parameters.AddWithValue("@notes", txtNotes.Text);

                NewClientId = Convert.ToInt32(clientCmd.ExecuteScalar());

                MessageBox.Show($"Клієнта успішно створено! ID: {NewClientId}");
                this.DialogResult = DialogResult.OK;
                this.Close();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при створенні клієнта: {ex.Message}");
        }
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        this.DialogResult = DialogResult.Cancel;
        this.Close();
    }
  }
}
using System;
using System.Data;
using System.Windows.Forms;
```

```
namespace FuelStation
{
    public partial class AddGoodsForm : Form
    {
        private DataTable goodsData;
        private DataTable selectedGoods;

        public AddGoodsForm()
        {
            InitializeComponent();
            LoadGoods();
            InitializeSelectedGoodsTable();
        }
        private void InitializeSelectedGoodsTable()
        {
            selectedGoods = new DataTable();
            selectedGoods.Columns.Add("ID_goods", typeof(int));
            selectedGoods.Columns.Add("Name", typeof(string));
            selectedGoods.Columns.Add("Price", typeof(decimal));
            selectedGoods.Columns.Add("Quantity", typeof(int));
            selectedGoods.Columns.Add("Total", typeof(decimal), "Price * Quantity");
        }
        private void LoadGoods()
        {
            string query = "SELECT ID_goods, name_goods_shop, price FROM
goods_shop";
            goodsData = DBHelper.ExecuteQuery(query);
```

```
            dataGridViewGoods.DataSource = goodsData;
            dataGridViewGoods.Columns["ID_goods"].Visible = false;
            dataGridViewGoods.Columns["name_goods_shop"].HeaderText = "Товар";
            dataGridViewGoods.Columns["price"].HeaderText = "Ціна";

            // Додаємо колонку для кількості
            DataGridViewTextBoxColumn colQuantity = new
DataGridViewTextBoxColumn();
            colQuantity.HeaderText = "Кількість";
            colQuantity.Name = "Quantity";
            dataGridViewGoods.Columns.Add(colQuantity);
        }
        public DataTable GetSelectedGoods()
        {
            return selectedGoods;
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            selectedGoods.Clear();

            foreach (DataGridViewRow row in dataGridViewGoods.Rows)
            {
                // Перевіряємо, чи введена кількість і чи вона більше 0
                if (row.Cells["Quantity"].Value != null &&
                    int.TryParse(row.Cells["Quantity"].Value.ToString(), out int quantity) &&
                    quantity > 0)
                {
```

```
                DataRow newRow = selectedGoods.NewRow();
                newRow["ID_goods"] = row.Cells["ID_goods"].Value;
                newRow["Name"] = row.Cells["name_goods_shop"].Value;
                newRow["Price"] = Convert.ToDecimal(row.Cells["price"].Value);
                newRow["Quantity"] = quantity;
                selectedGoods.Rows.Add(newRow);
            }
        }

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

    private void btnCancel_Click(object sender, EventArgs e)
    {
        this.DialogResult = DialogResult.Cancel;
        this.Close();
    }
  }
}
using FuelStation.AdditionalModules;
using System;
using System.Windows.Forms;

namespace FuelStation
{
  public partial class AdditionalDataForm : Form
  {
```

```
public AdditionalDataForm()
{
    InitializeComponent();
}
private void OpenForm(Form form)
{
    form.TopLevel = false;
    form.FormBorderStyle = FormBorderStyle.None;
    form.Dock = DockStyle.Fill;

    panelContent.Controls.Clear();
    panelContent.Controls.Add(form);
    form.Show();
}

private void btnReports_Click(object sender, EventArgs e)
{
    ReportsForm reportsForm = new ReportsForm();
    OpenForm(reportsForm);
}

private void btnFuelBalance_Click(object sender, EventArgs e)
{
    OpenForm(new FuelBalanceForm());
}

private void btnGoodsBalance_Click(object sender, EventArgs e)
{
```

```
    OpenForm(new GoodsBalanceForm());
}

private void btnClients_Click(object sender, EventArgs e)
{
    OpenForm(new ClientsForm());
}

private void btnEmployees_Click(object sender, EventArgs e)
{
    OpenForm(new EmployeesForm());
}

private void btnSuppliers_Click(object sender, EventArgs e)
{
    OpenForm(new SuppliersForm());
}

private void btnCash_Click(object sender, EventArgs e)
{
    MessageBox.Show("Функція 'Каса' тимчасово недоступна");
}

private void btnSafe_Click(object sender, EventArgs e)
{
    MessageBox.Show("Функція 'Сейф' тимчасово недоступна");
}
```

```
        private void btnCollection_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Функція 'Інкасація' тимчасово недоступна");
        }

        private void btnGoodsReceipt_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Функція 'Прийом товару' тимчасово недоступна");
        }

        private void btnFuelReceipt_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Функція 'Прийом бензовозу' тимчасово недоступна");
        }
    }
}using System;
using System.Data;
using System.Windows.Forms;

namespace FuelStation
{
    public partial class ClientCardForm : Form
    {
        public ClientCardForm()
        {
            InitializeComponent();
        }
```

```
private void btnOK_Click(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(txtClientId.Text))
    {
        MessageBox.Show("Будь ласка, введіть номер клієнта");
        return;
    }

    int clientId;
    if (!int.TryParse(txtClientId.Text, out clientId))
    {
        MessageBox.Show("Номер клієнта має бути числом");
        return;
    }

    string query = $@"
    SELECT c.ID_client, c.first_name, c.last_name,
           b.amount_bonus, b.preferences AS favorite_product
    FROM client c
    LEFT JOIN bonus_card b ON c.ID_bonus_card = b.ID_bonus_card
    WHERE c.ID_client = {clientId}";

    DataTable dt = DBHelper.ExecuteQuery(query);

    if (dt.Rows.Count == 0)
    {
        MessageBox.Show("Клієнта з таким номером не знайдено");
        return;
```

```
        }

        DataRow row = dt.Rows[0];
        Session.ClientId = clientId;
        Session.ClientName = $"{row["first_name"]} {row["last_name"]}";
        Session.BonusAmount = row["amount_bonus"] != DBNull.Value ? Convert.ToInt32(row["amount_bonus"]) : 0;
        Session.FavoriteProduct = row["favorite_product"] != DBNull.Value ? row["favorite_product"].ToString() : "Не вказано";

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

    private void btnCreate_Click(object sender, EventArgs e)
    {
        using (AddClientForm addClientForm = new AddClientForm())
        {
            if (addClientForm.ShowDialog() == DialogResult.OK)
            {
                // Отримуємо ID нового клієнта
                int newClientId = addClientForm.NewClientId;

                // Завантажуємо дані нового клієнта
                string query = $@"
                SELECT c.ID_client, c.first_name, c.last_name,
                       b.amount_bonus, b.preferences AS favorite_product
                FROM client c
```

```
            LEFT JOIN bonus_card b ON c.ID_bonus_card = b.ID_bonus_card
            WHERE c.ID_client = {newClientId}";

            DataTable dt = DBHelper.ExecuteQuery(query);

            if (dt.Rows.Count > 0)
            {
                DataRow row = dt.Rows[0];
                Session.ClientId = newClientId;
                Session.ClientName = $"{row["first_name"]} {row["last_name"]}";
                Session.BonusAmount = row["amount_bonus"] != DBNull.Value ? Convert.ToInt32(row["amount_bonus"]) : 0;
                Session.FavoriteProduct = row["favorite_product"] != DBNull.Value ? row["favorite_product"].ToString() : "Не вказано";

