РЕФЕРАТ

Кваліфікаційна робота на здобуття освітнього ступеню «бакалавр» за спеціальністю 121 – Інженерія програмного забезпечення. Тернопільський національний технічний університет ім. Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПс-43, 2023 рік. Пояснювальна записка до кваліфікаційної роботи на здобуття освітнього ступеню «бакалавр» містить: 79 с., 33 рис., 6 табл., 2 додатків.

Об'єкт дослідження – облік послуг споживачів Гусятинського РЕМ.

Мета роботи – проєктування бази даних, встановлення зв'язків між об'єктами предметної області та створення інформаційної системи для введення обліку послуг споживачів Гусятинського РЕМ.

Метод дослідження – опис предметної області шляхом опрацювання та аналізу документів; проєктування бази даних із використанням методів нормалізації; розробка і налагодження програми.

Отримані результати – розроблено базу даних та інформаційну систему обліку споживачів Гусятинського РЕМ..

Ключові слова: база даних, сутність, атрибут, зв'язок, скбд у visual studio 2019, нормалізація, споживачі, рем.

ANNOTATION

Qualification work for the degree of bachelor in the specialty 121 - Software Engineering. Ternopil Ivan Puluj National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group СПс-43, 2023. The explanatory note for the qualification work for the degree of bachelor contains: 79 pages, 33 figures, 6 tables, 2 appendices.

The object of the study is the accounting of the services of consumers of Husyatynsky REM.

The purpose of the work is to design a database, establish connections between objects of the subject area, and create an information system for entering the accounting of services of consumers of the Husyatinsky REM.

The research method is a description of the subject area by processing and analyzing documents; database design using normalization methods; program development and debugging.

The results obtained are the development of a database and an information system for accounting of consumers of Husyatynsky REM.

Keywords: database, entity, attribute, relationship, dbms in visual studio 2019, normalization, consumers, rem.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АІС – Автоматизована інформаційна система;

SQL – Structured query language.

1НФ – Перша нормальна форма;

2НФ – Друга нормальна форма;

3НФ – Третя нормальна форма;

ПЗ – Програмне забезпечення;

СКБД – Система керівання базами баних.

ВСТУП

Електроенергія відіграє важливу роль у повсякденному житті людей. Вона додає життя містам та селам, забезпечує рух транспорту, навіть традиційні автомобілі починають свій рух за допомоги електроенергії що запускає їх двигун. Також є повністю електричні автомобілі, тролейбуси, поїзди. Велику частину енергії використовує промисловість, електрична енергія потрібна для приведення у рух верстатів, які обробляють різні матеріали, розплавлення металу, зварювальних робіт. У побуті вона використовується для освітлення наших домівок, вулиць у містах та селах. Більшість медичних приладів використовують електроенергію для роботи. Увесь звичний для нас побут використовує електроенергію для роботи. Комп'ютери за якими ми працюємо та вчимося, електричні плити за якими ми готуємо, системи обігріву будинку. Весь цей потік енергії потрібно контролювати. Для цього використовують різні вимірювальні прилади. Більшість цих вимірювань збираються в ручну, і цей процес потребує поліпшень.

Було поставлено завдання створити програму та розробити базу даних для обліку електроенергії споживачів Гусятинського РЕМу. База даних повинна містити інформацію про споживачів, лічильники, трансформатори та кількість спожитої ними електроенергії.

Мета такого проєкту полягає в тому, щоб набути практичних навичок з аналізу об'єктів дослідження, проєктування бази даних, розробки та налагодження програми для організації роботи зі спроєктованою базою даних.

В якості системи управління базами даних вибрано СКБД у Visual Studio 2019, оскільки вона є провідною на ринку, а також має зручний інтерфейс.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Опис об’єкта автоматизації

Постійний розвиток інформаційного забезпечення є необхідним для успішного вирішення різноманітних завдань за допомогою комп'ютера оскільки комп'ютерні інформаційні технології нерозривно пов'язані з прогресом інформаційних систем що використовуються в економіці для автоматизованого вирішення завдань

Автоматизована інформаційна система [1] (АІС) є сукупністю виконавчих об'єктів та автоматично запрограмованих пристроїв, в якій функцію управління виконує людина. АІС є технічною системою, що за результатами завдання виконує процес автоматизації у різних галузях або їх поєднаннях.

АІС відіграє важливу роль у програмі, забезпечуючи ефективну обробку та управління інформацією. Вона забезпечує своєчасне та точне надання інформації органам управління, допомагаючи приймати обґрунтовані рішення. АІС збирає, аналізує, зберігає і візуалізує дані, пов'язані з економічним об'єктом, його станами та взаємодією. Вона допомагає забезпечити високу ефективність управлінської діяльності та сприяє оптимальному функціонуванню всієї системи. Завдяки АІС, програма має можливість реагувати на зміни, прогнозувати результати та забезпечувати високу якість прийнятих рішень.

Функціональна частина АІС [2] є ключовою, оскільки вона пов'язана з проблемними галузями та фактично втілює модель управління конкретним об'єктом. Вона включає елементи, що визначають її основний функціонал, такий як призначення, виконавчі функції та функції опрацювання інформації. Основними складовими функціональної частини АІС є підсистеми, блоки або комплекси задач, а також окремі задачі. Функціональна підсистема є відносно самостійною частиною системи, яка визначається спільною частиною головних ознак управління.

Інформаційне забезпечення є сукупністю методів та засобів для створення єдиної системи даних, організацією та узагальненням показників і документів, розробкою засобів формального опису даних. Воно є головним елементом АІС, яке надає задачам конкретний зміст. Успішність та ефективність виконавчих рішень частково залежить від якісної розробки програмного забезпечення.

1.2 Аналіз існуючих програмних систем

Спочатку необхідно чітко визначити вимоги до АІС, яку необхідно розробити. Це включає функціональність, масштабованість, безпеку, швидкодію, підтримку технологій та інші важливі аспекти. Вивчити ринок подібних програмних систем. Переглянути документацію, відгуки користувачів, порівняти функціональність та можливості кожної системи. Для аналізу було вибрано систему «АСТОР».

«АСТОР» – автоматизована система технічного обслуговування та ремонтів. Завдяки АСТОРу в автоматичному режимі формують потрібну технічну, звітну та аналітичну документацію по структурах і товариству загалом. «АСТОР» оптимізує роботу працівників, обліковує трудові затрати служб, дозволяє відкинути суб’єктивний фактор в оцінюванні технічного стану енергооб’єктів товариства. [3] Програма суттєво пришвидшує і спрощую процес формування поопорних схем ліній електропередач, схем підстанції, кабельних ліній, радіорелейного зв’язку, кабельних ліній зв’язку, релейного захисту та автоматики.

Автоматизована система комунального обліку витрат води «Есіон» дозволяє автоматично знімати покази квартирних лічильників води і передавати їх в центр обробки (керуючу компанію, товариство власників житла) з подальшим узагальненням результатів і поданням їх у вигляді рахунку-квитанції на оплату комунальних послуг (рис.1.1).

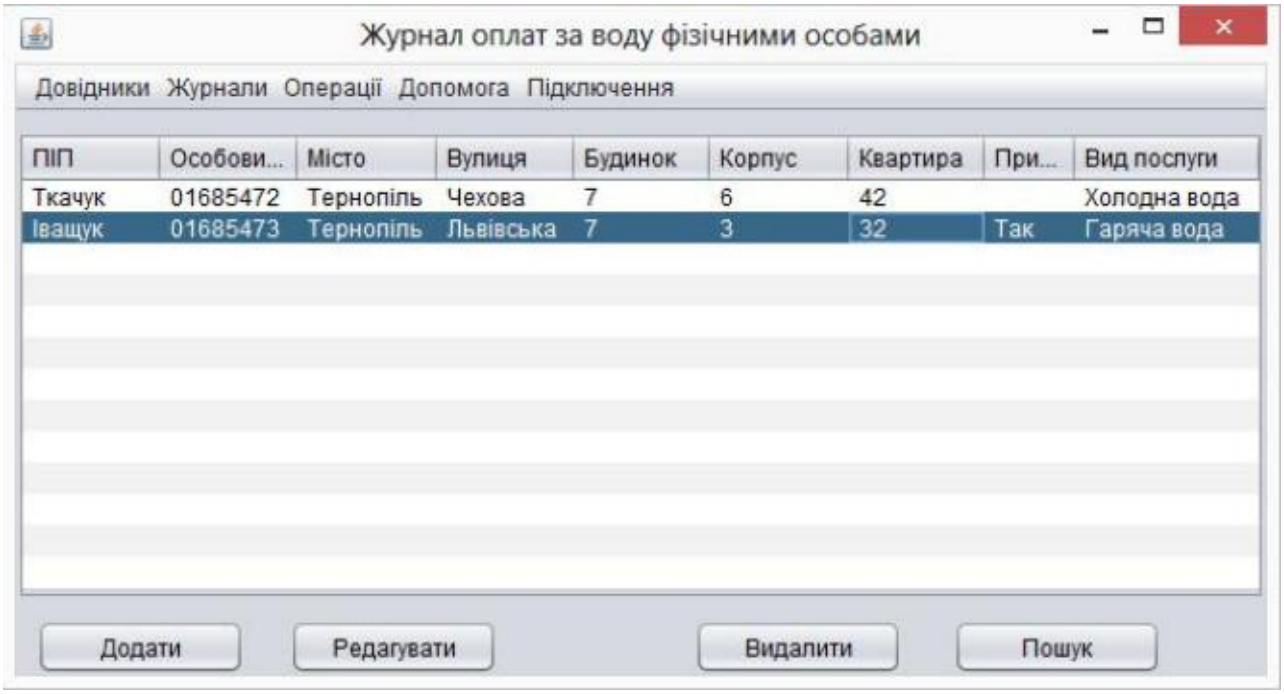


Рисунок 1.1 – Журнал оплат фізичних осіб

Залежно від налаштування системи (на вимогу оператора або через заданий період) прилад транслює поточні покази водолічильників в концентратор потоків даних. Останній їх конвертує і передає на автоматизоване робоче місце (АРМ) оператора. Передача даних від квартирного вузла обліку до концентратора є бездротовою. Для підвищення надійності системи передачі може використовуватися система ретрансляторів, що встановлюються в слабкострумових поверхових щитках.

Остаточна обробка інформації про споживаних ресурсах здійснюється засобами автоматизованого робочого місця оператора, включаючи підготовку рахунків-квитанцій на оплату комунальних послуг (рис.1.2 – 1.3).

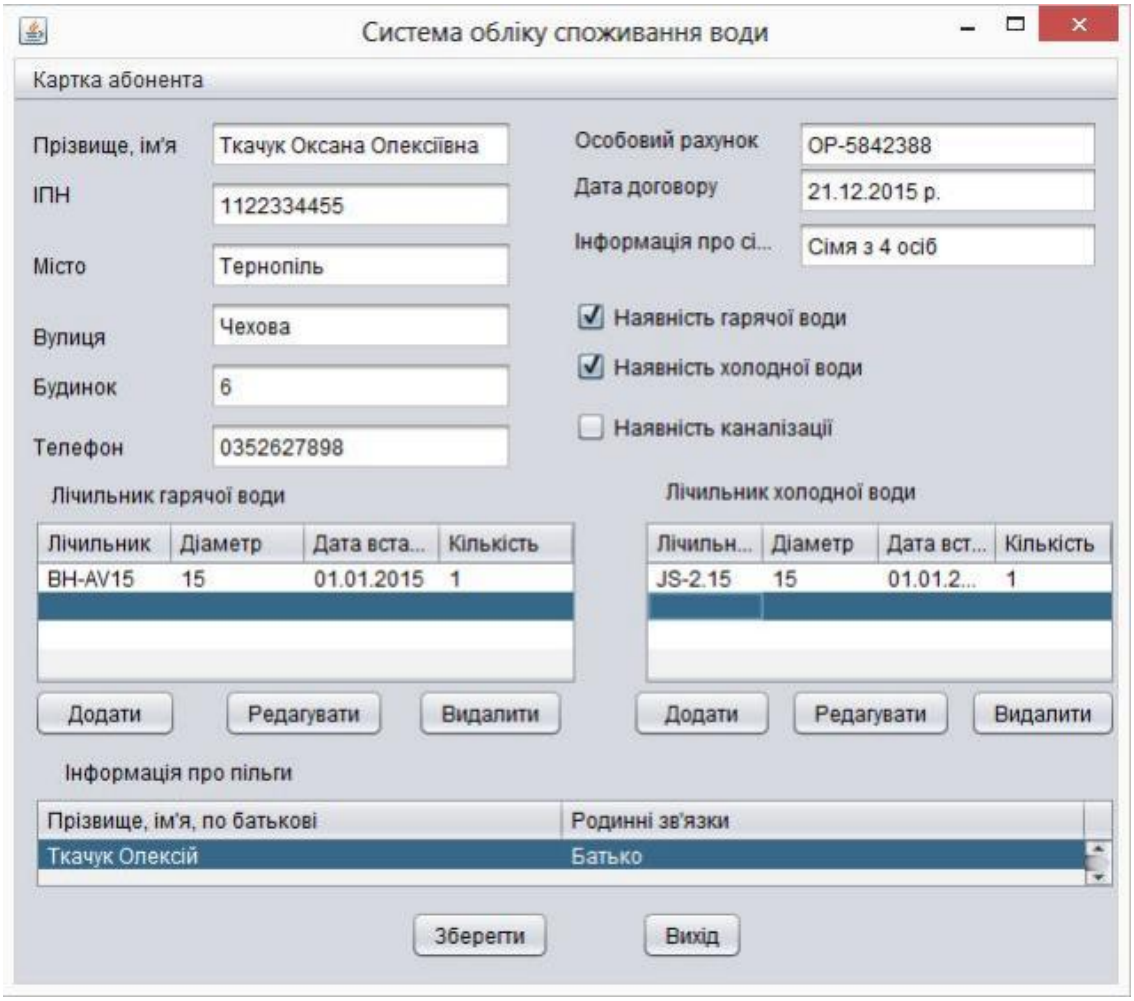


Рисунок 1.2 – Карта абонента

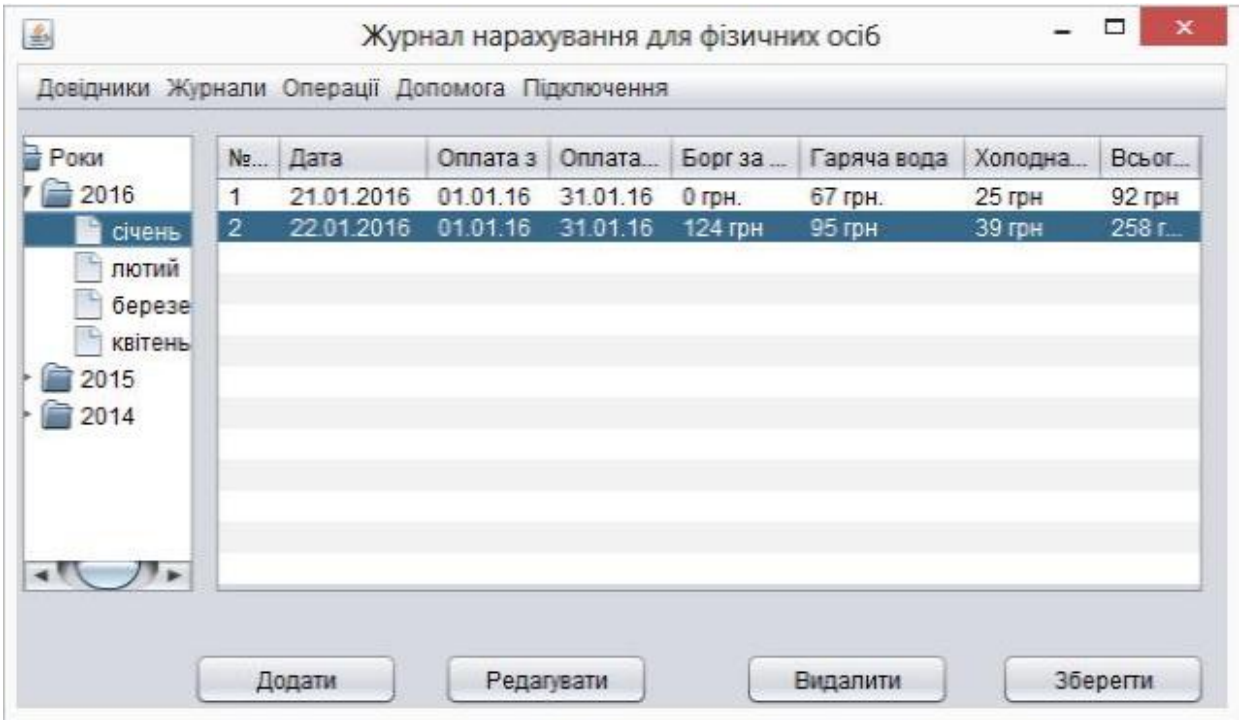


Рисунок 1.3 – Журнал нарахування для фізичних осіб

2 ПРОЄКТНА ЧАСТИНА

2.1 Постановка завдання

Реалізація завдання дипломної роботи є розробка системи автоматизації обліку послуг Гусятинського РЕМ що дозволить автоматизувати облік надання послуг. Цілі автоматизації – позбавити працівників паперової роботи, централізація даних, зменшення шансу на внесення помилкових даних, зменшення витрати часу на виконання пошуку, запису тощо.

Програма повинна підтримувати наступні функції:

- внесення даних про спожиту електроенергію;
- облік споживачів;
- видача завдань та документів про заміну або ремонт лічильника;
- видача завдань та документів про підключення підприємства чи будинку до мережі;
- перевірка балансу складу обладнання;
- облік техніки на балансі;
- облік працівників організації.

Згідно з поставленти завданням необхідно створити реляційні таблиці бази даних “DB” (Спроєктувати структури бази даних, спроєктувати структури таблиць та зв’язки між ними та забезпечити підтримання цілісності даних).

Після створення бази даних потрібно розробити алгоритм та структуру прикладної програми [4].

Виконати наступне:

- організувати облік споживачів;
- організувати облік працівників;
- розробити засіб доступу до бази даних через авторизацію;
- розробити засіб для видачі завдань на ремонт чи заміну лічильників і формування відповідних документів;

- розробити засіб для видачі завдань про підключення до мережі і формування відповідних документів;
- розробити засіб для обліку матеріалів на складі;
- розробити засіб для обліку наявної техніки.

Особливу увагу необхідно надати розробці інтерфейсу користувача. Тут необхідно враховувати сучасні стандарти (на сьогодні це «дружній» інтерфейс).

