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

Факультет компютерно-інформаційних систем та програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

На тему: «Розробка автоматизованої системи управління абонентським відділом комунального підприємства на платформі .NET»

Виконав: студент 4 курсу, групи СПс-41
спеціальності 121 «Інженерія програмного забезпечення»

(шифр і назва напряму підготовки, спеціальності)

	_____ (підпис)	Музика В.В. (прізвище та ініціали)
Керівник	_____ (підпис)	Михалик Д.М. (прізвище та ініціали)
Нормоконтроль	_____ (підпис)	Стоянов Ю. М. (прізвище та ініціали)
Завідувач кафедри	_____ (підпис)	Петрик М. Р. (прізвище та ініціали)
Рецензент	_____ (підпис)	_____ (прізвище та ініціали)

Тернопіль, 2024

АНОТАЦІЯ

Кваліфікаційна робота для здобуття ступеня «бакалавр» на тему: «Розробка автоматизованої системи управління абонентським відділом комунального підприємства на платформі .NET». Тернопільського національного технічного університету імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПс-41.

Відомість про обсяг: 74 сторінок, 26 рисунків, 9 таблиць, 5 додатків, 22 джерела.

Об’єктом дослідження даної роботи являється абонентський відділ комунального підприємства.

Метою роботи являється проектування бази даних та створення системи управління інформацією про абонентів та комунальні послуги.

Для дослідження була використана методика, що включає в себе аналіз предметної області та опрацювання інформації по поставленому завданню; проектування бази даних з використанням методів нормалізації; створення та тестування програми в середовищі Microsoft Visual Studio.

В результаті виконання поставленого завдання, було спроектовано та розроблено автоматизовану систему управління абонентами та послугами.

Ключові слова: База даних, АСУ, ER-діаграма, Нормалізація, Комунальне підприємство, Облік, Оплата, Послуги, Тарифи, Абоненти, Водопостачання, Вивезення сміття.

ABSTRACT

Bachelor's thesis on the topic "Development of an automated management system for the subscriber department of a municipal enterprise on the .NET platform", written by Vasyl Vasylovych Muziko, a student of Ivan Pul'uj Ternopil National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, group SPs-41.

Summary of content: 74 pages, 26 figures, 9 tables, 5 appendices, 22 references.

Research object – the subscriber department of a municipal enterprise.

The purpose of this work is to design a database and create an information system for accounting for municipal services with the aim of automating the process of accounting for subscribers and services.

Research method – description of the subject area through analysis and processing of information on the task; database design using normalization methods; writing and debugging a program in the Visual Studio environment.

Results obtained – designed and developed an information system for accounting for municipal services connected to the database.

Keywords: Database, DMS, Information System, ER-diagram, Normalization, Municipal Enterprise, Accounting, Payment, Services, Subscribers.

ЗМІСТ

ВСТУП

На сьогоднішній день в кожному населеному пункті проживає велика кількість громадян, кожному з яких необхідно надати комунальні послуги. Через це виникає потреба в автоматизації процесу надання та керування цими послуги. Тому виникає велика необхідність у полегшенні зберігання та опрацювання великих обсягів інформації про абонентів та їх послуги.

Опрацювання великих масивів інформації у паперовому варіанті не ефективно та затратно у часі. Через це виникає необхідність систематизувати процес зберігання та обробки інформації про абонентів, показники лічильників, формування квитанції з можливостю їх експорту та друку, та іншу інформацію про оплату комунальних послуг.

Створення такої системи дозволить значно зменшити кількість роботи та збільшити ефективність обробки інформації для працівників комунального підприємства.

Задачею абонентського відділу комунального підприємства являється опрацювання інформації про абонентів, збір показів лічильників, відстеження заборгованостей за надані послуги.

Метою дипломної роботи є створення системи керування, що дозволить автоматизувати роботу абонентського відділу комунального підприємства.

Для опрацювання інформації про абонентів та послуг, була побудована відповідна база даних. Також, використані стандартні методи для опрацювання даних. Взаємодія між системою та базою даних реалізовано за допомогою SQL запитів.

В створеному проекті також присутні додаткові вікна, такі як інформаційне вікно з інформацією про програму, довідка, та вікно для управління правами користувачів, які мають доступ до програми.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Опис комунального підприємства

Для забезпечення чистоти та комфорту для жителів, а також стабільного функціонування інфраструктури, велику роль відіграють комунальні підприємства. Їхні основні обов’язки полягають у наданні важливих комунальних послуг, серед яких життєво необхідними являється водопостачання та вивезення сміття.

Комунальні підприємства являються юридичними особами з майном, власним бюджетом, рахунками в банківських установах, печаткою підприємства, кутовими та іншими штампами з їхньою назвою та ідентифікаційним кодом. Права юридичної особи такі підприємства набувають з моменту державної реєстрації згідно з законодавством України. Вони виступають у господарських, цивільних та адміністративних правовідносинах від свого імені, набувають майнові права, беруть на себе зобов'язання, а також можуть бути позивачами чи відповідачами в судах загальної юрисдикції, господарських та адміністративних судах, несучи відповідальність за результати своєї господарської діяльності.

Як юридичні особи, комунальні підприємства мають право господарського відання щодо закріпленого за ними майна, можуть укладати договори, набувати майнові та немайнові права, брати на себе зобов'язання, бути позивачами та відповідачами в судах, та займатися діяльністю, що відповідає напрямкам, визначеним їх статутом.

Прибуток комунальних підприємств, який залишається після покриття матеріальних витрат, видання зарплат та інших сплат, залишається в бюджеті підприємства та використовується керівником відповідно до законодавства.

Керівник комунального підприємства може діяти від лиця підприємства, він виступає в ролі його представника в різноманітних організаціях, підприємствах, установах, органах. Він самостійно вирішує важливі питання, які можуть виникнути в процесі діяльності підприємства [1].

Комунальні підприємства виконують різноманітні функції, які є критично важливими для життєдіяльності громад. Насамперед, вони забезпечують:

– обслуговування об’єктів комунальної власності;

– обслуговування каналізаційних систем;

– збирання та вивезення сміття;

– організація різноманітних виступів та конференцій для покращення громадського устрою;

– підготовку будівельних ділянок;

– будівництво та ремонт комунальної інфраструктури, такої як дороги, мости, водовідведення, системи опалення тощо.

– організацію поховань;

– збереження та охорона природних ресурсів, таких як ліси, парки, водойми та повітря;

– установлення інженерного устаткування будівель та споруд.

Невід’ємною частиною комунальних підприємств є абонентський відділ. Він здійснює зберігання та обробку інформації про абонентів, у тому числі адреси, показники лічильників та здійснення оплати за надані послуги, тарифи на послуги, а також борги і нарахування по кожному абоненту. Оплата за комунальні послуги абонента здійснюється за унікальним особовим рахунком.

Абонент може отримати інформацію про надану послугу від працівника абонентського відділу у вигляді квитанції, в якій міститься інформація про борг, нарахування, перерахунок нарахувань при зміні інформації по особових рахунках, по кожному виду послуг.

Головними функціями абонентського відділу є:

– внесення, зберігання та обробка інформації про абонентів, адрес та показників лічильника;

– формування квитанцій по наданих послугах;

– внесення інформації про здійснені оплати комунальних послуг.

1.2 Огляд подібних проектних рішень

В даний час існують кілька аналогічних проектних рішень, на які слід звернути увагу, а саме: автоматизована система «Комунальна програма» та галузева програма «Облік SaaS».

Пакет прикладних програм «Комунальна програма» включає безліч необхідних і корисних функцій. Програма може обробляти будь-який обсяг даних та параметрів, дозволяючи одному додатку ефективно вести облік на великому підприємстві та у всіх його філіях. Завдяки сучасним технологіям, дані та параметри, які збирає та обробляє людина, можуть автоматично передаватися за допомогою інтернету, що дозволяє автоматизувати процес зібрання даних. Крім того, система забезпечує автоматичний розрахунок нарахувань за нормами споживання, кількістю проживаючих осіб або площею. При зміні тарифів програмне забезпечення самостійно перерахує всі дані за кілька хвилин і надішле нові платіжки споживачам електронною поштою [2].

Вигляд головної сторінки проектного рішення «Комунальна програма» подано на рисунку 1.1.

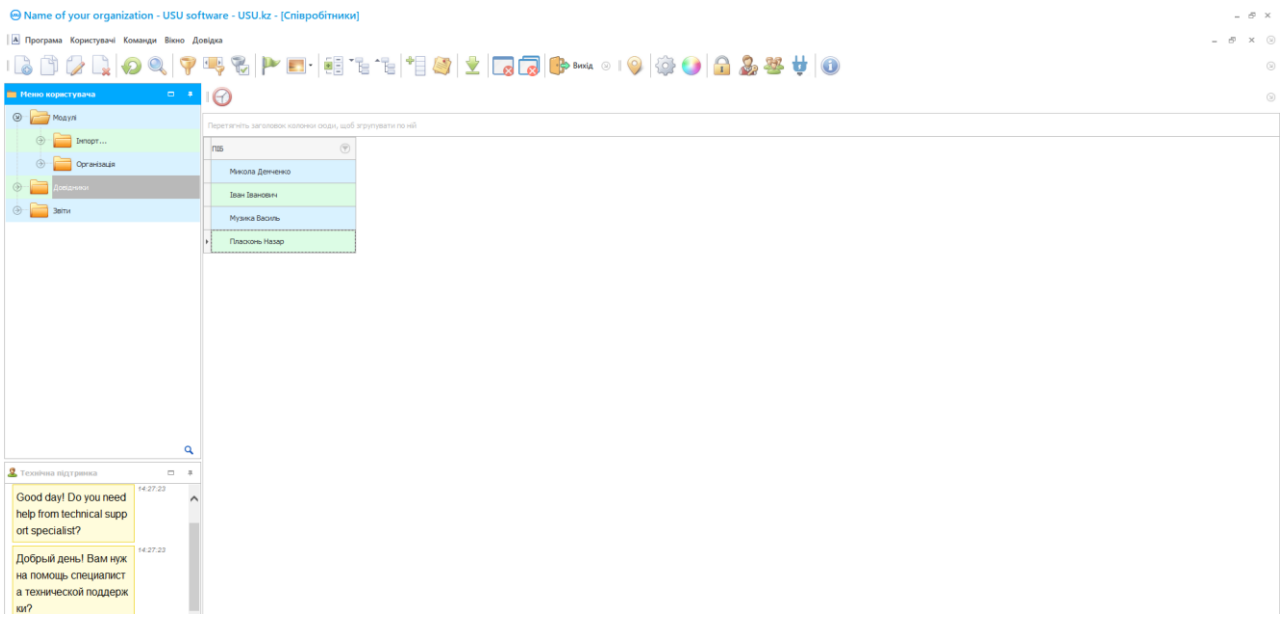


Рисунок 1.1 – Вигляд програми «Комунальна програма»

При більш детальному розгляді програмного продукту «Комунальна програма» визначимо перелік його переваг та недоліків, основними серед яких являється:

– високий рівень захисту інформації;

– присутність функції запровадження документації;

– використання сучасних засобів надсилання важливих повідомлень за допомогою sms, e-mail, viber;

– автоматизація процесу розрахунку нарахувань за поданими даними;

– легке встановлення програмного забезпечення.

Однак, це рішення також має недоліки, основним яким являється його вартість, а саме 20 000 грн. Також слід звернути увагу, що через великий функціонал програмного продукту, користувачу буде важко орієнтуватися без досвіду роботи в ньому.

В конфігурації проектного рішення «Облік SaaS» присутня спеціалізована функціональність, що враховує особливості житлово-комунальних господарств та об’єднань співвласників багатоквартирних будинків. Серед великої кількості функцій, слід звернути увагу на основні можливості:

– ведення переліку мешканців та розрахунки з ними за комунальні платежі;

– ведення переліку показів лічильників, тарифів по кожній наданій послузі, а також розрахунок вартості кожної послуги та формування рахунків на оплату за комунальні платежі;

– облік субсидій та пільг, що надаються мешканцям згідно законодавства, а також проведення взаєморозрахунків з бюджетом по компенсації цих пільг і субсидій;

– під час імпорту платежів від мешканців, присутня можливість автоматичного визначення платника за його особовим рахунком.

Вигляд проектного рішення «Облік Saas» зображено на рисунку 1.2.

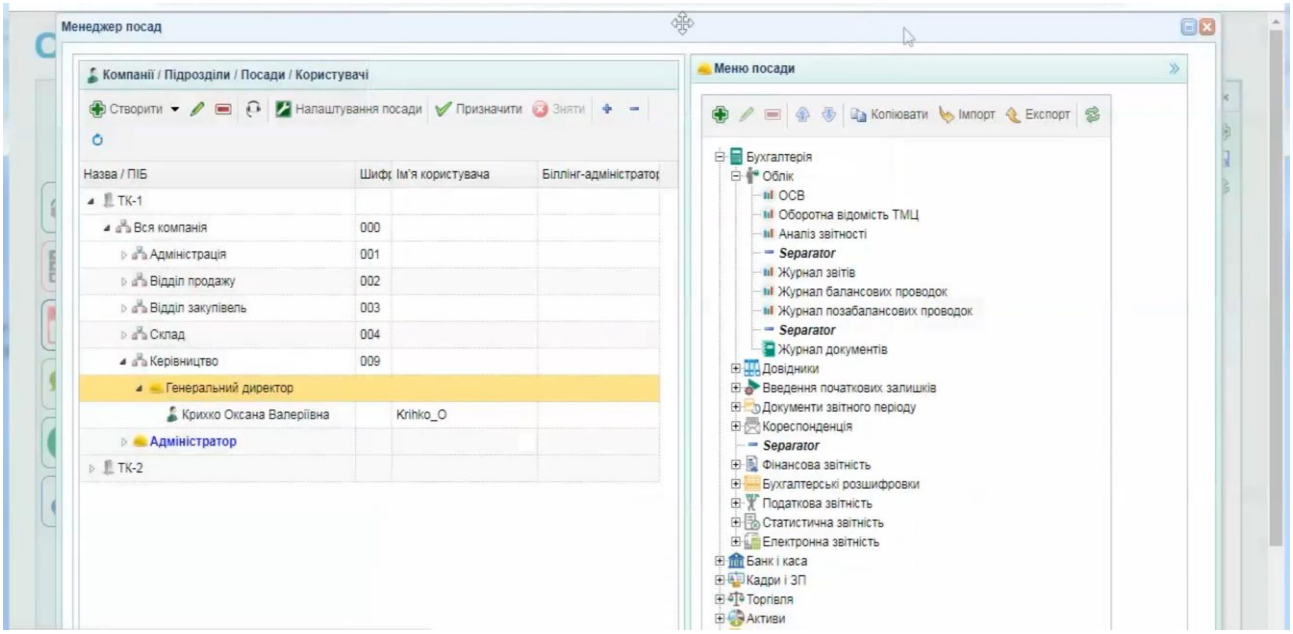


Рисунок 1.2 – Вигляд проектного рішення «Облік Saas»

До недоліків цього проектного рішення входить потреба у підключенні до інтернету, велика вартість та обмеження в кількості користувачів, які можуть отримати доступ до програми.

Проведений аналіз подібних проектних рішень дозволив виявити усі сильні та слабкі сторони продемонстрованих програмних продуктів. З поданих даних можна зробити висновок, що подібні проектні рішення мають свої як переваги, так і недоліки, а також, що головним недоліком є висока вартість програмного забезпечення.

Отже, у власному проектному рішенні буде створена автоматизована система управління абонентським відділом для малого комунального підприємства із простим функціоналом та невеликою конкурентною ціною.

2 ПРОЕКТНА ЧАСТИНА

2.1 Постановка завдання

Головним завданням дипломної роботи являється створення для системи управління абонентами та послугами, що дозволить полегшити та автоматизувати роботу з інформацією. Для цього було використано мову програмування C# та побудована база даних у системі керування базою даних MS SQL Server [3]. Взаємодія програми з базою даних відбувається за допомогою SQL запитів [4].

Створене проектне рішення повинно містити наступний функціонал:

– заставка;
– вікно, яке містить інформацію про програму;
– перегляд, додавання, редагування та видалення інформації в таблицях;
– автоматичний підрахунок необхідних даних в таблицях;
– пошук абонента за номером абонента, прізвищем, номером телефону;
– пошук адреси абонента за номером абонента, вулицею, особовим рахунком та прізвищем;
– формування квитанцій в Excel файлі;
– створення резервної копії бази даних;
– виведення повідомлень про помилки.

2.2 Проектування бази даних

Комунальне підприємство здійснює процес обліку абонентів, адрес, інформації про надані абонентам послуг по водопостачанню та вивезенню сміття.

Абоненти мають унікальний особовий рахунок, за допомогу яких вони можуть здійснювати оплату за надані комунальні послуги.

Розрахунок за послуги здійснюється наступним чином: в кожного абонента є послуги, які він отримує від комунального підприємства.

Комунальне підприємство надає абонентам послуги, за які, необхідно сплачувати кошти. Розрахунок суми до сплати за кожну надану послугу здійснює абонентський відділ. Оплата за послугу «Вивезення сміття», здійснюється наступним чином: кількість мешканців, що проживають за адресом, множиться на тариф. За послугу «Водопостачання» розрахунок проводиться за формулою: значення показника лічильника за поточний місяць віднімається від показника за попередній місяць, та отриману різницю множиться на тариф.

Бухгалтер проводить облік усіх абонентів, адрес, послуг, тарифів на послуги, а також боргів, та нарахувань по кожному абоненту. Абонент отримує інформацію про надані послуги у вигляді квитанції, в якій містяться дані про надану послугу, тариф, суми до сплати, борг, та період за який була надана послуга.

Оплату комунальних послуг приймає касир або секретар, в залежності від призначених обов'язків на комунальному підприємстві. Даній особі необхідно бачити інформацію про абонента і його баланс на сьогоднішній день, щоб прийняти оплату конкретного абонента.

Отже, сформулюємо основні операції бази даних:

– введення даних про абонентів, адреси, види послуг, показники, квитанції, оплати;

– введення даних про користувачів, які отримають доступ до програми;

– пошук абонентів за номером абонента, особовим рахунком, прізвищем, номером телефону;

– пошук адрес за номером абонента, особовим рахунком, прізвищем, вулицею;

– виведення інформації про житло шуканого абонента.

Після аналізу предметної області опишемо основні сутності бази даних, їх псевдоніми та особливості їх використання у таблиці 2.1.