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

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

```
}using MySql.Data.MySqlClient;
using System;
using System.Data;

namespace FuelStation
{
   public class DBHelper
   {
      public static string connectionString =
"server=localhost;database=fuelstation;user=root;password=;";

      public static DataTable ExecuteQuery(string query)
      {
         using (var conn = new MySqlConnection(connectionString))
         {
            conn.Open();
            using (var cmd = new MySqlCommand(query, conn))
            using (var da = new MySqlDataAdapter(cmd))
            {
               DataTable dt = new DataTable();
               da.Fill(dt);
               return dt;
            }
         }
      }
      public static bool ExecuteNonQuery(string query)
      {
         using (var conn = new MySqlConnection(connectionString))
```

```
        {
            conn.Open();
            using (var cmd = new MySqlCommand(query, conn))
            {
                int rowsAffected = cmd.ExecuteNonQuery();
                return rowsAffected > 0;
            }
        }
    }

    public static int ExecuteScalar(string query)
    {
        using (var conn = new MySqlConnection(connectionString))
        {
            conn.Open();
            using (var cmd = new MySqlCommand(query, conn))
            {
                object result = cmd.ExecuteScalar();
                return Convert.ToInt32(result);
            }
        }
    }
  }
}using ClosedXML.Excel;
using System;
using System.Windows.Forms;

namespace FuelStation
```

```
{
   public static class ExcelExporter
   {
      public static void ExportToExcel(DataGridView dataGridView)
      {
         try
         {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "Excel Files|*.xlsx";
            saveFileDialog.Title = "Зберегти звіт Excel";
            saveFileDialog.FileName = "Звіт_" +
DateTime.Now.ToString("yyyyMMdd_HHmmss");

            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
               using (var workbook = new XLWorkbook())
               {
                  var worksheet = workbook.Worksheets.Add("Звіт");

                  // Заголовки
                  for (int i = 0; i < dataGridView.Columns.Count; i++)
                  {
                     worksheet.Cell(1, i + 1).Value =
dataGridView.Columns[i].HeaderText;
                     worksheet.Cell(1, i + 1).Style.Font.Bold = true;
                  }

                  // Дані
```

```
                for (int i = 0; i < dataGridView.Rows.Count; i++)
                {
                    for (int j = 0; j < dataGridView.Columns.Count; j++)
                    {
                        worksheet.Cell(i + 2, j + 1).Value = dataGridView.Rows[i].Cells[j].Value?.ToString();
                    }
                }

                // Авто-розмір колонок
                worksheet.Columns().AdjustToContents();

                // Збереження
                workbook.SaveAs(saveFileDialog.FileName);

                MessageBox.Show($"Звіт успішно експортовано до файлу:\n{saveFileDialog.FileName}",
                                "Експорт завершено",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Information);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка при експорті до Excel: {ex.Message}",
                        "Помилка",
                        MessageBoxButtons.OK,
```

```
                    MessageBoxIcon.Error);
            }
        }
    }
}using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Windows.Forms;

namespace FuelStation
{
    public partial class FuelSaleForm : Form
    {
        private DataTable fuelTypes = new DataTable();
        private DataTable selectedGoods = new DataTable();
        private decimal fuelTotal = 0;
        private decimal goodsTotal = 0;
        private int currentFuelId = 0;
        private decimal fuelAvailable = 0;
        private decimal bonusToWidraw = 0;

        public FuelSaleForm()
        {
            InitializeComponent();
            InitializeSelectedGoodsTable();
            LoadFuelTypes();
            DisplayUserAndClientInfo();
        }
```

```
private void DisplayUserAndClientInfo()
{
    lblEmployeeName.Text = $"{Session.FullName}";
    lblEmployeePosition.Text = $"{Session.Position}";

    if (Session.ClientId > 0)
    {
        lblClientInfo.Text = $"{Session.ClientName}";
        lblBonusInfo.Text = $"{Session.BonusAmount}";
        lblFavoriteProduct.Text = $"{Session.FavoriteProduct}";
    }
    else
    {
        lblClientInfo.Text = "n/a";
        lblBonusInfo.Text = "n/a";
        lblFavoriteProduct.Text = "n/a";
    }
}
private void InitializeSelectedGoodsTable()
{
    selectedGoods.Columns.Add("ID_goods", typeof(int));
    selectedGoods.Columns.Add("Name", typeof(string));
    selectedGoods.Columns.Add("Price", typeof(decimal));
    selectedGoods.Columns.Add("Quantity", typeof(int));
    selectedGoods.Columns.Add("Total", typeof(decimal), "Price * Quantity");

    // Налаштування DataGridView для відображення товарів
    dataGridViewGoodsList.AutoGenerateColumns = false;
```

```
            dataGridViewGoodsList.Columns.Add("Name", "Товар");
            dataGridViewGoodsList.Columns["Name"].DataPropertyName = "Name";
            dataGridViewGoodsList.Columns.Add("Quantity", "Кількість");
            dataGridViewGoodsList.Columns["Quantity"].DataPropertyName = "Quantity";
            dataGridViewGoodsList.Columns.Add("Price", "Ціна");
            dataGridViewGoodsList.Columns["Price"].DataPropertyName = "Price";
            dataGridViewGoodsList.Columns["Price"].DefaultCellStyle.Format = "0.00";
            dataGridViewGoodsList.Columns.Add("Total", "Сума");
            dataGridViewGoodsList.Columns["Total"].DataPropertyName = "Total";
            dataGridViewGoodsList.Columns["Total"].DefaultCellStyle.Format = "0.00";
        }
        private void LoadFuelTypes()
        {
            string query = "SELECT ID_fuel, type_fuel AS FuelName, price FROM
goods_fuel";
            fuelTypes = DBHelper.ExecuteQuery(query);

            cmbFuelType.DisplayMember = "FuelName";
            cmbFuelType.ValueMember = "ID_fuel";
            cmbFuelType.DataSource = fuelTypes;
        }

        private void cmbFuelType_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cmbFuelType.SelectedValue != null &&
int.TryParse(cmbFuelType.SelectedValue.ToString(), out int fuelId))
            {
                currentFuelId = fuelId;
```

```
            decimal price =
Convert.ToDecimal(((DataRowView)cmbFuelType.SelectedItem)["price"]);
            txtPricePerLiter.Text = price.ToString("0.00");

            // Отримуємо залишки палива
            LoadFuelAvailability();
            CalculateFuelTotal();
        }
    }
    private void LoadFuelAvailability()
    {
        string query = $@"
        SELECT t.liters
        FROM tanks t
        WHERE t.ID_fuel = {currentFuelId}";

        DataTable dt = DBHelper.ExecuteQuery(query);

        if (dt.Rows.Count > 0 && dt.Rows[0][0] != DBNull.Value)
        {
            fuelAvailable = Convert.ToDecimal(dt.Rows[0][0]);
            txtFuelInTankAvailable.Text = fuelAvailable.ToString("0.00");
        }
        else
        {
            fuelAvailable = 0;
            txtFuelInTankAvailable.Text = "0.00";
        }
```

```
    }
    private void CalculateFuelTotal()
    {
        if (decimal.TryParse(txtLiters.Text, out decimal liters) &&
cmbFuelType.SelectedValue != null)
        {
            // Перевірка, чи достатньо палива
            if (liters > fuelAvailable)
            {
                MessageBox.Show($"Недостатньо палива! Доступно: {fuelAvailable}
л");
                txtLiters.Text = fuelAvailable.ToString("0.00");
                liters = fuelAvailable;
            }

            decimal price =
Convert.ToDecimal(((DataRowView)cmbFuelType.SelectedItem)["price"]);
            fuelTotal = liters * price;
            txtFuelTotal.Text = fuelTotal.ToString("0.00");
            UpdateGrandTotal();
        }
        else
        {
            txtFuelTotal.Text = "0.00";
            fuelTotal = 0;
            UpdateGrandTotal();
        }
    }
```

```
private void UpdateGrandTotal()
{
   decimal total = fuelTotal + goodsTotal;
   txtGrandTotal.Text = total.ToString("0.00");
}

private void btnOtherGoods_Click(object sender, EventArgs e)
{
   using (AddGoodsForm goodsForm = new AddGoodsForm())
   {
      if (goodsForm.ShowDialog() == DialogResult.OK)
      {
         // Отримуємо вибрані товари з форми
         DataTable selected = goodsForm.GetSelectedGoods();
         selectedGoods.Clear();
         foreach (DataRow row in selected.Rows)
         {
            selectedGoods.ImportRow(row);
         }

         // Оновлюємо список товарів у DataGridView
         dataGridViewGoodsList.DataSource = selectedGoods;
         goodsTotal = Convert.ToDecimal(selected.Compute("SUM(Total)", ""));
         lblGoodsTotal.Text = goodsTotal.ToString("0.00");
         UpdateGrandTotal();
      }
   }
}
```

```
    private void btnFullTank_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Функціонал буде реалізований згодом. Очікуйте оновлення ПЗ!");
    }

    private void btnPaymentByCash_Click(object sender, EventArgs e)
    {
        CompleteSale("cash");
    }

    private void btnPaymentByCard_Click(object sender, EventArgs e)
    {
        CompleteSale("card");
    }
    private void CompleteSale(string paymentType)
    {
        // Перевірка, чи обрано клієнта
        if (Session.ClientId == 0)
        {
            MessageBox.Show("Будь ласка, оберіть клієнта перед продажем");
            return;
        }

        // Перевірка, чи введена кількість літрів
        if (!decimal.TryParse(txtLiters.Text, out decimal liters) || liters <= 0)
        {
```

```
            MessageBox.Show("Будь ласка, введіть коректну кількість літрів");
            return;
        }

        // Перевірка залишків палива
        if (liters > fuelAvailable)
        {
            MessageBox.Show($"Недостатньо палива! Доступно: {fuelAvailable} л");
            return;
        }

        try
        {
            // Початок транзакції
            using (MySqlConnection conn = new MySqlConnection(DBHelper.connectionString))
            {
                conn.Open();
                MySqlTransaction transaction = conn.BeginTransaction();

                try
                {
                    // Додаємо запис про продаж
                    string saleQuery = @"
                    INSERT INTO sales (ID_employee, ID_client, sale_datetime, payment_type)
                    VALUES (@employeeId, @clientId, NOW(), @paymentType);
                    SELECT LAST_INSERT_ID();";
```

```
            MySqlCommand saleCmd = new MySqlCommand(saleQuery, conn,
transaction);
            saleCmd.Parameters.AddWithValue("@employeeId",
Session.EmployeeId);
            saleCmd.Parameters.AddWithValue("@clientId", Session.ClientId);
            saleCmd.Parameters.AddWithValue("@paymentType", paymentType);

            int saleId = Convert.ToInt32(saleCmd.ExecuteScalar());

            // Додаємо паливо до продажу
            string fuelQuery = @"
            INSERT INTO sale_fuel_items (ID_sale, ID_fuel, liters)
            VALUES (@saleId, @fuelId, @liters)";

            MySqlCommand fuelCmd = new MySqlCommand(fuelQuery, conn,
transaction);
            fuelCmd.Parameters.AddWithValue("@saleId", saleId);
            fuelCmd.Parameters.AddWithValue("@fuelId", currentFuelId);
            fuelCmd.Parameters.AddWithValue("@liters", liters);
            fuelCmd.ExecuteNonQuery();

            // Оновлюємо залишки палива
            string updateFuelQuery = @"
            UPDATE tanks