Після розробки програми необхідно протестувати програму на достатньому обсязі даних, який по можливості, враховує всі особливості цих даних та їх співвідношення. Після успішного тестування програми можна переходити до наступних етапів.

2.2 Проектування бази даних

РЕМ збирає та опрацьовує інформацію про споживачів. Якщо це фізична особа, то важлива інформація про прізвище, ім’я, по батькові споживача, його адресу проживання, особовий рахунок та будинки з лічильниками, які на нього зареєстровані. При роботі з юридичними особами важлива назва, рахунок, адреса офісу, зареєстровані на них підприємства.

Також РЕМ містить інформацію про персонал: прізвище, ім’я, по батькові працівника, адресу, займану ним посаду і до якої робочої групи він належить. І робочі групи: назва групи, працівники, які належать до неї, та виконані нею завдання. Ще зберігається інформація про трансформатори, а саме: назва, серійний номер, адреса та споживачі, які приєднані до нього.

Зберігається інформація про матеріали на складі, а саме: назва, одиниці виміру, його кількість. Інформація про техніку: назва, тип, номер. Інформація про завдання, така як: група, що його виконує, об’єкт завдання, тип завдання, виділена техніка, дата видачі, статус виконання.

Також наявна інформація про будинки та підприємства: адреса, лічильник, що встановлений, до якого трансформатора підключено, показники та власник.

Після аналізу процесу обліку послуг у Гусятинському РЕМ, визначив основні інформаційні об’єкти бази даних, їх властивості та обмеження, що накладаються на сутності. Задокументував опис сутностей бази даних DB у таблиці 2.1.

Таблиця 2.1 – Опис сутностей бази даних

Ім’я сутності	Опис	Псевдоніми	Особливості використання
Users	Зберігає логін та пароль для входу	Users	Кожен логін повинен бути унікальний
Transformers	Використовується для опису трансформаторів	Transformers	До кожного трансформатора може бути прив’язано декілька будинків
Physical_persons	Використовується для опису фізичних осіб споживачів електроенергії	Physical_persons	До кожного споживача може бути прив’язано декілька будинків
Juridical_persons	Використовується для опису юридичних осіб споживачів електроенергії	Juridical_persons	До кожного споживача може бути прив’язано декілька підприємств
Houses	Використовується для опису будинків	Houses	До кожного будинку може бути прив’язано лише один лічильник
Enterprises	Використовується для опису підприємств	Enterprises	До кожного підприємства може бути прив’язано лише один лічильник
HCounters	Використовується для опису лічильників будинків	HCounters	Може бути прив’язаний лише до одного будинку
ECounters	Використовується для опису лічильників підприємств	ECounters	Може бути прив’язаний лише до одного підприємства
EConsume	Зберігає показники використаної електроенерії підприємств	EConsume	Може бути прив’язано безліч до одного підприємства
HConsume	Зберігає показники використаної електроенерії будинків	HConsume	Може бути прив’язано безліч до одного будинку
Equipment	Використовується для опису техніки	Equipment	Може бути прив’язано до безлічі завдань
Materials	Використовується для опису матеріалів на складі	Materials	Немає
HTasks	Використовується для опису завдання для	HTasks	Може бути прив’язано безліч до одного будинку

Ім’я сутності	Опис	Псевдоніми	Особливості використання
	будинку		

Продовження таблиці 2.1

Ім’я сутності	Опис	Псевдоніми	Особливості використання
ETasks	Використовується для опису завдання для підприємства	ETasks	Може бути прив’язано безліч до одного підприємства
Employees	Використовується для опису працівників	Employees	Немає
Work_groups	Використовується для опис робочих груп	Work_groups	Може бути прив’язано безліч працівників до однієї групи
Towns	Використовується для опису міст	Towns	Немає
Streets	Використовується для опису вулиць	Streets	Може бути прив’язано безліч до міста

Опишемо набір атрибутів та їх характеристики для кожної сутності (див. Табл. 2.2).

Таблиця 2.2 – Опис атрибутів сутностей бази даних DB

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
Users	Id	Унікальний ідентифікатор користувача ПЗ	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Login	Логін користувача	Символьний тип	20 символів	-	-	Так
	Password	Пароль користувача	Символьний тип	20 символів	-	-	Так
Transformers	IdTransformer	Унікальний ідентифікатор трансформатора	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Model	Назва трансформатора	Символьний тип	20 символів	-	-	Так
	Power	Потужність трансформатора	Числовий тип, цілий	-	-	-	Так
	Voltage	Напруга трансформатора	Числовий тип, цілий	-	-	-	Так

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
	Adress	Адреса трансформатора	Числовий тип, цілий	-	-	-	Так

Продовження таблиці 2.2

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
Physical_persons	Id	Унікальний ідентифікатор споживача	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Name	Ім’я споживача	Символьний тип	20 символів	-	-	Так
	PB	По-батькові	Символьний тип	20 символів	-	-	Так
	Email	Email споживача	Символьний тип	50 символів	-	-	Так
	Phone	Телефон споживача	Символьний тип	10 символів	-	-	Так
	Invoice	Рахунок	Символьний тип	10 символів	-	-	Так
	House_number	Номер будинку	Числовий тип, цілий	-	-	-	Так
Juridical_persons	Id	Унікальний ідентифікатор споживача	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Name	Назва	Символьний тип	20 символів	-	-	Так
	Email	Email	Символьний тип	20 символів	-	-	Так
	Invoice	Рахунок	Символьний тип	20 символів	-	-	Так
	Phone	Номер телефону	Символьний тип	20 символів	-	-	Так
	Office_adress	Адреса офісу	Числовий тип, цілий	-	-	-	Так
Houses	Id	Унікальний ідентифікатор будинку	Числовий тип, цілий	Первинний ключ	-	-	Ні

	Owner	Власник будинку	Числовий тип, цілий	-	-	-	Так
	Adress	Адреса будинку	Числовий тип, цілий	-	-	-	Так
	Transformer	Трансформатор, до якого підключено	Числовий тип, цілий	-	-	-	Так

Продовження таблиці 2.2

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
	Number	Номер будинку	Числовий тип, цілий	-	-	-	Так
Enterprises	Id	Унікальний ідентифікатор будинку	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Owner	Власник підприємства	Числовий тип, цілий	-	-	-	Так
	Adress	Адреса підприємства	Числовий тип, цілий	-	-	-	Так
	Transformer	Трансформатор, до якого підключено	Числовий тип, цілий	-	-	-	Так
	Name	Назва підприємства	Символьний тип	20 символів	-	-	Так
HCounters	Id	Унікальний ідентифікатор лічильника	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Model	Назва моделі	Символьний тип	20 символів	-	-	Так
	House	Будинок де встановлено	Числовий тип, цілий	-	-	-	Так
	Install_date	Дата встановлення	Дата	-	-	-	Так
	About	Опис	Символьний тип	50 символів	-	-	Так
	Phases	Фази	Числовий тип, цілий	-	-	-	Так
ECounters	Id	Унікальний ідентифікатор лічильника	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Model	Назва моделі	Символьний тип	20 символів	-	-	Так
	House	Будинок де встановлено	Числовий тип, цілий	-	-	-	Так

	Install_date	Дата встановлення	Дата	-	-	-	Так
	About	Опис	Символьний тип	50 символів	-	-	Так
	Phases	Фази	Числовий тип, цілий	-	-	-	Так
HConsume	Id	Унікальний ідентифікатор показника споживання будинку	Числовий тип, цілий	Первинний ключ	-	-	Ні

Продовження таблиці 2.2

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
	House	Будинок	Числовий тип, цілий	-	-	-	Так
	Date	Дата	Дата	-	-	-	Так
	Day	Денне споживання	Числовий тип, цілий	-	-	-	Так
	Night	Нічне споживання	Числовий тип, цілий	-	-	-	Так
Consume	Id	Унікальний ідентифікатор показника споживання підприємства	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Enterprise	Підприємство	Числовий тип, цілий	-	-	-	Так
	Date	Дата	Дата	-	-	-	Так
	Night	Нічне споживання	Числовий тип, цілий	-	-	-	Так
Equipment	Id	Унікальний ідентифікатор техніки	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Name	Назва техніки	Символьний тип	20 символів	-	-	Так
	Type	Тип техніки	Символьний тип	20 символів	-	-	Так
	Nubmers	Номери	Символьний тип	8 символів	-	-	Так
Materails	Id	Унікальний ідентифікатор матеріалу	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Name	Назва матеріалу	Символьний тип	20 символів	-	-	Так
	Unit	Одиниці виміру	Символьний тип	20 символів	-	-	Так

	Count	Кількість матеріалу	Числовий тип, цілий	-	-	-	Так
HTasks	Id	Унікальний ідентифікатор завдання для будинку	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Work_group	Робоча група	Числовий тип, цілий	-	-	-	Так
	About	Опис	Символьний тип	100 символів	-	-	Так

Продовження таблиці 2.2

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
	Used_equipment	Використана техніка	Числовий тип, цілий	-	-	-	Так
	Used_spec_equipment	Використана спец техніка	Числовий тип, цілий	-	-	-	Так
	Target	Будинок	Числовий тип, цілий	-	-	-	Так
	Type	Тип завдання	Символьний тип	20 символів	-	-	Так
	Progress	Прогрес виконання	Числовий тип, цілий	-	-	-	Так
	Issue_date	Дата видачі	Дата	-	-	-	Так
	Complete_date	Дата виконання	Дата	-	-	-	Так
ETasks	Id	Унікальний ідентифікатор завдання для підприємства	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Work_group	Робоча група	Числовий тип, цілий	-	-	-	Так
	About	Опис	Символьний тип	100 символів	-	-	Так
	Used_equipment	Використана техніка	Числовий тип, цілий	-	-	-	Так
	Used_spec_equipment	Використана спец техніка	Числовий тип, цілий	-	-	-	Так
	Type	Тип завдання	Символьний тип	20 символів	-	-	Так
	Progress	Прогрес виконання	Числовий тип, цілий	-	-	-	Так
	Issue_date	Дата видачі	Дата	-	-	-	Так
	Complete_date	Дата виконання	Дата	-	-	-	Так
Employe	Id	Унікальний	Числовий	Первинний	-	-	Ні

es		ідентифікатор працівника	тип, цілий	ключ			
	Name	Ім’я працівника	Символьний тип	20 символів	-	-	Так
	Surname	Прізвище працівника	Символьний тип	20 символів	-	-	Так
	PB	По-батькові	Символьний тип	20 символів	-	-	Так
	Adress	Адреса	Числовий тип, цілий	-	-	-	Так
	Position	Посада	Символьний тип	20 символів	-	-	Так

Продовження таблиці 2.2

Сутність	Атрибут	Опис	Домен	Обмеження	Значення за замовчуванням	Псевдонім	Значення NULL
	Work_group	Робоча група	Числовий тип, цілий	-	-	-	Так
Work_groups	Id	Унікальний ідентифікатор групи	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Name	Назва групи	Символьний тип	10 символів	-	-	Так
Towns	Id	Унікальний ідентифікатор міста	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Name	Назва міста	Символьний тип	20 символів	-	-	Так
Street	Id	Унікальний ідентифікатор вулиці	Числовий тип, цілий	Первинний ключ	-	-	Ні
	Name	Назва міста	Символьний тип	20 символів	-	-	Так
	Town	Місто	Числовий тип, цілий	-	-	-	Так

На наступному етапі визначаємо типи зв’язків, що існують між сутностями. За вимогами користувачів зв’язки виражаються дієсловами. Також потрібно визначити кардинальність кожного зв’язку.

ER-діаграма використовується для представлення основних сутностей та типів зв’язків. Ще вона дозволяє графічно подати всі елементи інформаційної моделі за допомогою умовних позначок.

Опис зв’язків між сутностями наведено в таблиці 2.3, а ER-діаграму зв’язків – у додатку А.

Таблиця 2.3 – Опис зв’язків сутностей бази даних

Ім’я сутності 1	Ім’я зв’язку	Ім’я сутності 2	Тип зв’язку (кардинальність)
Houses	Зареєстровано на	Physical_persons	N:1
Physical_persons	Проживає на	Streets	N:1
Houses	Розташований на	Streets	N:1
Enterprises	Зареєстровано на	Juridical_persons	N:1

Продовження таблиці 2.3

Ім’я сутності 1	Ім’я зв’язку	Ім’я сутності 2	Тип зв’язку (кардинальність)
Juridical_persons	Офіс розташовано у	Streets	N:1
Enterprises	Розташовано на	Streets	N:1
Steets	Є у	Towns	N:1
HCounter	Встановлений у	Houses	1:1
ECounter	Встановлений у	Enterprises	1:1
HConsume	Спожито	Houses	N:1
EConsume	Спожито	Enterprises	N:1
ETasks	Виконати щодо	Enterprises	N:1
HTasks	Виконати щодо	Houses	N:1
Employees	Проживає на	Streets	N:1
Employees	Входить до	Work_groups	N:1
ETasks	Виконує	Work_groups	N:1
HTasks	Виконує	Work_groups	N:1
Equipment	Використано у	ETasks	1:1
Equipment	Використано у	HTasks	1:1
Houses	Підключено до	Transformers	N:1
Enterprises	Підключено до	Transformers	N:1
Transformers	Розташовано на	Streets	N:1

Для нормалізації спроєктованих відношень бази даних DB використаємо три нормальних форми [5].

Перша нормальна форма вимагає, щоб кожен атрибут відношення був неподільним і не містив груп даних, які повторюються. Розглянемо спроєктовані відношення Users, Transformers, Physical_persons, Juridical_persons, Houses, Enterprises, HCounters, ECounters, EConsume, HConsume, Equipment, Materials, HTasks, ETasks, Employees, Work_groups, Towns, Streets бази даних DB. Відношення не містять подільних атрибутів та груп даних, які повторюються, тому будуть нормалізованими до першої нормальної форми.

Друга нормальна форма передбачає перевірку існування функціональної залежності між описовими атрибутами та первинним ключем. Проаналізуємо кожне відношення бази даних DB та вияснимо, чи існує функціональна залежність або повна функціональна залежність у випадку складеного ключа між неключовими атрибутами та ключем. Кожне відношення містить простий первинний ключ, тому будемо розглядати функціональну залежність. Кожне з відношень бази даних DB містять простий первинний ключ. Описові атрибути цих відношень функціонально залежать від свого первинного ключа. Оскільки ці відношення відповідають першій нормальній формі та існує функціональна залежність між описовими атрибутами та первинним ключем, то вони будуть нормалізованими до другої нормальної форми.

Третя нормальна форма передбачає виконання вимог першої та другої нормальних форм та усунення транзитивних функціональних залежностей між атрибутами відношення. Проаналізуємо усі відношення спроєктованої бази даних DB. Транзитивних функціональних залежностей між атрибутами виділених відношень не існує. Тому усі відношення нормалізовані до третьої нормальної форми.

Отже, було проаналізоване кожне відношення бази даних, виключена надлишковість даних та проведена нормалізація до третьої нормальної форми. Розглянувши поставлену задачу, створимо базу даних, що буде містити такі таблиці: Users, Transformers, Physical_persons, Juridical_persons, Houses, Enterprises, HCounters, ECounters, EConsume, HConsume, Equipment, Materials, HTasks, ETasks, Employees, Work_groups, Towns, Streets [6].

Створимо таблицю Users за допомогою наступного SQL-запиту (див. Лістинг 2.1).Таблиця використовується для реєстрації та авторизації користувачів програми. Містить поля: Login, Password.

Лістинг 2.1 – Текст SQL-запиту для створення таблиці users бази даних

```
CREATE TABLE [dbo].[Users] (
    [Id]       INT        IDENTITY (1, 1) NOT NULL,
    [Login]    NCHAR (20) NULL,
    [Password] NCHAR (20) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
```

Створимо таблицю Transformers за допомогою наступного SQL-запиту (див. Лістинг 2.2). Таблиця зберігає дані про розташування трансформаторів на вулицях. Містить інформацію про: Model, Adress, Power, Voltage.

Лістинг 2.2 – Текст SQL-запиту для створення таблиці Transformers бази даних.

```
CREATE TABLE [dbo].[Transformers] (
    [Id]      INT        IDENTITY (1, 1) NOT NULL,
    [Model]   NCHAR (20) NULL,
    [Adress]  INT        NULL,
    [Power]   INT        NULL,
    [Voltage] INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Adress]) REFERENCES [dbo].[Streets] ([Id])
);
```

Створимо таблицю Physical_persons за допомогою наступного SQL-запиту (див. Лістинг 2.3). Ця таблиця призначена для збереження, перегляду та редагування інформації про споживачів електроенергії. Містить інформацію про: Name, Surname, PB, Email, Phone, Invoice, Adress, House_number.

Лістинг 2.3 – Текст SQL-запиту для створення таблиці Physical_persons бази даних.

```
CREATE TABLE [dbo].[Physical_persons] (
    [Id]           INT        IDENTITY (1, 1) NOT NULL,
    [Name]         NCHAR (20) NULL,
    [Surname]      NCHAR (20) NULL,
    [PB]           NCHAR (20) NULL,
    [Email]        NCHAR (50) NULL,
    [Phone]        NCHAR (10) NULL,
    [Invoice]      NCHAR (10) NULL,
    [Adress]       INT        NULL,
    [House_number] INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Adress]) REFERENCES [dbo].[Streets] ([Id]));
```

Створимо таблицю Juridical_persons за допомогою наступного SQL-запиту (див. Лістинг 2.4). Таблиця зберігає інформацію про компанії споживачі електроенергії та їх рахунки. Містить інформацію про: Name, Email, Phone, Invoice, Office_adress.

Лістинг 2.4 – Текст SQL-запиту для створення таблиці Juridical_persons бази даних.

```
CREATE TABLE [dbo].[Juridical_persons] (
    [Id]            INT        IDENTITY (1, 1) NOT NULL,
    [Name]          NCHAR (20) NULL,
    [Email]         NCHAR (50) NULL,
    [Phone]         NCHAR (10) NULL,
    [Invoice]       NCHAR (10) NULL,
    [Office_adress] INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Office_adress]) REFERENCES [dbo].[Streets] ([Id]));
```

Створимо таблицю Houses за допомогою наступного SQL-запиту (див. Лістинг 2.5). В таблиці зберігатимуться детальніша інформація про будинки споживачів електроенергії. Містить інформацію про: Adress, Owner, Transformer, Number.