Таблиця 2.1 – Опис сутностей бази даних

Ім’я сутності	Опис	Псевдоніми	Особливості використання
1	2	3	4
Користувачі	Користувачі, які мають доступ до програми	TableUser	Обов’язково потрібно вказати логін та пароль
Абоненти	Перелік усіх абонентів	TableAbonentu	Обов’язково потрібно вказати ПІБ та номер телефону
Адреса	Зберігаються дані про адреси абонентів	TableAdresa	Обов’язково потрібно вказати інформацію про житло
Види послуг	Містить види послуг комунального підприємства	TablePoslugu	Вказується назва послуги
Показники	Зберігаються показники лічильників абонентів	TablePokaznuku	Вказується показник лічильника, дата та період
Оплата	Зберігається інформація про суму нарахування абонентів	TableOplata	Обов’язково вказується період та дата
Квитанції на водопостачання	Зберігаються готові квитанції за воду для абонентів	TableKvVoda	Автоматично формується за допомогою зв’язків із іншими таблицями
Квитанції на вивіз сміття	Зберігаються готові квитанції за воду для абонентів	TableKvSm	Автоматично формується за допомогою зв’язків із іншими таблицями

Тепер розглянемо більш детальний опис атрибутів сутностей, їх доменів та властивостей бази даних «oblik_poslug», який наведено у таблиці 2.2.

Таблиця 2.2 – Опис атрибутів сутностей, їх доменів та властивостей бази даних «oblik_poslug»

Тип сутності	Атрибут	Опис	Тип даних, множина	Обмеження	Значення за замовчуванням	Псевдонім	Значення Null
1	2	3	4	5	6	7	8
Користувачі	id	Унікальний код користувача	Числовий	Первинний ключ	–	–	Ні
	Ulogin	Логін для входу	Текстовий	–	–	–	Ні
	Upassword	Пароль користувача	Текстовий	–	–	–	Ні
	UPosada	Посада	Текстовий	–	–	–	Ні

Продовження таблиці 2.2

1	2	3	4	5	6	7	8
Абоненти	Код	Унікальний код абонента	Числовий	Первинний ключ	–	–	Ні
	Прізвище	Прізвище	Текстовий	–	–	–	Ні
	Ім’я	Ім’я	Текстовий	–	–	–	Ні
	По батькові	По батькові	Текстовий	–	–	–	Ні
	Телефон	Номер телефона	Числовий	–	–	–	Ні
Адреса	Код	Унікальний код абонента	Числовий	Первинний ключ	–	–	Ні
	Населений пункт	Населений пункт	Текстовий	–	–	–	Ні
	Вулиця	Назва вулиці абонента	Текстовий	–	–	–	Ні
	Будинок	Номер будинку абонента	Числовий	–	–	–	Так
	Квартира	Номер квартири абонента	Числовий	–	–	–	Так
	Кількість жильців	Кількість мешканців в житлі	Числовий	–	–	–	Ні
	Площа	Площа житла (квадратних метрів)	Числовий	–	–	–	Ні
	Власник	Код власника житла	Числовий	Зовнішній ключ	–	–	Ні
Послуги	Код	Унікальний код послуги	Числовий	Первинний ключ	–	–	Ні
	Тариф	Тариф за послугу	Числовий	–	–	–	Ні
	Назва	Назва послуги	Текстовий	–	–	–	Ні

Продовження таблиці 2.2

1	2	3	4	5	6	7	8
Показники	Код	Унікальний код показника лічильника	Числовий	Первинний ключ	–	–	Ні
	Показник	Значення показника лічильника	Числовий	–	–	–	Ні
	Дата	Дата вказання показника лічильника	Дата	–	–	–	Ні
	Період	Період за який вказаний показник лічильника	Текстовий	–	–	–	Ні
	Адреса	Адреса звідки вказаний показник лічильника	Числовий	Зовнішній ключ	–	–	Ні
Оплата	Код	Унікальний код абонента	Числовий	Первинний ключ	–	–	Ні
	Адреса	Адреса по якій проводиться оплата	Числовий	–	–	–	Ні
	Період	Період за який проводиться оплата	Текстовий	–	–	–	Ні
	Дата	Дата здійснення оплати за послугу	Дата	–	–	–	Ні
	До оплати	Сума до оплати	Числовий	–	–	–	Ні
	Вид послуги	Вид послуги за який проводиться оплата	Текстовий	Зовнішній ключ	–	–	Ні
	Борг	Заборгованість абонента	Числовий	–	–	–	Ні

Продовження таблиці 2.2

1	2	3	4	5	6	7	8
Квитанція на водопостачання	Код	Унікальний код	Числовий	Первинний ключ	–	–	Ні
	Прізвище	Прізвище абонента	Текстовий	–	–	–	Ні
	Ім’я	Ім’я абонента	Текстовий	–	–	–	Ні
	По батькові	По батькові абонента	Текстовий	–	–	–	Ні
	Адреса	Адреса абонента	Числовий	–	–	–	Ні
	Період	Період за який оплачується	Текстовий	–	–	–	Ні
	Показник лічильника (поточний)	Значення показника лічильника за поточний місяць	Числовий	–	–	–	Ні
	Показник лічильника (попередній)	Значення показника лічильника за попередній місяць	Числовий	–	–	–	Ні
	Різниця	Різниця показників	Числовий	–	–	–	Ні
	Тариф	Тариф на водопостачання	Числовий	–	–	–	Ні
	До оплати	Сума до оплати	Числовий	–	–	–	Ні
Квитанція на вивіз сміття	Код	Унікальний код абонента	Числовий	Первинний ключ	–	–	Ні
	Прізвище	Прізвище абонента	Текстовий	–	–	–	Ні
	Ім’я	Ім’я абонента	Текстовий	–	–	–	Ні
	По батькові	По батькові абонента	Текстовий	–	–	–	Ні
	Адреса	Адреса абонента	Числовий	–	–	–	Ні
	Період	Період оплати	Текстовий	–	–	–	Ні
	Кількість жильців	Кількість жильців в житлі	Числовий	–	–	–	Ні
	До оплати	Сума до оплати	Числовий	–	–	–	Ні

Для того, щоб почати створення сутностей бази даних, спочатку потрібно визначити атрибути, які будуть виступати в ролі ідентифікатора. В базі даних в якості ідентифікатора виступає так званий «первинний ключ» [5].

Отже, наступні атрибути будуть виступати в ролі первинних ключів:

– Користувачі –> Код;
– Абоненти –> Код;
– Адреса –> Код;
– Види послуг –> Код;
– Показники –> Код;
– Оплата –> Код;
– Квитанція на водопостачання –> Код;
– Квитанція на вивіз сміття –> Код.

Відношення між сутностями бази даних наведено в таблиці 2.3. ER-діаграма наведена у додатку А.

Таблиця 2.3 – Опис зв’язків між сутностями бази даних «oblik_poslug»

Ім’я сутності 1	Ім’я сутності 2	Ім’я зв’язку	Тип зв’язку
Абоненти	Адреса	Має	1:N
Вид послуги	Оплата	Має	1:N
Адреса	Оплата	Має	1:N
Адреса	Показники	Містить	1:N
Адреса	Квитанція на водопостачання	Містить	1:N
Адреса	Квитанція на вивіз сміття	Містить	1:N

Проаналізувавши вимоги користувачів, приступаємо до проектування логічної моделі даних. Спочатку потрібно перетворити концептуальну модель у логічну. Перш за все, ми визначаємо відношення, які несумісні з реляційною моделлю. В цей момент проводяться операції вилучення складних зв'язків, в тому числі вилучення зв'язків "багато-до-багатьох" та множинних атрибутів [6].

В результаті аналізу відношень між таблицями, в базі даних не знайдено складних зв’язків або типу «багато-до-багатьох».

Отже, можна зробити висновок, що концептуальна модель даних успішно перетворена у логічну.

Нормалізація відношень – це процес поетапного розбиття вихідних відношень бази даних на більш прості. На цьому етапі схема відношень бази даних перетворюється в послідовні нормальні форми, кожна з яких має кращі властивості, ніж попередня. Відношення відповідає першій нормальній формі, якщо кожне значення в кожному стовпці і рядку є елементарним, тобто не містить групових або повторюваних значень [7].

Усі сутності мають прості, неподільні атрибути. Це означає, що процес приведення відношень до першої нормальної форми завершено.

Відношення перебуває у другій нормальній формі, якщо виконуються обмеження першої нормальної форми, та всі описові атрибути функціонально залежать від первинного ключа.

Атрибути відношень повністю задовольняють функціональну і логічну залежність від первинного ключа. У результаті логічна модель переходить у другу нормальну форму.

Відношення знаходиться у третій нормальній формі, якщо кожен описовий атрибут є взаємно незалежним від інших описових атрибутів і повністю залежить від первинного ключа. Це означає, що ні один описовий атрибут не залежить від ключа та всі описові атрибути функціонально повністю визначені за значенням первинного ключа і не взаємно залежать між собою [8].

Проаналізувавши відношення бази даних, можна зробити висновок, що вони будуть нормалізованими до третьої нормальної форми.

Проектування бази даних потрібно починати із створення структури, тобто переліку таблиць та їх стовпців, а також побудови зв’язків між таблицями.

Розпочнемо створення таблиці «TableAbonentu», яка буде містити інформацію про абонентів. Для цього виконуємо запит, вигляд якого наведено на рисунку 2.1.

```
use [oblik_poslug]
create table TableAbonentu (
id int not null primary key,
prizvuwe varchar(30) not null,
imya varchar(25) not null,
pobotakovi varchar(25) not null,
phone varchar(25) not null)
```

Рисунок 2.1 - Код SQL-запиту для створення таблиці «TableAbonentu» в базі даних «oblik_poslug»

Далі створюємо таблицю «TableAdresa», для чого виконаємо SQL-запит, який зображено на рисунку 2.2.

```
use [oblik_poslug]
create table TableAdresa (
id int not null primary key,
pynkt varchar(25) not null,
streetname varchar(25) not null,
bydunok int,
kvartura int,
kilk_osob int not null,
plowa int not null,
vlasnuk int not null,
Foreign key (vlasnuk)
References TableAbonentu(id))
```

Рисунок 2.2 - Код SQL-запиту для створення таблиці «TableAdresa» в базі даних «oblik_poslug»

Наступним кроком буде створення таблиці «TableOplata», як зображено на рисунку 2.3.

```
use [oblik_poslug]
create table TableOplata (
id int not null primary key,
adresa int not null,
Foreign Key (adresa) References TableAdresa (id),
period varchar(25) not null,
data date not null,
do_oplatu float not null,
vud_poslugu int not null,
Foreign Key (vud_poslugu) References TablePoslugu (id),
borg float
not null)
```

Рисунок 2.3 - Код SQL-запиту для створення таблиці «TableOplata» в базі даних «oblik_poslug»

Для створення таблиці «TablePoslugu» виконуємо запит, вигляд якого зображено на рисунку 2.4.

```
use [oblik_poslug]
create table TablePoslugu (
id int not null primary key,
name varchar(35)
not  null)
```

Рисунок 2.4 - Код SQL-запиту для створення таблиці «TablePoslugu» в базі даних «oblik_poslug»

Таблицю «TableUser» створюємо за допомогою запиту зображеного на рисунку 2.5.

```
use [oblik_poslug]
create table TableUser (
id int not null primary key,
Ulogin varchar(35) not  null,
UPassword varchar(35) not null,
UPosada varchar(35) not  null)
```

Рисунок 2.5 - Код SQL-запиту для створення таблиці «TableUser» в базі даних «oblik_poslug»

Далі створюємо таблицю «TablePokaznuku» за допомогою SQL-запиту, зображеного на рисунку 2.6.

```
use [oblik_poslug]
create table TablePokaznuku (
id int not null primary key,
pokaznuk bigint not null,
data date not null,
period varchar(25) not null,
adresa int not null,
Foreign Key (adresa) References TableAdresa (id))
```

Рисунок 2.6 - Код SQL-запиту для створення таблиці «TablePokaznuku» в базі даних «oblik_poslug»

Після цього створюємо таблицю «TableKvVoda» за допомогою SQL-запиту, вигляд якого зображено на рисунку 2.7.

```
use [oblik_poslug]
create table TableKvVoda (
id int not null primary key,
prizvuwe varchar(30) not null,
imya varchar(25) not null,
pobatkovi varchar(25) not null,
adresa int not null,
Foreign Key (adresa) References TableAdresa (id),
period varchar(25) not null,
pokaznukzaraz bigint not null,
pokaznukpoperedniy bigint not null,
riznuca float not null,
taruf float not null,
do_oplatu float not null,
borg float not null)
```

Рисунок 2.7 - Код SQL-запиту для створення таблиці «TableKvVoda» в базі даних «oblik_poslug»

Для створення таблиці «TableKvSm» виконуємо наступний запит, вигляд якого зображено на рисунку 2.8.

```
use [oblik_poslug]
create table TableKvSm (
id int not null primary key,
prizvuwe varchar(30) not null,
imya varchar(25) not null,
pobatkovi varchar(25) not null,
adresa int not null,
Foreign Key (adresa) References TableAdresa (id),
period varchar(25) not null,
taruf int not null,
kilk_osob int not null,
do_oplatu float not null)
```

Рисунок 2.8 - Код SQL-запиту для створення таблиці «TableKvSm» в базі даних «oblik_poslug»

Таким чином, за допомогою наведених SQL-запитів створюються необхідні таблиці, які в подальшому будуть використовуватися для зберігання даних.

Після створення таблиць побудуємо діаграму зв’язків між таблицями в базі даних «oblik_poslug» (див. Дод. Б).

2.3 Проектування інформаційної системи

Важливим кроком для створення програми є проектування. Для цього використовується уніфікована мова моделювання – UML (Unified Modeling Language).

Для розуміння процесу роботи системи використовується опис функцій системи через варіанти використання (прецеденти або ж UseCase).

Варіанти використання – це інструкція для виконання дій, що робить система, коли користувачі або інші програмні системи взаємодіють з нею.

Для того щоб спроектувати систему управління, необхідно побудувати наступні діаграми:

- діаграма використання (Use case);
- діаграма класів (Class diagram);
- діаграма послідовності (Sequence diagram).

Діаграма використання – це інструмент, який показує, як користувачі взаємодіють із системою. Вона ілюструє різні сценарії, в яких користувачі використовують функції програмного забезпечення для досягнення певних цілей. Така діаграма допомагає зрозуміти вимоги до системи і спланувати її функціональні можливості. Кожен варіант використання визначає послідовність дій, які повинні бути виконані спроектованою системою при взаємодії її з відповідним актором [9].

Діаграма використання для системи управління абонентським відділом комунального підприємства наведена на рисунку 2.9.

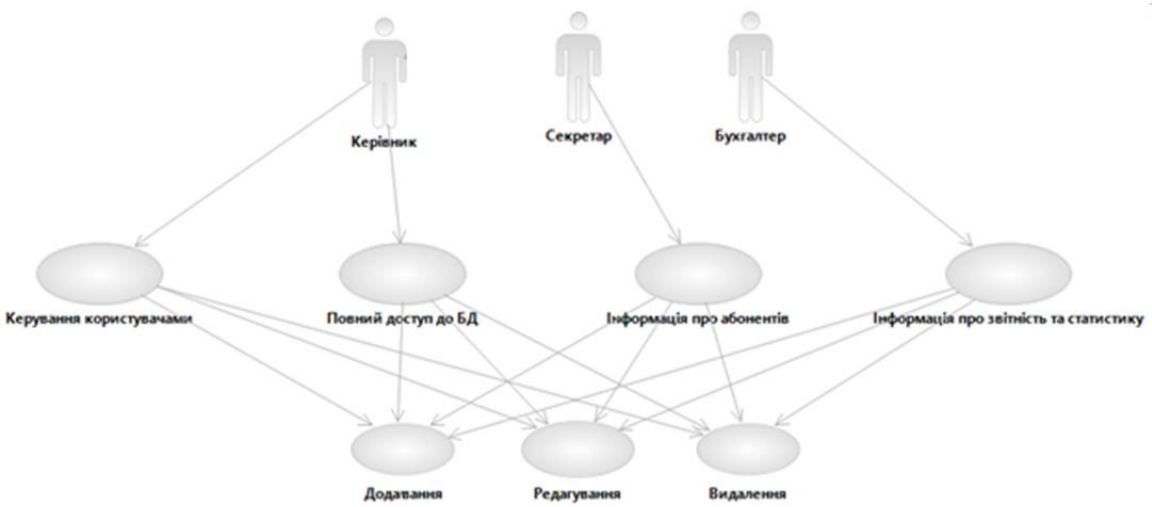


Рисунок 2.9 – Діаграма використання для обліку комунальних послуг

Діаграма класів – це представлення таблиць, їх атрибутів та відношень між ними.

Діаграма представляє собою набір символів класів, відомих як іконки, та зв'язків між ними. Іконка – це стандартизоване візуальне зображення певного поняття, яке має фіксовану форму та легко розпізнається. Іконка класу виглядає як прямокутник, який може бути розділений на дві або три частини. Верхня частина є обов'язковою і містить назву класу. Друга і третя частини, які можуть бути відсутні, містять: друга – список атрибутів класу, третя – список операцій класу. Іншими словами, діаграма класів може відображати лише назви класів, або назви разом з атрибутами, або назви, атрибути та операції класів [10].

Діаграма класів для програми обліку комунальних послуг наведена на рисунку 2.10.

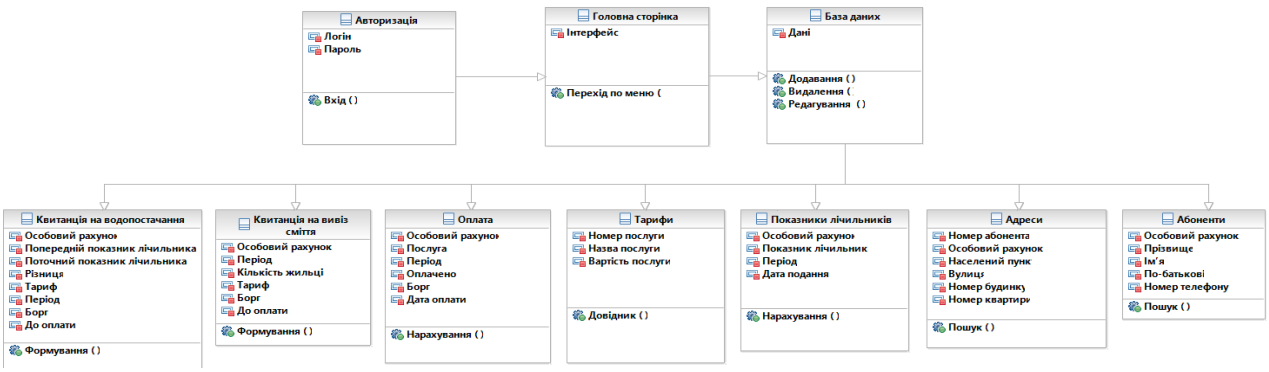


Рисунок 2.10 – Діаграма класів для програми обліку комунальних послуг

Діаграма взаємодії є ефективним засобом для подання поведінки об'єктів у сценаріях. У цій нотації кожному об'єкту сценарію відповідає лінія життя, яка ілюструє його активність та взаємодію з іншими об'єктами протягом часу. Це дозволяє розуміти, як об'єкти взаємодіють між собою та як змінюється їх стан протягом виконання конкретного сценарію.

На діаграмі цей елемент зображується вертикальною пунктирною лінією, на верхівці якої в прямокутнику вказується назва об'єкта між моментами його створення та знищення. Крайня права лінія представляє зовнішнє середовище, з яким взаємодіє сценарій, тобто його зовнішній інтерфейс [10].