            SET liters = liters - @liters
            WHERE ID_fuel = @fuelId";
```

```
                MySqlCommand updateFuelCmd = new
MySqlCommand(updateFuelQuery, conn, transaction);
                updateFuelCmd.Parameters.AddWithValue("@liters", liters);
                updateFuelCmd.Parameters.AddWithValue("@fuelId", currentFuelId);
                updateFuelCmd.ExecuteNonQuery();

                // Додаємо товари до продажу
                foreach (DataRow row in selectedGoods.Rows)
                {
                    int goodsId = Convert.ToInt32(row["ID_goods"]);
                    int quantity = Convert.ToInt32(row["Quantity"]);

                    string goodsQuery = @"
                    INSERT INTO sale_shop_items (ID_sale, ID_goods, quantity)
                    VALUES (@saleId, @goodsId, @quantity)";

                    MySqlCommand goodsCmd = new MySqlCommand(goodsQuery,
conn, transaction);
                    goodsCmd.Parameters.AddWithValue("@saleId", saleId);
                    goodsCmd.Parameters.AddWithValue("@goodsId", goodsId);
                    goodsCmd.Parameters.AddWithValue("@quantity", quantity);
                    goodsCmd.ExecuteNonQuery();

                    // Оновлюємо залишки товару
                    string updateGoodsQuery = @"
                    UPDATE goods_shop
                    SET amount = amount - @quantity
                    WHERE ID_goods = @goodsId";
```

```
                MySqlCommand updateGoodsCmd = new
MySqlCommand(updateGoodsQuery, conn, transaction);
                updateGoodsCmd.Parameters.AddWithValue("@quantity", quantity);
                updateGoodsCmd.Parameters.AddWithValue("@goodsId", goodsId);
                updateGoodsCmd.ExecuteNonQuery();
            }

            // Зберігаємо зміни
            transaction.Commit();

            // Показуємо повідомлення про успіх
            SuccessForm successForm = new SuccessForm();
            successForm.ShowDialog();

            // Очищаємо дані клієнта в сесії
            Session.ClearClientData();

            // Закриваємо форму
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        catch (Exception ex)
        {
            transaction.Rollback();
            MessageBox.Show($"Помилка при збереженні продажу:
{ex.Message}");
        }
```

```
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка підключення до БД: {ex.Message}");
        }
    }

    private void txtLiters_TextChanged(object sender, EventArgs e)
    {
        CalculateFuelTotal();
    }

    private void btnScanCard_Click(object sender, EventArgs e)
    {
        // Заглушка
        MessageBox.Show("Функціонал буде реалізований згодом. Очікуйте оновлення ПЗ!");
    }

    private void button1_Click(object sender, EventArgs e)
    {
        bonusToWidraw = Session.BonusAmount;

        if (lblGoodsTotal.Text != "..." || lblGoodsTotal.Text != null)
        {
            if (goodsTotal >= bonusToWidraw)
            {
```

```
            lblGoodsTotal.Text = (goodsTotal - bonusToWidraw).ToString("0.00");
            Session.BonusAmount = 0;

            string bonusUpdateQuery = @"UPDATE clients SET bonus_amount = 0
WHERE ID_client = @clientId";
            using (MySqlConnection conn = new
MySqlConnection(DBHelper.connectionString))
            {
                conn.Open();
                MySqlCommand cmd = new MySqlCommand(bonusUpdateQuery,
conn);
                cmd.Parameters.AddWithValue("@clientId", Session.ClientId);
                cmd.ExecuteNonQuery();
            }
        }
    }

    private void btnDiscardCheck_Click(object sender, EventArgs e)
    {
        Close();
    }
  }
}
using System;
using System.Data;
using System.Windows.Forms;
```

```
namespace FuelStation
{
   public partial class LoginForm : Form
   {
      private int employeeId;

      public LoginForm()
      {
         InitializeComponent();
         txtPassword.PasswordChar = '*';
      }

      private void LoadEmployeeData(int employeeId)
      {
         string query = $"SELECT * FROM employees WHERE ID_employee = {employeeId}";
         DataTable dt = DBHelper.ExecuteQuery(query);

         if (dt.Rows.Count > 0)
         {
            DataRow row = dt.Rows[0];
            Session.EmployeeId = employeeId;
            Session.FullName = $"{row["first_name"]} {row["last_name"]}";
            Session.Position = row["position"].ToString();

            this.employeeId = employeeId;
         }
      }
```

```
private void btnLogin_Click_1(object sender, EventArgs e)
{
    string username = txtUsername.Text;
    string password = txtPassword.Text;

    string query = $@"
        SELECT userauth.*, employees.first_name, employees.last_name, employees.position
        FROM userauth
        INNER JOIN employees ON userauth.employee_id = employees.ID_employee
        WHERE username = '{username}' AND password = '{password}'";
    DataTable dt = DBHelper.ExecuteQuery(query);

    if (dt.Rows.Count > 0)
    {
        DataRow row = dt.Rows[0];

        int employeeId = Convert.ToInt32(row["employee_id"]);
        LoadEmployeeData(employeeId);
        Session.EmployeeId = employeeId;
        Session.FullName = $"{row["first_name"]} {row["last_name"]}";
        Session.Position = row["position"].ToString();

        // Визначаємо, чи є користувач адміністратором
        Session.IsAdmin = (Session.Position.ToLower() == "адміністратор");
        this.Hide();
```

```
            new MainForm().Show();
        }
        else
        {
            MessageBox.Show("Невірний логін або пароль");
        }
    }
  }
}
using System;
using System.Windows.Forms;

namespace FuelStation
{
  public partial class MainForm : Form
  {
    public MainForm()
    {
      InitializeComponent();
      UpdateUserInfo();
      UpdateClientInfo();
    }
    private void UpdateUserInfo()
    {
      lblUserInfo.Text = Session.IsAuthenticated ? $"{Session.FullName}" : "Не
автентифіковано";
      lblUserPosition.Text = Session.IsAuthenticated ? $"{Session.Position}" : "Не
автентифіковано";
```

```
}
private void btnLogout_Click(object sender, EventArgs e)
{
    Session.EmployeeId = 0;
    Session.ClearClientData();
    this.Hide();
    new LoginForm().Show();
}
private void UpdateClientInfo()
{
    if (Session.ClientId > 0)
    {
        lblClientName.Text = $"Клієнт: {Session.ClientName}";
        lblBonusAmount.Text = $"{Session.BonusAmount}";
        lblFavoriteProduct.Text = $"{Session.FavoriteProduct}";
    }
    else
    {
        lblClientName.Text = "n/a";
        lblBonusAmount.Text = "n/a";
        lblFavoriteProduct.Text = "n/a";
    }
}

private void btnCard_Click(object sender, EventArgs e)
{
    using (var clientForm = new ClientCardForm())
    {
```

```
            if (clientForm.ShowDialog() == DialogResult.OK)
            {
                UpdateClientInfo();
            }
        }
    }

    private void ShowFuelSaleForm()
    {
        FuelSaleForm form = new FuelSaleForm();
        form.ShowDialog();
    }

    private void btn1stFuelColumn_Click(object sender, EventArgs e) =>
ShowFuelSaleForm();

    private void btn2ndFuelColumn_Click(object sender, EventArgs e) =>
ShowFuelSaleForm();

    private void btn3rdFuelColumn_Click(object sender, EventArgs e) =>
ShowFuelSaleForm();

    private void btn4thFuelColumn_Click(object sender, EventArgs e) =>
ShowFuelSaleForm();

    private void btn5thFuelColumn_Click(object sender, EventArgs e) =>
ShowFuelSaleForm();
```

```
    private void btn6thFuelColumn_Click(object sender, EventArgs e) => ShowFuelSaleForm();

    private void btnAdmin_Click(object sender, EventArgs e)
    {
        if (Session.IsAdmin)
        {
            AdditionalDataForm adminForm = new AdditionalDataForm();
            adminForm.ShowDialog();
        }
        else
        {
            MessageBox.Show("Відмовлено в доступі. Тільки адміністратори можуть використовувати цю функцію.",
                            "Обмежений доступ",
                            MessageBoxButtons.OK,
                            MessageBoxIcon.Warning);
        }
    }

    private void btnNonFuelOperation_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Ця функція наразі не реалізована.", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
  }
}
using System;
```

```
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation
{
    public partial class ReportsForm : Form
    {
        private ComboBox cmbReportType;
        private DateTimePicker dtpStartDate;
        private DateTimePicker dtpEndDate;
        private Button btnGenerate;
        private DataGridView dgvReport;
        private Button btnExport;

        public ReportsForm()
        {
            Text = "Система звітів";
            InitializeComponent();
            InitializeLayout();
        }

        private void InitializeLayout()
        {
            this.Size = new Size(1200, 700);

            // Оптимізовані розміри лейблів
            Label lblReportType = new Label();
```

```
        lblReportType.Text = "Тип звіту:";
        lblReportType.AutoSize = false;
        lblReportType.Size = new Size(80, 20);  // Фіксована ширина
        lblReportType.Location = new Point(10, 20);
        this.Controls.Add(lblReportType);

        cmbReportType = new ComboBox();
        cmbReportType.Location = new Point(95, 20);  // Оптимальне розміщення
        cmbReportType.Size = new Size(250, 21);
        cmbReportType.DropDownStyle = ComboBoxStyle.DropDownList;
        cmbReportType.Items.AddRange(new string[] {
    "Продажі палива за період",
    "Продажі товарів за період",
    "Залишки палива",
    "Клієнтська база",
    "ТОП-10 товарів",
    "Продажі по працівникам",
    "Залишки товарів"
});
        cmbReportType.SelectedIndex = 0;
        this.Controls.Add(cmbReportType);

        Label lblPeriod = new Label();
        lblPeriod.Text = "Період:";
        lblPeriod.AutoSize = false;
        lblPeriod.Size = new Size(80, 20);
        lblPeriod.Location = new Point(10, 50);
        this.Controls.Add(lblPeriod);
```

```
dtpStartDate = new DateTimePicker();
dtpStartDate.Location = new Point(95, 50);
dtpStartDate.Size = new Size(100, 20);
dtpStartDate.Format = DateTimePickerFormat.Short;
dtpStartDate.Value = DateTime.Today.AddMonths(-1);
this.Controls.Add(dtpStartDate);

dtpEndDate = new DateTimePicker();
dtpEndDate.Location = new Point(200, 50);
dtpEndDate.Size = new Size(100, 20);
dtpEndDate.Format = DateTimePickerFormat.Short;
dtpEndDate.Value = DateTime.Today;
this.Controls.Add(dtpEndDate);

btnGenerate = new Button();
btnGenerate.Location = new Point(305, 50);
btnGenerate.Size = new Size(90, 23);
btnGenerate.Text = "Згенерувати";
btnGenerate.Click += BtnGenerate_Click;
this.Controls.Add(btnGenerate);

btnExport = new Button();
btnExport.Location = new Point(400, 50);
btnExport.Size = new Size(90, 23);
btnExport.Text = "Експорт";
btnExport.Click += BtnExport_Click;
this.Controls.Add(btnExport);
```

```
        dgvReport = new DataGridView();
        dgvReport.Location = new Point(10, 80);
        dgvReport.Size = new Size(1170, 610);
        dgvReport.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
                           AnchorStyles.Left | AnchorStyles.Right;
        dgvReport.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
        dgvReport.ReadOnly = true;
        this.Controls.Add(dgvReport);
    }
    private void BtnGenerate_Click(object sender, EventArgs e)
    {
        try
        {
            string reportType = cmbReportType.SelectedItem.ToString();
            string query = "";

            switch (reportType)
            {
                case "Продажі палива за період":
                    query = $@"
                    SELECT
                        f.type_fuel AS 'Тип палива',
                        SUM(sfi.liters) AS 'Літри',