Лістинг 2.5 – Текст SQL-запиту для створення таблиці Houses бази даних.

```
CREATE TABLE [dbo].[Houses] (
    [Id]          INT IDENTITY (1, 1) NOT NULL,
    [Adress]      INT NULL,
    [Owner]       INT NULL,
    [Transformer] INT NULL,
    [Number]      INT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Adress]) REFERENCES [dbo].[Streets] ([Id]),
    FOREIGN KEY ([Owner]) REFERENCES [dbo].[Physical_persons] ([Id]),
    FOREIGN KEY ([Transformer]) REFERENCES [dbo].[Transformers] ([Id]));
```

Створимо таблицю Enterprises за допомогою наступного SQL-запиту (див. Лістинг 2.6). Таблиця призначення для збереження розширеної інформації про компанії. Містить поля: Adress, Owner, Transformer, Name.

Лістинг 2.6 – Текст SQL-запиту для створення таблиці Enterprises бази даних.

```
CREATE TABLE [dbo].[Enterprises] (
    [Id]          INT        IDENTITY (1, 1) NOT NULL,
    [Adress]      INT        NULL,
    [Owner]       INT        NULL,
    [Transformer] INT        NULL,
    [Name]        NCHAR (20) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Adress]) REFERENCES [dbo].[Streets] ([Id]),
    FOREIGN KEY ([Owner]) REFERENCES [dbo].[Juridical_persons] ([Id]),
    FOREIGN KEY ([Transformer]) REFERENCES [dbo].[Transformers] ([Id]));
```

Створимо таблицю HCounters за допомогою наступного SQL-запиту (див. Лістинг 2.7). Таблиця містить дані про лічильники, будинок в якому вони встановлені та дату встановлення. Містить поля: Model, House, Install_date, About, Phases.

Лістинг 2.7 – Текст SQL-запиту для створення таблиці HCounters бази даних.

```
CREATE TABLE [dbo].[HCounters] (
    [Id]           INT        IDENTITY (1, 1) NOT NULL,
    [Model]        NCHAR (20) NULL,
    [House]        INT        NULL,
    [Install_date] DATETIME   NULL,
    [About]        NCHAR (50) NULL,
    [Phases]       INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([House]) REFERENCES [dbo].[Houses] ([Id]));
```

Створимо таблицю ECounters за допомогою наступного SQL-запиту (див. Лістинг 2.8). Таблиця містить детальну інформацію про трансформатори. Містить поля: Model, Enterprise, Install_date, About, Phases.

Лістинг 2.8 – Текст SQL-запиту для створення таблиці ECounters бази даних.

```
CREATE TABLE [dbo].[ECounters] (
    [Id]           INT        IDENTITY (1, 1) NOT NULL,
    [Model]        NCHAR (20) NULL,
    [Enterprise]   INT        NULL,
    [Install_date] DATETIME   NULL,
    [About]        NCHAR (50) NULL,
    [Phases]       INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
```

Створимо таблицю HConsume за допомогою наступного SQL-запиту (див. Лістинг 2.9).Таблиця містить дані про спожиту електроенергію за денний та нічний період кожним будинком. Містить поля: House, Date, Day, Night.

Лістинг 2.9 – Текст SQL-запиту для створення таблиці HConsume бази даних.

```
CREATE TABLE [dbo].[HConsume] (
    [Id]    INT      IDENTITY (1, 1) NOT NULL,
    [House] INT      NULL,
    [Date]  DATETIME NULL,
    [Day]   INT      NULL,
    [Night] INT      NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([House]) REFERENCES [dbo].[Houses] ([Id]));
```

Створимо таблицю EConsume за допомогою наступного SQL-запиту (див. Лістинг 2.10).У таблиці зберігається інформація про спожиту електроенергію компаніями за денний та нічний час. Містить поля: Enterprise, Date, Day, Night.

Лістинг 2.10 – Текст SQL-запиту для створення таблиці EConsume бази даних.

```
CREATE TABLE [dbo].[EConsume] (
    [Id]         INT      IDENTITY (1, 1) NOT NULL,
    [Enterprise] INT      NULL,
    [Date]       DATETIME NULL,
    [Day]        INT      NULL,
    [Night]      INT      NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
```

Створимо таблицю Equipment за допомогою наступного SQL-запиту (див. Лістинг 2.11). Таблиця зберігає інформацію про інвентар працівників. Містить поля: Name, Type, Numbers.

Лістинг 2.11 – Текст SQL-запиту для створення таблиці Equipment бази даних.

```
CREATE TABLE [dbo].[Equipment] (
    [Id]      INT        IDENTITY (1, 1) NOT NULL,
    [Name]    NCHAR (20) NULL,
    [Type]    NCHAR (20) NULL,
    [Numbers] NCHAR (8)  NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC));
```

Створимо таблицю Materials за допомогою наступного SQL-запиту (див. Лістинг 2.12). У таблиці зберігається інформація про матеріали та їх кількість які зберігаються на складі. Містить поля: Name, Unit, Count.

Лістинг 2.12 – Текст SQL-запиту для створення таблиці Materials бази даних.

```
CREATE TABLE [dbo].[Materials] (
    [Id]    INT        IDENTITY (1, 1) NOT NULL,
    [Name]  NCHAR (20) NULL,
    [Unit]  NCHAR (20) NULL,
    [Count] INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC));
```

Створимо таблицю HTasks за допомогою наступного SQL-запиту (див. Лістинг 2.13). У таблиці зберігаються завдання та накази на виконання роботи по трансформаторах та лічильниках. Також є інформація про використаний інвентар із складу. Містить поля: Work_group, About, Used_equipment, Used_spec_equipment, Target, Type, Progress, Issue_date, Complete_date.

Лістинг 2.13 – Текст SQL-запиту для створення таблиці HTasks бази даних.

```
CREATE TABLE [dbo].[HTasks] (
    [Id]                 INT         IDENTITY (1, 1) NOT NULL,
    [Work_group]         INT         NULL,
    [About]              NCHAR (100) NULL,
    [Used_equipment]     INT         NULL,
    [Used_spec_equipment] INT        NULL,
    [Target]             INT         NULL,
    [Type]               NCHAR (20)  NULL,
    [Progress]           INT         NULL,
    [Issue_date]         DATE        NULL,
    [Complete_date]      DATE        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Work_group]) REFERENCES [dbo].[Work_groups] ([Id]),
    FOREIGN KEY ([Used_equipment]) REFERENCES [dbo].[Equipment] ([Id]),
    FOREIGN KEY ([Target]) REFERENCES [dbo].[Houses] ([Id]),
    FOREIGN KEY ([Used_spec_equipment]) REFERENCES [dbo].[Equipment] ([Id]));
```

Створимо таблицю ETasks за допомогою наступного SQL-запиту (див. Лістинг 2.14). Таблиця містить інформацію про завершені завдання. Містить поля: Work_group, About, Used_equipment, Used_spec_equipment, Target, Type, Progress, Issue_date, Complete_date.

Лістинг 2.14 – Текст SQL-запиту для створення таблиці ETasks бази даних.

```
CREATE TABLE [dbo].[ETasks] (
    [Id]                 INT         IDENTITY (1, 1) NOT NULL,
    [Work_group]         INT         NULL,
    [About]              NCHAR (100) NULL,
    [Used_equipment]     INT         NULL,
    [Used_spec_equipment] INT        NULL,
    [Target]             INT         NULL,
    [Type]               NCHAR (20)  NULL,
    [Progress]           INT         NULL,
    [Issue_date]         DATE        NULL,
    [Complete_date]      DATE        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Work_group]) REFERENCES [dbo].[Work_groups] ([Id]), ([Id]));
```

Створимо таблицю Employees за допомогою наступного SQL-запиту (див. Лістинг 2.15). У таблиці зберігається інформацію про працівників, посади та їх групи. Містить поля: Name Surname PB Adress Adress Position WorkGroup.

Лістинг 2.15 – Текст SQL-запиту для створення таблиці Employees бази даних.

```
CREATE TABLE [dbo].[Employees] (
    [Id]        INT        IDENTITY (1, 1) NOT NULL,
    [Name]      NCHAR (20) NULL,
    [Surname]   NCHAR (20) NULL,
    [PB]        NCHAR (20) NULL,
    [Adress]    INT        NULL,
    [Position]  NCHAR (20) NULL,
    [WorkGroup] INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([WorkGroup]) REFERENCES [dbo].[Work_groups] ([Id]),
    FOREIGN KEY ([Adress]) REFERENCES [dbo].[Streets] ([Id]));
```

Створимо таблицю Work_groups за допомогою наступного SQL-запиту (див. Лістинг 2.16).Таблиця містить інформацію про робочі групи. Містить поля: Name.

Лістинг 2.16 – Текст SQL-запиту для створення таблиці Work_groups бази даних.

```
CREATE TABLE [dbo].[Work_groups] (
    [Id]   INT        IDENTITY (1, 1) NOT NULL,
    [Name] NCHAR (10) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC));
```

Створимо таблицю Towns за допомогою наступного SQL-запиту (див. Лістинг 2.17). У таблиці зберігається інформація про населені пункти. Містить поля: Name.

Лістинг 2.17 – Текст SQL-запиту для створення таблиці Towns бази даних.

```
CREATE TABLE [dbo].[Towns] (
    [Id]   INT        IDENTITY (1, 1) NOT NULL,
    [Name] NCHAR (20) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC));
```

Створимо таблицю Streets за допомогою наступного SQL-запиту (див. Лістинг 2.18). У таблиця збережена інформація про вулиці та населені пункти в яких вони знаходяться. Містить поля: Name, Town.

Лістинг 2.18 – Текст SQL-запиту для створення таблиці Streets бази даних.

```
CREATE TABLE [dbo].[Streets] (
    [Id]   INT        IDENTITY (1, 1) NOT NULL,
    [Name] NCHAR (20) NULL,
    [Town] INT        NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    FOREIGN KEY ([Town]) REFERENCES [dbo].[Towns] ([Id]));
```

За допомогою SQL запитів ми створили таблиці для збереження даних та коректної роботи програми.

2.3 Проєктування інформаційної системи

Для проєктування потрібно побудувати три діаграми: діаграма прецедентів (use-case diagram), діаграма класів (class diagram) та діаграма послідовностей (sequence diagram).

Діаграма прецедентів відображає відношення між користувачами та системою. Основні елементи діаграми прецедентів є діючі особи, варіанти використання й відносини між ними. Зв’язки між ними вказують за допомогою стрілок. Головна мета діаграми прецедентів показати функціональність системи з точки зору користувача та їх взаємодії з системою. Вона допомагає зрозуміти основні функції системи, виявити проблемні ситуації та вимоги до системи. Діаграма прецедентів є важливим інструментом для забезпечення сприйняття вимог до програмного забезпечення [7].

Для створення діаграми прецедентів було визначено акторів: працівник та користувач. Працівник має повний доступ до всіх функцій ПЗ, та до БД. Користувач може переглянути певні дані. Діаграма прецедентів наведена на рисунку 2.1.

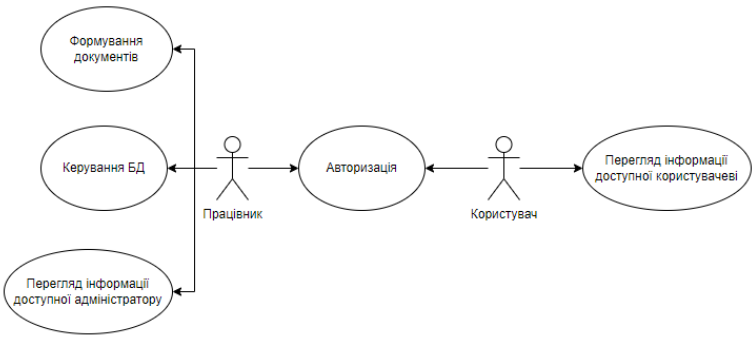


Рисунок 2.1 – Діаграма прецедентів

Діаграма класів – складається з прямокутних форм з назвами класів та методів які є в ньому а також зв’язків між ними [7]. На стадії проектування

діаграми класів застосовують, щоб передати структуру класів, які формують структуру системи. Кожен клас має ім'я, яке повинно бути унікальним.

Робота починається з головної сторінки. Головна сторінка взаємодіє із базою даних. Інформація, що відображається на вкладках підбирається із БД. Діаграма класів наведена на рисунку 2.2.

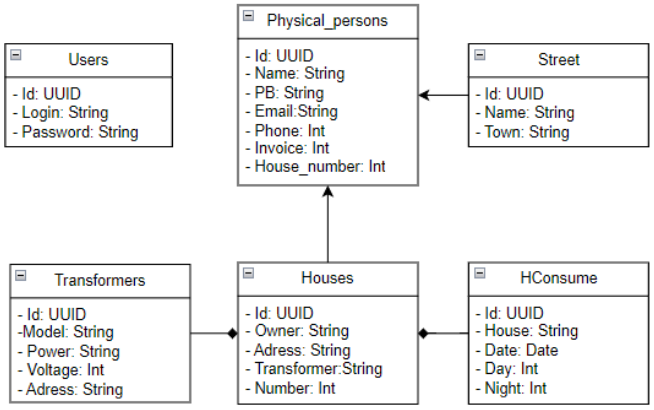


Рисунок 2.2 – Діаграма класів

Діаграма послідовностей відображає процес роботи працівника [7] . Він має спершу пройти авторизацію, перш ніж приступити до своєї роботи. Діаграма послідовності наведена на рисунку 2.3.

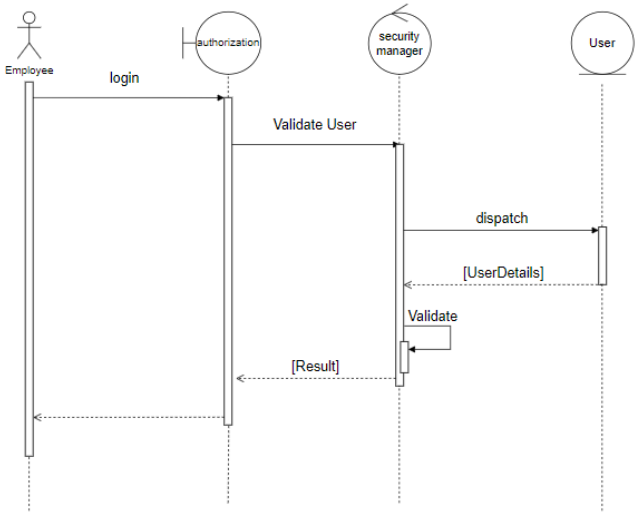


Рисунок 2.3 – Діаграма послідовностей

Отже, на етапі проєктування в онлайн редакторі Visual Paradigm з використанням уніфікованої мови моделювання побудовано діаграму прецедентів, діаграму класів та діаграму послідовності.

3 КОНСТРУЮВАННЯ ТА ТУСТУВАННЯ ПРОГРАМИ

3.1 Проектування інтерфейсу користувача

Графічний інтерфейс користувача - це спосіб взаємодії між користувачем і комп'ютерною системою, використовуючи графічні елементи, такі як вікна, кнопки, значки та інші візуальні елементи. Він дозволяє користувачеві взаємодіяти з програмами та операційною системою шляхом використання миші, клавіатури або сенсорного екрану [8].

Графічний інтерфейс користувача робить взаємодію з комп'ютером більш інтуїтивною та зручною. Він дає змогу користувачеві візуально сприймати та керувати інформацією шляхом взаємодії з графічними об'єктами.

Для роботи з АІС було розроблено інтуїтивно зрозумілий інтерфейс, з відображення усієї необхідної інформації у текстових рядках, таблицях та документах [9].

Після запуску програми, на екрані відображається вікно авторизації (див. рис. 3.1), де користувач може зайти під свій обліковий запис.

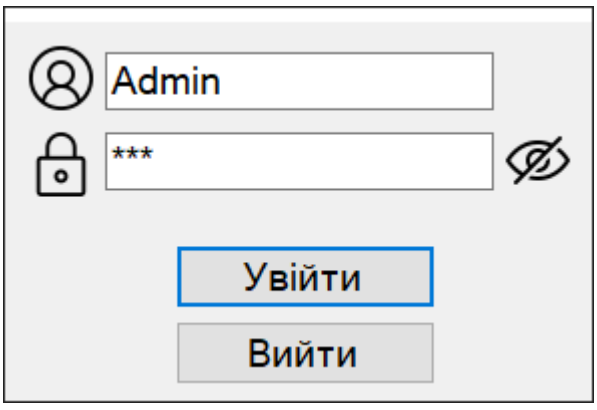


Рисунок 3.1 – Вікно авторизації

Після авторизації облікового запису здійснюється перехід на головне вікно (див. рис. 3.2). Головне вікно містить меню навігації та таблицю із вкладками, при виборі яких можна здійснювати дії, які необхідно користувачу.

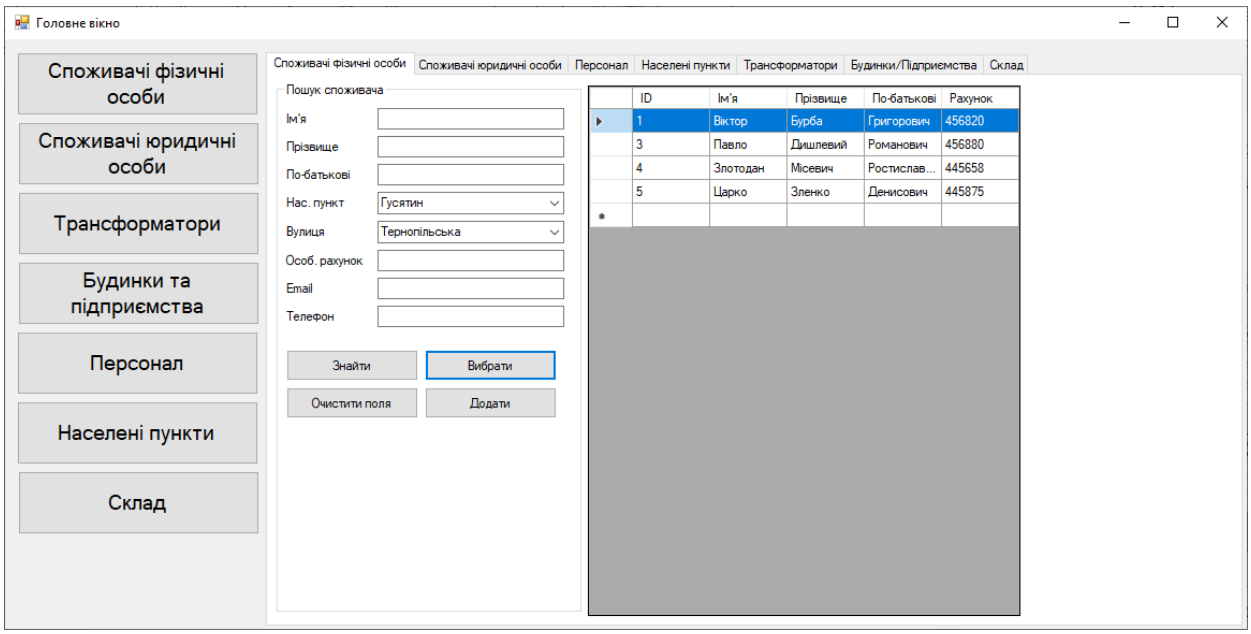


Рисунок 3.2 – Головне вікно

На головній формі відбувається пошук фізичних осіб споживачів, та їх вибір для роботи, при цьому відкривається вікно для роботи з ним(див. рис. 3.3).