Діаграма послідовностей для системи управління абонентським відділом комунального підприємства наведена на рисунку 2.11.

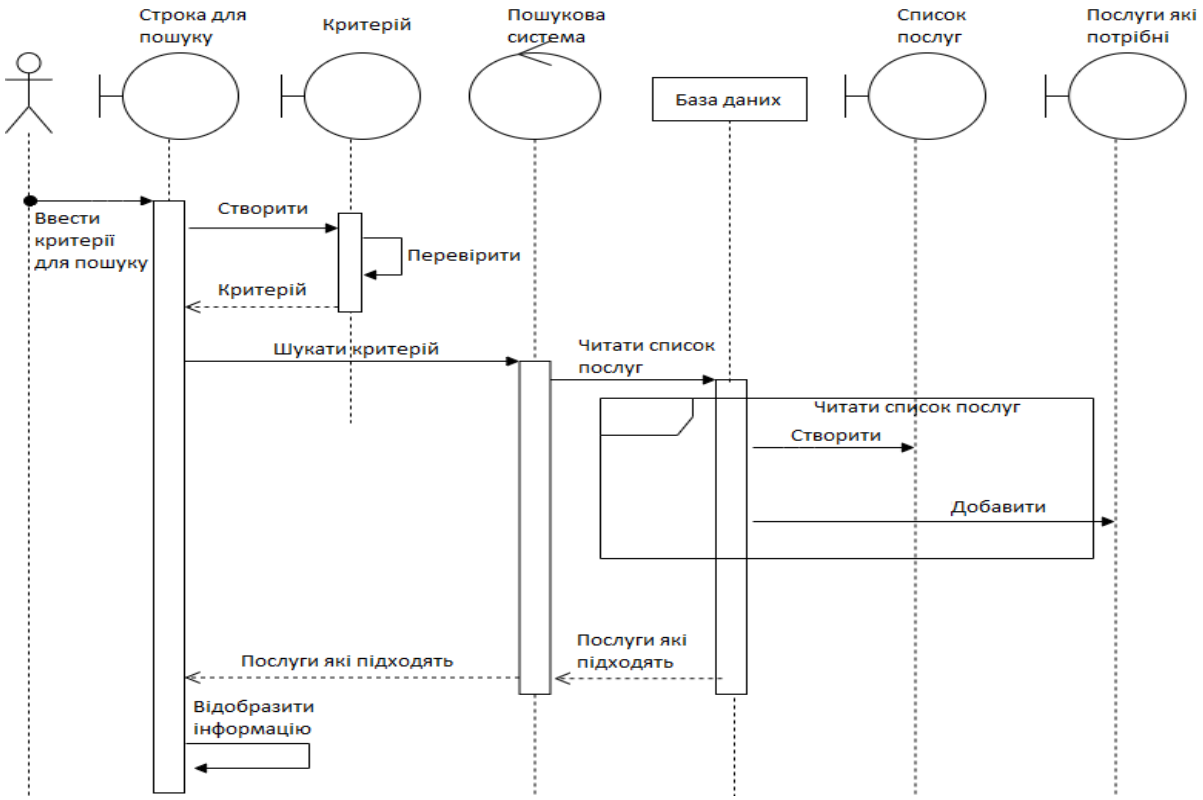


Рисунок 2.11 – Діаграма послідовності для програми обліку комунальних послуг

На зображеній вище діаграмі можна побачити взаємодію об'єктів між собою протягом часу роботи. Це наглядно демонструє взаємодію об'єктів між собою, та як саме з часом змінюється їх стан.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Проектування інтерфейсу користувача

Після запуску програму з’являється вікно заставки (див. Рис. 3.1).

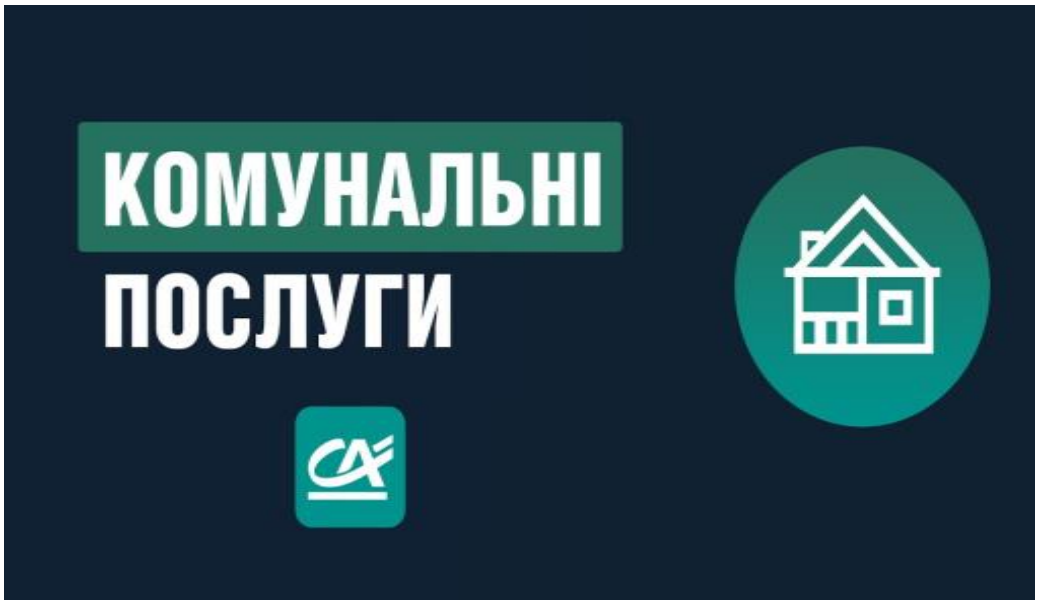


Рисунок 3.1 – Заставка програми

Вікно із заставкою плавно ставновить прозорим та зникає, а на його місця появляється вікн авторизації. Вигляд вікна наведено на рисунку 3.2.

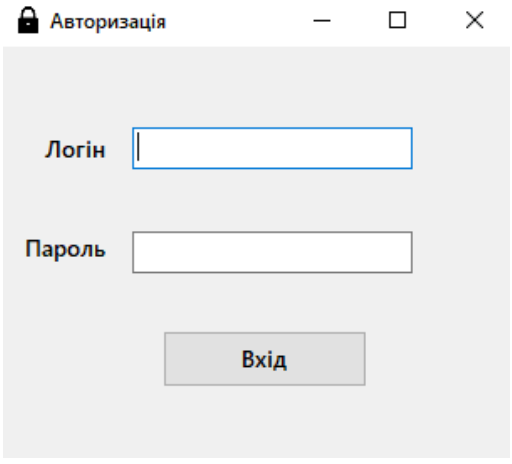


Рисунок 3.2 – Вигляд форми авторизації

Якщо користувач пройшов процес авторизації, з’являється головне вікно програми із переліком абонентів (див. Рис. 3.3).

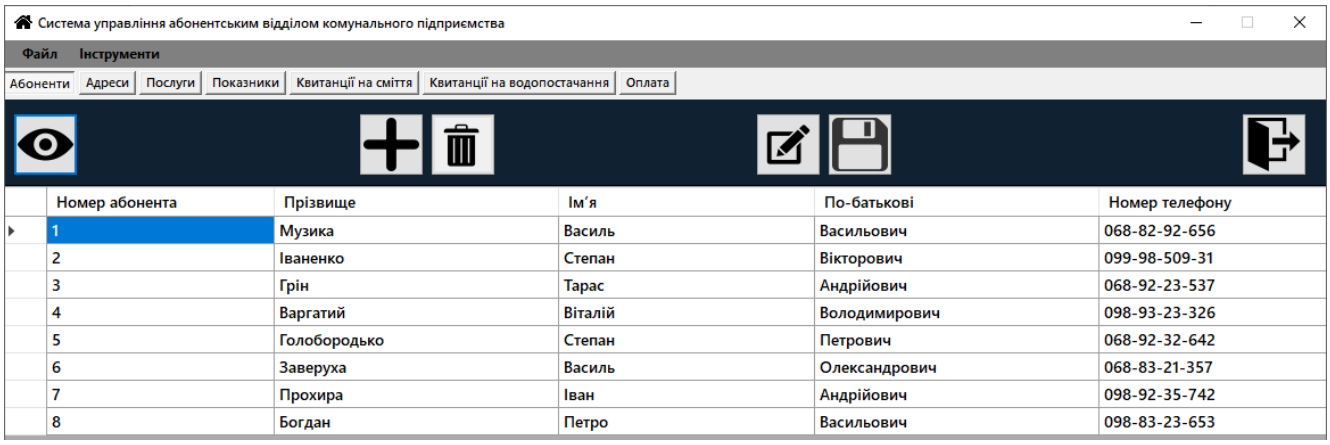


Рисунок 3.3 – Вигляд вкладки «Абоненти»

Над таблицею з даними знаходяться кнопки керування (зліва направо): Показати, Додати, Видалити, Редагування, Зберегти, Вихід. Кнопки позначені стандартними іконками, що робить інтерфейс програми простим та зрозумілим.

При натисканні на кнопку Додати, з’являється вікно для добавлення нового абонента (див. Рис. 3.4). Програмний код опрацювання головних подій цього та інших вікон наведено у Додатку В.

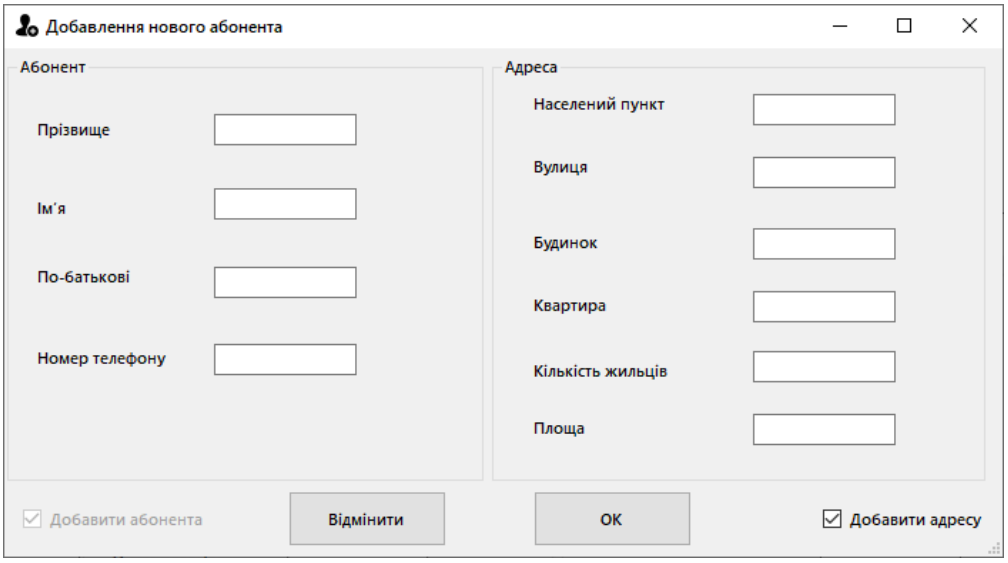


Рисунок 3.4 – Вигляд вікна добавлення нового абонента

Також, в вікні добавлення абонента присутня можливість при потребі добавити його адресу.

Інформація про адреси абонентів знаходиться в вкладці «Адреси», вигляд якого зображено на рисунку 3.5.

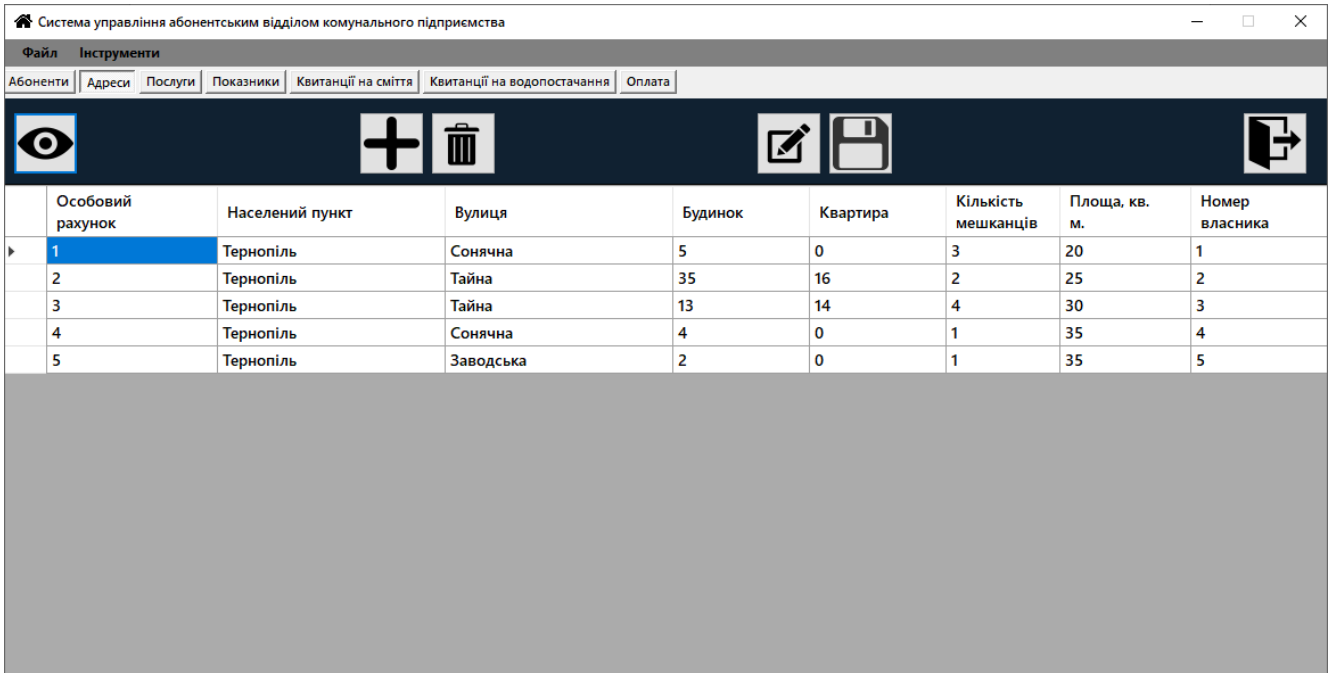


Рисунок 3.5 – Вигляд вкладки «Адреси»

Інформація про послуги, які надає комунальне підприємство знаходиться в вкладці «Послуги» (див. Рис. 3.6).

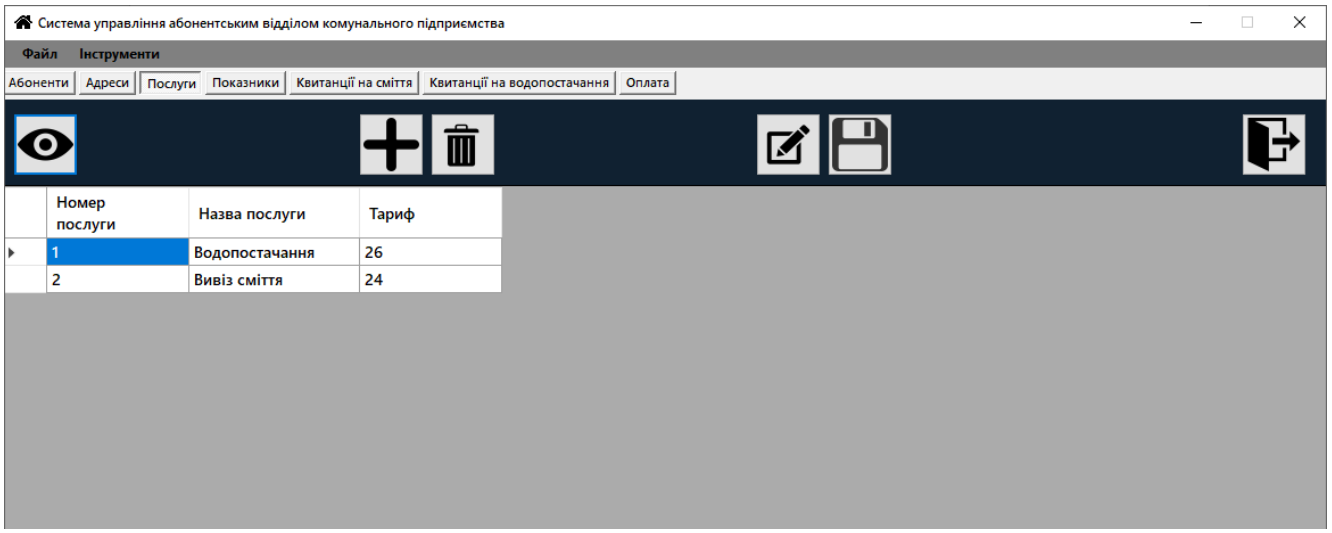
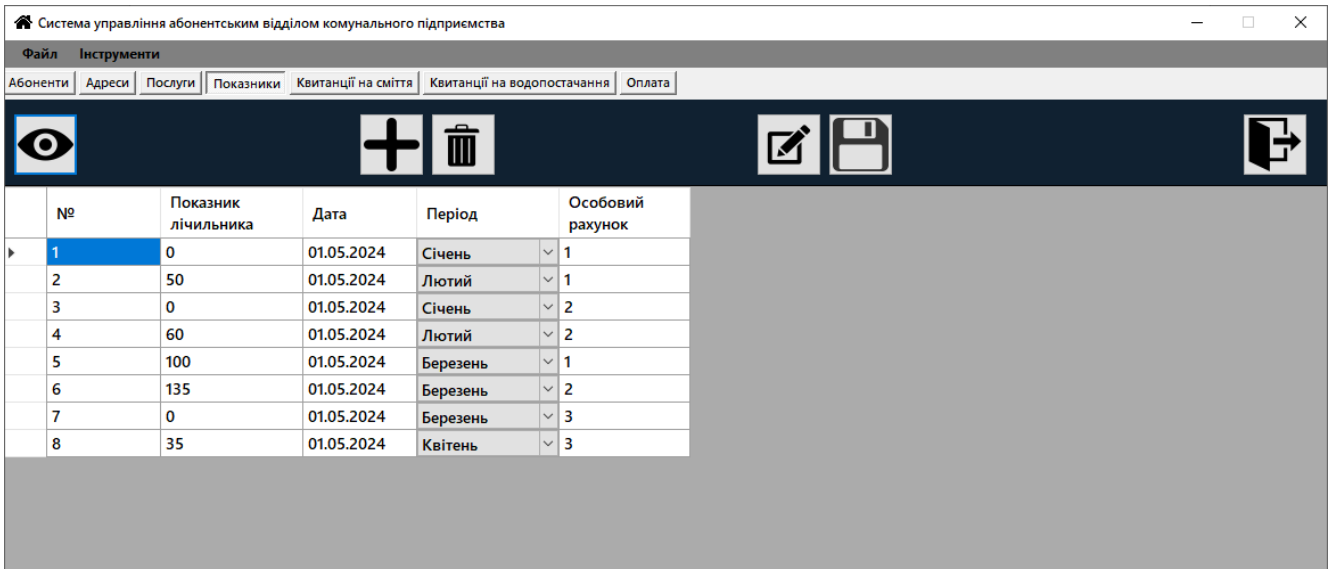