                        SUM(sfi.liters * f.price) AS 'Сума'
                    FROM sale_fuel_items sfi
                    JOIN goods_fuel f ON sfi.ID_fuel = f.ID_fuel
```

```
                JOIN sales s ON sfi.ID_sale = s.ID_sale
                WHERE s.sale_datetime BETWEEN '{dtpStartDate.Value:yyyy-MM-
dd}' AND '{dtpEndDate.Value:yyyy-MM-dd 23:59:59}'
                GROUP BY f.type_fuel
                ORDER BY SUM(sfi.liters) DESC";
                break;

            case "Продажі товарів за період":
                query = $@"
                SELECT
                    g.name_goods_shop AS 'Товар',
                    SUM(ssi.quantity) AS 'Кількість',
                    SUM(ssi.quantity * g.price) AS 'Сума'
                FROM sale_shop_items ssi
                JOIN goods_shop g ON ssi.ID_goods = g.ID_goods
                JOIN sales s ON ssi.ID_sale = s.ID_sale
                WHERE s.sale_datetime BETWEEN '{dtpStartDate.Value:yyyy-MM-
dd}' AND '{dtpEndDate.Value:yyyy-MM-dd 23:59:59}'
                GROUP BY g.name_goods_shop
                ORDER BY SUM(ssi.quantity) DESC";
                break;

            case "Залишки палива":
                query = $@"
                SELECT
                    t.ID_tanks AS 'ID цистерни',
                    type_fuel AS 'Тип палива',
                    t.volume AS 'Макс. місткість',
```

```
        t.liters AS 'Поточний залишок',
        ROUND((t.liters / t.volume) * 100, 2) AS 'Заповненість (%)'
    FROM tanks t
    JOIN goods_fuel f ON t.ID_fuel = f.ID_fuel
    ORDER BY t.ID_tanks";
    break;

case "Клієнтська база":
    query = $@"
    SELECT
        c.ID_client AS 'ID',
        CONCAT(c.last_name, ' ', c.first_name) AS 'Клієнт',
        c.phone AS 'Телефон',
        c.birth AS 'Дата народження',
        b.amount_bonus AS 'Бонуси',
        b.preferences AS 'Улюблений товар'
    FROM client c
    LEFT JOIN bonus_card b ON c.ID_bonus_card = b.ID_bonus_card
    ORDER BY c.last_name";
    break;

case "ТОП-10 товарів":
    query = $@"
    SELECT
        g.name_goods_shop AS 'Товар',
        SUM(ssi.quantity) AS 'Кількість продажів',
        SUM(ssi.quantity * g.price) AS 'Загальна сума'
    FROM sale_shop_items ssi
```

```
            JOIN goods_shop g ON ssi.ID_goods = g.ID_goods
            GROUP BY g.name_goods_shop
            ORDER BY SUM(ssi.quantity) DESC
            LIMIT 10";
            break;

        case "Продажі по працівникам":
            query = $@"
            SELECT
                CONCAT(e.last_name, ' ', e.first_name) AS 'Працівник',
                COUNT(s.ID_sale) AS 'Кількість продажів',
                SUM(sf.total_fuel + ss.total_goods) AS 'Загальна сума'
            FROM sales s
            JOIN employees e ON s.ID_employee = e.ID_employee
            LEFT JOIN (
                SELECT sfi.ID_sale, SUM(sfi.liters * f.price) AS total_fuel
                FROM sale_fuel_items sfi
                JOIN goods_fuel f ON sfi.ID_fuel = f.ID_fuel
                GROUP BY sfi.ID_sale
            ) sf ON s.ID_sale = sf.ID_sale
            LEFT JOIN (
                SELECT ssi.ID_sale, SUM(ssi.quantity * g.price) AS total_goods
                FROM sale_shop_items ssi
                JOIN goods_shop g ON ssi.ID_goods = g.ID_goods
                GROUP BY ssi.ID_sale
            ) ss ON s.ID_sale = ss.ID_sale
            WHERE s.sale_datetime BETWEEN '{dtpStartDate.Value:yyyy-MM-
dd}' AND '{dtpEndDate.Value:yyyy-MM-dd 23:59:59}'
```

```
        GROUP BY e.ID_employee
        ORDER BY SUM(sf.total_fuel + ss.total_goods) DESC";
        break;

    case "Залишки товарів":
        query = $@"
        SELECT
            name_goods_shop AS 'Товар',
            amount AS 'Залишок',
            price AS 'Ціна',
            discount AS 'Знижка (%)'
        FROM goods_shop
        ORDER BY name_goods_shop";
        break;
}

DataTable dt = DBHelper.ExecuteQuery(query);
dgvReport.DataSource = dt;

// Форматування числових колонок
foreach (DataGridViewColumn column in dgvReport.Columns)
{
    if (column.HeaderText.Contains("Сума") ||
        column.HeaderText.Contains("Ціна") ||
        column.HeaderText.Contains("Бонуси"))
    {
        column.DefaultCellStyle.Format = "N2";
```

```
                column.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            }

            if (column.HeaderText.Contains("Кількість") ||
                column.HeaderText.Contains("Залишок") ||
                column.HeaderText.Contains("Літри"))
            {
                column.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка при генерації звіту: {ex.Message}",
                        "Помилка",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
}

private void BtnExport_Click(object sender, EventArgs e)
{
    if (dgvReport.Rows.Count == 0)
    {
        MessageBox.Show("Немає даних для експорту. Спочатку згенеруйте
звіт.",
```

```
                    "Попередження",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Warning);
                return;
            }

            ExcelExporter.ExportToExcel(dgvReport);
        }
    }
}
namespace FuelStation
{
    public static class Session
    {
        // Дані працівника
        public static int EmployeeId { get; set; }
        public static string FullName { get; set; }
        public static string Position { get; set; }
        public static bool IsAuthenticated => EmployeeId > 0;
        public static bool IsAdmin { get; set; }

        // Дані клієнта
        public static int ClientId { get; set; }
        public static string ClientName { get; set; }
        public static int BonusAmount { get; set; }
        public static string FavoriteProduct { get; set; }

        public static void ClearClientData()
```

```
        {
            ClientId = 0;
            ClientName = "";
            BonusAmount = 0;
            FavoriteProduct = "";
        }
        public static void Clear()
        {
            EmployeeId = 0;
            FullName = "";
            Position = "";
            IsAdmin = false;
            ClearClientData();
        }
    }
}
using System;
using System.Windows.Forms;

namespace FuelStation
{
    public partial class SuccessForm : Form
    {
        public SuccessForm()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
        }
```

```
        private void SuccessForm_Load(object sender, EventArgs e)
        {
            // Запускаємо таймер для автоматичного закриття
            timerClose.Start();
        }

        private void timerClose_Tick(object sender, EventArgs e)
        {
            timerClose.Stop();
            this.Close();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
    public partial class ClientEditForm : Form
```

```
{
    private int clientId = -1;
    private int bonusCardId = -1;

    private TextBox txtLastName;
    private TextBox txtFirstName;
    private TextBox txtMiddleName;
    private TextBox txtPhone;
    private DateTimePicker dtpBirth;
    private TextBox txtNotes;
    private NumericUpDown numBonusAmount;
    private TextBox txtFavoriteProduct;
    private Button btnSave;
    private Button btnCancel;

    public ClientEditForm(int id = -1)
    {
        clientId = id;
        InitializeComponent();
        InitializeLayout();
        if (clientId > 0) LoadData();
        this.Text = clientId > 0 ? "Редагувати клієнта" : "Додати клієнта";
        this.Size = new Size(450, 400);
    }

    private void InitializeLayout()
    {
        // Прізвище
```

```
Label lblLastName = new Label();
lblLastName.Text = "Прізвище:";
lblLastName.Location = new Point(20, 20);
this.Controls.Add(lblLastName);

txtLastName = new TextBox();
txtLastName.Location = new Point(150, 20);
txtLastName.Size = new Size(250, 20);
this.Controls.Add(txtLastName);

// Ім'я
Label lblFirstName = new Label();
lblFirstName.Text = "Ім'я:";
lblFirstName.Location = new Point(20, 50);
this.Controls.Add(lblFirstName);

txtFirstName = new TextBox();
txtFirstName.Location = new Point(150, 50);
txtFirstName.Size = new Size(250, 20);
this.Controls.Add(txtFirstName);

// По батькові
Label lblMiddleName = new Label();
lblMiddleName.Text = "По батькові:";
lblMiddleName.Location = new Point(20, 80);
this.Controls.Add(lblMiddleName);

txtMiddleName = new TextBox();
```

```
txtMiddleName.Location = new Point(150, 80);
txtMiddleName.Size = new Size(250, 20);
this.Controls.Add(txtMiddleName);

// Телефон
Label lblPhone = new Label();
lblPhone.Text = "Телефон:";
lblPhone.Location = new Point(20, 110);
this.Controls.Add(lblPhone);

txtPhone = new TextBox();
txtPhone.Location = new Point(150, 110);
txtPhone.Size = new Size(250, 20);
this.Controls.Add(txtPhone);

// Дата народження
Label lblBirth = new Label();
lblBirth.Text = "Дата народження:";
lblBirth.Location = new Point(20, 140);
this.Controls.Add(lblBirth);

dtpBirth = new DateTimePicker();
dtpBirth.Location = new Point(150, 140);
dtpBirth.Size = new Size(150, 20);
dtpBirth.Format = DateTimePickerFormat.Short;
this.Controls.Add(dtpBirth);

// Примітки
```

```
Label lblNotes = new Label();
lblNotes.Text = "Примітки:";
lblNotes.Location = new Point(20, 170);
this.Controls.Add(lblNotes);

txtNotes = new TextBox();
txtNotes.Location = new Point(150, 170);
txtNotes.Size = new Size(250, 60);
txtNotes.Multiline = true;
this.Controls.Add(txtNotes);

// Бонуси
Label lblBonusAmount = new Label();
lblBonusAmount.Text = "Бонуси:";
lblBonusAmount.Location = new Point(20, 240);
this.Controls.Add(lblBonusAmount);

numBonusAmount = new NumericUpDown();
numBonusAmount.Location = new Point(150, 240);
numBonusAmount.Size = new Size(100, 20);
numBonusAmount.DecimalPlaces = 2;
numBonusAmount.Minimum = 0;
numBonusAmount.Maximum = 100000;
this.Controls.Add(numBonusAmount);

// Улюблений товар
Label lblFavoriteProduct = new Label();
lblFavoriteProduct.Text = "Улюблений товар:";
```

```
        lblFavoriteProduct.Location = new Point(20, 270);
        this.Controls.Add(lblFavoriteProduct);

        txtFavoriteProduct = new TextBox();
        txtFavoriteProduct.Location = new Point(150, 270);
        txtFavoriteProduct.Size = new Size(250, 20);
        this.Controls.Add(txtFavoriteProduct);

        // Кнопки
        btnSave = new Button();
        btnSave.Location = new Point(150, 310);
        btnSave.Size = new Size(100, 30);
        btnSave.Text = "Зберегти";
        btnSave.Click += BtnSave_Click;
        this.Controls.Add(btnSave);

        btnCancel = new Button();
        btnCancel.Location = new Point(260, 310);
        btnCancel.Size = new Size(100, 30);
        btnCancel.Text = "Скасувати";
        btnCancel.Click += (s, e) => this.DialogResult = DialogResult.Cancel;
        this.Controls.Add(btnCancel);
    }
    private void LoadData()
    {
        string query = $@"
        SELECT c.*, b.amount_bonus AS 'Бонуси', b.preferences AS 'Улюблений
товар'
```

```
        FROM client c
        JOIN bonus_card b ON c.ID_bonus_card = b.ID_bonus_card
        WHERE c.ID_client = {clientId}";

        DataTable dt = DBHelper.ExecuteQuery(query);

        if (dt.Rows.Count > 0)
        {
            DataRow row = dt.Rows[0];
            txtLastName.Text = row["last_name"].ToString();
            txtFirstName.Text = row["first_name"].ToString();
            txtMiddleName.Text = row["middle_name"].ToString();
            txtPhone.Text = row["phone"].ToString();
            dtpBirth.Value = Convert.ToDateTime(row["birth"]);
            txtNotes.Text = row["notes"].ToString();
            numBonusAmount.Value = Convert.ToDecimal(row["amount_bonus"]);
            txtFavoriteProduct.Text = row["preferences"].ToString();
            bonusCardId = Convert.ToInt32(row["ID_bonus_card"]);
        }
    }

    private void BtnSave_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtLastName.Text) ||