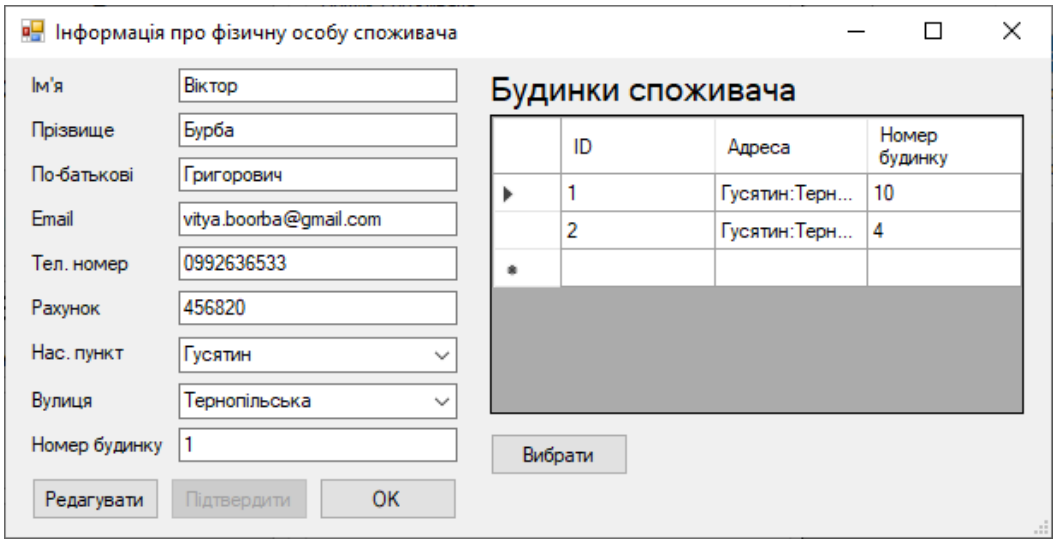


Рисунок 3.3 – Вікно для робити зі споживачем

У цьому вікні можна переглянути інформацію про споживача, редагувати її та дізнатися, які будинки зареєстровані на нього, та виконувати дії з цими будинками. При виборі будинку відкривається вікно для роботи з ним(див. рис. 3.4).

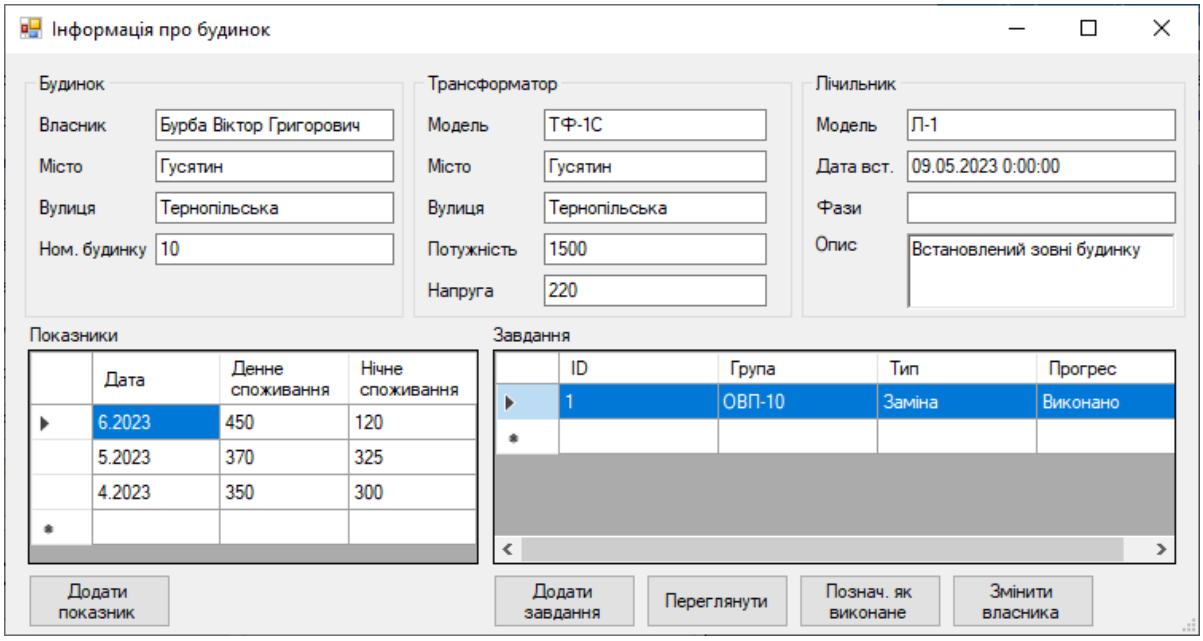


Рисунок 3.4 – Вікно для роботи з будинком

У цьому вікні можна переглянути інформацію про будинок, його власника, трансформатор, до якого підключено, встановлений лічильник, показники споживання електроенергії, та завдання для цього будинку. Можна додати показник(див. рис. 3.5) та завдання(див. рис. 3.6)

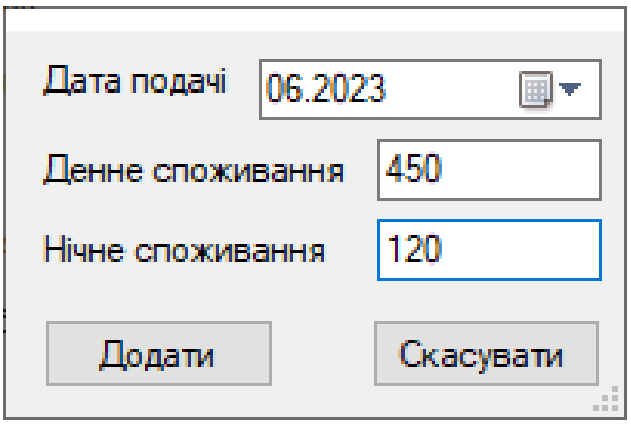


Рисунок 3.5 – Вікно для додання показника

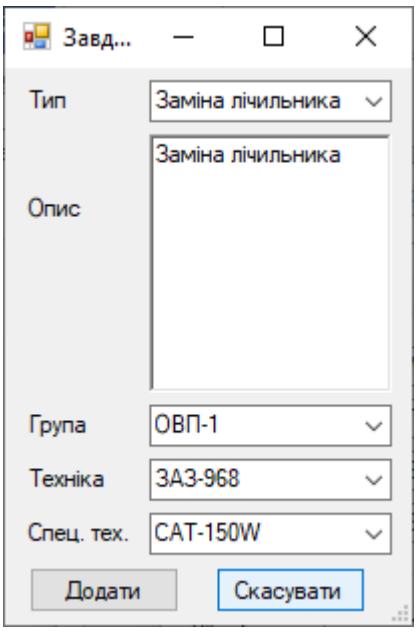


Рисунок 3.6 – Вікно для додавання завдання

Після додання завдання можна його переглянути (див. рис. 3.7).

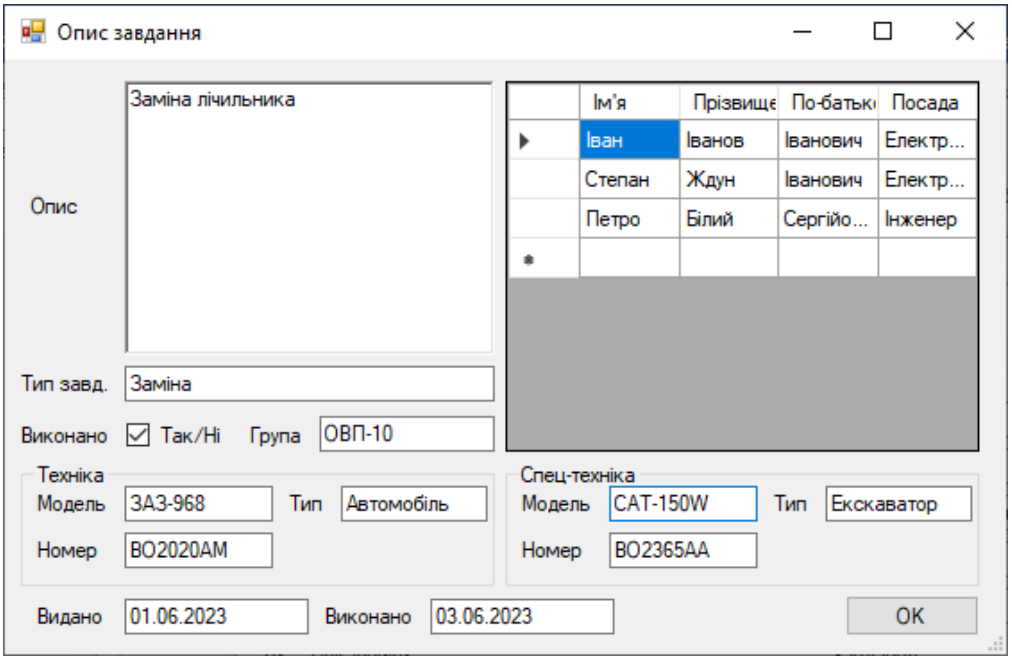


Рисунок 3.7 – Вікно опису завдання

При роботі з юридичними особами функціонал аналогічний.

На вкладці персонал можна виконувати дії з працівниками та робочими групами (див. рис. 3.8).

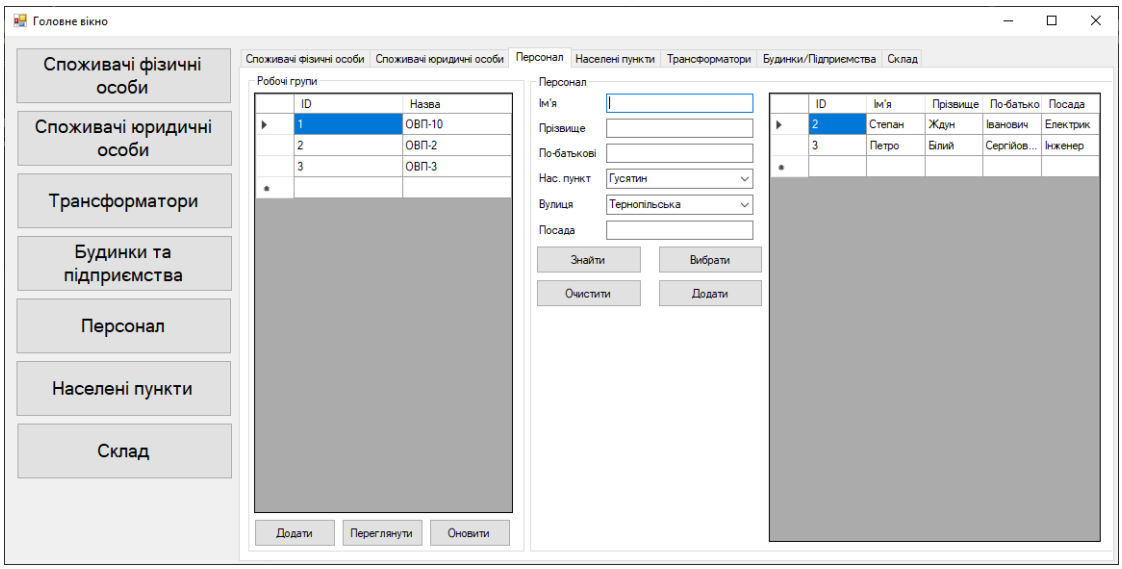


Рисунок 3.8 – Вікно персоналу

Тут можна переглянути склад робочої групи та виконані ними завдання (див. рис. 3.9). та додати нову (див. рис. 3.10).

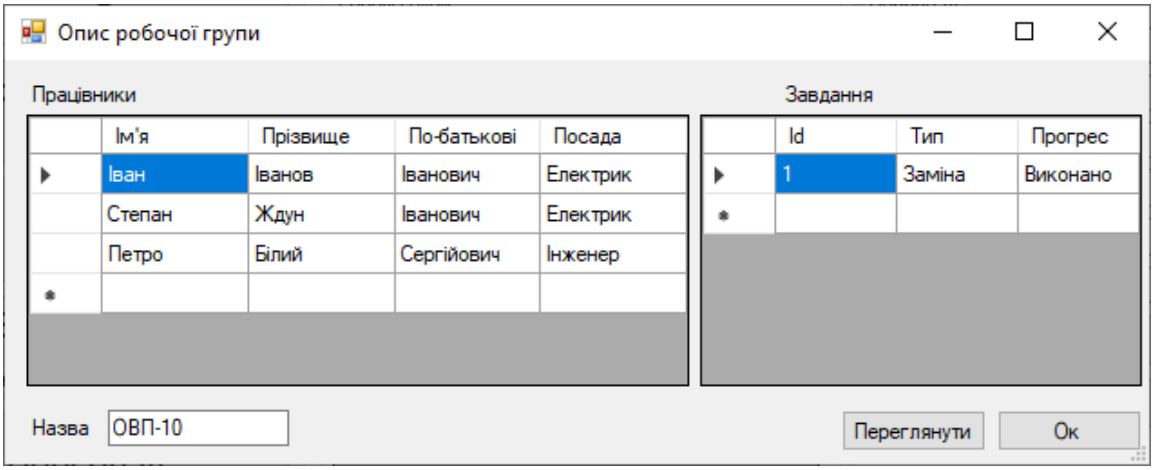


Рисунок 3.9 – Вікно опису робочої групи

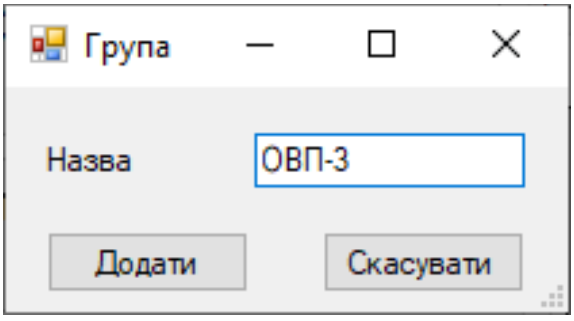


Рисунок 3.10 – Вікно додання робочої групи

При виборі працівника відкривається вікно, де можна переглянути інформацію про нього, та редагувати її (див. рис. 3.11).

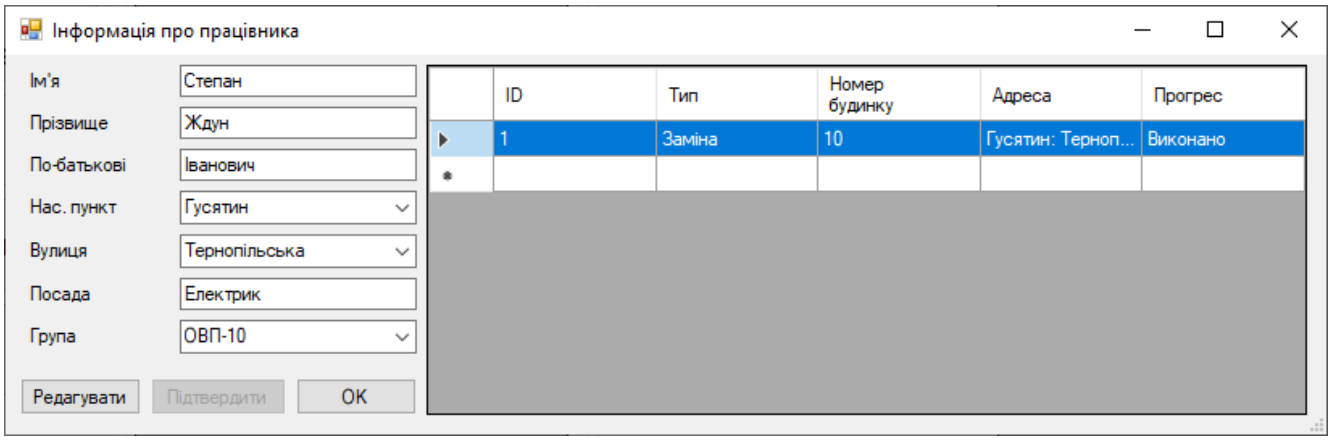


Рисунок 3.11 – Вікно працівника

На вкладці населені пункти можна переглянути, додати та редагувати міста і вулиці (див. рис. 3.12).

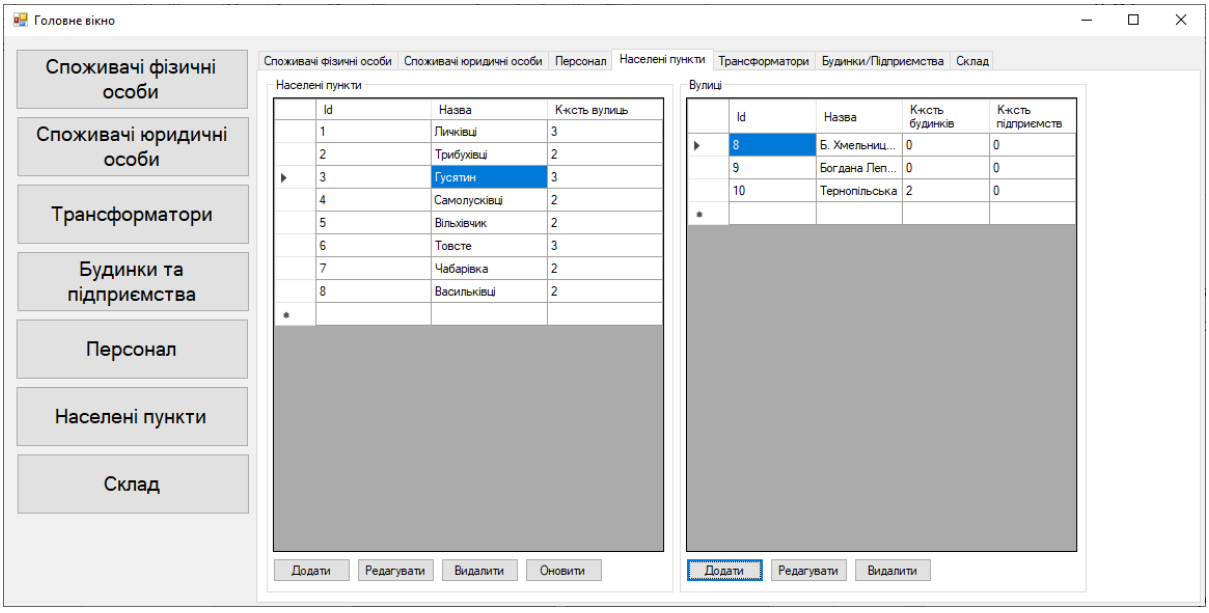


Рисунок 3.12 – Вікно населені пункти

На вкладці трансформатори можна переглянути інформацію про трансформатори (див. рис. 3.13).