Рисунок 3.6 – Вигляд вкладки «Послуги»

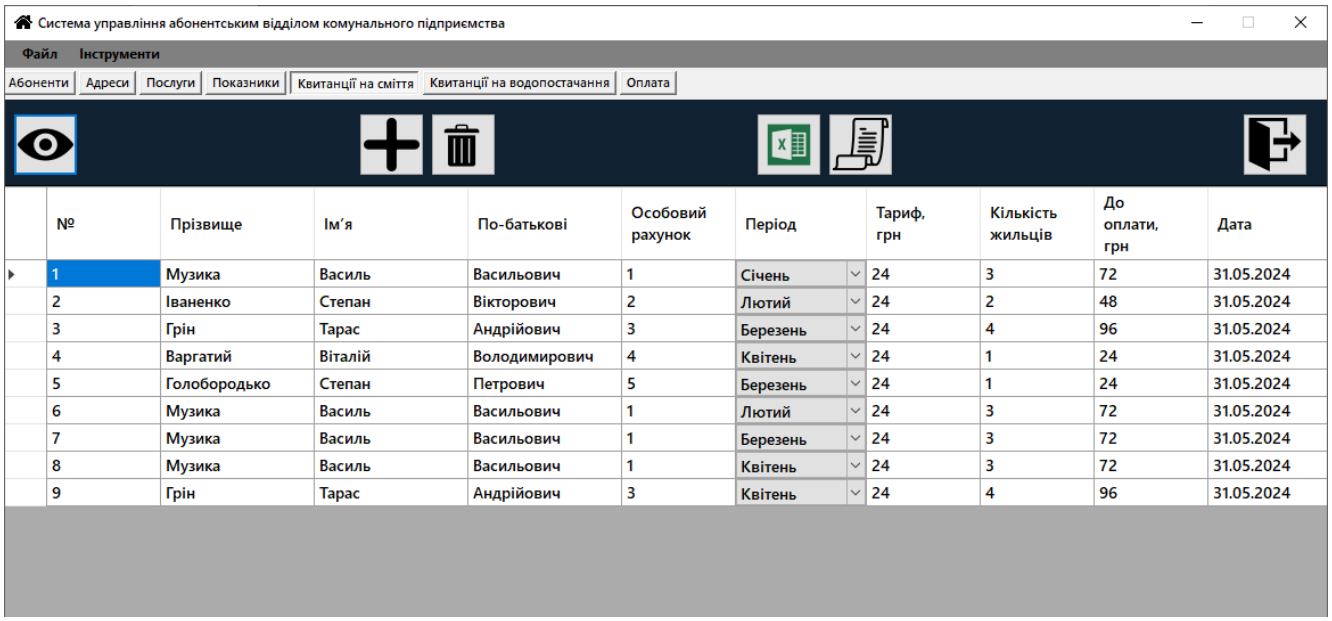
Для послуги «Водопостачання» необхідно вносити дані про показники лічильника, які зберігаються у вкладці «Показники» (див. Рис. 3.7).



№	Показник лічильника	Дата	Період	Особовий рахунок
1	0	01.05.2024	Січень	1
2	50	01.05.2024	Лютий	1
3	0	01.05.2024	Січень	2
4	60	01.05.2024	Лютий	2
5	100	01.05.2024	Березень	1
6	135	01.05.2024	Березень	2
7	0	01.05.2024	Березень	3
8	35	01.05.2024	Квітень	3

Рисунок 3.7 – Вигляд вкладки «Показники»

На вкладці «Квитанції на сміття» здійснюється керування, та формування квитанцій на вивезення сміття. Також, в цьому розділі присутня функція експорту даних в файл Microsoft Excel (див. Рис. 3.8). При натисканні на кнопку добавлення квитанції з’являється вікно для введення необхідних даних (див. Рис. 3.9).



№	Прізвище	Ім’я	По-батькові	Особовий рахунок	Період	Тариф, грн	Кількість жильців	До оплати, грн	Дата
1	Музика	Василь	Васильович	1	Січень	24	3	72	31.05.2024
2	Іваненко	Степан	Вікторович	2	Лютий	24	2	48	31.05.2024
3	Грін	Тарас	Андрійович	3	Березень	24	4	96	31.05.2024
4	Варгатий	Віталій	Володимирович	4	Квітень	24	1	24	31.05.2024
5	Голобородько	Степан	Петрович	5	Березень	24	1	24	31.05.2024
6	Музика	Василь	Васильович	1	Лютий	24	3	72	31.05.2024
7	Музика	Василь	Васильович	1	Березень	24	3	72	31.05.2024
8	Музика	Василь	Васильович	1	Квітень	24	3	72	31.05.2024
9	Грін	Тарас	Андрійович	3	Квітень	24	4	96	31.05.2024

Рисунок 3.8 – Вигляд вкладки «Квитанції на сміття»

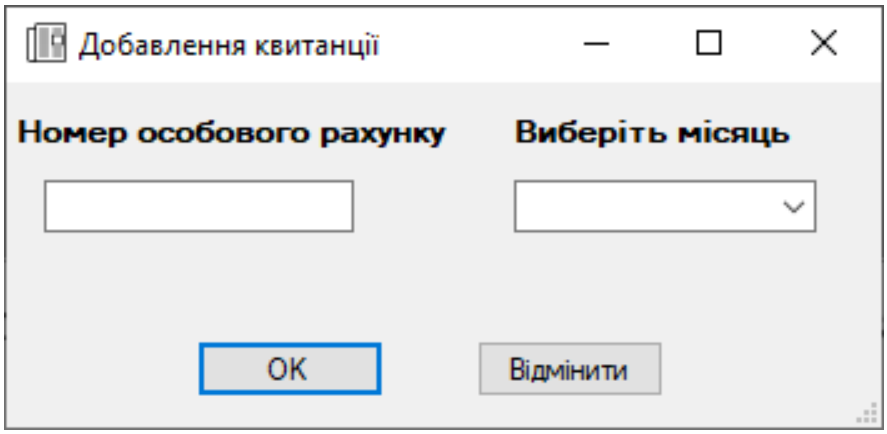
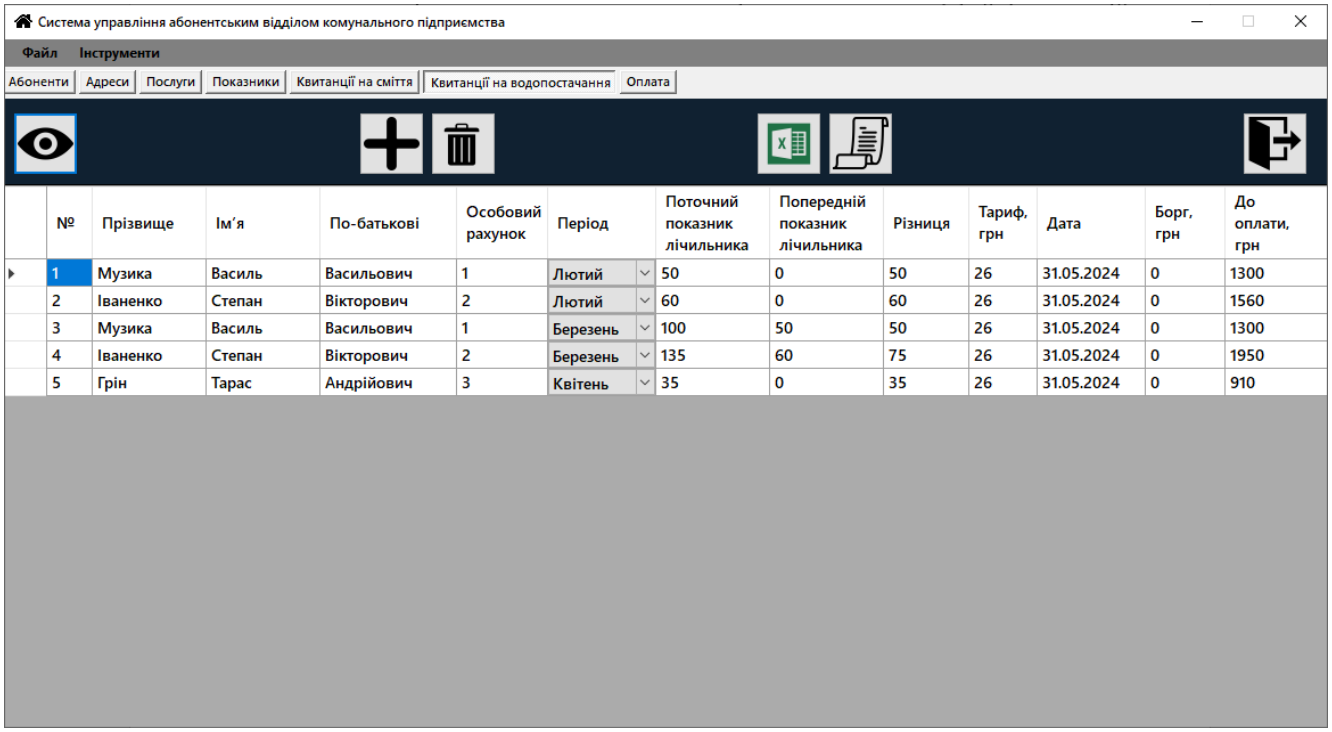


Рисунок 3.9 – Вигляд вікна формування квитанції на вивезення сміття

Вкладка "Квитанції водопостачання" дозволяє додавати, видаляти та формувати квитанції в файлі Microsoft Excel (див. Рис. 3.10). Зображення вікна формування квитанції можна побачити на рисунку 3.11.



№	Прізвище	Ім'я	По-батькові	Особовий рахунок	Період	Поточний показник лічильника	Попередній показник лічильника	Різниця	Тариф, грн	Дата	Борг, грн	До оплати, грн
1	Музика	Василь	Васильович	1	Лютий	50	0	50	26	31.05.2024	0	1300
2	Іваненко	Степан	Вікторович	2	Лютий	60	0	60	26	31.05.2024	0	1560
3	Музика	Василь	Васильович	1	Березень	100	50	50	26	31.05.2024	0	1300
4	Іваненко	Степан	Вікторович	2	Березень	135	60	75	26	31.05.2024	0	1950
5	Грін	Тарас	Андрійович	3	Квітень	35	0	35	26	31.05.2024	0	910

Рисунок 3.10 – Вигляд вкладки «Квитанції водопостачання»

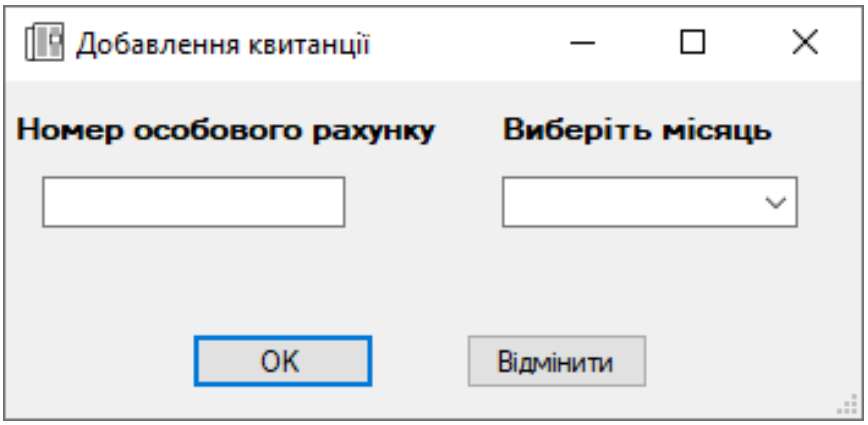
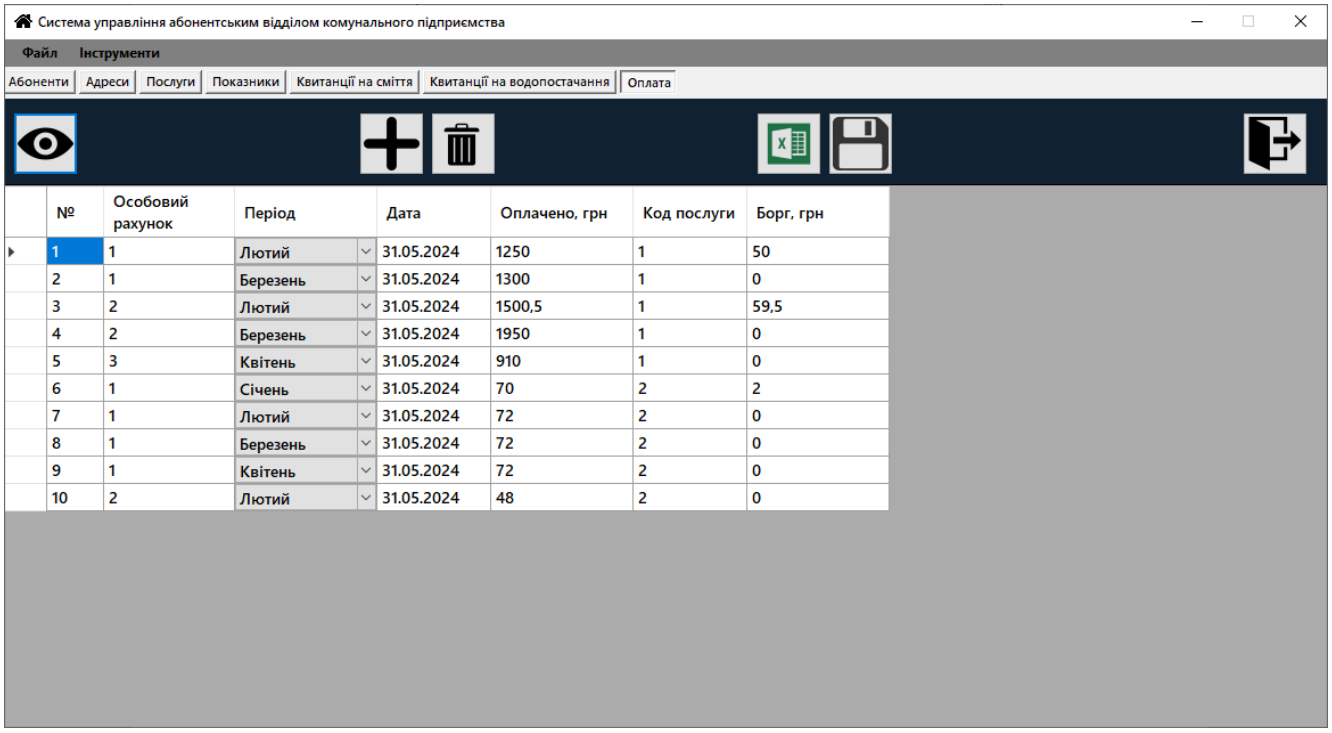


Рисунок 3.11 – Вигляд вікна формування квитанції на водопостачання

Дані про успішні оплати за надані послуги знаходяться в вкладці «Оплата», вигляд та функціонал якої зображено на рисунку 3.12.



№	Особовий рахунок	Період	Дата	Оплачено, грн	Код послуги	Борг, грн
1	1	Лютий	31.05.2024	1250	1	50
2	1	Березень	31.05.2024	1300	1	0
3	2	Лютий	31.05.2024	1500,5	1	59,5
4	2	Березень	31.05.2024	1950	1	0
5	3	Квітень	31.05.2024	910	1	0
6	1	Січень	31.05.2024	70	2	2
7	1	Лютий	31.05.2024	72	2	0
8	1	Березень	31.05.2024	72	2	0
9	1	Квітень	31.05.2024	72	2	0
10	2	Лютий	31.05.2024	48	2	0

Рисунок 3.12 – Вигляд вкладки «Оплата»

В меню програми в розділі «Інструменти» можна відкрити вікно «Налаштування прав користувачів», в якому зберігаються дані про користувачів, які мають доступ до програми. Вигляд цього вікна зображено на рисунку 3.13.

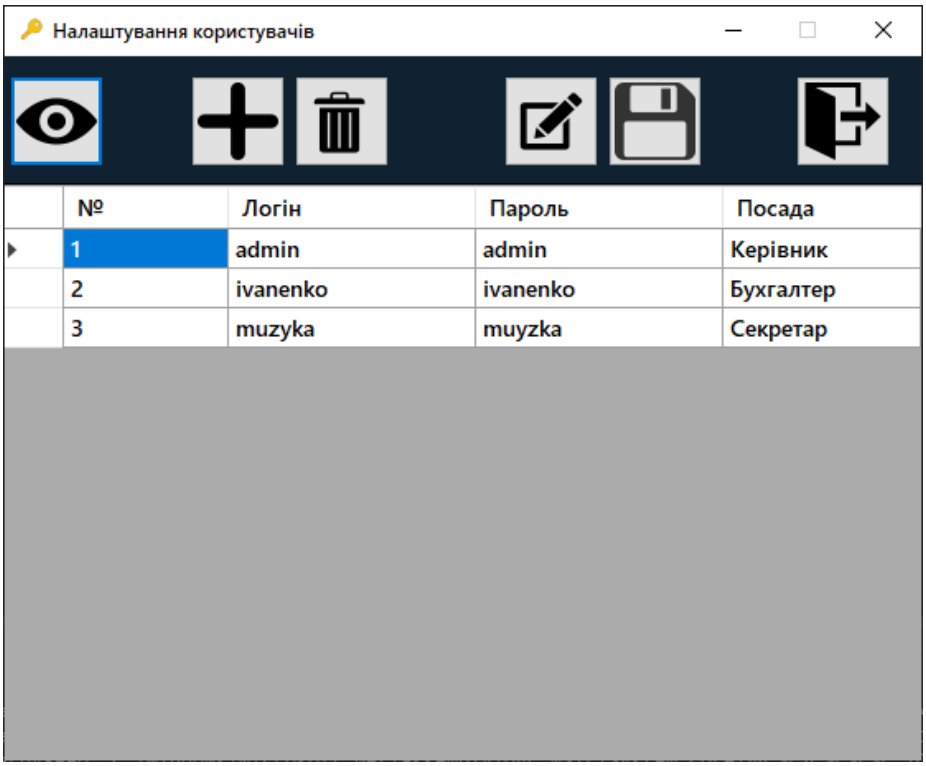


Рисунок 3.13 – Вигляд вікна налаштування користувачів

За допомогою меню програми, а саме розділу «Інструменти», можна відкрити вікно для пошуку даних, вигляд якого на рисунку 3.14.