string.IsNullOrWhiteSpace(txtFirstName.Text))
        {
            MessageBox.Show("Будь ласка, заповніть обов'язкові поля (Прізвище та
Ім'я)");
```

```
            return;
        }

        try
        {
            // Оновлення або створення бонусної карти
            string bonusQuery;
            if (bonusCardId > 0)
            {
                bonusQuery = $@"
                UPDATE bonus_card
                SET amount_bonus = {numBonusAmount.Value},
                    preferences = '{txtFavoriteProduct.Text.Replace("'", "''")}'
                WHERE ID_bonus_card = {bonusCardId}";
            }
            else
            {
                bonusQuery = $@"
                INSERT INTO bonus_card (amount_bonus, preferences)
                VALUES ({numBonusAmount.Value},
'{txtFavoriteProduct.Text.Replace("'", "''")}');
                SELECT LAST_INSERT_ID();";
            }

            // Якщо це новий клієнт, створюємо бонусну карту і отримуємо її ID
            if (clientId <= 0)
            {
                bonusCardId = DBHelper.ExecuteScalar(bonusQuery);
```

```
            }
            else
            {
                DBHelper.ExecuteNonQuery(bonusQuery);
            }

            // Оновлення або створення клієнта
            string clientQuery;
            if (clientId > 0)
            {
                clientQuery = $@"
                UPDATE client SET
                    last_name = '{txtLastName.Text.Replace("'", "''")}',
                    first_name = '{txtFirstName.Text.Replace("'", "''")}',
                    middle_name = '{txtMiddleName.Text.Replace("'", "''")}',
                    phone = '{txtPhone.Text}',
                    birth = '{dtpBirth.Value:yyyy-MM-dd}',
                    notes = '{txtNotes.Text.Replace("'", "''")}'
                WHERE ID_client = {clientId}";
            }
            else
            {
                clientQuery = $@"
                INSERT INTO client
                    (last_name, first_name, middle_name, phone, ID_bonus_card, birth,
notes)
                VALUES
```

```
                ('{txtLastName.Text.Replace("'", "''")}', '{txtFirstName.Text.Replace("'",
"''")}', '{txtMiddleName.Text.Replace("'", "''")}',
                '{txtPhone.Text}', {bonusCardId}, '{dtpBirth.Value:yyyy-MM-dd}',
'{txtNotes.Text.Replace("'", "''")}')";
            }

            bool success = DBHelper.ExecuteNonQuery(clientQuery);
            if (success)
            {
                MessageBox.Show("Дані клієнта збережено успішно");
                this.DialogResult = DialogResult.OK;
            }
            else
            {
                MessageBox.Show("Помилка при збереженні даних клієнта");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при збереженні: {ex.Message}");
        }
    }
}using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
```

```
namespace FuelStation.AdditionalModules
{
    public partial class ClientsForm : Form
    {
        private DataGridView dataGridView;
        private Button btnAdd;
        private Button btnEdit;
        private Button btnDelete;
        private TextBox txtSearch;
        private Button btnSearch;

        public ClientsForm()
        {
            Text = "Управління клієнтами";
            InitializeComponent();
            InitializeLayout();
        }

        private void InitializeLayout()
        {
            this.Size = new Size(1000, 600);

            // Пошук
            txtSearch = new TextBox();
            txtSearch.Location = new Point(150, 20);
            txtSearch.Size = new Size(300, 20);
            this.Controls.Add(txtSearch);
```

```
btnSearch = new Button();
btnSearch.Location = new Point(460, 20);
btnSearch.Size = new Size(100, 23);
btnSearch.Text = "Пошук";
btnSearch.Click += BtnSearch_Click;
this.Controls.Add(btnSearch);

// Кнопки керування
btnAdd = new Button();
btnAdd.Location = new Point(20, 60);
btnAdd.Size = new Size(100, 30);
btnAdd.Text = "Додати";
btnAdd.Click += BtnAdd_Click;
this.Controls.Add(btnAdd);

btnEdit = new Button();
btnEdit.Location = new Point(130, 60);
btnEdit.Size = new Size(100, 30);
btnEdit.Text = "Редагувати";
btnEdit.Click += BtnEdit_Click;
this.Controls.Add(btnEdit);

btnDelete = new Button();
btnDelete.Location = new Point(240, 60);
btnDelete.Size = new Size(100, 30);
btnDelete.Text = "Видалити";
btnDelete.Click += BtnDelete_Click;
this.Controls.Add(btnDelete);
```

```
        // Таблиця даних
        dataGridView = new DataGridView();
        dataGridView.Location = new Point(20, 100);
        dataGridView.Size = new Size(950, 450);
        dataGridView.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
AnchorStyles.Left | AnchorStyles.Right;
        dataGridView.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
        dataGridView.ReadOnly = true;
        dataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
        this.Controls.Add(dataGridView);

        dataGridView.AutoResizeColumnHeadersHeight();
        dataGridView.AutoResizeRows();
        dataGridView.AutoResizeColumns();
    }
    private void LoadData(string search = "")
    {
        string query = @"
        SELECT c.ID_client, c.last_name AS 'Прізвище', c.first_name 'Ім'я', c.phone AS
'Телефон', c.birth AS 'Дата народження',
               b.amount_bonus AS 'Бонуси', b.preferences AS 'Вподобання'
        FROM client c
        LEFT JOIN bonus_card b ON c.ID_bonus_card = b.ID_bonus_card";

        if (!string.IsNullOrWhiteSpace(search))
        {
```

```
            query += $" WHERE c.last_name LIKE '%{search}%' OR c.first_name LIKE
'%{search}%' OR c.phone LIKE '%{search}%'";
        }

        DataTable dt = DBHelper.ExecuteQuery(query);
        dataGridView.DataSource = dt;
    }
    private void BtnSearch_Click(object sender, EventArgs e)
    {
        LoadData(txtSearch.Text);
    }

    private void BtnAdd_Click(object sender, EventArgs e)
    {
        using (ClientEditForm editForm = new ClientEditForm())
        {
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }
    }

    private void BtnEdit_Click(object sender, EventArgs e)
    {
        if (dataGridView.SelectedRows.Count == 0)
        {
            MessageBox.Show("Будь ласка, оберіть клієнта для редагування");
```

```
            return;
        }

        int clientId = Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_client"].Value);
        using (ClientEditForm editForm = new ClientEditForm(clientId))
        {
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }
    }

    private void BtnDelete_Click(object sender, EventArgs e)
    {
        if (dataGridView.SelectedRows.Count == 0)
        {
            MessageBox.Show("Будь ласка, оберіть клієнта для видалення");
            return;
        }

        int clientId = Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_client"].Value);
        string lastName = dataGridView.SelectedRows[0].Cells["last_name"].Value.ToString();
        string firstName = dataGridView.SelectedRows[0].Cells["first_name"].Value.ToString();
```

```
        if (MessageBox.Show($"Ви дійсно бажаєте видалити клієнта '{lastName} {firstName}'?",
            "Підтвердження видалення", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            // Спочатку отримаємо ID бонусної карти для видалення
            string getBonusCardIdQuery = $"SELECT ID_bonus_card FROM client WHERE ID_client = {clientId}";
            int bonusCardId = DBHelper.ExecuteScalar(getBonusCardIdQuery);

            // Видаляємо клієнта
            string deleteClientQuery = $"DELETE FROM client WHERE ID_client = {clientId}";
            bool clientDeleted = DBHelper.ExecuteNonQuery(deleteClientQuery);

            if (clientDeleted)
            {
                // Тепер видаляємо бонусну карту
                string deleteBonusCardQuery = $"DELETE FROM bonus_card WHERE ID_bonus_card = {bonusCardId}";
                DBHelper.ExecuteNonQuery(deleteBonusCardQuery);

                MessageBox.Show("Клієнта успішно видалено");
                LoadData();
            }
            else
            {
```

```
                MessageBox.Show("Помилка при видаленні клієнта");
            }
        }
    }
}using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
    public partial class EmployeeEditForm : Form
    {
        private int employeeId = -1;

        private TextBox txtLastName;
        private TextBox txtFirstName;
        private TextBox txtPosition;
        private TextBox txtPhone;
        private TextBox txtBankCard;
        private DateTimePicker dtpWorkStart;
        private Button btnSave;
        private Button btnCancel;

        public EmployeeEditForm(int id = -1)
        {
            employeeId = id;
```

```
    InitializeComponent();
    InitializeLayout();
    if (employeeId > 0) LoadData();
    this.Text = employeeId > 0 ? "Редагувати працівника" : "Додати працівника";
    this.Size = new Size(450, 350);
}

private void InitializeLayout()
{
    // Прізвище
    Label lblLastName = new Label();
    lblLastName.Text = "Прізвище:";
    lblLastName.Location = new Point(20, 20);
    this.Controls.Add(lblLastName);

    txtLastName = new TextBox();
    txtLastName.Location = new Point(150, 20);
    txtLastName.Size = new Size(250, 20);
    this.Controls.Add(txtLastName);

    // Ім'я
    Label lblFirstName = new Label();
    lblFirstName.Text = "Ім'я:";
    lblFirstName.Location = new Point(20, 50);
    this.Controls.Add(lblFirstName);

    txtFirstName = new TextBox();
    txtFirstName.Location = new Point(150, 50);
```

```
txtFirstName.Size = new Size(250, 20);
this.Controls.Add(txtFirstName);

// Посада
Label lblPosition = new Label();
lblPosition.Text = "Посада:";
lblPosition.Location = new Point(20, 80);
this.Controls.Add(lblPosition);

txtPosition = new TextBox();
txtPosition.Location = new Point(150, 80);
txtPosition.Size = new Size(250, 20);
this.Controls.Add(txtPosition);

// Телефон
Label lblPhone = new Label();
lblPhone.Text = "Телефон:";
lblPhone.Location = new Point(20, 110);
this.Controls.Add(lblPhone);

txtPhone = new TextBox();
txtPhone.Location = new Point(150, 110);
txtPhone.Size = new Size(250, 20);
this.Controls.Add(txtPhone);

// Банківська карта
Label lblBankCard = new Label();
lblBankCard.Text = "Банк. карта:";
```

```
lblBankCard.Location = new Point(20, 140);
this.Controls.Add(lblBankCard);

txtBankCard = new TextBox();
txtBankCard.Location = new Point(150, 140);
txtBankCard.Size = new Size(250, 20);
this.Controls.Add(txtBankCard);

// Дата початку роботи
Label lblWorkStart = new Label();
lblWorkStart.Text = "Дата початку:";
lblWorkStart.Location = new Point(20, 170);
this.Controls.Add(lblWorkStart);

dtpWorkStart = new DateTimePicker();
dtpWorkStart.Location = new Point(150, 170);
dtpWorkStart.Size = new Size(150, 20);
dtpWorkStart.Format = DateTimePickerFormat.Short;
this.Controls.Add(dtpWorkStart);

// Кнопки
btnSave = new Button();
btnSave.Location = new Point(150, 220);
btnSave.Size = new Size(100, 30);
btnSave.Text = "Зберегти";
btnSave.Click += BtnSave_Click;
this.Controls.Add(btnSave);
```

```
        btnCancel = new Button();
        btnCancel.Location = new Point(260, 220);
        btnCancel.Size = new Size(100, 30);
        btnCancel.Text = "Скасувати";
        btnCancel.Click += (s, e) => this.DialogResult = DialogResult.Cancel;
        this.Controls.Add(btnCancel);
    }
    private void LoadData()
    {
        string query = $"SELECT * FROM employees WHERE ID_employee = {employeeId}";
        DataTable dt = DBHelper.ExecuteQuery(query);

        if (dt.Rows.Count > 0)
        {
            DataRow row = dt.Rows[0];
            txtLastName.Text = row["last_name"].ToString();
            txtFirstName.Text = row["first_name"].ToString();