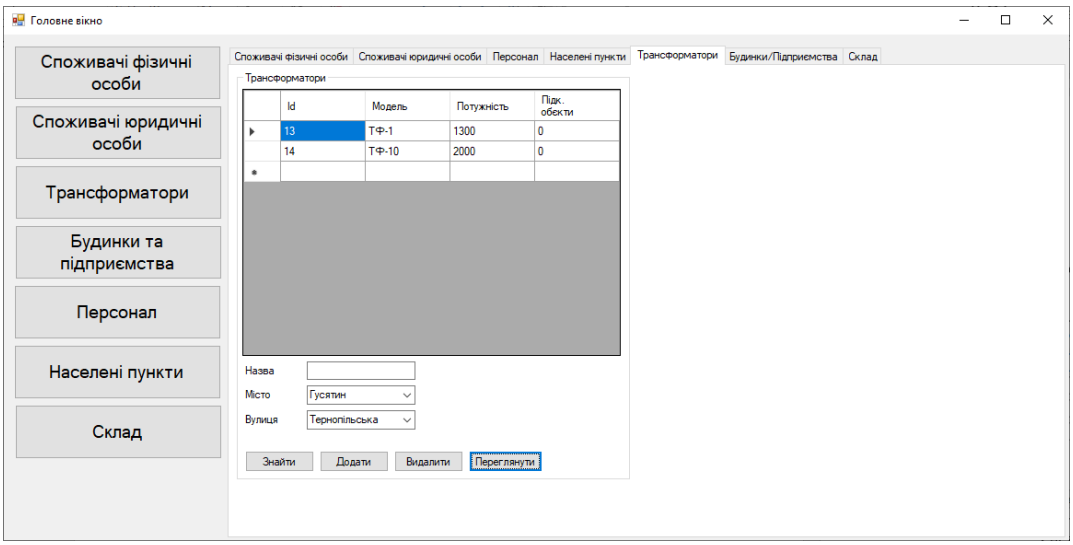


Рисунок 3.13 – Вікно трансформатори

Також тут можна додати новий трансформатор (див. рис. 3.14). та переглянути про нього інформацію (див. рис. 3.15).

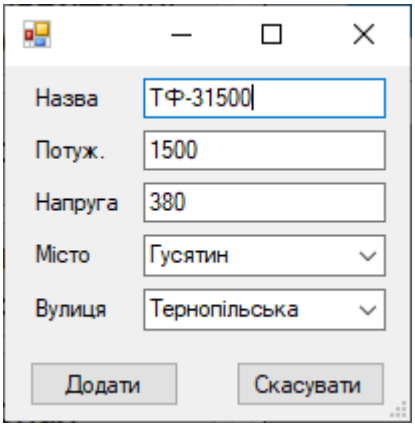


Рисунок 3.14 – Додання трансформатора

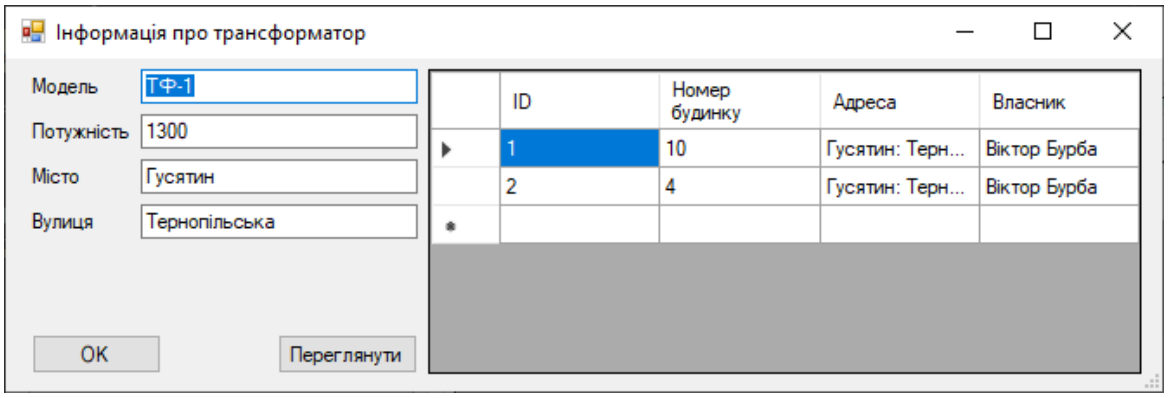


Рисунок 3.15 – Інформація про трансформатор

Наступна вкладка це будинки та підприємства (див. рис. 3.16)

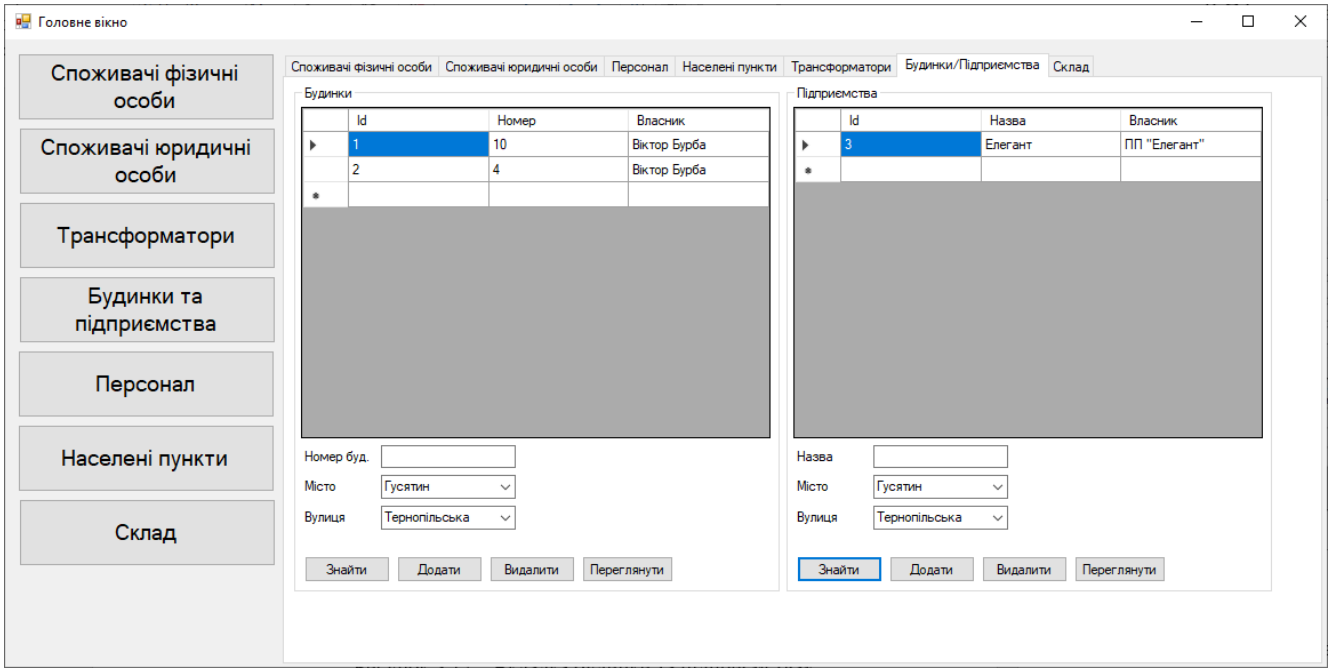


Рисунок 3.16 – Вкладка будинки та підприємства

Тут можна додати будинок (див. рис.3.17) та переглянути про нього інформацію (див. рис.3.18). У підприємств функціонал аналогічний.

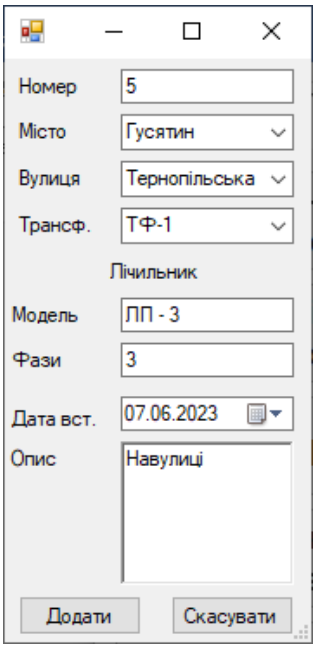


Рисунок 3.17 – Додання будинку

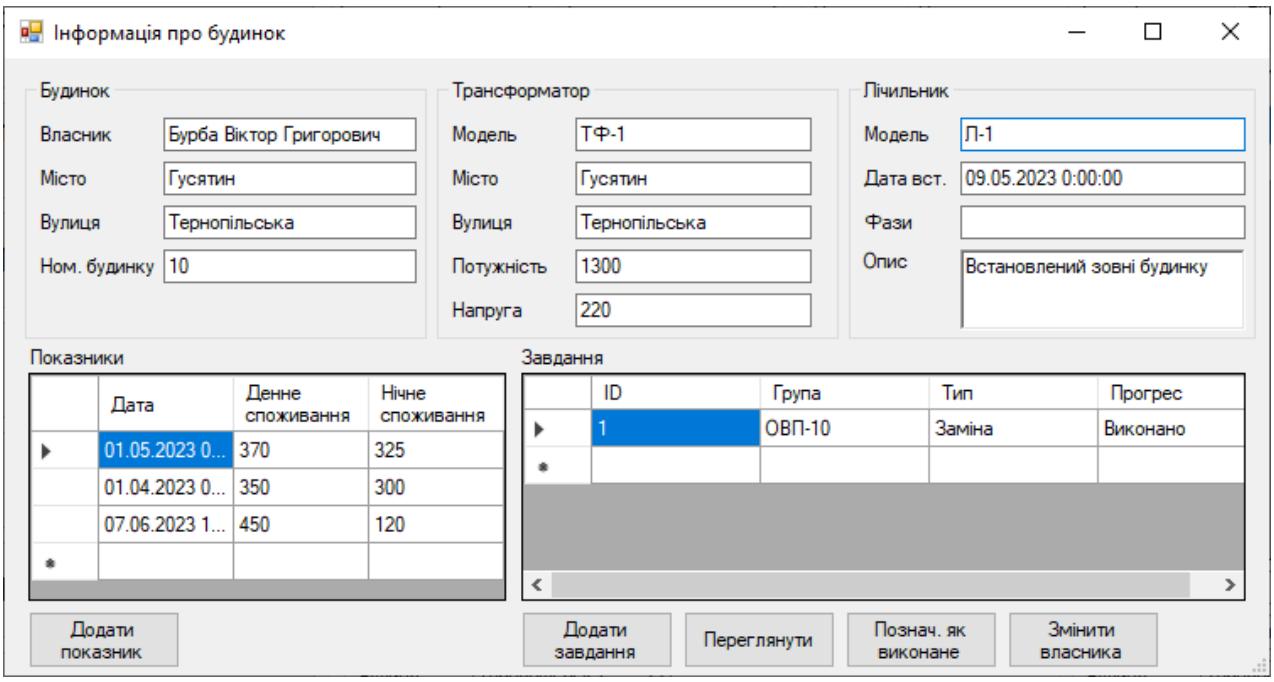


Рисунок 3.18 – Перегляд інформації

Наступна вкладка це склад. Тут можна переглянути матеріали на складі та техніку на балансі (див. рис. 3.19).

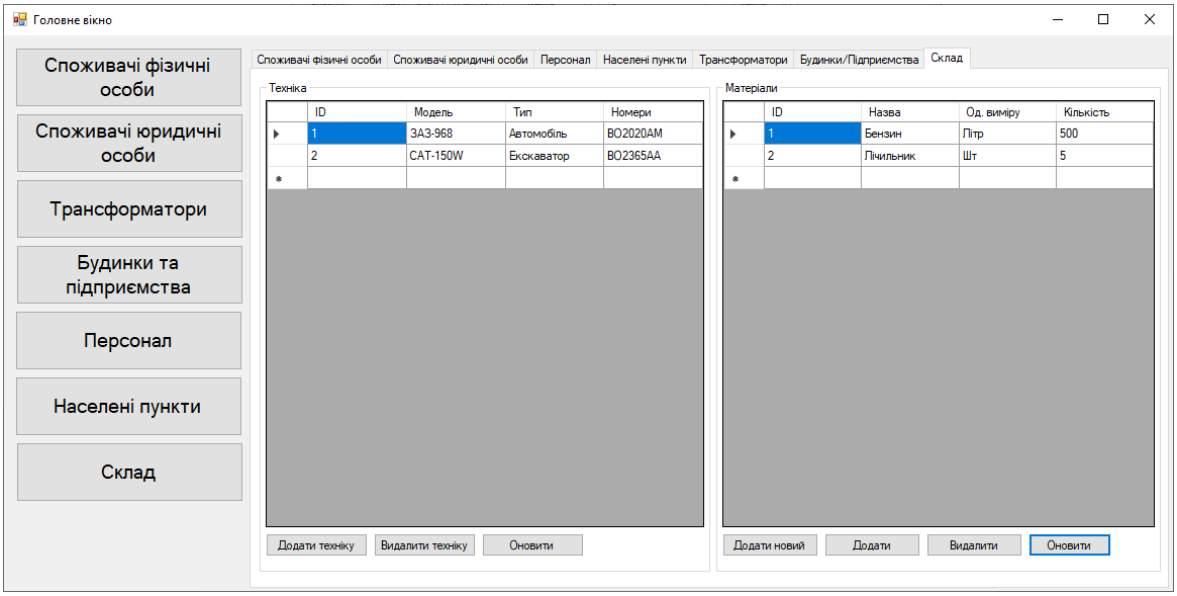


Рисунок 3.19 – Вікно складу

Також можна додати нову техніку (див. рис. 3.20) та матеріали (див. рис. 3.21) на склад.

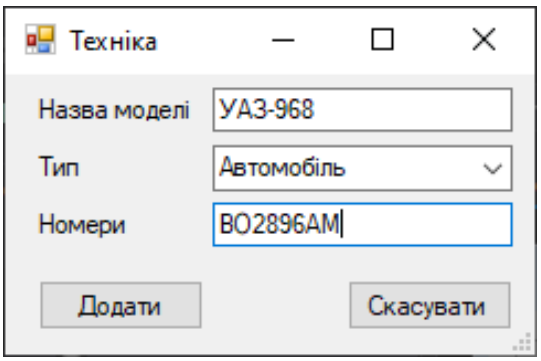


Рисунок 3.20 – Додання нової техніки

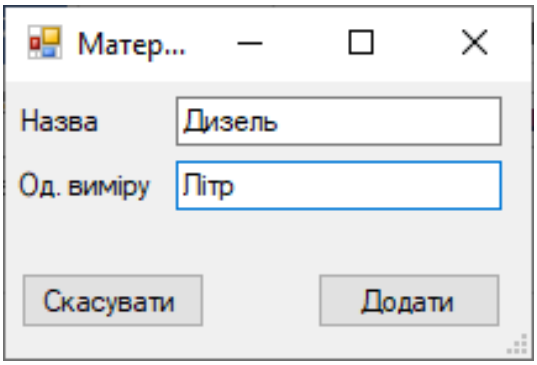


Рисунок 3.21 – Додання матеріалів

На рисунках зображені головний інтерфейс програми та показаний основний функціонал програми.

3.2 Опис програмних модулів

Програма для обліку споживання електроенергії створена з використанням середовища розробки Visual Studio 2019 та мови програмування C#.

У процесі розробки, було необхідно створити базу даних, яку було зручно використовувати в програмі. Для цього було використано вбудовану Систему Керування Базами Даних (СКБД) [10] в Visual Studio 2019. За допомогою неї, було налагоджено зв'язки між таблицями, використовуючи систему первинних - зовнішніх ключів, що дозволяє ефективно організувати та зберігати дані. Створена база даних під назвою "DB" стала основою для отримання та обробки

даних в програмі. Для цього був використаний елемент "Model", який дозволяє створювати об'єкти з бази даних. За допомогою цих об'єктів можна виконувати різноманітні операції, такі як видалення, зміна та додавання нових записів.

У проекті також було використано потужний набір технологій під назвою Entity Framework 6 [11]. Ця структура об'єктно-реляційного відображення (ORM) [12] з відкритим кодом надає розширені можливості для взаємодії з базою даних через ADO.NET. Вона стала невід'ємною частиною розробки програм на платформі .NET Framework.

Насамкінець, використання компонента Microsoft .NET Framework LINQ (Language Integrated Query) [13] додало ще більше гнучкості та зручності до роботи з даними. LINQ дозволяє здійснювати запити до різноманітних джерел даних, таких як масиви, XML документи та реляційні бази даних, за допомогою виразів, подібних до SQL. Це дозволяє виконувати різноманітні операції з даними з легкістю та ефективністю. Отже, цей проект об'єднує в собі потужність технологій Visual Studio 2019, вбудованої СУБД, Entity Framework та LINQ, щоб забезпечити ефективний облік споживання електроенергії зручним та сучасним способом.

Завдяки використанню .NET Framework, ця програма може працювати як звичайний десктопний застосунок, так і веб-додаток, що робить її універсальною та доступною для різних сценаріїв використання.

Для роботи програми використовуються наступні модулі, опис модулів програми наведено у таблиці 3.1.