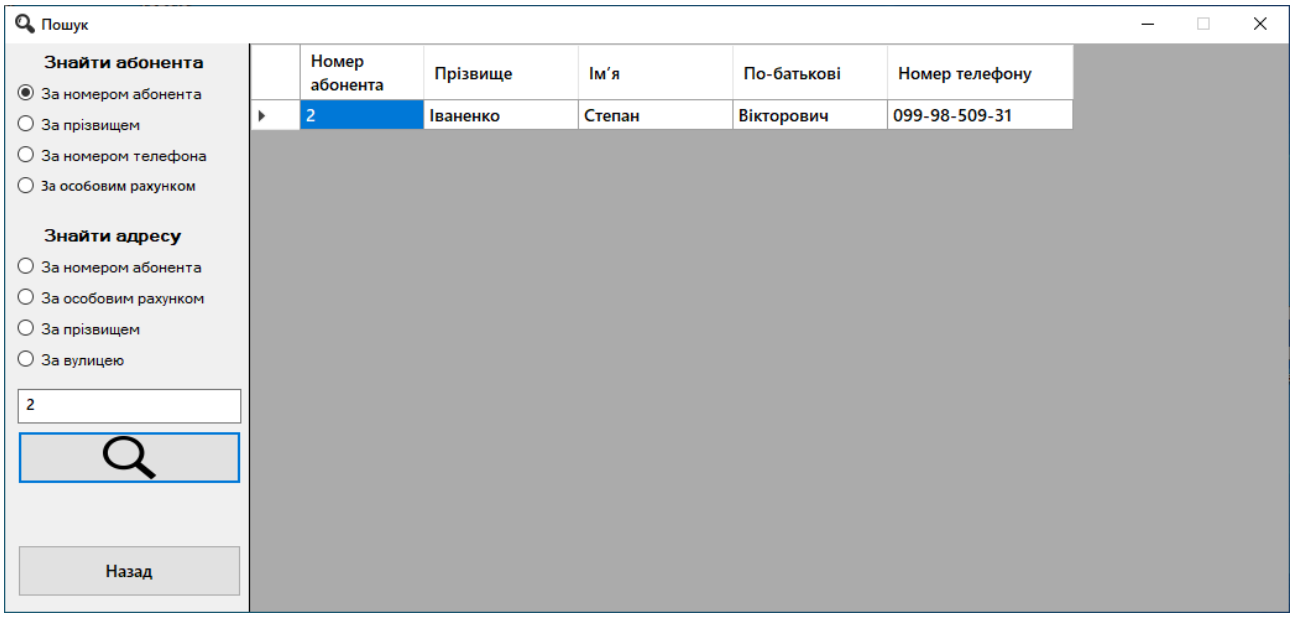


Рисунок 3.14 – Вигляд вікна для пошуку даних

В створеній системі управління також присутні додаткові вікна, а саме: «Інформація про програму» (див. Рис. 3.15), та вікно з довідкою, в якому можна отримати допомогу щодо особливостей користування програмою (див. Рис. 3.16).

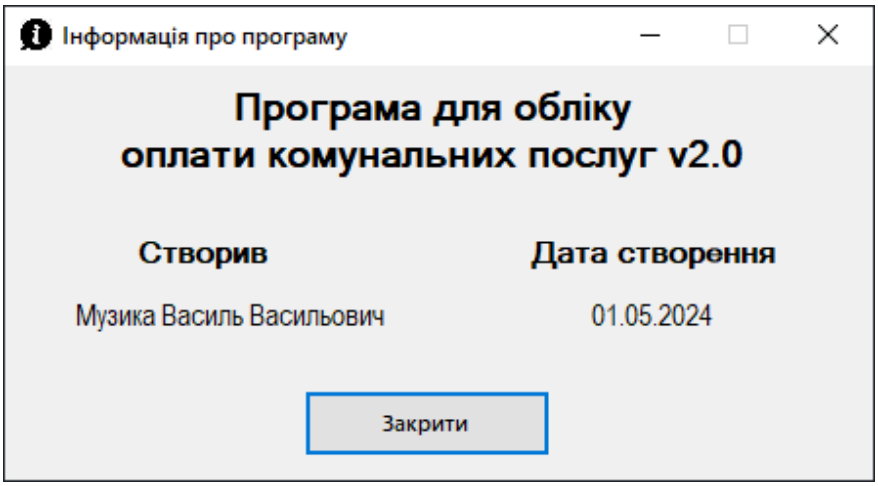


Рисунок 3.15 – Вікно з інформацією про програму

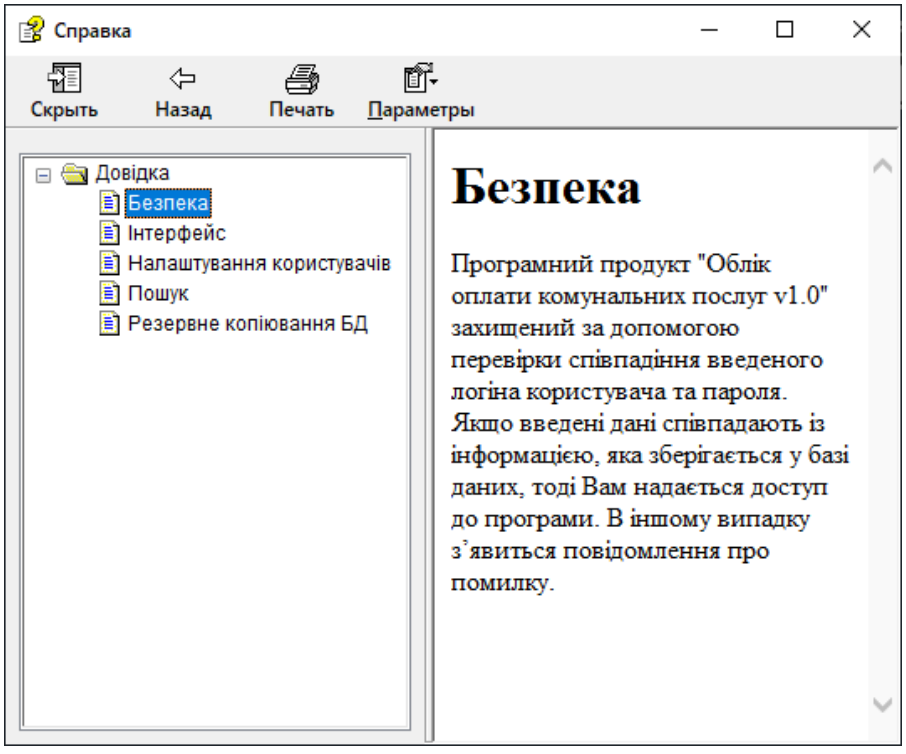


Рисунок 3.16 – Довідка програми

В довідці знаходиться детальна інформація про програму, а саме: безпека та функціонал програми.

3.2 Опис програмних модулів

Проаналізувавши технічне завдання та вимоги користувачів, сформуємо та опишемо основні програмні модулі. Для реалізації системи використовуємо середовище розробки Microsoft Visual Studio, мову програмування C# та MS SQL Server для побудови структури бази даних.

Microsoft Visual Studio – це програмне забезпечення від фірми Майкрософт, яке надає інтегроване середовище розробки програмного забезпечення з великою кількістю інструментів [11].

C# – це об'єктно-орієнтована мова програмування, яка дозволяє розробникам створювати програми для комп'ютерів та інших пристроїв. Вона має простий синтаксис, що полегшує написання коду, і використовується для створення різноманітних програм, від веб-додатків до ігор [12].

Microsoft SQL Server – це програмне забезпечення, розроблене компанією Microsoft, яке дозволяє управляти базами даних [13].

Використовуючи описане програмне забезпечення та технології, було визначено та описано необхідні програмні модулі.

Перелік програмних модулів проекту наведено та описано у таблиці 3.1.

Таблиця 3.1 – Опис модулів програми «oblik_poslug»

№	Назва програмного модулю	Короткий опис
1	2	3
1	Zastavka	Форма для появлення заставки при запуску програми
2	Login	Модуль для здійснення процесу авторизації. Вводиться логін та пароль та натискається кнопка «Вхід»
3	Form1	Основне вікно програми, на якому знаходяться таблиці із даними.
4	AddAbonents	Цей модуль потрібний для добавлення нового абонента
5	AddKvVoda	Необхідний для створення квитанції за послугу «Водопостачання»
6	AddKvSm	Використовується для створення квитанції за послугу вивезення сміття
7	info	Відображає інформацію про програму та її власника

Продовження таблиці 3.1

1	2	3
8	KvSmittya	Модуль потрібний для попереднього перегляду квитанції за вивезення сміття, а також її друку
9	KvVoda	Модуль потрібний для попереднього перегляду квитанції на водопостачання та її друку
10	Search	Необхідний модуль для здійснення пошуку абонентів, а також їх адрес
11	TableUsers	Модуль для управління користувачами, які мають доступ до програми

3.3 Опис результатів тестування

Після створення програмного проекту необхідно провести його тестування різноманітними методами.

Для проведення тестування з метою виявлення помилок будуть використовуватись наступні методи:

– ручне тестування;

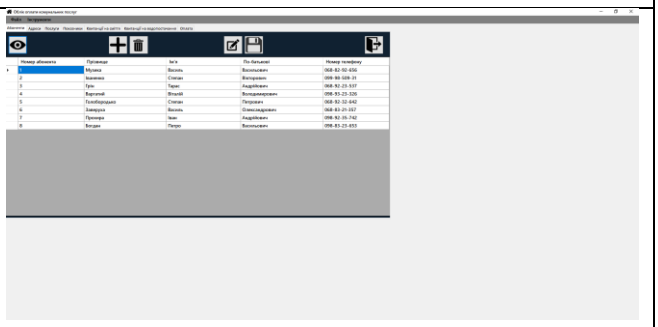
– функціональне тестування.

Ручне тестування – це тип тестування, в якому тестування виконується людиною вручну. Вона виступає користувачем програми й досліджує її на наявність помилок. Для професійного тестування часто використовують написаний план тестування з варіантами тестування, який також ще називають «тест-кейс» (test cases).

Результатом проведення ручного тестування являється написання структури баг-репорту (Bug-report).

Структуру баг-репорту наведено у таблиці 3.1.

Таблиця 3.2 - Структура Bug-report

Система управління абонентського відділу комунального підприємства	
Короткий опис (Summary)	При натисканні на кнопку для розгортання вікна, воно некоректно розгортається
Проект (Project)	WindowsFormsApplication1
Компонент додатку (Component)	Notification
Номер версії (Version)	1.0.1
Серйозність дефекту (Severity)	S5 Тривіальний (Trivial)
Пріоритет дефекту (Priority)	P3 Низький (Low)
Автор (Author)	Музика Василь Васильович
Призначення (Assigned to)	Музика Василь Васильович
Середовище	
ОС /Service Pack / Version Pack / Апаратні характеристики	Windows 10 Professional / Intel Core i5-8300H / RAM 8 Гб / NVIDIA GeForce GTX1050
Опис	
Кроки відтворення (Steps to reproduce)	1. Натискання кнопки для розгортання вікна в панелі керування вікном
Фактичний результат (Result)	Вікно збільшилося, проте розмір елементів на ньому залишились однаковими
Очікуваний результат (Expected result)	Вікно не змінить свого розміру
Додатково	
Прикріплений файл (Attachment)	

Для виправлення даної помилки, властивості «MaximizeBox» було задане значення негативне значення (False).

Тепер необхідно провести функціональне тестування.

Functional testing (Функціональне тестування) – це вид тестування, процес якого полягає в перевірці правильності отриманих даних при наданні певних вхідних даних, тобто порівняння одержаного результату з очікуваним [14]. Функціональне тестування імітує фактичне використання системи.

Результатом проведення функціонального тестування є набір тестів відомих як «test-case».

Результати тестування програми функціональним методом наведено у таблиці 3.3, розширений опис тест-кейсу наведено у таблиці 3.4.

Таблиця 3.3 – Короткий опис тест-кейсів

ID	Тип	Опис	Очікувано	Реально	Pass/ Fail
01	positive	Перевірка функції експорту даних в файл Microsoft Excel	Дані будуть експортовані в файл Microsoft Excel	Дані успішно експортовано в файл Microsoft Excel	Pass
02	positive	Добавлення адреси за допомогою вікна створення абонента	Добавлення наступної інформації: Тернопіль, Заводська, 2, 0, 1, 35, 5	Добавлення наступної інформації: Тернопіль, Заводська, 2, 0, 1, 35, 5	Pass
03	positive	Добавлення нової квитанції за вивезення сміття, вхідні дані при добавленні: номер особового рахунку 3, місяць Квітень	Добавлення наступної інформації: Грін, Тарас, Андрійович, 3, Квітень, 24, 4, 96, 31.05.2024	Добавлення наступної інформації: Грін, Тарас, Андрійович, 3, Квітень, 24, 4, 96, 31.05.2024	Pass
04	positive	Добавлення нової квитанції за водопостачання, вхідні дані при добавленні: номер особового рахунку 1, місяць Березень	Добавлення наступної інформації: Музика Василь Васильович, 1, Березень, 100, 50, 50, 26, 31.05.2024, 0, 1300	Добавлення наступної інформації: Музика Василь Васильович, 1, Березень, 100, 50, 50, 26, 31.05.2024, 0, 1300	Pass

Таблиця 3.4 – Розширений опис тест-кейсу

Назва тесту, що перевіряється	Здійснення перевірки добавлення інформації про абонентів, їх адреси, показники лічильників, квитанції та оплату комунальних послуг		
Функція	System.Data.SqlClient		
Дата тесту	30/05/2024		
Дія	Очікуваний результат	Реальний результат	PASS/FAIL/BLOCK
Добавлення інформації про абонента	Добавлення нового абонента з наступними даними: 1, Музика, Василь, Васильович, 068-82-92-535	Добавлення нового абонента з наступними даними: 1, Музика, Василь, Васильович, 068-82-92-535	Pass

Продовження таблиці 3.4

Добавлення інформації про адрес абонента	Добавлення наступних даних: 2, Тернопіль, Тайна, 35, 16, 2, 25, 2	Добавлення наступних даних: 2, Тернопіль, Тайна, 35, 16, 2, 25, 2	Pass
Добавлення даних про показник лічильника	Добавлення наступних даних: 2, 50, 01.05.2024, Лютий, 1	Добавлення наступних даних: 2, 50, 01.05.2024, Лютий, 1	Pass
Формування квитанції за послугу вивезення сміття	Добавлення наступних даних: 5, Голобородько, Степан, Петрович, 5, Березень, 24, 1, 24, 31.05.2024	Добавлення наступних даних: 5, Голобородько, Степан, Петрович, 5, Березень, 24, 1, 24, 31.05.2024	Pass
Формування квитанції за послугу водопостачання	Добавлення наступних даних: 1, Музика, Василь, Васильович, 1, Лютий, 50, 0, 50, 26, 31.05.2024, 0, 1300	Добавлення наступних даних: 1, Музика, Василь, Васильович, 1, Лютий, 50, 0, 50, 26, 31.05.2024, 0, 1300	Pass
Добавлення даних про здійснену оплату за послугу	Добавлення наступних даних: 6, 1, Січень, 31.05.2024, 70, 2, 2	Добавлення наступних даних: 6, 1, Січень, 31.05.2024, 70, 2, 2	Pass

Отож, проведемо далі модульне тестування, суть якого полягає в перевірці правильності роботи певної частини коду. В об'єктно-орієнтованих системах це здебільшого тестування класів, причому базові модульні тести включають перевірку конструкторів і деструкторів.

Такі тести зазвичай створюються розробниками під час написання коду (метод «білої скриньки»), щоб переконатися, що функції працюють відповідно до очікувань. Для однієї функції може бути написано декілька тестів, щоб охопити всі можливі сценарії її використання. Модульне тестування не перевіряє роботу програмного забезпечення в цілому, але гарантує, що окремі компоненти працюють коректно й автономно [15].

Модульне тестування є етапом створення програмного забезпечення, який передбачає одночасне використання різноманітних методів для запобігання дефектам і визначення підходів, що дозволяють знизити ризики, час і витрати на

розробку. Його виконують розробники або інженери програмного забезпечення під час етапу створення у життєвому циклі розробки. Основна мета модульного тестування полягає у виявленні та виправленні помилок проектування, що дозволяє забезпечити необхідний рівень якості програмного забезпечення згідно з вимогами процесу контролю якості.

Залежно від структури та потреб процесу розробки програмного забезпечення, модульне тестування може виконувати аналіз коду статично, аналізувати потоки даних, оцінку метрик, експертний огляд коду, перевірку покриття коду та інші методи тестування.

Для створення unit-тесту в графічному середовищі Visual Studio, спочатку необхідно створити проект для модульного тестування (див. Рис. 3.17).

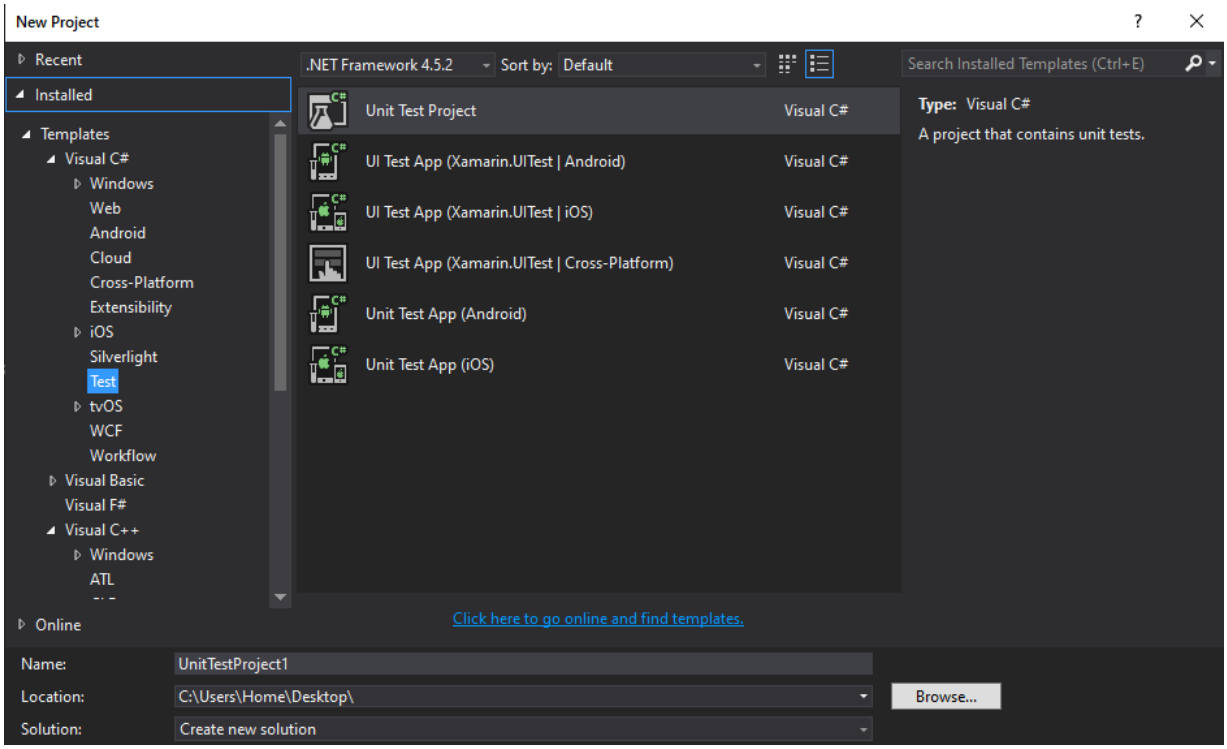


Рисунок 3.17 – Створення нового unit-тесту

Для здійснення прив’язки створеного проекту тестування із проектом десктопного додатку потрібно обрати посилання на цей проект (див. Рис. 3.18). Це необхідно для того, щоб отримана доступ до класів.