            txtPosition.Text = row["position"].ToString();
            txtPhone.Text = row["phone"].ToString();
            txtBankCard.Text = row["bank_card"].ToString();
            dtpWorkStart.Value = Convert.ToDateTime(row["date_work_start"]);
        }
    }

    private void BtnSave_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtLastName.Text) ||
```

```
            string.IsNullOrWhiteSpace(txtFirstName.Text) ||
            string.IsNullOrWhiteSpace(txtPosition.Text))
        {
            MessageBox.Show("Будь ласка, заповніть обов'язкові поля (Прізвище, Ім'я, Посада)");
            return;
        }

        try
        {
            string query;
            if (employeeId > 0)
            {
                query = $@"
                UPDATE employees SET
                    last_name = '{txtLastName.Text.Replace("'", "''")}',
                    first_name = '{txtFirstName.Text.Replace("'", "''")}',
                    position = '{txtPosition.Text.Replace("'", "''")}',
                    phone = '{txtPhone.Text}',
                    bank_card = '{txtBankCard.Text}',
                    date_work_start = '{dtpWorkStart.Value:yyyy-MM-dd}'
                WHERE ID_employee = {employeeId}";
            }
            else
            {
                query = $@"
                INSERT INTO employees
                    (last_name, first_name, position, phone, bank_card, date_work_start)
```

```
                VALUES
                    ('{txtLastName.Text.Replace("", "")}', '{txtFirstName.Text.Replace("",
"")}',
                    '{txtPosition.Text.Replace("", "")}', '{txtPhone.Text}',
                    '{txtBankCard.Text}', '{dtpWorkStart.Value:yyyy-MM-dd}')";
            }

            bool success = DBHelper.ExecuteNonQuery(query);
            if (success)
            {
                MessageBox.Show("Дані працівника збережено успішно");
                this.DialogResult = DialogResult.OK;
            }
            else
            {
                MessageBox.Show("Помилка при збереженні даних працівника");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при збереженні: {ex.Message}");
        }
    }
  }
}
using System;
using System.Data;
using System.Drawing;
```

```
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
    public partial class EmployeesForm : Form
    {
        private DataGridView dataGridView;
        private Button btnAdd;
        private Button btnEdit;
        private Button btnDelete;
        private TextBox txtSearch;
        private Button btnSearch;

        public EmployeesForm()
        {
            Text = "Управління працівниками";
            InitializeComponent();
            InitializeLayout();
            LoadData();
        }

        private void InitializeLayout()
        {
            this.Size = new Size(1000, 600);

            // Пошук
            txtSearch = new TextBox();
            txtSearch.Location = new Point(150, 20);
```

```
txtSearch.Size = new Size(300, 20);
this.Controls.Add(txtSearch);

btnSearch = new Button();
btnSearch.Location = new Point(460, 20);
btnSearch.Size = new Size(100, 23);
btnSearch.Text = "Пошук";
btnSearch.Click += BtnSearch_Click;
this.Controls.Add(btnSearch);

// Кнопки керування
btnAdd = new Button();
btnAdd.Location = new Point(20, 60);
btnAdd.Size = new Size(100, 30);
btnAdd.Text = "Додати";
btnAdd.Click += BtnAdd_Click;
this.Controls.Add(btnAdd);

btnEdit = new Button();
btnEdit.Location = new Point(130, 60);
btnEdit.Size = new Size(100, 30);
btnEdit.Text = "Редагувати";
btnEdit.Click += BtnEdit_Click;
this.Controls.Add(btnEdit);

btnDelete = new Button();
btnDelete.Location = new Point(240, 60);
btnDelete.Size = new Size(100, 30);
```

```
        btnDelete.Text = "Видалити";
        btnDelete.Click += BtnDelete_Click;
        this.Controls.Add(btnDelete);

        // Таблиця даних
        dataGridView = new DataGridView();
        dataGridView.Location = new Point(20, 100);
        dataGridView.Size = new Size(950, 450);
        dataGridView.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
AnchorStyles.Left | AnchorStyles.Right;
        dataGridView.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
        dataGridView.ReadOnly = true;
        dataGridView.AutoSizeColumnsMode                                    =
DataGridViewAutoSizeColumnsMode.Fill;
        this.Controls.Add(dataGridView);
    }
    private void LoadData(string search = "")
    {
        string query = "SELECT ID_employee, last_name AS 'Прізвище', first_name
AS 'Імя', position AS 'Посада', phone AS 'Телефон', " +
            "bank_card AS 'Банк. карта', date_work_start AS 'Дата поч. роботи' FROM
employees";

        if (!string.IsNullOrWhiteSpace(search))
        {
            query += $" WHERE last_name LIKE '%{search}%' OR first_name LIKE
'%{search}%' OR position LIKE '%{search}%'";
        }
```

```
    DataTable dt = DBHelper.ExecuteQuery(query);
    dataGridView.DataSource = dt;
}

private void BtnSearch_Click(object sender, EventArgs e)
{
    LoadData(txtSearch.Text);
}

private void BtnAdd_Click(object sender, EventArgs e)
{
    using (EmployeeEditForm editForm = new EmployeeEditForm())
    {
        if (editForm.ShowDialog() == DialogResult.OK)
        {
            LoadData();
        }
    }
}

private void BtnEdit_Click(object sender, EventArgs e)
{
    if (dataGridView.SelectedRows.Count == 0)
    {
        MessageBox.Show("Будь ласка, оберіть працівника для редагування");
        return;
    }
```

```
        int employeeId = Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_employee"].Value);
        using (EmployeeEditForm editForm = new EmployeeEditForm(employeeId))
        {
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }
    }

    private void BtnDelete_Click(object sender, EventArgs e)
    {
        if (dataGridView.SelectedRows.Count == 0)
        {
            MessageBox.Show("Будь ласка, оберіть працівника для видалення");
            return;
        }

        int employeeId = Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_employee"].Value);
        string lastName = dataGridView.SelectedRows[0].Cells["last_name"].Value.ToString();
        string firstName = dataGridView.SelectedRows[0].Cells["first_name"].Value.ToString();
```

```
        if (MessageBox.Show($"Ви дійсно бажаєте видалити працівника '{lastName} {firstName}'?",
            "Підтвердження видалення", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            string query = $"DELETE FROM employees WHERE ID_employee = {employeeId}";
            if (DBHelper.ExecuteNonQuery(query))
            {
                MessageBox.Show("Працівника успішно видалено");
                LoadData();
            }
            else
            {
                MessageBox.Show("Помилка при видаленні працівника");
            }
        }
    }
  }
}
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
  public partial class FuelBalanceForm : Form
  {
```

```
private DataGridView dataGridView;

public FuelBalanceForm()
{
    Text = "Залишки палива";
    InitializeComponent();
    InitializeLayout();
    LoadFuelData();
}
protected void InitializeLayout()
{
    dataGridView = new DataGridView();
    dataGridView.Dock = DockStyle.Fill;
    dataGridView.Location = new Point(0, 0);
    dataGridView.Size = new Size(900, 500);
    dataGridView.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
                          AnchorStyles.Left | AnchorStyles.Right;
    dataGridView.AutoResizeColumnHeadersHeight();
    dataGridView.AutoResizeRows();
    dataGridView.AutoResizeColumns();

    this.Controls.Add(dataGridView);
    dataGridView.BringToFront();
}
private void LoadFuelData()
{
    string query = @"
    SELECT t.ID_tanks,
```

```
            CONCAT('Цистерна ', t.ID_tanks) AS 'Назва цистерни',
            f.type_fuel AS 'Тип палива',
            t.liters AS 'Залишок',
            t.volume AS 'Макс. місткість',
            ROUND((t.liters / t.volume) * 100, 2) AS '%'
        FROM tanks t
        JOIN goods_fuel f ON t.ID_fuel = f.ID_fuel";

        DataTable dt = DBHelper.ExecuteQuery(query);
        dataGridView.DataSource = dt;
    }
  }
}using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
  public partial class GoodsBalanceForm : Form
  {
    private DataGridView dataGridView;
    private Button btnAdd;
    private Button btnEdit;
    private Button btnDelete;
    private TextBox txtSearch;
    private Button btnSearch;
```

```
public GoodsBalanceForm()
{
    Text = "Залишки товарів";
    InitializeComponent();
    InitializeLayout();
}
private void InitializeLayout()
{
    // Розміри форми
    this.Size = new Size(1000, 600);

    // Пошук
    txtSearch = new TextBox();
    txtSearch.Location = new Point(150, 20);
    txtSearch.Size = new Size(300, 20);
    this.Controls.Add(txtSearch);

    btnSearch = new Button();
    btnSearch.Location = new Point(460, 20);
    btnSearch.Size = new Size(100, 23);
    btnSearch.Text = "Пошук";
    btnSearch.Click += BtnSearch_Click;
    this.Controls.Add(btnSearch);

    // Кнопки керування
    btnAdd = new Button();
    btnAdd.Location = new Point(20, 60);
    btnAdd.Size = new Size(100, 30);
```

```
        btnAdd.Text = "Додати";
        btnAdd.Click += BtnAdd_Click;
        this.Controls.Add(btnAdd);

        btnEdit = new Button();
        btnEdit.Location = new Point(130, 60);
        btnEdit.Size = new Size(100, 30);
        btnEdit.Text = "Редагувати";
        btnEdit.Click += BtnEdit_Click;
        this.Controls.Add(btnEdit);

        btnDelete = new Button();
        btnDelete.Location = new Point(240, 60);
        btnDelete.Size = new Size(100, 30);
        btnDelete.Text = "Видалити";
        btnDelete.Click += BtnDelete_Click;
        this.Controls.Add(btnDelete);

        // Таблиця даних
        dataGridView = new DataGridView();
        dataGridView.Location = new Point(20, 100);
        dataGridView.Size = new Size(950, 450);
        dataGridView.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
                              AnchorStyles.Left | AnchorStyles.Right;
        dataGridView.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
        dataGridView.ReadOnly = true;
        dataGridView.AutoSizeColumnsMode                                    =
DataGridViewAutoSizeColumnsMode.Fill;
```

```
        this.Controls.Add(dataGridView);

        dataGridView.AutoResizeColumnHeadersHeight();
        dataGridView.AutoResizeRows();
        dataGridView.AutoResizeColumns();
    }
    private void LoadData(string search = "")
    {
        string query = "SELECT ID_goods, name_goods_shop AS 'Назва', amount AS 'К-сть', price AS 'Ціна', discount AS 'Знижка' FROM goods_shop";

        if (!string.IsNullOrWhiteSpace(search))
        {
            query += $" WHERE name_goods_shop LIKE '%{search}%'";
        }

        DataTable dt = DBHelper.ExecuteQuery(query);
        dataGridView.DataSource = dt;
    }

    private void BtnSearch_Click(object sender, EventArgs e)
    {
        LoadData(txtSearch.Text);
    }

    private void BtnAdd_Click(object sender, EventArgs e)
    {
        using (GoodsEditForm editForm = new GoodsEditForm())
```

```
        {
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }
    }

    private void BtnEdit_Click(object sender, EventArgs e)
    {
        if (dataGridView.SelectedRows.Count == 0)
        {
            MessageBox.Show("Будь ласка, оберіть товар для редагування");
            return;
        }

        int goodsId =
Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_goods"].Value);
        using (GoodsEditForm editForm = new GoodsEditForm(goodsId))
        {
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }
    }

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