Таблиця 3.1 – Опис модулів програми

№	Назва програмного модулю	Короткий опис
1	LoginForm	Використовується для авторизації в програму.
2	MainForm	Головна форма програми. На ній відображаються таблиці із даними
3	AddPPersonForm	Форма для додання фізичної особи споживача
4	AddJPersonForm	Форма для додання юридичної особи споживача
5	AddEmployeeForm	Форма для додання працівника
6	AddTask	Форма для додання завдання
7	AddTown	Форма для додання міста
8	AddStreet	Форма для додання вулиці
9	AddWorkGroup	Форма для додання робочої групи
10	AddTransformerForm	Форма для додання трансформатора
11	AddHouse	Форма для додання будинку
12	AddEquipment	Форма для додання техніки
13	AddNewMaterialForm	Форма для додання нового матеріалу на склад
14	AddMoreMaterialForm	Форма для додання певної кількості матеріалу на склад
15	AboutEmployeeForm	Форма для роботи з працівником
16	AboutTask	Форма опису завдання
17	AboutTransformerForm	Форма для роботи з трансформатором
18	HouseWorkingForm	Форма для роботи з будинком
19	EnterprisesWorkingForm	Форма для роботи з підприємством
20	ChangeConsumerForm	Форма для зміни власника будинку чи підприємства
21	ChangeCounter	Форма для зміни лічильника
22	ChangeTransformer	Форма для зміни трансформатора
23	AddEnterprises	Форма для додання підприємства

3.3 Опис результатів тестування

Перед введенням програми на експлуатацію, потрібно обов’язково провести тестування, щоб виправити помилки, які були допущені в процесі розробки.

Від якості тестування залежить зручність та ефективність використання програми. Під час тестування програма перевіряється на відповідність технічному завданню, перевіряється її функціонал [14]. Проведемо функціональне тестування При запуску програми відкривається вікно авторизації (див. рис. 3.23). Щоб увійти необхідно ввести дані користувача а саме логін та пароль. Необхідно переконатися чи виконується авторизація коректно.

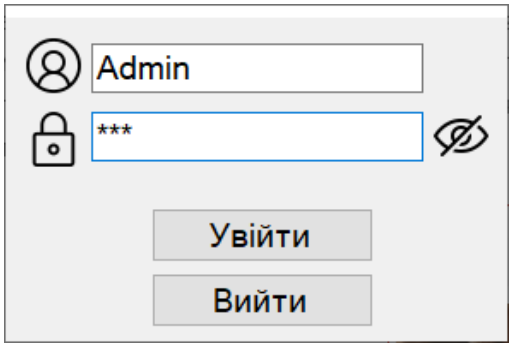


Рисунок 3.23 – Вікно авторизації

При не правильному логіні чи паролі з’являється наступне повідомлення(див. рис. 3.24).

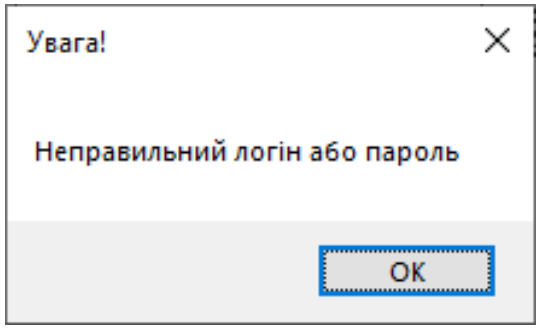


Рисунок 3.24 – Повідомлення про неправильний логін чи пароль

При правильному введенні відкриється головна форма(див. рис. 3.25).

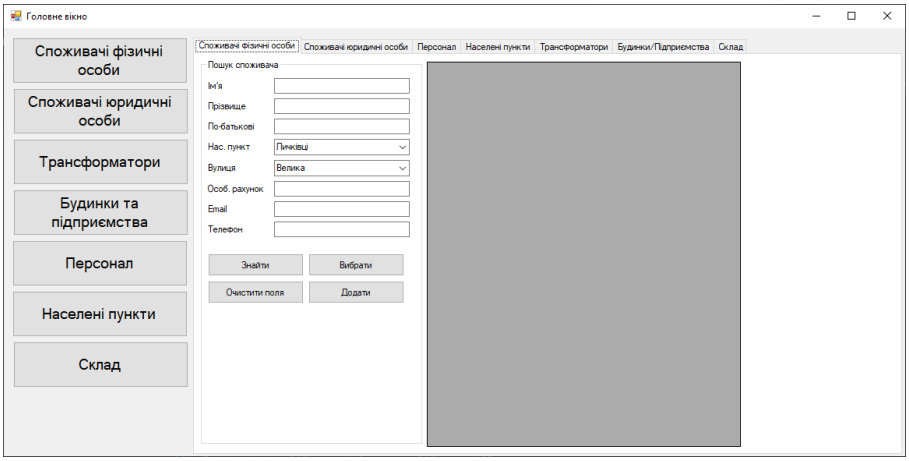


Рисунок 3.25 – Головна форма

Надалі необхідно переконатися чи працює додання споживачів. Необхідно натиснути кнопку “Додати” та з'явиться форма для додання споживача (див. рис. 3.26).

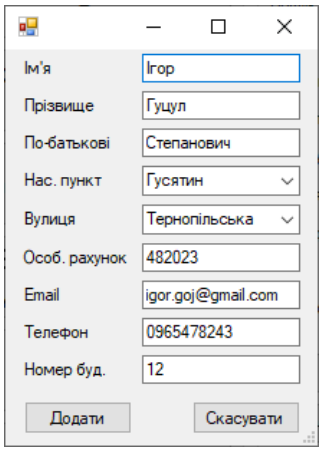


Рисунок 3.26 – форма для додання споживача

При успішному доданні виникне наступне повідомлення (див. рис. 7).

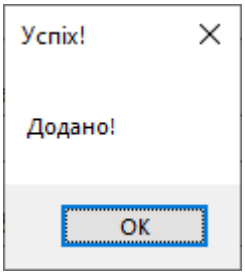
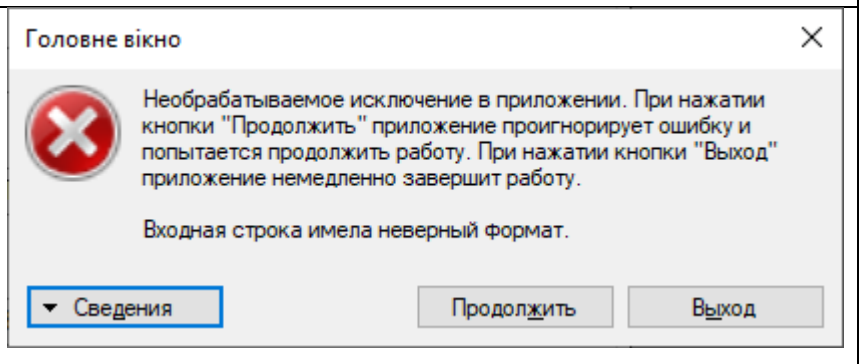


Рисунок 3.27 – Повідомлення про додання

При виконанні цієї дії знайдено помилку. Було створено Bug-report

Таблиця 3.2 – Bug-report

АІС “Гусятинський РЕМ”	
Короткий опис	При доданні запису виникає помилка
Проєкт	АІС “Гусятинський РЕМ”
Компонент програми	Форма для додання споживача
Номер версії	V 1.0
Серйозність	S3(Значний)
Пріоритет	S3(Значний)
Автор	Бурба Віктор
Призначення	Developer – Бурба Віктор
Середовище	
ОС	Windows 10 Home, NET Framework 4.7.2, ОЗП 8 Гб, Intel Core I5 2.7 GHz
Опис	
Кроки відтворення	1. Запуск програми 2. Вхід 3. Відкриття форми для додання споживача 4. Натиснути кнопку додати
Фактичний результат	Виліт програми
Очікуваний результат	Додання запису
Додатково	
Прикріплений файл	

Баг був виправлений заміною коду на наступний

Лістинг 3.1 – Код додання споживача

```
private void AddPPerson_Click(object sender, EventArgs e)
{
    db = new Model1();
    AddPPersonForm add = new AddPPersonForm();
    Physical_persons person = new Physical_persons();

    add.PPersonTown.DataSource = db.Towns.ToList();
    add.PPersonTown.DisplayMember = "Name";
    add.PPersonTown.ValueMember = "Id";

    DialogResult dialogResult = add.ShowDialog(this);
    if (dialogResult == DialogResult.OK)
    {
        person.Name = add.PPersonName.Text.Trim();
        person.Surname = add.PPersonSurname.Text.Trim();
        person.PB = add.PPersonPB.Text.Trim();
        person.Surname = add.PPersonSurname.Text.Trim();
        person.Email = add.PPersonEmail.Text.Trim();
        person.Phone = add.PPersonPhoneNumber.Text.Trim();
        person.Invoice = add.PPersonRahunok.Text.Trim();
        person.House_number = Convert.ToInt32(add.HouseNumber.Text);
        person.Adress = Convert.ToInt32(add.PPersonStreet.SelectedValue);
        db.Physical_persons.Add(person);
        db.SaveChanges();
        MessageBox.Show("Додано!","Успіх!");
    }
}
```

Виправлення багу дозволило продовжити функціональне тестування.

Надалі необхідно переконатися чи працює пошук, знайдемо доданого споживача. Необхідно ввести в поля значення для пошуку та натиснути кнопку “Знайти”(див. рис. 3.28).

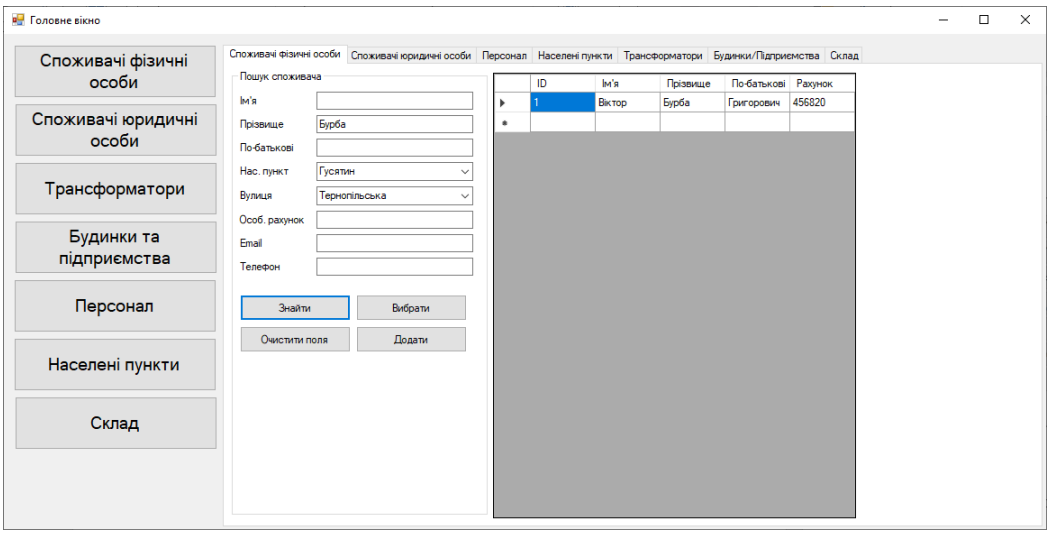


Рисунок 3.28 – Результат пошуку

В таблиці 3.2 наведені тест-кейси.

Таблиця 3.2 – Тест-кейси

ID	Тип	Опис	Очікувано	Реально	Pass/Fail
01	Positive	Перевірка на коректну авторизацію	При введенні правильного логін та паролю увійти, якщо не вони не правильні , то вивести повідомлення про помилку	При введенні вірних логіну та паролю здійснено вхід, при невірних повідомлення про помилку	Pass
02	Positive	Перевірка на додання запису	Додання запису у базу даних	Запис у базу даних доданий	Pass
03	Positive	Вибірка даних з бази даних	Виведення даних	Дані виведенно	Pass

Як показали тести, усі основні функції працюють. Було знайдено помилку та виправлено її.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ураженні електричним струмом.

Небезпека ураження струмом може чекати людину як вдома, так і на вулиці. Виявити пошкоджений або оголений провід, що знаходиться під напругою дуже важко – ні за звуком, ні за запахом, ні візуально провід під напругою не відрізняється від того, який не заживлений у мережі. Тому вкрай необхідно пам’ятати правила електробезпеки, щоб уникнути травматизму.

Людина, яка піддається дії струму, не може покликати на допомогу та самостійно звільнитись від предмету, через який її ударило струмом. Дотик до струмопровідних частин у більшості випадків призводить до судом м’язів, які обмежують здатність рухатись чи говорити. Як правило, про те, що людина у небезпеці свідчить її несподіване падіння на вулиці або неприродне відкидання від джерела струму невидимою силою, раптова втрата свідомості, судоми, яскраво виражене мимовільне скорочення м’язів, опіки на тілі з різко окресленими межами.

Перші дії для допомоги потерпілому від ураження струмом. При ураженні електричним струмом в першу чергу потрібно звільнити потерпілого від струмопровідних частин обладнання. Необхідно швидко відключити від мережі ту частину електрообладнання, до якої доторкається людина. Будь-яке зволікання при наданні допомоги призводить до загибелі людини, яка знаходиться під дією струму. При звільненні потерпілих від струмопровідних частин або проводу вимикають струм, використовуючи сухий одяг, палицю, дошку, шапку, сухі рукавиці, рукав одягу, діелектричні рукавиці. Провідники перерізають інструментом з ізольованими ручками, перерубують сокирою з дерев’яним сухим топорищем [15, 16].

Необхідно захистити себе, відтягуючи потерпілого від джерела струму. Потерпілого можна відтягнути від струмопровідних частин за сухий одяг. При цьому, людина, яка відтягує потерпілого повинна уникати дотику до навколишніх

металевих предметів та до відкритих частин тіла потерпілого. Відтягуючи потерпілого за ноги, не можна торкатися його взуття, оскільки воно може бути сирим і стає провідником електричного струму. Той, хто надає допомогу, повинен одягнути діелектричні рукавиці або обмотати руки шарфом, натягнути на них рукав піджака або пальта. Можна також ізолювати себе, ставши на гумовий килимок, суху дошку тощо.

Необхідно надати першу медичну допомогу ураженій струмом людині. Після звільнення потерпілого від дії струму потрібно відразу ж надати йому першу медичну допомогу.

Негайно сповістити службу екстреної медичної допомоги. Почніть серцево-легеневу реанімацію у будь-якої людини, що не подає ознак життя.

Якісний непрямий масаж серця (якомога швидше починайте виконувати непрямий масаж серця, виконуйте натискання у нижній половині грудини («в центрі грудної клітки»), натискайте на глибину не менше 5, але не більше 6 см, натискайте на грудну клітку зі швидкістю 100–120 натисків/хв з якомога меншою кількістю переривань, дайте грудній клітці повністю випрямитися після кожного стиснення; не спирайтеся на груди (грудну клітку); виконуйте натискання на грудну клітку на твердій поверхні.

Штучне дихання (чергуйте 30 натискань та 2 вдихи;якщо ви не можете забезпечити штучну вентиляцію легенів, виконуйте безперервне натискання на грудну клітку) [17, 18].

Автоматичний зовнішній дефібрилятор (як тільки АЗД надійде (буде доступний), або якщо такий вже є на місці, де наявний постраждалий з зупинкою серця, увімкніть його; прикріпіть електродні прокладки (наліпки) до оголеної грудної клітки постраждалого відповідно до положення, вказаного на АЗД або на прокладках (наліпках); дотримуйтесь голосових (та/або візуальних) підказок АЗД).

Лікаря необхідно викликати незалежно від того чи притомний потерпілий. Номер телефону швидкої допомоги – 103. Якщо потерпілий після звільнення від дії електричного струму і надання медичної допомоги прийшов до тями, його не

слід самого відпускати додому. Над таким потерпілим встановлюють спостереження у лікарні, так як наслідки від впливу електричного струму можуть проявитися через кілька годин і призвести до більш важких наслідків.

4.2 Особливості заходів електробезпеки на підприємствах.

Електричний струм – невидимий для людини, не має запаху та кольору. Тому його важко виявити. Ураження відбувається коли людина, з одного боку, доторкається неізольованого проводу, металевого корпусу електроприладу з несправною ізоляцією або металевого предмета під напругою, а з іншого боку землі, заземлених предметів, труб тощо. На підприємствах електробезпека відіграє важливу роль у запобіганні нещасних випадків, збереженні життя та забезпеченні безпеки працівників. Основні заходи електробезпеки на підприємствах включають наступне:

Виділяють три системи засобів і заходів забезпечення електробезпеки:

- система технічних засобів і заходів;
- система електрозахисних засобів;
- система організаційно-технічних заходів і засобів.

Система технічних засобів і заходів електробезпеки

Технічні засоби та заходи з електробезпеки реалізуються в конструкції електроустановок при їх розробці, виготовленні і монтажі відповідно до чинних нормативів. За своїми функціями технічні засоби і заходи забезпечення електробезпеки поділяються на дві групи:

- технічні заходи і засоби забезпечення електробезпеки при нормальному режимі роботи електроустановок;
- технічні заходи і засоби забезпечення електробезпеки при аварійних режимах роботи електроустановок.

Основні технічні засоби і заходи забезпечення електробезпеки при нормальному режимі роботи електроустановок включають:

- ізоляцію струмовідних частин;
- недоступність струмовідних частин;
- блоківки безпеки;
- засоби орієнтації в електроустановках;
- виконання електроустановок, ізольованих від землі;
- захисне розділення електричних мереж;
- компенсацію ємнісних струмів замикання на землю;
- вирівнювання потенціалів.

Із метою підвищення рівня безпеки, залежно від призначення, умов експлуатації і конструкції, в електроустановках застосовується одночасно більшість з перерахованих технічних засобів і заходів.

Ізоляція струмовідних частин. Забезпечує технічну працездатність електроустановок, зменшує вірогідність потраплянь людини під напругу, замикань на землю і на корпус електроустановок, зменшує струм через людину при доторканні до неізольованих струмовідних частин в електроустановках, що живляться від ізольованої від землі мережі за умови відсутності фаз із пошкодженою ізоляцією [19]:

- робочу – забезпечує нормальну роботу електроустановок і захист від ураження електричним струмом;
- додаткову – забезпечує захист від ураження електричним струмом на випадок пошкодження робочої ізоляції;
- подвійну – складається з робочої і додаткової;
- підсилену – поліпшена робоча ізоляція, яка забезпечує такий рівень захисту як і подвійна.

Також на підприємствах використовують додаткові системи забезпечення безпеки:

Електромагнітні блоківки безпеки вимикачів, роз'єднувачів, заземлюючих ножів використовуються на відкритих і закритих розподільних пристроях з метою

забезпечення необхідної послідовності вмикання і вимикання обладнання. Вони виконуються, переважно, у вигляді стержневих електромагнітів.

Виконання електричних мереж, ізольованих від землі. В мережах, ізольованих від землі, при однофазному включенні людини під напругу і відсутності пошкодження ізоляції інших фаз, величина струму через людину визначається опором ізоляції фаз відносно землі, який, щонайменше, становить 105 Ом. Таким чином, виконання мереж, ізольованих від землі, обмежує величину струму через людину за рахунок опору ізоляції фаз відносно землі при умові забезпечення необхідного стану ізоляції.