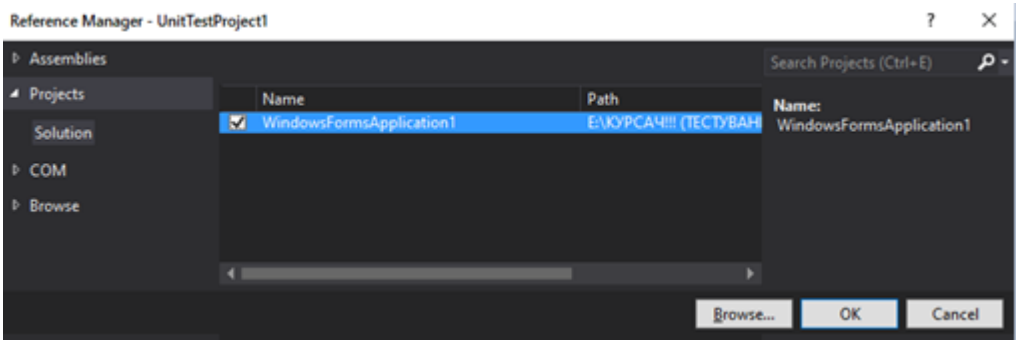


Рисунок 3.18 – Вікно обрання посилання на десктопний проект

Після виконаних дій, модульні тести успішно добавляться у файл UnitTest1.cs в проекті WindowsFormsApplication1.Test. Код добавленого тесту зображено на рисунку 3.19.

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using EssentialTools.Models;
namespace EssentialTools.Tests
{
    [TestClass]
    public class UnitTest1
    {
        private IDLessZero getTestObject()
        {
            return new MaxID();
        }
        [TestMethod]
        public void ID_less_zero()
        { // arrange (організація)
            IDLessZero target = getTestObject();
            decimal total = 1;
            // act (акт)
            var IDTotal = target.ApplyID(total);
            // assert (підтвердження)
            Assert.AreEqual(total + 1, IDTotal);
        }
    }
}
```

Рисунок 3.19 – Код тесту для перевірки правильності генерування ID

Щоб дізнатися результат виконання тесту, потрібно відкрити віно Test Explorer, в якому описано назву тесту та його результат. (див. Рис. 3.20).

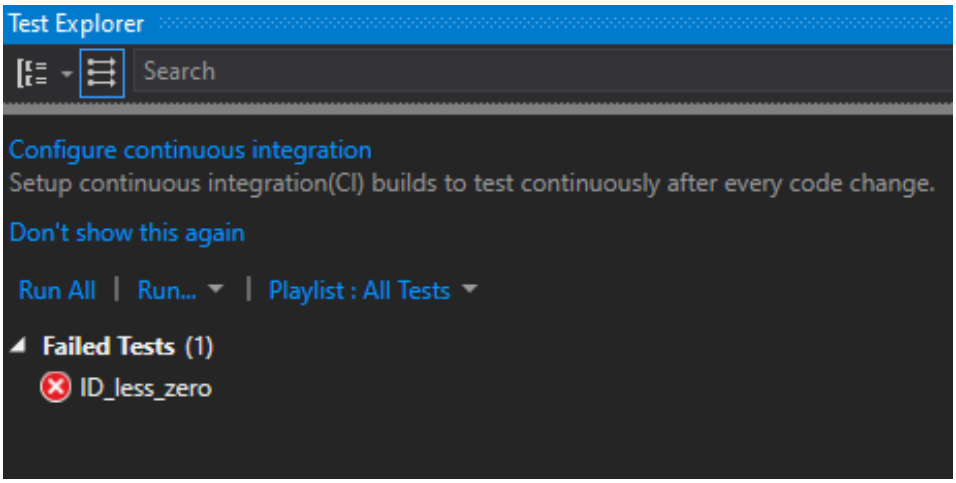


Рисунок 3.20 – Вигляд вікна для відображення результату тесту

Далі напишемо код для тестування, суть якого буде полягати в перевірці значення «ID», щоб воно було більше нуля. Для цього добавимо умову перевірки в реалізацію класу MaxID (див. Рис. 3.21).

```
using System;
namespace EssentialTools.Models
{
      public class MaxID : IDLessZero
      {
            public decimal ApplyID(decimal totalParam)
            {
                  if (ID < 0)
                  {
                        throw new ArgumentOutOfRangeException();
                  }
                  else if (ID > 0)
                  {
                        return ID;
                  }
            }
      }
}
```

Рисунок 3.21 – Тест для перевірки значення ID

Результат виконання тесту на перевірку значення «ID» можна переглянути на рисунку 3.22.

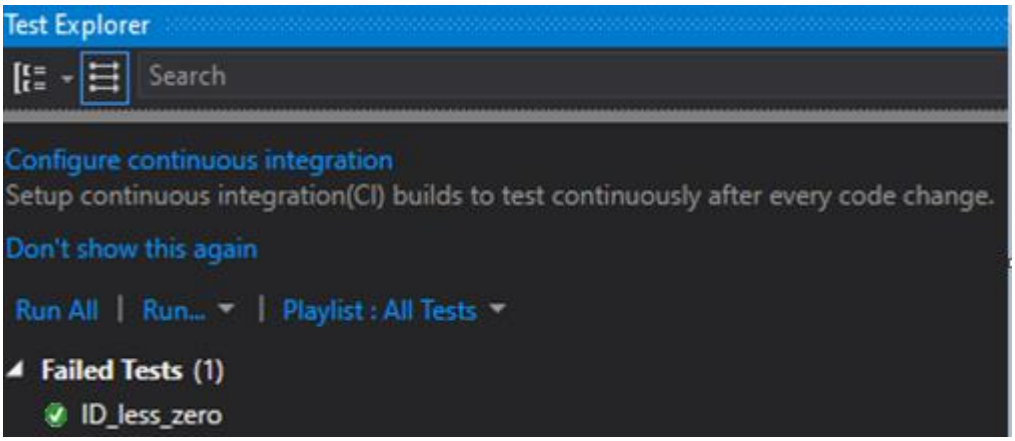


Рисунок 3.22 – Результат виконання тесту на перевірку значення «ID»

Даний вид тестування дозволяє надійно перевірити якість написаного коду.

Проходження цього типу тестування можна здійснювати при абсолютно любій зміні структури коду, в тому числі при добавленні або видаленні частини коду в написаної функції у класі.

Отож, провівши різні видів тестування в створеній системі управління, можна зробити висновок, що дуже часто в створеній програмі будуть присутні помилки або так звані «баги». Найчастіше за все, помилки виникають через неуважність розробника при написанні коду. Зазвичай, в першій версії створеного програмного забезпечення часто знаходять помилки, тому існує таке поняття як «бета-версія», суть якого полягає в тестуванні версії програми перед тим, як робити її основною.

Підводячи підсумки, в ході виконання кількох видів тестування, в роботі програми були знайдені недоліки в кількох частинах системи, а точніше: при проектуванні користувацького інтерфейсу, при роботі з функціями в класі. Останній вид виконаного тестування демонструє нам, що значна частина програмного коду системи написана допустимо, та помилки майже відсутні.

Написання різних типів тестування для системи та їх виконання дозволило знайти кілька недоліків та помилок, які були успішно виправлені.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬПОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Вплив діяльності людини на довкілля.

Сучасне суспільство, яке характеризується високим рівнем споживання та інтенсивним промисловим розвитком, має значний вплив на навколишнє середовище. На жаль, ця діяльність людини часто призводить до серйозного забруднення та деградації природних екосистем.

Однією з найвагоміших проблем є забруднення довкілля. Промислові викиди, автомобільні вихлопи та побутові відходи стали постійними супутниками нашого повсякденного життя. Особливо гостро стоїть питання забруднення повітря, яке призводить до глобального потепління, кислотних дощів та погіршення якості життя людей. Так, за даними ООН, близько 7 мільйонів людей щорічно помирають передчасно через забруднення повітря. Ця проблема особливо актуальна для великих міст, де скупчення промислових підприємств та інтенсивний рух транспорту створюють несприятливу екологічну ситуацію [16].

Не менш важливою є проблема забруднення водних ресурсів. Промислові та побутові стоки, нафтові й інші хімічні забруднення, потрапляючи у річки, озера та моря, порушують природну рівновагу водних екосистем. Це призводить до зникнення багатьох видів флори та фауни, погіршення якості питної води та небезпечних наслідків для здоров'я людей. Одним з найбільш вражаючих прикладів катастрофічного впливу людини на водойми є Аральське море, яке за кілька десятиліть перетворилося з четвертого за величиною озера світу на практично повністю висохлий басейн.

Не можна оминути й проблему утилізації побутових відходів. Сучасна модель споживання, яка заохочує постійне придбання нових товарів, призводить до стрімкого накопичення сміття. За даними ООН, щороку у світі утворюється понад 2 мільярди тон твердих побутових відходів, і лише невелика частка з них переробляється або утилізується екологічно безпечним способом. Решта потрапляє на стихійні звалища чи спалюється, забруднюючи навколишнє середовище [17].

Окрім зазначених вище проблем, діяльність людини негативно впливає на біорізноманіття, виснажує природні ресурси, провокує виникнення природних катастроф тощо. Все це в сукупності створює реальну загрозу для існування людства та планети Земля загалом.

Серед інших важливих аспектів впливу діяльності людини на довкілля є деградація ґрунтів. Інтенсивне сільське господарство, використання пестицидів та добрив, вирубка лісів і надмірний випас худоби призводять до втрати родючості ґрунтів та їх ерозії. Це знижує продуктивність аграрних угідь, що в свою чергу загрожує продовольчій безпеці. За даними ООН, кожного року через ерозію втрачається близько 24 мільярдів тонн родючого ґрунту, що становить значну загрозу для сільськогосподарських регіонів.

Ще однією проблемою є викиди парникових газів, що сприяють зміні клімату. Збільшення концентрації вуглекислого газу (CO2), метану (CH4) та оксиду азоту (N2O) в атмосфері призводить до підвищення середньої температури на планеті. Це викликає глобальні зміни клімату, включаючи екстремальні погодні явища, такі як посухи, повені, урагани і тайфуни. Зміни клімату також впливають на рівень моря, викликаючи його підвищення через танення полярних льодовиків та термічне розширення океанів.

Вплив діяльності людини на лісові екосистеми також є значним. Дефорестація, або вирубка лісів, має катастрофічні наслідки для біорізноманіття та клімату. Ліси є важливими регуляторами клімату, оскільки вони поглинають вуглекислий газ і виробляють кисень. Втім, знищення лісів, особливо тропічних, призводить до втрати життєвого середовища для багатьох видів флори і фауни, збільшення викидів CO2 в атмосферу та погіршення кліматичних умов.

Не можна забувати і про забруднення ґрунтів важкими металами та токсичними речовинами, яке відбувається через діяльність гірничодобувної промисловості, неправильне поводження з відходами та застосування хімікатів у сільському господарстві. Забруднення ґрунтів негативно впливає на здоров'я людей та тварин, знижує врожайність культур і може мати довготривалі наслідки для екосистем.

Для вирішення глобальних екологічних проблем необхідні скоординовані зусилля як на державному, так і на індивідуальному рівні. Це включає впровадження екологічних технологій у промисловості, розвиток альтернативної енергетики та удосконалення системи поводження з відходами. Одним з найефективніших шляхів мінімізації негативного впливу людської діяльності на довкілля є впровадження стійких практик та технологій, таких як використання відновлюваних джерел енергії, розвиток циркулярної економіки та зменшення використання традиційних джерел енергії, які значно забруднюють довкілля. Підвищення екологічної свідомості громадян також є важливим фактором у боротьбі з екологічними проблемами. Тільки спільними зусиллями можна зберегти нашу планету для майбутніх поколінь.

Отже, підвищення екологічної обізнаності та свідомості серед населення є ключовим фактором у боротьбі з екологічними проблемами. Освіта та інформування про наслідки людської діяльності для довкілля можуть сприяти зміні поведінки та прийняттю більш екологічно відповідальних рішень як на індивідуальному, так і на суспільному рівнях. Екологічна освіта має починатися з раннього віку і бути інтегрованою в усі рівні освітньої системи.

Загалом, збереження довкілля вимагає комплексного підходу, що включає як наукові дослідження і технологічні інновації, так і зміну поведінки суспільства та ефективну екологічну політику. Тільки таким чином ми зможемо забезпечити стійкий розвиток та збереження нашої планети для майбутніх поколінь.

Розпочинати дії для збереження навколишнього середовища потрібна якнайшвидше, оскільки з кожним роком ця проблема стає більш актуальнішою. На мою думку, перш за все необхідно донести важливість підтримання навколишнього середовища серед учнів та студентів, оскільки вони являються наступним поколінням, тому дана проблема напряму стосується їх.

4.2 Особливості заходів електробезпеки на підприємствах.

На сучасних підприємствах електробезпека має велике значення з точки зору забезпечення безпеки працівників та нормального функціонування технологічних процесів. Особливості заходів електробезпеки на підприємствах пов'язані з різноманітними аспектами, включаючи організаційні, технічні та професійно-виробничі аспекти.

Організаційні аспекти включають в себе розробку та впровадження ефективних систем управління електробезпекою. Це означає встановлення чітких процедур щодо роботи з електрообладнанням, навчання персоналу правилам безпеки та регулярне проведення перевірок та аудитів систем безпеки. Крім того, важливо забезпечити належний рівень доступу до електрообладнання лише кваліфікованим персоналом, який має відповідні знання та навички.

Технічні аспекти електробезпеки передбачають застосування спеціалізованого електрообладнання та засобів захисту. Це включає в себе використання автоматичних вимикачів, розеток з захисними кришками, заземлення обладнання та ізоляцію проводів. Крім того, слід регулярно проводити технічний огляд та обслуговування електрообладнання для виявлення та усунення можливих несправностей.

Професійно-виробничі аспекти електробезпеки включають навчання персоналу правилам безпеки та надання їм необхідних знань та навичок щодо роботи з електрообладнанням. Це означає проведення регулярних тренувань та інструктажів з безпеки, а також надання персоналу необхідних засобів індивідуального захисту, таких як ізолюючі рукавички та окуляри.

Одним з аспектів електробезпеки на підприємствах є безпечне користування комп'ютерами та принтерами. Для цього необхідно дотримуватися таких правил:

– Встановлюйте комп’ютери в сухих, добре вентильованих приміщеннях.

– Перед підключенням або відключенням комп'ютера чи принтера переконайтеся, що вони вимкнені. Використовуйте лише правильно заземлені розетки та відповідні кабелі.

– Уникайте перевантаження електричних мереж. Не підключайте до одного розетки занадто багато пристроїв одночасно, оскільки це може призвести до перевантаження електричної мережі та виникнення пожежі.

– Проводьте регулярне технічне обслуговування комп'ютерів та принтерів для виявлення та усунення можливих несправностей.

– Для захисту від різких перепадів напруги в електричних мережах використовуйте стабілізатори напруги.

– Не використовуйте пошкоджені кабелі або обладнання, оскільки це може призвести до короткого замикання та виникнення пожежі [18].

Врахування цих аспектів дозволить забезпечити безпеку користування електрообладнанням на підприємствах та уникнути можливих негативних наслідків.

Враховуючи всі ці аспекти, заходи електробезпеки на підприємствах є складними та багатогранними, вимагаючи поєднання організаційних, технічних та професійно-виробничих заходів для забезпечення безпеки працівників та ефективного функціонування підприємства.

Отже, можна зробити висновок, що є безліч особливостей заходів електробезпеки на підприємствах, яких необхідно дотримуватися. Незважаючи на можливі витрати на них, слід пам’ятати, що здоров’я працівників завжди важливіше любих фінансових витрат.

ВИСНОВКИ

Швидкий прогрес інформаційних технологій призвів до інтенсивного використання комп'ютерної техніки та інформаційних систем у повсякденному житті людей.

Щодня з'являється все більше автоматизованих систем для різних сфер діяльності, які спрощують процес опрацювання інформації, та частково допомагають в управлінні технологічними процесами на підприємствах. Це сприяє зростанню автоматизації багатьох процесів.

В ході виконання кваліфікаційної роботи бакалавра було створено автоматизовану систему управління абонентським відділом комунального підприємства. Розроблена система призначена для використання в комунальних підприємствах. Ця робота стала результатом навчання в університеті імені Івана Пулюя, основою якого являється вивчення проектування баз даних, діаграм, освоєнням мови програмування C#, та проведенням різних видів тестування.

Даний проект містить основні функції для роботи з базою даних та SQL запитами. Також, виконання цієї кваліфікаційної роботи дозволила покращити знання з мови програмування C#, а також надала змогу здобути більше досвіду по роботі з C#, базами даних, SQL запитами, а також проведенням різних видів тестування програмного забезпечення.

Розроблений проект після незначних доопрацювання може масштабно використовуватися в різних комунальних підприємствах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт КП «Тернопіль водоканал» [Електронний ресурс] – Режим доступу: http://www.vodokanal.te.ua/ (дата звернення: 01.05.2024).

2. Комунальна програма [Електронний ресурс] – Режим доступу: https://ususoft.com.ua/uk/kommunalnaya_programma.php (дата звернення: 02.05.2024).

3. SQL Server Management Studio (SSMS) [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16 (дата звернення 03.05.2024).

4. Огляд основних SQL запитів - Блог ITVDN [Електронний ресурс] – Режим доступу: https://itvdn.com/ua/blog/article/m-sql (дата звернення 04.05.2024).

5. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань: навч. посіб. Львів, Магнолія-2006. 2012. 584 с.

6. Шаров С.В., Осадчий В.В. Бази даних та інформаційні системи: навч. посіб. Мелітополь : МДПУ ім. Б. Хмельницького, 2014. 352 с.

7. Пасічник В. В., Резніченко В. А. Організація баз даних та знань: підручник. Київ : Видавнича група BHV, 2006. 384 с.

8. О. М. Бевз, В. М. Папінов, Ю. А. Скидан. Проектування програмних засобів систем управління: навч. посіб. Вінниця, ВНТУ. 2010. 124 с.

9. Розробка uml діаграми варіантів використання [Електронний ресурс] – Режим доступу: https://studfiles.net/preview/5200239/page:6/ (дата звернення 05.05.2024).

10. Схеми бази даних: компоненти, типи та проектування [Електронний ресурс] – Режим доступу: https://foxminded.ua/skhemy-bazy-danyh/ (дата звернення 08.05.2024).

11. Visual Studio documentation | Microsoft Learn [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022 (дата звернення 09.05.2024).

12. Програмування на C# [Електронний ресурс] – Режим доступу: http://programer.in.ua/index.php/prohramuvannia/prohramuvannia-na-c (дата звернення: 10.05.2024).

13. Microsoft SQL Server – Вікіпедія [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення 11.05.2024).

14. Функціональне тестування – Вікіпедія [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Функціональне_тестування (дата звернення 12.05.2024).

15. Тестування програмного забезпечення: етапи та методи [Електронний ресурс] – Режим доступу: https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/ (дата звернення 13.05.2024).

16. World Health Organization. Ambient (outdoor) air pollution [Електронний ресурс] – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health (дата звернення: 07.06.2024).

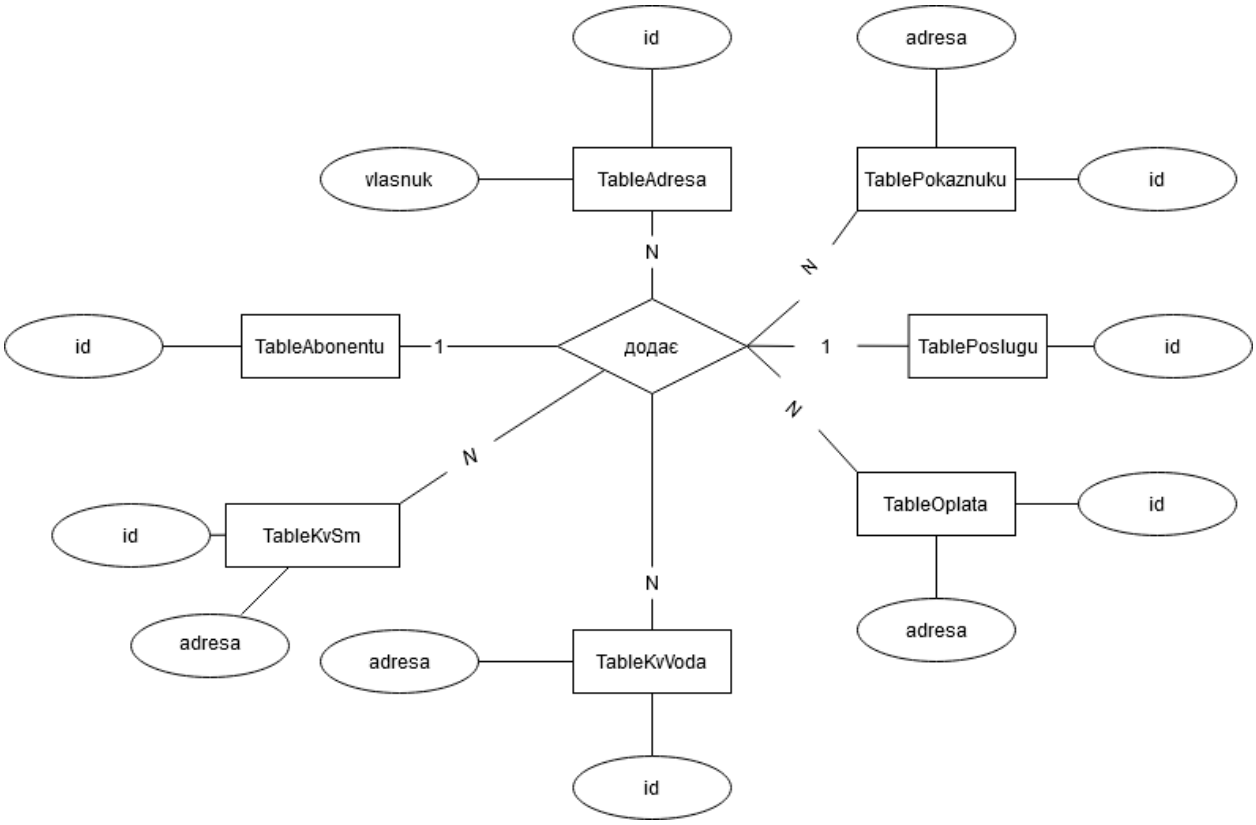
17. Kaza, S., Yao, L., Bhada-Tata, P., Van Woerden, F. What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050. World Bank. [Електронний ресурс] – Режим доступу: https://datatopics.worldbank.org/what-a-waste/ (дата звернення: 07.06.2024).

18. Електробезпека на підприємстві та в офісі [Електронний ресурс] – Режим доступу: https://profiteh.ua/elektrobezpeka-na-pidpryiemstvi-ta-ofisi/ (дата звернення 07.06.2024).

ДОДАТКИ

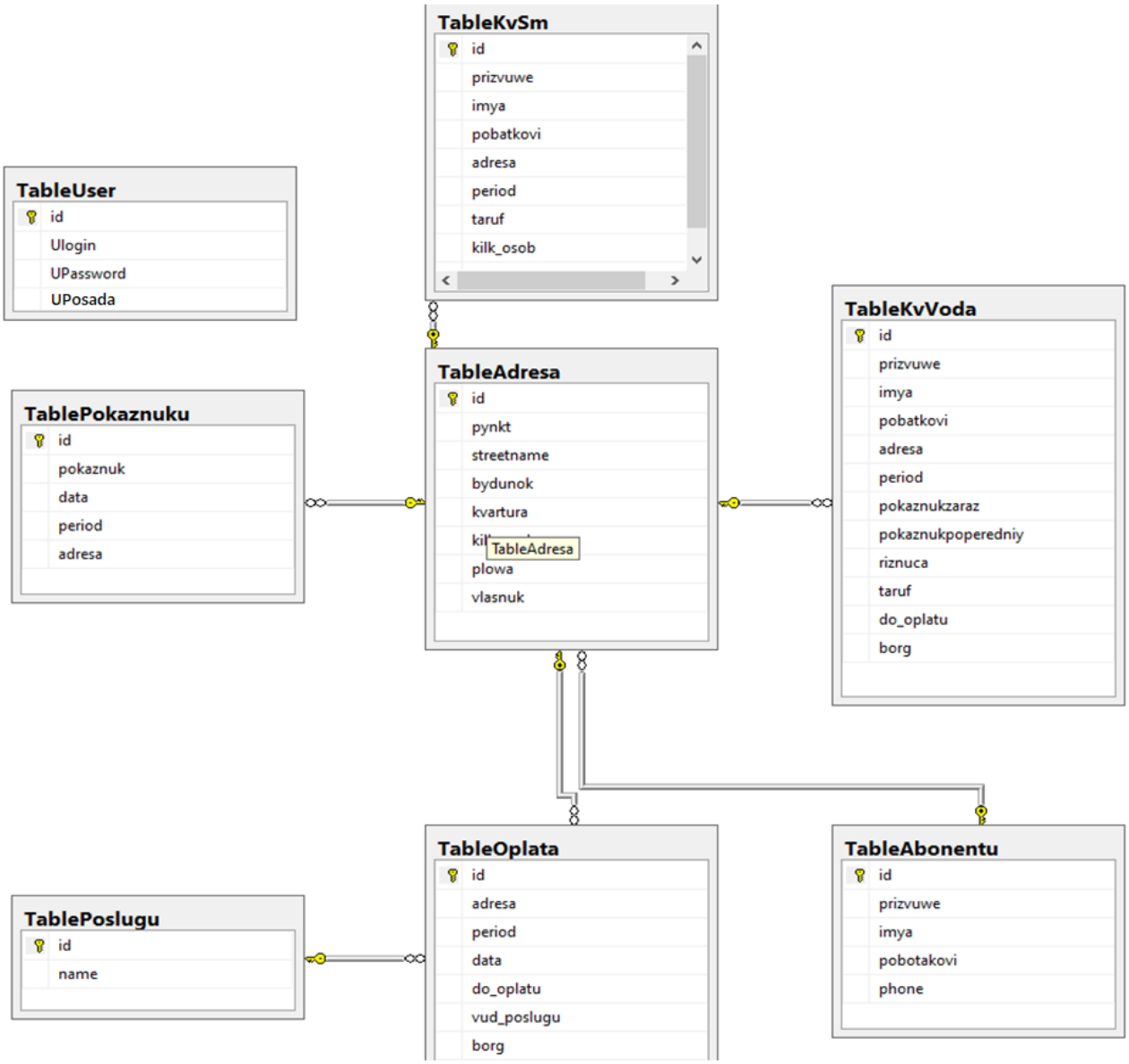
Додаток А

Діаграма «сутність-зв’язок»



Додаток Б

Діаграма зв’язків між таблицями бази даних



Додаток В

Основний код програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.Office.Interop.Excel;
using System.IO;
using System.Diagnostics;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            table1buttons(false);
            TableAdresaButtons(false);
        }
        public SqlConnection connectionString = new SqlConnection("Data
Source=VASYA\\SQLEXPRESS; Integrated Security=SSPI; Initial Catalog=oblik_poslug_copy");

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        protected override void OnFormClosing(FormClosingEventArgs e)
        {
            base.OnFormClosing(e);
            System.Windows.Forms.Application.Exit();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.Application.Exit();
        }
        private async void AbonentuShowTable()
        {
            try
            {
                dataGridView1.Rows.Clear();
                connectionString.Open();
                SqlCommand command = new SqlCommand("Select id,prizvuwe,imya,pobotakovi,phone
From TableAbonentu", connectionString);

                SqlDataReader reader = await command.ExecuteReaderAsync();               if
                (reader.HasRows)
                {
```