```
    {
        if (dataGridView.SelectedRows.Count == 0)
        {
            MessageBox.Show("Будь ласка, оберіть товар для видалення");
            return;
        }

        int goodsId = Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_goods"].Value);
        string name = dataGridView.SelectedRows[0].Cells["name_goods_shop"].Value.ToString();

        if (MessageBox.Show($"Ви дійсно бажаєте видалити товар '{name}'?",
            "Підтвердження видалення", MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            string query = $"DELETE FROM goods_shop WHERE ID_goods = {goodsId}";
            // Потенційна помилка:
            if (DBHelper.ExecuteNonQuery(query))
            {
                MessageBox.Show("Товар успішно видалено");
                LoadData();
            }
        }
    }
}using System;
```

```
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
    public partial class GoodsEditForm : Form
    {
        private int goodsId = -1;
        private TextBox txtName;
        private NumericUpDown numAmount;
        private NumericUpDown numPrice;
        private NumericUpDown numDiscount;
        private Button btnSave;
        private Button btnCancel;

        public GoodsEditForm(int id = -1)
        {
            goodsId = id;
            InitializeComponent();
            InitializeLayout();
            if (goodsId > 0) LoadData();
            this.Text = goodsId > 0 ? "Редагувати товар" : "Додати товар";
            this.Size = new Size(400, 250);
        }

        private void InitializeLayout()
        {
```

```
// Назва товару
Label lblName = new Label();
lblName.Text = "Назва товару:";
lblName.Location = new Point(20, 20);
this.Controls.Add(lblName);

txtName = new TextBox();
txtName.Location = new Point(150, 20);
txtName.Size = new Size(200, 20);
this.Controls.Add(txtName);

// Кількість
Label lblAmount = new Label();
lblAmount.Text = "Кількість:";
lblAmount.Location = new Point(20, 50);
this.Controls.Add(lblAmount);

numAmount = new NumericUpDown();
numAmount.Location = new Point(150, 50);
numAmount.Size = new Size(100, 20);
numAmount.Minimum = 0;
numAmount.Maximum = 10000;
this.Controls.Add(numAmount);

// Ціна
Label lblPrice = new Label();
lblPrice.Text = "Ціна:";
lblPrice.Location = new Point(20, 80);
```

```
this.Controls.Add(lblPrice);

numPrice = new NumericUpDown();
numPrice.Location = new Point(150, 80);
numPrice.Size = new Size(100, 20);
numPrice.DecimalPlaces = 2;
numPrice.Minimum = 0;
numPrice.Maximum = 100000;
this.Controls.Add(numPrice);

// Знижка
Label lblDiscount = new Label();
lblDiscount.Text = "Знижка (%):";
lblDiscount.Location = new Point(20, 110);
this.Controls.Add(lblDiscount);

numDiscount = new NumericUpDown();
numDiscount.Location = new Point(150, 110);
numDiscount.Size = new Size(100, 20);
numDiscount.DecimalPlaces = 2;
numDiscount.Minimum = 0;
numDiscount.Maximum = 100;
this.Controls.Add(numDiscount);

// Кнопки
btnSave = new Button();
btnSave.Location = new Point(150, 150);
btnSave.Size = new Size(80, 30);
```

```
        btnSave.Text = "Зберегти";
        btnSave.Click += BtnSave_Click;
        this.Controls.Add(btnSave);

        btnCancel = new Button();
        btnCancel.Location = new Point(240, 150);
        btnCancel.Size = new Size(80, 30);
        btnCancel.Text = "Скасувати";
        btnCancel.Click += (s, e) => this.DialogResult = DialogResult.Cancel;
        this.Controls.Add(btnCancel);
    }
    private void LoadData()
    {
        string query = $"SELECT * FROM goods_shop WHERE ID_goods = {goodsId}";
        DataTable dt = DBHelper.ExecuteQuery(query);

        if (dt.Rows.Count > 0)
        {
            DataRow row = dt.Rows[0];
            txtName.Text = row["name_goods_shop"].ToString();
            numAmount.Value = Convert.ToDecimal(row["amount"]);
            numPrice.Value = Convert.ToDecimal(row["price"]);
            numDiscount.Value = Convert.ToDecimal(row["discount"]);
        }
    }