Захисне розділення електричних мереж. Загальний опір ізоляції проводів електричної мережі відносно землі і ємкісна складова струму замикання на землю залежать від протяжності мережі і її розгалуженості. Зі збільшенням протяжності і розгалуженості мережі зменшується паралельна робота ізоляторів (накопичення дефектів) і збільшується ємкість.

Застосування малих напруг. До малих напруг належать напруги 42 В і менше змінного струму частотою 50 Гц і 110 В і менше постійного струму. Чинні нормативні документи виділяють два діапазони малих напруг змінного струму: 12 В і 42 В. Напруга до 42 В змінного і до 110 В постійного струму застосовується в приміщеннях з підвищеною небезпекою електротравм, особливо небезпечних і поза приміщеннями для живлення ручного електрифікованого інструменту, ручних переносних ламп, світильників місцевого освітлення з лампами розжарювання, в яких конструктивно не виключена можливість контакту сторонніх осіб зі струмовідними частинами, світильників загального освітлення з лампами розжарювання при висоті підвісу світильників меншій 2,5 м.

Вирівнювання потенціалів. Застосовується з метою зниження можливих напруг дотику і кроку при експлуатації електроустановок або попаданні людини під ці напруги за інших обставин. Вирівнювання потенціалів досягається за рахунок навмисного підвищення потенціалу опорної поверхні, на якій може стояти людина, до рівня потенціалу струмовідних частин, яких вона може

торкатися, або за рахунок зменшення перепаду потенціалів на поверхні землі чи підлозі приміщень в зоні можливого розтікання струму [20, 21, 22].

ВИСНОВКИ

У ході виконання дипломного проекту було розроблено програму та базу даних для обліку послуг споживачів електроенергії Гусятинського РЕМу. База даних була структурована та містити інформацію про споживачів, лічильники, трансформатори та кількість спожитої ними електроенергії, а також про надані послуги та виконані ремонтні роботи.

Створена програма показує високу ефективність та функціональність, що дозволяє зручно та швидко виконувати облік споживачів електроенергії та їхніх показників споживання. Вона надає можливість вести точний облік, зберігати та оновлювати дані, а також генерувати звіти для забезпечення ефективного управління роботою Гусятинського РЕМу.

Розроблена база даних та програма сприятимуть покращенню процесу обліку та надання послуг споживачам електроенергії. Це дозволить забезпечити більш точний та ефективний контроль, виявляти можливі проблеми та здійснювати оперативні заходи для їх вирішення. Такий підхід сприятиме покращенню якості надання послуг та задоволенню потреб споживачів електроенергії в регіоні, що в свою чергу позитивно позначиться на репутації та розвитку Гусятинського РЕМу.

В якості системи управління базами даних було вибрано вбудовану СКБД у Visual Studio 2019, оскільки вона є провідною на ринку, а також має зручний інтерфейс.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Сучасні автоматизовані інформаційні системи та їх використання в економічному аналізі й управлінні. Pidru4niki. URL: https://pidru4niki.com/18421120/ (дата звернення: 25.05.2023).

2. Складові частини інформаційних систем: Характеристика. Освіта.UA. URL: https://osvita.ua/vnz/reports/econom_pidpr/19058/ (дата звернення: 26.05.2023).

3. Посібник щодо використання комплексу програмного забезпечення АСТОР / Посадова інструкція пакету галузевого програмного забезпечення. К.: ТОВ «Інфо-прайм», 2020. 237 ст.

4. Гайна Г. А. Основи проектування баз даних. Київ: КНУБА, 2005.204 с.

5. Реляційні бази даних. Relational databases. URL: https://rdb.dp.ua/uk/chapter_03 (дата звернення: 01.06.2023).

6. SQL SELECT Statement. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/sql/sql_select.asp (дата звернення: 04.06.2023).

7. П30 Петрик М.Р. Моделювання програмного забезпечення : науково методичний посібник / М.Р. Петрик, О.Ю. Петрик – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2015. – 200 с.

8. Графічний інтерфейс користувача. — Wiki TNEU. Wiki TNEU. URL: http://wiki.tneu.edu.ua/index.php?title=Графічний_інтерфейс_користувача. (дата звернення: 06.06.2023).

9. Інтуїтивно зрозумілий користувальницький інтерфейс & як він може допомогти покращити UX / UI / UX. Кращі уроки по веб-розробці. URL: https://ua.phhsnews.com/articles/uiux/intuitive-ui-how-it-can-help-improve-ux.html (дата звернення: 08.06.2023).

10. Поняття про БД та СКБД. URL: https://studopedia.com.ua/1_160172_ponyattya-pro-bd-ta-skbd.html (дата звернення: 08.06.2023)

11. Настанова по ADO.NET entity framework 6. METANIT.COM. URL: https://metanit.com/sharp/entityframework/ (дата звернення: 09.06.2023).

12. Object relational mapping (ORM) data access :: Spring framework. URL: https://docs.spring.io/spring-framework/reference/data-access/orm.html (дата звернення: 09.06.2023).

13. Language-Integrated Query (LINQ) (C#). Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/(дата звернення: 09.06.2023).

14. Введення в тестування програмного забезпечення. Навчальний ресурс з тестування програмного забезпечення. URL: https://qalearning.com.ua/theory/lectures/material/testing-intro/ (дата звернення: 11.06.2023)

15. Бедрій І.Я., Нечай В.Я. Безпека життєдіяльності. Навчальний посібник. – Львів: Манголія 2006, 2007. 499 с.

16. Безпека життєдіяльності. Навчальний посібник. - К.: Основа, 2011.

17. Скобло Ю.С., Соколовська Т.Б., Морозенко Д.І. та ін. Безпека життєдіяльності. Навчальний посібник для вищих навчальних закладів 3-4 рівнів акредитації. – К.: Кондор, 2003. 424 с.

18. Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці. – К.: Каравела, 2007. 408 с.

19. Д. Заікіна та В. Глива, Основи охорони праці та безпека життєдіяльності. RS Global S. z O.O.

20. ДСТУ 7237:2011 Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту Київ 2011 10 с. (Інформація та документація).

21. ПУЕ Правила улаштування електроустановок (перше переглянуте, перероблене, доповнене та адаптоване до умов України видання) Київ 2017 617 с.

22. Охорона праці в галузі: Загальні вимоги. Навчальний посібник. «Основа». 2011. 551 с.

ДОДАТКИ

ДОДАТОК А Програмний код

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Project.Model;
using Word = Microsoft.Office.Interop.Word;
namespace Project
{
    public partial class MainForm : Form
    {
        Model1 db;

        public void ReplaceWordStub(string StubToReplace, string
text, Word.Document document)
        {
            var range = document.Content;
            range.Find.ClearFormatting();
            range.Find.Execute(FindText: StubToReplace, ReplaceWith:
text);
        }

        public MainForm()
        {
            InitializeComponent();
        }
        private void MainForm_Load(object sender, EventArgs e)
        {
            db = new Model1();
            //Список міст
            PPersonTown.DataSource = db.Towns.ToList();
            PPersonTown.DisplayMember = "Name";
            PPersonTown.ValueMember = "Id";

            EmployeeTownBox.DataSource = db.Towns.ToList();
            EmployeeTownBox.DisplayMember = "Name";
            EmployeeTownBox.ValueMember = "Id";

            HouseTownBox.DataSource = db.Towns.ToList();
            HouseTownBox.DisplayMember = "Name";
            HouseTownBox.ValueMember = "Id";
            TransformerTownBox.DataSource = db.Towns.ToList();
            TransformerTownBox.DisplayMember = "Name";
            TransformerTownBox.ValueMember = "Id";
```

Продовження додатку А

```
            JPersonTown.DataSource = db.Towns.ToList();
            JPersonTown.DisplayMember = "Name";
            JPersonTown.ValueMember = "Id";
        }
        private void PPersonTown_SelectedValueChanged(object sender,
EventArgs e)
        {
            //Виведення вулиця за містом
            db = new Model1();
            List<Streets> streets = new List<Streets>();
            foreach (Streets streets1 in db.Streets)
            {
                if (streets1.Town.Equals(PPersonTown.SelectedValue))
                {
                    streets.Add(streets1);
                }
            }
            PPersonStreet.DataSource = streets;
            PPersonStreet.DisplayMember = "Name".Trim();
            PPersonStreet.ValueMember = "Id";
        }
        private void PPersonFindButton_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            PPersonView.Rows.Clear();
            PPersonView.Columns.Clear();
            string Command = "Select * from Physical_persons Where
";
            int k = 0;
            if (PPersonName.Text!="")
            {
                if (k>0)
                {
                    Command += " AND ";
                }
                Command += "Name Like N'%" +PPersonName.Text.Trim()
+"%'";
                k++;
            }
            if (PPersonSurname.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Surname Like N'%" +
PPersonSurname.Text.Trim()+"%'";
                k++;
            }
            if (PPersonPB.Text != "")
```

Продовження додатку А

```
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "PB Like N'%" +
PPersonPB.Text.Trim()+"%'";
                k++;
            }
            if (PPersonEmail.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Email Like N'%" +
PPersonEmail.Text.Trim() + "%'";
                k++;
            }
            if (PPersonPhoneNumber.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Phone Like N'%" +
PPersonPhoneNumber.Text.Trim() + "%'";
                k++;
            }
            if (PPersonRahunok.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Invoice Like N'%" +
PPersonRahunok.Text.Trim() + "%'";
                k++;
            }
            if (PPersonStreet.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Adress ="
+Convert.ToInt32(PPersonStreet.SelectedValue);
                k++;
            if (k==0)
            {
               MessageBox.Show("Введіть дані для пошуку!");
```

Продовження додатку А

```
                return;
            }
            List<Physical_persons> list =
db.Database.SqlQuery<Physical_persons>(Command).ToList();

            int rownumber = 0;
            PPersonView.Columns.Add("Id", "ID");
            PPersonView.Columns.Add("Name", "Ім'я");
            PPersonView.Columns.Add("Surname", "Прізвище");
            PPersonView.Columns.Add("PB", "По-батькові");
            PPersonView.Columns.Add("Rahunok", "Рахунок");
            foreach (Physical_persons item in list)
            {
                PPersonView.Rows.Add();
                PPersonView.Rows[rownumber].Cells["Id"].Value =
item.Id;
                PPersonView.Rows[rownumber].Cells["Name"].Value =
item.Name.Trim();
                PPersonView.Rows[rownumber].Cells["Surname"].Value =
item.Surname.Trim();
                PPersonView.Rows[rownumber].Cells["PB"].Value =
item.PB.Trim();
                PPersonView.Rows[rownumber].Cells["Rahunok"].Value =
item.Invoice.Trim();
                rownumber++;
            }
        }
        private void PPersonWorkButton_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            PPersonWorkingForm form = new PPersonWorkingForm();
            form.StreetBox.DataSource = db.Streets.ToList();
            form.StreetBox.DisplayMember = "Name";
            form.StreetBox.ValueMember = "Id";
            form.TownBox.DataSource = db.Towns.ToList();
            form.TownBox.DisplayMember = "Name";
            form.TownBox.ValueMember = "Id";
            int id;
            if (PPersonView.RowCount != 0 &&
PPersonView.CurrentRow.Cells[0].Value != null )
            {
                id =
Convert.ToInt32(PPersonView.CurrentRow.Cells[0].Value);
            }
            else
            {
                MessageBox.Show("Виберіть споживача!","Увага!");
                return;
            }
            Physical_persons item = db.Physical_persons.Find(id);
```

Продовження додатку А

```
            form.IdConsumerBox.Text = Convert.ToString(item.Id);
            form.NameBox.Text = item.Name;
            form.SurnameBox.Text = item.Surname;
            form.PBBox.Text = item.PB;
            form.EmailBox.Text = item.Email;
            form.NumberBox.Text = item.Phone;
            form.StreetBox.SelectedValue = item.Adress;
            form.TownBox.SelectedValue = item.Streets.Towns.Id;
            form.RahunokBox.Text = item.Invoice;
            form.HouseNumberBox.Text =
Convert.ToString(item.House_number);
            form.ConsumerHousesView.Columns.Add("Id", "ID");
            form.ConsumerHousesView.Columns.Add("Adress", "Адреса");
            form.ConsumerHousesView.Columns.Add("Number", "Номер
будинку");
            int rownumber = 0;
            foreach (Houses houses in item.Houses)
            {
                form.ConsumerHousesView.Rows.Add();

form.ConsumerHousesView.Rows[rownumber].Cells["Id"].Value =
houses.Id;

form.ConsumerHousesView.Rows[rownumber].Cells["Adress"].Value =
houses.Streets.Towns.Name.Trim() + ":" + houses.Streets.Name.Trim();

form.ConsumerHousesView.Rows[rownumber].Cells["Number"].Value =
houses.Number;
                rownumber++;
            }
            form.ShowDialog();
        }
        private void AddVehicle_Click(object sender, EventArgs e)
        {
        }
        private void AddMaterialButton_Click(object sender,
EventArgs e)
        {
        }
        private void EmployeeTownBox_SelectedValueChanged(object
sender, EventArgs e)
        {
            db = new Model1();
            List<Streets> streets = new List<Streets>();
            foreach (Streets streets1 in db.Streets)
            {
                if
(streets1.Town.Equals(EmployeeTownBox.SelectedValue))
                {
                    streets.Add(streets1);
                }
```

Продовження додатку А

```
            }
            EmployeeStreetBox.DataSource = streets;
            EmployeeStreetBox.DisplayMember = "Name".Trim();
            EmployeeStreetBox.ValueMember = "Id";
        }
        private void PPersonClearButton_Click(object sender,
EventArgs e)
        {
            PPersonName.Clear();
            PPersonSurname.Clear();
            PPersonPB.Clear();
            PPersonEmail.Clear();
            PPersonPhoneNumber.Clear();
            PPersonRahunok.Clear();
            PPersonTown.Text = "";
            PPersonStreet.Text = "";
        }
        private void UpdateGroupButton_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            GroupsView.Rows.Clear();
            GroupsView.Columns.Clear();
            GroupsView.Columns.Add("Id", "ID");
            GroupsView.Columns.Add("Name", "Назва");
            int rownumber = 0;
            foreach (Work_groups equipment in db.Work_groups)
            {
                GroupsView.Rows.Add();
                GroupsView.Rows[rownumber].Cells["Id"].Value =
equipment.Id;
                GroupsView.Rows[rownumber].Cells["Name"].Value =
equipment.Name.Trim();

                rownumber++;
            }
        }

        private void AddGroupButton_Click(object sender, EventArgs
e)
        {
            db = new Model1();
            AddWorkGroup add = new AddWorkGroup();
            Work_groups group = new Work_groups();
            DialogResult dialogResult = add.ShowDialog(this);
            if (dialogResult == DialogResult.OK)
            {
                group.Name = add.GroupName.Text.Trim();
                db.Work_groups.Add(group);
                db.SaveChanges();
                MessageBox.Show("Додано!","Успіх");
```

Продовження додатку А

```
            }
        }

        private void AddPPerson_Click(object sender, EventArgs e)
        {
            db = new Model1();
            AddPPersonForm add = new AddPPersonForm();
            Physical_persons person = new Physical_persons();

            add.PPersonTown.DataSource = db.Towns.ToList();
            add.PPersonTown.DisplayMember = "Name";
            add.PPersonTown.ValueMember = "Id";

            DialogResult dialogResult = add.ShowDialog(this);
            if (dialogResult == DialogResult.OK)
            {
                person.Name = add.PPersonName.Text.Trim();
                person.Surname = add.PPersonSurname.Text.Trim();
                person.PB = add.PPersonPB.Text.Trim();
                person.Surname = add.PPersonSurname.Text.Trim();
                person.Email = add.PPersonEmail.Text.Trim();
                person.Phone = add.PPersonPhoneNumber.Text.Trim();
                person.Invoice = add.PPersonRahunok.Text.Trim();
                person.House_number =
Convert.ToInt32(add.HouseNumber.Text);
                person.Adress =
Convert.ToInt32(add.PPersonStreet.SelectedValue);
                db.Physical_persons.Add(person);
                db.SaveChanges();
                MessageBox.Show("Додано!","Успіх!");
            }
        }

        private void FindEmployeeButton_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            PersonalView.Rows.Clear();
            PersonalView.Columns.Clear();
            string Command = "Select * from Employees Where ";
            int k = 0;

            if (EmployeeNameBox.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Name Like N'%" +
EmployeeNameBox.Text.Trim() + "%'";
                k++;
```

Продовження додатку А

```
            }
            if (EmployeeSurnameBox.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Surname Like N'%" +
EmployeeSurnameBox.Text.Trim() + "%'";
                k++;
            }

            if (EmployeePBBox.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "PB Like N'%" + EmployeePBBox.Text.Trim()
+ "%'";
                k++;
            }

            if (EmployeePositionBox.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Position Like N'%" +
EmployeePositionBox.Text.Trim() + "%'";
                k++;
            }

            if (EmployeeStreetBox.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Adress =" +
Convert.ToInt32(EmployeeStreetBox.SelectedValue);
                k++;
            }

            if (k == 0)
            {
                MessageBox.Show("Введіть дані для пошуку!");
                return;
            }
```

Продовження додатку А

```
            List<Employees> list =
db.Database.SqlQuery<Employees>(Command).ToList();
            int rownumber = 0;
            PersonalView.Columns.Add("Id", "ID");
            PersonalView.Columns.Add("Name", "Ім'я");
            PersonalView.Columns.Add("Surname", "Прізвище");
            PersonalView.Columns.Add("PB", "По-батькові");
            PersonalView.Columns.Add("Posada", "Посада");
            foreach (Employees item in list)
            {
                PersonalView.Rows.Add();
                PersonalView.Rows[rownumber].Cells["Id"].Value =
item.Id;
                PersonalView.Rows[rownumber].Cells["Name"].Value =
item.Name.Trim();
                PersonalView.Rows[rownumber].Cells["Surname"].Value
= item.Surname.Trim();
                PersonalView.Rows[rownumber].Cells["PB"].Value =
item.PB.Trim();
                PersonalView.Rows[rownumber].Cells["Posada"].Value =
item.Position.Trim();
                rownumber++;
            }
        }
        private void ChooseGroupButton_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            AboutWorkGroupForm about = new AboutWorkGroupForm();
            int id = 0;
            if (GroupsView.RowCount != 0)
            {
                if (GroupsView.CurrentRow.Cells[0].Value != null)
                {
                    id =
Convert.ToInt32(GroupsView.CurrentRow.Cells[0].Value);
                }
                else
                {
                    MessageBox.Show("Виберіть групу!", "Увага!");
                    return;
                }
            }
            else
            {
                MessageBox.Show("Виберіть групу!", "Увага!");
                return;
            }
            Work_groups group = db.Work_groups.Find(id);
            about.NameGroup.Text = group.Name.Trim();
           // about.EmployeesView.Columns.Add("Id", "Id");
```

Продовження додатку А