Продовження Додатку В

```
                    while (reader.Read())
                    {
                        dataGridView1.Rows.Add(reader["id"], reader["prizvuwe"],
reader["imya"], reader["pobotakovi"], reader["phone"]);
                    }
                }

                if (dataGridView1.Visible == true)
                {
                    dataGridView1.Visible = false;
                    table1buttons(false);
                }
                else
                {
                    dataGridView1.Visible = true;
                    table1buttons(true);
                }
                connectionString.Close();
            }
            catch { connectionString.Close(); };
        }

        private void button2_Click(object sender, EventArgs e)
        {
            AbonentuShowTable();
        }

        private void refreshInfo()
        {
            try
            {
                connectionString.Open();
            SqlCommand command = new SqlCommand("Select id,prizvuwe,imya,pobotakovi,phone From
TableAbonentu", connectionString);



            dataGridView1.Rows.Clear();
            command.CommandText = "Select * From TableAbonentu";
            SqlDataReader dataReader = command.ExecuteReader();

            while (dataReader.Read())
            {
                dataGridView1.Rows.Add(dataReader["id"], dataReader["prizvuwe"],
dataReader["imya"], dataReader["pobotakovi"], dataReader["phone"]);
            }
            dataReader.Close();
            connectionString.Close();
            }
            catch
            {
                connectionString.Close();
            }
        }

        private void Table1Save()
        {
            try
            {
```

Продовження Додатку В

```
                connectionString.Open();
                SqlCommand command = connectionString.CreateCommand();

            int kol = dataGridView1.Rows.GetRowCount(DataGridViewElementStates.Visible);
            for (int i = 0; i < kol; i++)
            {
                int id = Convert.ToInt32(dataGridView1[0, i].Value);
                string prizvuwe = Convert.ToString(dataGridView1[1, i].Value);
                string imya = Convert.ToString(dataGridView1[2, i].Value);
                string pobotakovi = Convert.ToString(dataGridView1[3, i].Value);
                string phone = Convert.ToString(dataGridView1[4, i].Value);

                command.CommandText = "UPDATE TableAbonentu SET prizvuwe='" + prizvuwe + "',
imya='" + imya + "', pobotakovi='" + pobotakovi + "', phone='" + phone + "' WHERE id=" + id;

                Console.WriteLine(command.CommandText);

                command.ExecuteNonQuery();


            }
            connectionString.Close();
            }
            catch
            {
                MessageBox.Show("Некоректні дані!", "Помилка", MessageBoxButtons.OK);
                connectionString.Close();
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Table1Save();
        }

        private void db_execute(string command)
        {
            try
            {
                connectionString.Open();
                Console.WriteLine(command);
                SqlCommand command1 = new SqlCommand(command, connectionString);

                SqlDataReader reader = command1.ExecuteReader();
                connectionString.Close();
            }
            catch { MessageBox.Show("Помилка!", "Помилка"); connectionString.Close(); }
        }

        private void RefreshAllTable()
        {
            try
            {
                dataGridView1.Visible = true;
                dataGridView2.Visible = true;
                dataGridView3.Visible = true;
                dataGridView4.Visible = true;
                dataGridView5.Visible = true;
                refreshInfo();
```

Продовження Додатку В

```
                refreshInfoAuth();
                refreshinfotable2();
                RefreshPuslugu();
                refreshTableOplata();
                dataGridView1.Visible = true;
                dataGridView2.Visible = false;
                dataGridView3.Visible = false;
                dataGridView4.Visible = false;
                dataGridView5.Visible = false;
            }
            catch { }

        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                AddAbonents AddAbonent_form = new AddAbonents();
                this.Hide();
                connectionString.Open();
                AddAbonent_form.ShowDialog();
                int nomer;
                int nomerAbonentu;
                SqlCommand count = new SqlCommand("Select max(id) From TableAdresa",
connectionString);
                SqlCommand countAbonentu = new SqlCommand("Select max(id) From TableAbonentu",
connectionString);

                try
                {

                    nomer = Convert.ToInt32(count.ExecuteScalar().ToString());
                }
                catch { nomer = 0; }

                try
                {
                    nomerAbonentu = Convert.ToInt32(countAbonentu.ExecuteScalar().ToString());
                }
                catch { nomerAbonentu = 0; }

                if (nomerAbonentu > nomer) nomer = nomerAbonentu;

                try
                {
                    if (AddAbonent_form.DialogResult == DialogResult.OK)
                    {
                        if (((AddAbonent_form.textBox1.Text == "" ||
AddAbonent_form.textBox2.Text == "") || (AddAbonent_form.textBox3.Text == "" ||
AddAbonent_form.textBox4.Text == "")))
                        {
                            MessageBox.Show("Введіть не усі дані!", "Помилка");

                            Form1 gl_forma = new Form1();
                            gl_forma.Show();
                        }
                        else
```

Продовження Додатку В

```
                        {
                            nomer++;
                            SqlCommand command = connectionString.CreateCommand();

                            command.CommandText = "INSERT INTO TableAbonentu VALUES(" + nomer
+ ", '" + AddAbonent_form.textBox1.Text + "', '" +
                AddAbonent_form.textBox2.Text + "', '" + AddAbonent_form.textBox3.Text + "', '"
+ AddAbonent_form.textBox4.Text + "')";
                            Console.WriteLine(command.CommandText);
                            command.ExecuteNonQuery();

                            //
                            if (AddAbonent_form.checkBox1.Checked == true)
                            {
                                try
                                {
                                    command.CommandText = "INSERT INTO TableAdresa VALUES(" +
nomer + ", '" + AddAbonent_form.textBox5.Text + "', '" +
                 AddAbonent_form.textBox6.Text + "', '" + AddAbonent_form.textBox7.Text + "',
'" + AddAbonent_form.textBox8.Text + "', '" + AddAbonent_form.textBox9.Text + "', '" +
StrReplace(AddAbonent_form.textBox10.Text) + "', '" + nomer + "')";
                                    Console.WriteLine(command.CommandText);
                                    command.ExecuteNonQuery();

                                    AddFirstPokaznuk(nomer);
                                }
                                catch
                                {
                                    command.CommandText = "DELETE FROM TableAbonentu WHERE
id=" + nomer;
                                    Console.WriteLine(command.CommandText);
                                    command.ExecuteNonQuery();
                                    MessageBox.Show("Введені дані про адресу некоректні!",
"Помилка");
                                    connectionString.Close();
                                    AddAbonent_form.Close();
                                    this.Show();

                                }
                            }
                            //

                            AddAbonent_form.Close();
                            this.Show();
                            connectionString.Close();
                            RefreshAllTable();

                        }
                    }
                    if (AddAbonent_form.DialogResult == DialogResult.Cancel)
                    {
                        AddAbonent_form.Close();
                        this.Show();
                        connectionString.Close();
                    }
                }
                catch
                {
                    MessageBox.Show("Введені дані некоректні!", "Помилка");
```

Продовження Додатку В

```
                connectionString.Close();
                AddAbonent_form.Close();
                this.Show();
            }
        }
        catch { connectionString.Close(); }

    }

    private void button5_Click(object sender, EventArgs e)
    {
        try
        {

            Table1Save();
        connectionString.Open();

        int osobovuy;
        int IdAbonenta = Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value);


        SqlCommand count = new SqlCommand("Select id From TableAdresa WHERE vlasnuk=" +
IdAbonenta, connectionString);
        count.CommandText = "Select id From TableAdresa WHERE vlasnuk=" + IdAbonenta;
        try
        {
            osobovuy = Convert.ToInt32(count.ExecuteScalar().ToString());
        }
        catch { osobovuy = 1; }

        connectionString.Close();
        db_execute("DELETE FROM TableKvSm WHERE adresa=" + osobovuy);
        db_execute("DELETE FROM TableKvVoda WHERE adresa=" + osobovuy);
        db_execute("DELETE FROM TableOplata WHERE adresa=" + osobovuy);
        db_execute("DELETE FROM TablePokaznuku WHERE adresa=" + osobovuy);
        db_execute("DELETE FROM TableAdresa WHERE vlasnuk=" + osobovuy);
        db_execute("DELETE FROM TableAbonentu WHERE id=" + IdAbonenta);

        //refreshInfo();
        RefreshAllTable();
        table1buttons(true);
        }
        catch
        {
            connectionString.Close();
        }
    }

    private void button6_Click(object sender, EventArgs e)
    {
        try
        {
            if (dataGridView1.ReadOnly == true)
            {
                dataGridView1.ReadOnly = false;
                dataGridView1.Columns[0].ReadOnly = true;

            }
            else
```

Продовження Додатку В

```
                {
                    dataGridView1.ReadOnly = true;
                    dataGridView1.Columns[0].ReadOnly = true;

                }
            }
            catch
            {
            }
        }

        private void dataGridView3_CellContentClick(object sender, DataGridViewCellEventArgs
e)
        {

        }

        private async void ShowUser_Click(object sender, EventArgs e)
        {
            try
            {

            dataGridView2.Rows.Clear();
            connectionString.Open();
            SqlCommand command = new SqlCommand("Select
id,pynkt,streetname,bydunok,kvartura,kilk_osob,plowa,vlasnuk From TableAdresa",
connectionString);

            SqlDataReader reader = await command.ExecuteReaderAsync();
            if (reader.HasRows)
            {
                while (reader.Read())
                {
                    dataGridView2.Rows.Add(reader["id"], reader["pynkt"],
reader["streetname"], reader["bydunok"], reader["kvartura"], reader["kilk_osob"],
reader["plowa"], reader["vlasnuk"]);
                }
            }

            if (dataGridView2.Visible == true)
            {
                dataGridView2.Visible = false;
                TableAdresaButtons(false);
            }
            else
            {
                dataGridView2.Visible = true;
                TableAdresaButtons(true);
            }
            connectionString.Close();

            }
            catch
            {
                connectionString.Close();
            }
        }

        private void table1buttons(bool OffON)
```