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

```
{
    if (string.IsNullOrWhiteSpace(txtName.Text))
    {
        MessageBox.Show("Будь ласка, введіть назву товару");
        return;
    }

    try
    {
        string query;
        if (goodsId > 0)
        {
            query = $@"UPDATE goods_shop SET
            name_goods_shop = '{txtName.Text}',
            amount = {numAmount.Value},
            price = {numPrice.Value},
            discount = {numDiscount.Value}
            WHERE ID_goods = {goodsId}";
        }
        else
        {
            query = $@"INSERT INTO goods_shop
            (name_goods_shop, amount, price, discount)
            VALUES ('{txtName.Text}', {numAmount.Value},
            {numPrice.Value}, {numDiscount.Value})";
        }

        if (DBHelper.ExecuteNonQuery(query))
```

```
            {
                MessageBox.Show("Дані збережено успішно");
                this.DialogResult = DialogResult.OK;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при збереженні: {ex.Message}");
        }
    }
  }
}using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
  public partial class SupplierEditForm : Form
  {
    private int supplierId = -1;

    private TextBox txtName;
    private TextBox txtPhone;
    private TextBox txtAddress;
    private TextBox txtNotes;
    private Button btnSave;
    private Button btnCancel;
```

```
    public SupplierEditForm(int id = -1)
    {
        supplierId = id;
        InitializeComponent();
        InitializeLayout();
        if (supplierId > 0) LoadData();
        this.Text = supplierId > 0 ? "Редагувати постачальника" : "Додати постачальника";
        this.Size = new Size(450, 300);
    }

    private void InitializeLayout()
    {
        // Назва
        Label lblName = new Label();
        lblName.Text = "Назва:";
        lblName.Location = new Point(20, 20);
        this.Controls.Add(lblName);

        txtName = new TextBox();
        txtName.Location = new Point(150, 20);
        txtName.Size = new Size(250, 20);
        this.Controls.Add(txtName);

        // Телефон
        Label lblPhone = new Label();
        lblPhone.Text = "Телефон:";
```

```
lblPhone.Location = new Point(20, 50);
this.Controls.Add(lblPhone);

txtPhone = new TextBox();
txtPhone.Location = new Point(150, 50);
txtPhone.Size = new Size(250, 20);
this.Controls.Add(txtPhone);

// Адреса
Label lblAddress = new Label();
lblAddress.Text = "Адреса:";
lblAddress.Location = new Point(20, 80);
this.Controls.Add(lblAddress);

txtAddress = new TextBox();
txtAddress.Location = new Point(150, 80);
txtAddress.Size = new Size(250, 20);
this.Controls.Add(txtAddress);

// Примітки
Label lblNotes = new Label();
lblNotes.Text = "Примітки:";
lblNotes.Location = new Point(20, 110);
this.Controls.Add(lblNotes);

txtNotes = new TextBox();
txtNotes.Location = new Point(150, 110);
txtNotes.Size = new Size(250, 60);
```

```
        txtNotes.Multiline = true;
        this.Controls.Add(txtNotes);

        // Кнопки
        btnSave = new Button();
        btnSave.Location = new Point(150, 180);
        btnSave.Size = new Size(100, 30);
        btnSave.Text = "Зберегти";
        btnSave.Click += BtnSave_Click;
        this.Controls.Add(btnSave);

        btnCancel = new Button();
        btnCancel.Location = new Point(260, 180);
        btnCancel.Size = new Size(100, 30);
        btnCancel.Text = "Скасувати";
        btnCancel.Click += (s, e) => this.DialogResult = DialogResult.Cancel;
        this.Controls.Add(btnCancel);
    }
    private void LoadData()
    {
        string query = $"SELECT * FROM supplier_fuel WHERE ID_supplier =
{supplierId}";
        DataTable dt = DBHelper.ExecuteQuery(query);

        if (dt.Rows.Count > 0)
        {
            DataRow row = dt.Rows[0];
            txtName.Text = row["name_supplier"].ToString();
```

```
        txtPhone.Text = row["phone"].ToString();
        txtAddress.Text = row["adress"].ToString();
        txtNotes.Text = row["notes"].ToString();
    }
}

private void BtnSave_Click(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(txtName.Text))
    {
        MessageBox.Show("Будь ласка, введіть назву постачальника");
        return;
    }

    try
    {
        string query;
        if (supplierId > 0)
        {
            query = $@"
            UPDATE supplier_fuel SET
                name_supplier = '{txtName.Text.Replace("'", "''")}',
                phone = '{txtPhone.Text}',
                adress = '{txtAddress.Text.Replace("'", "''")}',
                notes = '{txtNotes.Text.Replace("'", "''")}'
            WHERE ID_supplier = {supplierId}";
        }
        else
```

```
            {
                query = $@"
                INSERT INTO supplier_fuel
                    (name_supplier, phone, adress, notes)
                VALUES
                    ('{txtName.Text.Replace("'", "''")}', '{txtPhone.Text}',
                     '{txtAddress.Text.Replace("'",       "''")}',       '{txtNotes.Text.Replace("'",
"''")}')";
            }

            bool success = DBHelper.ExecuteNonQuery(query);
            if (success)
            {
                MessageBox.Show("Дані постачальника збережено успішно");
                this.DialogResult = DialogResult.OK;
            }
            else
            {
                MessageBox.Show("Помилка при збереженні даних постачальника");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при збереженні: {ex.Message}");
        }
    }
}using System;
```

```
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace FuelStation.AdditionalModules
{
    public partial class SuppliersForm : Form
    {
        private DataGridView dataGridView;
        private Button btnAdd;
        private Button btnEdit;
        private Button btnDelete;
        private TextBox txtSearch;
        private Button btnSearch;

        public SuppliersForm()
        {
            Text = "Управління постачальниками";
            InitializeComponent();
            InitializeLayout();

            // Автоматичне створення постачальників при першому відкритті
            if (DBHelper.ExecuteScalar("SELECT COUNT(*) FROM supplier_fuel") == 0)
            {
                CreateDefaultSuppliers();
            }

            LoadData();
```

```
    }
    private void CreateDefaultSuppliers()
    {
        string[] queries = {
        "INSERT INTO supplier_fuel (name_supplier, phone, adress, notes) VALUES ('Постачальник бензину', '+380991234561', 'Київ, вул. Бензинова 1', 'Основний постачальник бензину')",
        "INSERT INTO supplier_fuel (name_supplier, phone, adress, notes) VALUES ('Постачальник газу', '+380991234562', 'Київ, вул. Газова 5', 'Постачальник газу')",
        "INSERT INTO supplier_fuel (name_supplier, phone, adress, notes) VALUES ('Постачальник дизпалива', '+380991234563', 'Київ, вул. Дизельна 10', 'Постачальник дизельного палива')"
    };

        foreach (string query in queries)
        {
            DBHelper.ExecuteNonQuery(query);
        }
    }

    private void InitializeLayout()
    {
        this.Size = new Size(1000, 600);

        // Пошук
        txtSearch = new TextBox();
        txtSearch.Location = new Point(150, 20);
        txtSearch.Size = new Size(300, 20);
```

```
this.Controls.Add(txtSearch);

btnSearch = new Button();
btnSearch.Location = new Point(460, 20);
btnSearch.Size = new Size(100, 23);
btnSearch.Text = "Пошук";
btnSearch.Click += BtnSearch_Click;
this.Controls.Add(btnSearch);

// Кнопки керування
btnAdd = new Button();
btnAdd.Location = new Point(20, 60);
btnAdd.Size = new Size(100, 30);
btnAdd.Text = "Додати";
btnAdd.Click += BtnAdd_Click;
this.Controls.Add(btnAdd);

btnEdit = new Button();
btnEdit.Location = new Point(130, 60);
btnEdit.Size = new Size(100, 30);
btnEdit.Text = "Редагувати";
btnEdit.Click += BtnEdit_Click;
this.Controls.Add(btnEdit);

btnDelete = new Button();
btnDelete.Location = new Point(240, 60);
btnDelete.Size = new Size(100, 30);
btnDelete.Text = "Видалити";
```

```
        btnDelete.Click += BtnDelete_Click;
        this.Controls.Add(btnDelete);

        // Таблиця даних
        dataGridView = new DataGridView();
        dataGridView.Location = new Point(20, 100);
        dataGridView.Size = new Size(950, 450);
        dataGridView.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
AnchorStyles.Left | AnchorStyles.Right;
        dataGridView.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
        dataGridView.ReadOnly = true;
        dataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
        this.Controls.Add(dataGridView);

        dataGridView.AutoResizeColumnHeadersHeight();
        dataGridView.AutoResizeRows();
        dataGridView.AutoResizeColumns();
    }
    private void LoadData(string search = "")
    {
        string query = "SELECT ID_supplier, name_supplier AS 'Назва', phone AS
'Контакти', adress AS 'Адреса', notes AS 'Примітки' FROM supplier_fuel";

        if (!string.IsNullOrWhiteSpace(search))
        {
            query += $" WHERE name_supplier LIKE '%{search}%' OR phone LIKE
'%{search}%'";
```

```
    }

    DataTable dt = DBHelper.ExecuteQuery(query);
    dataGridView.DataSource = dt;
}

private void BtnSearch_Click(object sender, EventArgs e)
{
    LoadData(txtSearch.Text);
}

private void BtnAdd_Click(object sender, EventArgs e)
{
    using (SupplierEditForm editForm = new SupplierEditForm())
    {
        if (editForm.ShowDialog() == DialogResult.OK)
        {
            LoadData();
        }
    }
}

private void BtnEdit_Click(object sender, EventArgs e)
{
    if (dataGridView.SelectedRows.Count == 0)
    {
        MessageBox.Show("Будь ласка, оберіть постачальника для редагування");
        return;
```

```
        }

        int supplierId = Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_supplier"].Value);
        using (SupplierEditForm editForm = new SupplierEditForm(supplierId))
        {
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }
    }

    private void BtnDelete_Click(object sender, EventArgs e)
    {
        if (dataGridView.SelectedRows.Count == 0)
        {
            MessageBox.Show("Будь ласка, оберіть постачальника для видалення");
            return;
        }

        int supplierId = Convert.ToInt32(dataGridView.SelectedRows[0].Cells["ID_supplier"].Value);
        string name = dataGridView.SelectedRows[0].Cells["name_supplier"].Value.ToString();

        if (MessageBox.Show($"Ви дійсно бажаєте видалити постачальника '{name}'?",
```

```
            "Підтвердження видалення", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
        {
            string query = $"DELETE FROM supplier_fuel WHERE ID_supplier =
{supplierId}";
            if (DBHelper.ExecuteNonQuery(query))
            {
                MessageBox.Show("Постачальника успішно видалено");
                LoadData();
            }
            else
            {
                MessageBox.Show("Помилка при видаленні постачальника");
            }
        }
    }
}
```