```
            about.EmployeesView.Columns.Add("Name", "Ім'я");
            about.EmployeesView.Columns.Add("Surname", "Прізвище");
            about.EmployeesView.Columns.Add("PB", "По-батькові");
            about.EmployeesView.Columns.Add("Position", "Посада");
            int rownumber = 0;
            List<Employees> list = group.Employees.ToList();
            foreach (Employees item in list)
            {
                about.EmployeesView.Rows.Add();

//about.EmployeesView.Rows[rownumber].Cells["Id"].Value = item.Id;

about.EmployeesView.Rows[rownumber].Cells["Name"].Value =
item.Name.Trim();

about.EmployeesView.Rows[rownumber].Cells["Surname"].Value =
item.Surname.Trim();

about.EmployeesView.Rows[rownumber].Cells["PB"].Value =
item.PB.Trim();

about.EmployeesView.Rows[rownumber].Cells["Position"].Value =
item.Position.Trim();
                rownumber++;
            }
            rownumber = 0;
            about.TaskView.Columns.Add("Id", "Id");
            about.TaskView.Columns.Add("Type", "Тип");
            about.TaskView.Columns.Add("Progress", "Прогрес");
            List<HTasks> list1 = group.HTasks.ToList();
            foreach (HTasks item in list1)
            {
                about.TaskView.Rows.Add();
                about.TaskView.Rows[rownumber].Cells["Id"].Value =
item.Id;
                about.TaskView.Rows[rownumber].Cells["Type"].Value =
item.Type.Trim();
                if (Convert.ToBoolean(item.Progress) == true)
                {

about.TaskView.Rows[rownumber].Cells["Progress"].Value = "Виконано";
                }
                else
                {

about.TaskView.Rows[rownumber].Cells["Progress"].Value = "Не
виконано";
                }
                rownumber++;
            }
            about.ShowDialog();
```

Продовження додатку А

```
        }
        private void AddEmployees_Click(object sender, EventArgs e)
        {
            AddEmployeeForm add = new AddEmployeeForm();
            Employees item = new Employees();
            db = new Model1();
            add.EmployeeTown.DataSource = db.Towns.ToList();
            add.EmployeeTown.DisplayMember = "Name";
            add.EmployeeTown.ValueMember = "Id";
            add.GroupBox.DataSource = db.Work_groups.ToList();
            add.GroupBox.DisplayMember = "Name";
            add.GroupBox.ValueMember = "Id";
            DialogResult dialogResult = add.ShowDialog(this);
            if (dialogResult == DialogResult.OK)
            {
                item.Name = add.EmployeeName.Text.Trim();
                item.Surname = add.EmployeeSurname.Text.Trim();
                item.PB= add.EmployeePB.Text.Trim();
                item.Position = add.EmployeePosition.Text.Trim();
                item.WorkGroup =
Convert.ToInt32(add.GroupBox.SelectedValue);
                item.Adress =
Convert.ToInt32(add.EmployeeStreet.SelectedValue);
                db.Employees.Add(item);
                db.SaveChanges();
                MessageBox.Show("Додано!", "Успіх!");
            }
        }
        private void UpdateTownView_Click(object sender, EventArgs
e)
        {
            db = new Model1();
            int rownumber = 0;
            TownsView.Columns.Clear();
            TownsView.Rows.Clear();
            TownsView.Columns.Add("Id", "Id");
            TownsView.Columns.Add("Name", "Назва");
            TownsView.Columns.Add("Count", "К-ксть вулиць");
            List<Towns> list = db.Towns.ToList();
            foreach (Towns town in list)
            {
                TownsView.Rows.Add();
                TownsView.Rows[rownumber].Cells["Id"].Value =
town.Id;
                TownsView.Rows[rownumber].Cells["Name"].Value =
town.Name.Trim();
                TownsView.Rows[rownumber].Cells["Count"].Value =
town.Streets.Count;
                rownumber++;
            }
        }
```

Продовження додатку А

```
        private void TownsView_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            db = new Model1();
            if (TownsView.CurrentRow.Cells[0].Value != null)
            {
                int id =
Convert.ToInt32(TownsView.CurrentRow.Cells[0].Value);
                Towns town = db.Towns.Find(id);
                int rownumber = 0;
                StreetsView.Columns.Clear();
                StreetsView.Rows.Clear();
                StreetsView.Columns.Add("Id", "Id");
                StreetsView.Columns.Add("Name", "Назва");
                StreetsView.Columns.Add("HCount", "К-кість
будинків");
                StreetsView.Columns.Add("ECount", "К-кість
підприємств");
                List<Streets> list = town.Streets.ToList();
                foreach (Streets street in list)
                {
                    StreetsView.Rows.Add();
                    StreetsView.Rows[rownumber].Cells["Id"].Value =
street.Id;
                    StreetsView.Rows[rownumber].Cells["Name"].Value
= street.Name.Trim();

StreetsView.Rows[rownumber].Cells["HCount"].Value =
street.Houses.Count;

StreetsView.Rows[rownumber].Cells["ECount"].Value =
street.Enterprises.Count;
                    rownumber++;
                }
            }
            else
            {
                MessageBox.Show("Виберіть місто","Увага!");
            }
        }

        private void AddTownButton_Click(object sender, EventArgs e)
        {
            db = new Model1();
            Towns town = new Towns();
            AddTown add = new AddTown();
            add.Text = "Додання міста";
            DialogResult dialogResult = add.ShowDialog(this);
            if (dialogResult == DialogResult.OK)
            {
                town.Name = add.NameBox.Text.Trim();
```

Продовження додатку А

```
                db.Towns.Add(town);
                db.SaveChanges();
                MessageBox.Show("Додано!","Успіх!");
            }
        }

        private void AddStreetButton_Click(object sender, EventArgs
e)
        {
            db = new Model1();
            Streets street = new Streets();
            AddStreet add = new AddStreet();
            add.TownBox.DataSource = db.Towns.ToList();
            add.TownBox.DisplayMember = "Name";
            add.TownBox.ValueMember = "Id";

            add.Text = "Додання вулиці";
            DialogResult dialogResult = add.ShowDialog(this);
            if (dialogResult == DialogResult.OK)
            {
                street.Name = add.NameBox.Text.Trim();
                street.Town =
Convert.ToInt32(add.TownBox.SelectedValue);
                db.Streets.Add(street);
                db.SaveChanges();
                MessageBox.Show("Додано!", "Успіх!");
            }
        }

        private void HouseTownBox_SelectedValueChanged(object
sender, EventArgs e)
        {
            db = new Model1();
            List<Streets> streets = new List<Streets>();
            foreach (Streets streets1 in db.Streets)
            {
                if
(streets1.Town.Equals(HouseTownBox.SelectedValue))
                {
                    streets.Add(streets1);
                }
            }
            HouseStreetBox.DataSource = streets;
            HouseStreetBox.DisplayMember = "Name".Trim();
            HouseStreetBox.ValueMember = "Id";
        }

        private void TransformerTownBox_SelectedValueChanged(object
sender, EventArgs e)
        {
            db = new Model1();
```

Продовження додатку А

```
            List<Streets> streets = new List<Streets>();
            foreach (Streets streets1 in db.Streets)
            {
                if
(streets1.Town.Equals(TransformerTownBox.SelectedValue))
                {
                    streets.Add(streets1);
                }
            }
            TransformerStreetBox.DataSource = streets;
            TransformerStreetBox.DisplayMember = "Name".Trim();
            TransformerStreetBox.ValueMember = "Id";
        }

        private void AddTransformerButton_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            AddTransformerForm add = new AddTransformerForm();
            add.TownBox.DataSource = db.Towns.ToList();
           add.TownBox.DisplayMember = "Name";
            add.TownBox.ValueMember = "Id";

            DialogResult dialogResult = add.ShowDialog(this);
            if (dialogResult == DialogResult.OK)
            {
                Transformers item = new Transformers();
                item.Model = add.NameBox.Text.Trim();
                item.Power =
Convert.ToInt32(add.PowerBox.Text.Trim());
                item.Adress =
Convert.ToInt32(add.StreetBox.SelectedValue);
                db.Transformers.Add(item);
                db.SaveChanges();
                MessageBox.Show("Додано!", "Успіх!");
            }
        }
        private void UpdateTransformerView_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            List<Transformers> list = new List<Transformers>();
            list = db.Transformers.ToList();
            int row = 0;
            TransformerView.Columns.Add("Id","Id");
            TransformerView.Columns.Add("Model","Модель");
            TransformerView.Columns.Add("Power","Потужність");
            TransformerView.Columns.Add("Count","Підк. обєкти");
            foreach (Transformers item in list)
            {
                TransformerView.Rows.Add();
```

Продовження додатку А

```
                TransformerView.Rows[row].Cells["Id"].Value =
item.Id;
                TransformerView.Rows[row].Cells["Model"].Value =
item.Model;
                TransformerView.Rows[row].Cells["Power"].Value =
item.Power;
                TransformerView.Rows[row].Cells["Count"].Value =
item.Houses.Count + item.Enterprises.Count;
                row++;
            }
        }
        private void AddMoreMaterialButton_Click(object sender,
EventArgs e)
        {
            AddMoreMaterialForm add = new AddMoreMaterialForm();
            db = new Model1();
            Materials item =
db.Materials.Find(Convert.ToInt32(MaterialView.CurrentRow.Cells[0].V
alue));
            DialogResult dialogResult = add.ShowDialog(this);

            if (dialogResult == DialogResult.OK)
            {
                var name = item.Name.Trim();
                var oldcount = item.Count;
                var added = Convert.ToInt32(add.CountBox.Text);
                var price = Convert.ToInt32(add.PriceBox.Text);
                var newcount = item.Count += added;
                var allpice = added * price;
                item.Count += Convert.ToInt32(add.CountBox.Text);
                db.Entry(item).State = EntityState.Modified;
                db.SaveChanges();
                MessageBox.Show("Додано");

                string Template =
@"D:\Project\Templates\AddMoreMaterialExample.docx";
                var WordApp = new Word.Application();
                WordApp.Visible = false;
                var WordDocument = WordApp.Documents.Open(Template);
                string NewPath =
@"D:\Project\Documents\Nakladna.docx";
                /*string NewPath = @"D:\Project\Documents\Nakladna
"+DateTime.Now.ToShortDateString()+".docx";
                MessageBox.Show(NewPath);*/
                WordDocument.SaveAs(NewPath);
                WordDocument = WordApp.Documents.Open(NewPath);
                ReplaceWordStub("{name}", name, WordDocument);

ReplaceWordStub("{oldcount}",oldcount.ToString(),WordDocument);
                ReplaceWordStub("{added}",added.ToString(),
WordDocument);
```

Продовження додатку А

```
                ReplaceWordStub("{price}",price.ToString(),
WordDocument);
                ReplaceWordStub("{newcount}",newcount.ToString(),
WordDocument);
                ReplaceWordStub("{allprice}",allpice.ToString(),
WordDocument);

ReplaceWordStub("{date}",DateTime.Now.ToShortDateString(),WordDocume
nt);
                WordDocument.Save();
                WordApp.Visible = true;

            }
        }

        private void UpdateMaterials_Click(object sender, EventArgs
e)
        {
            db = new Model1();
            MaterialView.Rows.Clear();
            MaterialView.Columns.Clear();
            List<Materials> list = db.Materials.ToList();
            MaterialView.Columns.Add("Id", "ID");
            MaterialView.Columns.Add("Name", "Назва");
            MaterialView.Columns.Add("Unit", "Од. виміру");
            MaterialView.Columns.Add("Count", "Кількість");
            int rownumber = 0;
            foreach (Materials item in list)
            {
                MaterialView.Rows.Add();
                MaterialView.Rows[rownumber].Cells["Id"].Value =
item.Id;
                MaterialView.Rows[rownumber].Cells["Name"].Value =
item.Name.Trim();
                MaterialView.Rows[rownumber].Cells["Unit"].Value =
item.Unit.Trim();
                MaterialView.Rows[rownumber].Cells["Count"].Value =
item.Count;
                rownumber++;
            }
        }

        private void FindJPersonButton_Click(object sender,
EventArgs e)
        {
            db = new Model1();
            JPersonView.Rows.Clear();
            JPersonView.Columns.Clear();
            string Command = "Select * from Juridical_persons Where
";
            int k = 0;
```

Продовження додатку А

```
            if (JPersonName.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Name Like N'%" + JPersonName.Text.Trim()
+ "%'";
                k++;
            }

            if (JPersonEmail.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Email Like N'%" +
JPersonEmail.Text.Trim() + "%'";
                k++;
            }
            if (JPersonPhone.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Phone Like N'%" +
JPersonPhone.Text.Trim() + "%'";
                k++;
            }
            if (JPersonInvoice.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Invoice Like N'%" +
JPersonInvoice.Text.Trim() + "%'";
                k++;
            }
            if (JPersonStreet.Text != "")
            {
                if (k > 0)
                {
                    Command += " AND ";
                }
                Command += "Office_adress =" +
Convert.ToInt32(JPersonStreet.SelectedValue);
                k++;
            }
```

Продовження додатку А

```
            if (k == 0)
            {
                MessageBox.Show("Введіть дані для пошуку!");
                return;
            }
            List<Juridical_persons> list =
db.Database.SqlQuery<Juridical_persons>(Command).ToList();
            int rownumber = 0;
            JPersonView.Columns.Add("Id", "ID");
            JPersonView.Columns.Add("Name", "Назва");
            JPersonView.Columns.Add("Phone", "Телефон");
            JPersonView.Columns.Add("Invoice", "Рахунок");
            foreach (Juridical_persons item in list)
            {
                JPersonView.Rows.Add();
                JPersonView.Rows[rownumber].Cells["Id"].Value =
item.Id;
                JPersonView.Rows[rownumber].Cells["Name"].Value =
item.Name.Trim();
                JPersonView.Rows[rownumber].Cells["Phone"].Value =
item.Phone.Trim();
                JPersonView.Rows[rownumber].Cells["Invoice"].Value =
item.Invoice.Trim();
                rownumber++;
            }
        }
        private void JPersonTown_SelectedValueChanged(object sender,
EventArgs e)
        {
            db = new Model1();
            List<Streets> streets = new List<Streets>();
            foreach (Streets streets1 in db.Streets)
            {
                if (streets1.Town.Equals(JPersonTown.SelectedValue))
                {
                    streets.Add(streets1);
                }
            }
            JPersonStreet.DataSource = streets;
            JPersonStreet.DisplayMember = "Name".Trim();
            JPersonStreet.ValueMember = "Id";
        }
        private void ChooseJPerson_Click(object sender, EventArgs e)
        {
            db = new Model1();
            int id=0;
            if (JPersonView.RowCount>0)
            {
                if (JPersonView.CurrentRow.Cells[0].Value!=null)
                {
```

Продовження додатку А

```
                    id =
Convert.ToInt32(JPersonView.CurrentRow.Cells[0].Value);
                }
                else
                {
                    MessageBox.Show("Виберіть споживача!");
                    return;
                }
            }
            else
            {
                MessageBox.Show("Виберіть споживача!");
                return;
            }
            Juridical_persons person =
db.Juridical_persons.Find(id);
            JPersonWorkingForm form = new JPersonWorkingForm();
            form.IdConsumerBox.Text = Convert.ToString(person.Id);
            form.NameBox.Text = person.Name.Trim();
            form.NumberBox.Text = person.Phone.Trim();
            form.EmailBox.Text = person.Email.Trim();
            form.RahunokBox.Text = person.Invoice.Trim();
            form.TownBox.DataSource = db.Towns.ToList();
            form.TownBox.DisplayMember = "Name";
            form.TownBox.ValueMember = "Id";
            form.TownBox.SelectedValue = person.Streets.Towns.Id;
            form.ConsumerEnterprisesView.Columns.Add("ID", "Id");
            form.ConsumerEnterprisesView.Columns.Add("Name",
"Назва");
            form.ConsumerEnterprisesView.Columns.Add("Adress",
"Адреса");
            int rownumber = 0;
            foreach (Enterprises item in person.Enterprises)
            {
                form.ConsumerEnterprisesView.Rows.Add();

form.ConsumerEnterprisesView.Rows[rownumber].Cells["ID"].Value =
item.Id;

form.ConsumerEnterprisesView.Rows[rownumber].Cells["Name"].Value =
item.Name.Trim();

form.ConsumerEnterprisesView.Rows[rownumber].Cells["Adress"].Value =
item.Streets.Towns.Name.Trim() + ": "
                    + item.Streets.Name.Trim();
                rownumber++;
            }
            form.ShowDialog();
        }
        private void AddVehicleButton_Click(object sender, EventArgs
e)
```

Продовження додатку А

```
        {
            db = new Model1();
            AddEquipment form = new AddEquipment();
            DialogResult dialogResult = form.ShowDialog(this);
            if (dialogResult == DialogResult.OK)
            {
                Equipment item = new Equipment();
                item.Name = form.ModelName.Text.Trim();
                item.Type = form.TypeBox.Text.Trim();
                item.Numbers = form.Number.Text.Trim();
                db.Equipment.Add(item);
                db.SaveChanges();
                MessageBox.Show("Додано!");
            }
        }
        private void UpdateEquip_Click(object sender, EventArgs e)
        {
            db = new Model1();
            VehicleView.Rows.Clear();
            VehicleView.Columns.Clear();
            VehicleView.Columns.Add("Id","ID");
            VehicleView.Columns.Add("Model","Модель");
            VehicleView.Columns.Add("Type","Тип");
            VehicleView.Columns.Add("Numbers","Номери");
            int rownumber = 0;
            foreach (Equipment item in db.Equipment)
            {
                VehicleView.Rows.Add();
                VehicleView.Rows[rownumber].Cells["Id"].Value =
item.Id;
                VehicleView.Rows[rownumber].Cells["Model"].Value =
item.Name.Trim();
                VehicleView.Rows[rownumber].Cells["Type"].Value =
item.Type.Trim();
                VehicleView.Rows[rownumber].Cells["Numbers"].Value =
item.Numbers.Trim();
                rownumber++;
            }
        }
    }
}
```

ДОДАТОК Б Диск з програмою