Продовження Додатку В

```
        {
            button3.Enabled = OffON;
            button4.Enabled = OffON;
            button5.Enabled = OffON;
            button6.Enabled = OffON;        }

        private void TableAdresaButtons(bool OffON)
        {
            AddUser.Enabled = OffON;
            RemoveUser.Enabled = OffON;
            EditModeForUser.Enabled = OffON;
            SaveUser.Enabled = OffON;
        }

        private void refreshInfoAuth()
        {
            try
            {

                connectionString.Open();
            SqlCommand command = new SqlCommand("Select
id,pynkt,streetname,bydunok,kvartura,kilk_osob,plowa,vlasnuk From TableAdresa",
connectionString);



            dataGridView2.Rows.Clear();
            command.CommandText = "Select * From TableAdresa";
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                dataGridView2.Rows.Add(reader["id"], reader["pynkt"], reader["streetname"],
reader["bydunok"], reader["kvartura"], reader["kilk_osob"], reader["plowa"],
reader["vlasnuk"]);
            }
            reader.Close();
            connectionString.Close();

            }
            catch
            {
                connectionString.Close();
            }
        }


        private void AddUser_Click(object sender, EventArgs e)
        {
            try
            {
                SaveAllUser();

            connectionString.Open();
            int maxIDAdresa = 1, maxIDAbonentu = 1;
            SqlCommand command = connectionString.CreateCommand();

            try
            {
```

Продовження Додатку В

```
                command.CommandText = "SELECT max(id) FROM TableAdresa";
                maxIDAdresa = (Convert.ToInt32(command.ExecuteScalar().ToString())) + 1;

            }
            catch
            {
                maxIDAdresa = 1;
            }

            try
            {
                command.CommandText = "SELECT max(id) FROM TableAbonentu";
                maxIDAbonentu = (Convert.ToInt32(command.ExecuteScalar().ToString())) + 1;
            }
            catch
            {
                maxIDAbonentu = 1;
            }

            if (maxIDAdresa < maxIDAbonentu) maxIDAdresa = maxIDAbonentu;

            Table2Save();

            db_execute("INSERT INTO TableAdresa VALUES ('" + maxIDAdresa + "', 'Населений
пункт', 'Вулиця', '1', '1', '0', '0', '" + (maxIDAbonentu - 1) + "')");
            connectionString.Close();
            refreshInfoAuth();

            }
            catch
            {
                connectionString.Close();
            }
        }

        private void RemoveUser_Click(object sender, EventArgs e)
        {
            try
            {
                SaveAllUser();
            int tmp = Convert.ToInt32(dataGridView2.CurrentRow.Cells[0].Value);
            db_execute("DELETE FROM TableKvSm WHERE adresa=" + tmp);
            db_execute("DELETE FROM TableKvVoda WHERE adresa=" + tmp);
            db_execute("DELETE FROM TableOplata WHERE adresa=" + tmp);
            db_execute("DELETE FROM TablePokaznuku WHERE adresa=" + tmp);
            db_execute("DELETE FROM TableAdresa WHERE id=" + tmp);

            RefreshAllTable();
            dataGridView2.Visible = true;
            table1buttons(true);
            TableAdresaButtons(true);
                //refreshInfoAuth();
            }
            catch
            {
            }
        }
```

Продовження Додатку В

```
        private void SaveAllUser()
        {

            try
            {
                connectionString.Open();
                SqlCommand command = connectionString.CreateCommand();


                int kol = dataGridView2.Rows.GetRowCount(DataGridViewElementStates.Visible);
                for (int i = 0; i < kol; i++)
                {
                    int id = Convert.ToInt32(dataGridView2[0, i].Value);
                    string pynkt = Convert.ToString(dataGridView2[1, i].Value);
                    string streetname = Convert.ToString(dataGridView2[2, i].Value);
                    string bydunok = Convert.ToString(dataGridView2[3, i].Value);
                    string kvartura = Convert.ToString(dataGridView2[4, i].Value);
                    int kilk_osob = Convert.ToInt32(dataGridView2[5, i].Value);
                    string plowa = Convert.ToString(dataGridView2[6, i].Value);
                    int vlasnuk = Convert.ToInt32(dataGridView2[7, i].Value);
                    plowa = StrReplace(plowa);

                    command.CommandText = "UPDATE TableAdresa SET pynkt='" + pynkt + "',
streetname='" + streetname + "', bydunok='" + bydunok + "', kvartura='" + kvartura + "',
kilk_osob='" + kilk_osob + "', plowa='" + plowa + "', vlasnuk='" + vlasnuk + "' WHERE id=" +
id;
                    //Console.WriteLine(command.CommandText);
                    command.ExecuteNonQuery();
                }

            }
            catch
            {
                MessageBox.Show("Некоректні дані!", "Помилка", MessageBoxButtons.OK);
            }
            connectionString.Close();
        }

        private void SaveUser_Click(object sender, EventArgs e)
        {
            SaveAllUser();
        }

        private void EditModeForUser_Click(object sender, EventArgs e)
        {
            try
            {
                if (dataGridView2.ReadOnly == true)
                {
                    dataGridView2.ReadOnly = false;
                   // dataGridView2.Columns[0].ReadOnly = true;
                }
                else
                {
                    dataGridView2.ReadOnly = true;
                   // dataGridView2.Columns[0].ReadOnly = true;
                }
            }
```

Продовження Додатку В

```
            catch
            {
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.Application.Exit();
        }

        private void table2buttons(bool OFFON)
        {
            //peremuliv_rozrah.Enabled = OFFON;
            button9.Enabled = OFFON;
            button10.Enabled = OFFON;
            button11.Enabled = OFFON;
            button12.Enabled = OFFON;
            //label5.Visible = OFFON;
            //label6.Visible = OFFON;
            //label7.Visible = OFFON;
            //label8.Visible = OFFON;
        }

        private async void PokaznukuShowTable()
        {
            try
            {
                dataGridView3.Rows.Clear();
                connectionString.Open();
                SqlCommand command = new SqlCommand("Select id,pokaznuk,data,period,adresa
From TablePokaznuku", connectionString);
               SqlDataReader reader = await command.ExecuteReaderAsync();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        dataGridView3.Rows.Add(reader["id"], reader["pokaznuk"],
((DateTime)reader["data"]).ToShortDateString(), reader["period"], reader["adresa"]);
                    }
                }

                if (dataGridView3.Visible == true)
                {
                    dataGridView3.Visible = false;
                    table2buttons(false);
                }
                else
                {
                    dataGridView3.Visible = true;
                    table2buttons(true);
                }
            }
            catch { }
            connectionString.Close();
        }

        private void button8_Click(object sender, EventArgs e)
        {
```

Продовження Додатку В

```
            PokaznukuShowTable();
        }

        private void refreshinfotable2()
        {
            try
            {

                connectionString.Open();
            SqlCommand command = new SqlCommand("Select * From TablePokaznuku",
connectionString);

            dataGridView3.Rows.Clear();
            command.CommandText = "Select * From TablePokaznuku";
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                dataGridView3.Rows.Add(reader["id"], reader["pokaznuk"],
((DateTime)reader["data"]).ToShortDateString(), reader["period"], reader["adresa"]);
            }
            reader.Close();

            }
            catch
            {
            }
            connectionString.Close();
        }

        private void button9_Click(object sender, EventArgs e)
        {
            try
            {
                connectionString.Open();
            int rowid, osob_rah;
            SqlCommand command = connectionString.CreateCommand();

            command.CommandText = "SELECT max(id) FROM TablePokaznuku";
            try
            {
                rowid = (Convert.ToInt32(command.ExecuteScalar().ToString())) + 1;
            }
            catch { rowid = 1; }

            Console.WriteLine("rowid: " + rowid);

            command.CommandText = "SELECT max(id) FROM TableAdresa";
            try
            {
                osob_rah = Convert.ToInt32(command.ExecuteScalar().ToString());
            }
            catch { osob_rah = 1; }

            connectionString.Close();
            Table2Save();
            db_execute("INSERT INTO TablePokaznuku VALUES ('" + rowid + "', '0', '" +
DateTime.Today.ToString("dd.MM.yyyy") + "', 'Січень', '" + osob_rah + "')");
            refreshinfotable2();
```

Продовження Додатку В

```
    }
    catch
    {
        connectionString.Close();
    }
}

private void button10_Click(object sender, EventArgs e)
{
    try
    {
        Table2Save();
        int tmp = Convert.ToInt32(dataGridView3.CurrentRow.Cells[0].Value);
        db_execute("DELETE FROM TablePokaznuku WHERE id=" + tmp);
        refreshinfotable2();
    }
    catch
    {
    }
}

private void button11_Click(object sender, EventArgs e)
{
    try
    {
        if (dataGridView3.ReadOnly == true)
        {
            dataGridView3.ReadOnly = false;
            dataGridView3.Columns[0].ReadOnly = true;
        }
        else
        {
            dataGridView3.ReadOnly = true;
            dataGridView3.Columns[0].ReadOnly = true;
        }
    }
    catch
    {
    }
}

private void Table2Save()
{
    try
    {
        connectionString.Open();
        SqlCommand command = connectionString.CreateCommand();

        int kol = dataGridView3.Rows.GetRowCount(DataGridViewElementStates.Visible);
        for (int i = 0; i < kol; i++)
        {
            int id = Convert.ToInt32(dataGridView3[0, i].Value);
            int pokaznuk = Convert.ToInt32(dataGridView3[1, i].Value);
            string data = Convert.ToString(dataGridView3[2, i].Value);
            string period = Convert.ToString(dataGridView3[3, i].Value);
            int adresa = Convert.ToInt32(dataGridView3[4, i].Value);
```

Продовження Додатку В

```
                    command.CommandText = "UPDATE TablePokaznuku SET pokaznuk='" + pokaznuk +
"', data='" + data + "', period='" + period + "', adresa='" + adresa + "' WHERE id=" + id;

                    Console.WriteLine(command.CommandText);

                    command.ExecuteNonQuery();

                }
            }
            catch
            {
                connectionString.Close();
            }
            connectionString.Close();
        }

        private void button12_Click(object sender, EventArgs e)
        {
            Table2Save();
        }

        private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.Application.Exit();
        }

        private void інформаціяПроПрограмуToolStripMenuItem_Click(object sender, EventArgs e)
        {

            info forminf = new info();
            forminf.Show();
        }
        private void hlopivkaExit_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.Application.Exit();
        }
        private string NumberMounthToNameMounth(int mounth)
        {
            string returnZN = "Січень";
            switch (mounth)
            {
                case 1: returnZN = "Січень"; break;
                case 2: returnZN = "Лютий"; break;
                case 3: returnZN = "Березень"; break;
                case 4: returnZN = "Квітень"; break;
                case 5: returnZN = "Травень"; break;
                case 6: returnZN = "Червень"; break;
                case 7: returnZN = "Липень"; break;
                case 8: returnZN = "Серпень"; break;
                case 9: returnZN = "Вересень"; break;
                case 10: returnZN = "Жовтень"; break;
                case 11: returnZN = "Листопад"; break;
                case 12: returnZN = "Грудень"; break;
                default: break;
            }
            return returnZN;
        }
```

Продовження Додатку В

```
        private void AddFirstPokaznuk(int osob_rah)
        {
            string misyac = NumberMounthToNameMounth((DateTime.Now).Month);

            try
            {
                int rowid;
                SqlCommand command = connectionString.CreateCommand();

                command.CommandText = "SELECT max(id) FROM TablePokaznuku";
                try
                {
                    rowid = (Convert.ToInt32(command.ExecuteScalar().ToString())) + 1;
                }
                catch { rowid = 1; }
                connectionString.Close();
                Console.WriteLine("Zaput:  " + "INSERT INTO TablePokaznuku VALUES ('" + rowid
+ "', '0', '" + DateTime.Today.ToString("dd.MM.yyyy") + "', '" + Mounthminus1(misyac) + "', '"
+ osob_rah + "')");
                db_execute("INSERT INTO TablePokaznuku VALUES ('" + rowid + "', '0', '" +
DateTime.Today.ToString("dd.MM.yyyy") + "', '" + Mounthminus1(misyac) + "', '" + osob_rah +
"')");
            }
            catch
            {
            }
        }
        private void table3buttons(bool OFFON)
        {
            hlopivkaAdd.Enabled = OFFON;
            hlopivkaRemove.Enabled = OFFON;
            hlopivkaEdit.Enabled = OFFON;
            hlopivkaSave.Enabled = OFFON;
        }
        private async void PosluguShowTable()
        {
            try
            {
                dataGridView4.Rows.Clear();
                connectionString.Open();
                SqlCommand command = new SqlCommand("Select * From TablePoslugu",
connectionString);
               SqlDataReader reader = await command.ExecuteReaderAsync();
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        dataGridView4.Rows.Add(reader["id"], reader["name"], reader["taruf"]);
                    }
                }
                if (dataGridView4.Visible == true)
                {
                    dataGridView4.Visible = false;
                    table3buttons(false);
                }
                else
                {
                    dataGridView4.Visible = true;
```

Продовження Додатку В

```
                    table3buttons(true);
                }
            }
            catch { }
            connectionString.Close();
        }
        private void hlopivkaShow_Click(object sender, EventArgs e)
        {
            PosluguShowTable();
        }
        private void PosluguSave()
        {
            try
            {
                connectionString.Open();
                SqlCommand command = connectionString.CreateCommand();
            int kol = dataGridView4.Rows.GetRowCount(DataGridViewElementStates.Visible);
            for (int i = 0; i < kol; i++)
            {
                int id = Convert.ToInt32(dataGridView4[0, i].Value);
                string name = Convert.ToString(dataGridView4[1, i].Value);
                string taruf = Convert.ToString(dataGridView4[2, i].Value);
                taruf = StrReplace(taruf);
                command.CommandText = "UPDATE TablePoslugu SET name='" + name + "', taruf='" +
taruf + "' WHERE id=" + id;
                Console.WriteLine(command.CommandText);
                command.ExecuteNonQuery();
            }
            connectionString.Close();
            }
            catch
            {
                MessageBox.Show("Некоректні дані!", "Помилка", MessageBoxButtons.OK);
connectionString.Close();
            }
        }
        public void RefreshPuslugu()
        {
            try
            {
                connectionString.Open();
            SqlCommand command = new SqlCommand("Select * From TablePoslugu",
connectionString);

            dataGridView4.Rows.Clear();
            command.CommandText = "Select * From TablePoslugu";
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                dataGridView4.Rows.Add(reader["id"], reader["name"], reader["taruf"]);
            }
            reader.Close();
            connectionString.Close();
            }
            catch
            {
                connectionString.Close();
            }
```

Продовження Додатку В

```
        }
        private void hlopivkaAdd_Click(object sender, EventArgs e)
        {
            try
            {
            PosluguSave();
            int last_id;
            int row_count = dataGridView4.Rows.GetRowCount(DataGridViewElementStates.Visible);
            if (dataGridView4.Rows.GetRowCount(DataGridViewElementStates.Visible) == 0)
last_id = 0;
            else
            {
                last_id = Convert.ToInt32(dataGridView4.Rows[row_count - 1].Cells[0].Value);
            }
            db_execute("INSERT INTO TablePoslugu VALUES ('" + (last_id + 1) + "', 'Назва
послуги', '" + 1 + "')");
            RefreshPuslugu();
            }
            catch
            {
            }
        }
```

Додаток Г

Публікація студентській науково-технічній конференції

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

25-26 квітня 2024 р.

(збірник тез конференції)

Тернопіль 2024

ББК 72+34 (Укр)
М34

Матеріали VII Міжнародної студентської науково - технічної конференції / Тернопіль: Тернопільський національний технічний університет ім. І.Пулюя (м. Тернопіль, 25-26 квітня 2024 р.), 2024.- 308 с.

В збірнику друкуються матеріали VII Міжнародної студентської науково-технічної конференції. Тернопіль. – ТНТУ ім. І. Пулюя (25-26 квітня 2024 р.) за наступними науковими напрямками:
культура і мистецтво; гуманітарні науки; соціальні та поведінкові науки; управління та адміністрування; природничі науки; математика та статистика; інформаційні технології; механічна інженерія; електрична інженерія; автоматизація та приладобудування; хімічна та біоінженерія; електроніка та телекомунікації; виробництво та технології; архітектура та будівництво; аграрні науки та продовольство; сфера обслуговування; транспорт.

Адреса конференції:
46001, м. Тернопіль, вул. Руська, 56
Тернопільський національний технічний університет ім. Івана Пулюя
e-mail: snt@tntu.edu.ua
Тернопільський національний технічний університет ім. Івана Пулюя

УДК 004.41

Музика В. – ст. гр. СПс-41

Тернопільський національний технічний університет імені Івана Пулюя

АСИНХРОННІСТЬ ПРОГРАМУВАННЯ В РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Науковий керівник: канд. техн. наук., доц. Михалик Д. М.

Muzyka V. – postgraduate

Ternopil Ivan Puluj National Technical University

ASYNCHRONOUS PROGRAMMING IN SOFTWARE DEVELOPMENT

Supervisor: Assoc. Prof. Dr. Myhalyk D. M.

Ключові слова: Асинхронність, програмування, розробка.

Key words: Asynchrony, programming, development.

У розробці програмного забезпечення, асинхронне програмування відіграє значну роль, коли йдеться про написання гнучкого, продуктивного, та багатопотокового коду. Існує безліч переваг використання асинхронних функцій та методів, тому кожен сучасний розробник повинен знати та вміти використовувати їх в програмуванні.

Цей підхід до програмування став широко використовуваним завдяки появі асинхронних програмних інтерфейсів (API) та бібліотек, що його підтримують, а також завдяки розвитку мов програмування, які надають зручні інструменти для асинхронного програмування.

Суть асинхронного програмування полягає в здатності програми виконувати різні операції паралельно, без чекання завершення попередніх операцій перед продовженням виконання.

Таким чином, програми можуть ефективно використовувати час та ресурси, забезпечуючи швидке та реактивне виконання операцій навіть у випадках, коли деякі операції займають багато часу або потребують зовнішніх даних.

Основні аспекти асинхронного програмування, які стають ключовими для розробки програмного забезпечення:

1. Неблокуючі операції: замість блокування виконання коду під час очікування на завершення певних операцій (наприклад, вводу-виводу або мережевих запитів), програма може переходити до виконання інших завдань.

2. Паралельне виконання: асинхронне програмування дозволяє виконувати різні частини програми паралельно, тобто одночасно, забезпечуючи більш ефективне використання ресурсів системи.

3. Обробка подій та зворотні виклики: результати асинхронних операцій зазвичай передаються через зворотні виклики або обробляються як події. Це дозволяє програмі бути реактивною на події та зміни, що відбуваються у середовищі виконання.

Література

1. Albahari J., Albahari B. C# 7.0 in a Nutshell: The Definitive Reference / O'Reilly Media, 2017. – 1072 с.

2. Asynchronous programming with async and await (C#) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/.

3. Freeman A. Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming. O'Reilly Media, 2014. – 208 с.

4. Albahari J., Albahari B. Threading in C# [Електронний ресурс] – Режим доступу до ресурсу: https://www.albahari.com/threading/

Додаток Д

Диск із кваліфікаційною роботою бакалавра