МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи освітнього ступеня «фаховий молодший бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
на тему:

«Програма для автоматизації обліку комунальних платежів»

Виконав здобувач освіти групи П-42
Харчук Нікіта Дмитрович

Керівник роботи:
Устименко Ярослав Іванович

Рецензент:
Устименко Людмила Миколаївна

Житомир – 2025

Зміст

РЕФЕРАТ

Дипломна робота на тему «Програма для автоматизації обліку комунальних платежів» (57 сторінок, 19 рисунків, 1 додаток, 27 джерел) присвячена розробці настільного додатка для управління комунальними платежами з використанням Windows Forms, .NET 8 та SQLite. Метою є створення зручного інструменту для автоматизації введення, зберігання та аналізу даних про платежі.

У роботі проаналізовано аналоги, що підтвердило потребу в спеціалізованому рішенні. Обґрунтовано вибір технологій: Windows Forms для інтерфейсу, .NET 8 для продуктивності, SQLite для локального зберігання. Спроєктовано базу даних із таблицями Services і Bills, реалізовано класи (Service, UtilityBill, MainForm, ServiceForm) та алгоритми (лінійна фільтрація, QuickSort). Програма підтримує додавання, редагування, видалення платежів і послуг, автоматичне заповнення тарифів і показників, фільтрацію та сортування.

Інтерфейс є інтуїтивним, із динамічним відображенням полів. Інструкція описує встановлення та роботу з програмою. Системні вимоги мінімальні: Windows 10, 2 ГБ ОЗП, 100 МБ на диску. Програма перевершує аналоги за спеціалізацією та простотою, придатна для домашнього використання. Перспективи розвитку включають аналітику та портування на інші платформи.

Ключові слова: автоматизація, комунальні платежі, Windows Forms, .NET 8, SQLite, облік, інтерфейс користувача.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – Система управління базами даних.

SQL – Structured Query Language (мова структурованих запитів), мова для роботи з базами даних.

GUI – Graphical User Interface (графічний інтерфейс користувача).

.NET – Платформа розробки від Microsoft для створення додатків.

Windows Forms – Технологія для створення графічних інтерфейсів у Windows.

SQLite – Легка вбудована реляційна база даних.

O(n) – Позначення часової складності алгоритму, де n – розмір вхідних даних.

O(log n) – Позначення логарифмічної часової складності.

O(n log n) – Позначення часової складності алгоритмів сортування, таких як QuickSort.

API – Application Programming Interface (інтерфейс програмування додатків).

CSV – Comma-Separated Values (формат файлу для зберігання табличних даних).

ID – Ідентифікатор, унікальний номер для запису в базі даних.

UI – User Interface (інтерфейс користувача).

ООП – Об’єктно-орієнтоване програмування.

C – Мова програмування, використана в .NET.

ВСТУП

Сучасне суспільство характеризується високим рівнем автоматизації та цифровізації повсякденних процесів, що значно полегшує управління побутовими завданнями. Одним із таких завдань є облік комунальних платежів, який включає фіксацію витрат на електроенергію, газ, воду, опалення та інші послуги. Для приватних домогосподарств цей процес часто є трудомістким через необхідність ручного збору даних, розрахунків та контролю своєчасності оплат. Відсутність зручного інструменту для автоматизації обліку може призводити до помилок, втрати часу та фінансових незручностей.

Розвиток інформаційних технологій відкриває широкі можливості для створення програмних рішень, здатних оптимізувати облік комунальних платежів. Використання сучасних технологій, таких як Windows Forms для створення зручного графічного інтерфейсу, платформа .NET 8 для забезпечення продуктивності та масштабованості, а також SQLite як легкої та ефективної бази даних, дозволяє розробити надійну програму, адаптовану до потреб користувачів. Такі рішення забезпечують автоматизацію введення даних, зберігання історії платежів, фільтрацію за різними критеріями та швидкий доступ до актуальної інформації.

Актуальність теми дипломної роботи зумовлена зростаючою потребою в ефективних інструментах для управління особистими фінансами, зокрема в сфері комунальних платежів. Автоматизація цього процесу не лише економить час, але й сприяє підвищенню фінансової дисципліни та точності розрахунків. Розробка програми, яка поєднує простоту використання, функціональність і доступність, є важливим кроком у напрямку цифровізації побутового менеджменту.

Мета роботи полягає в розробці програми для автоматизації обліку комунальних платежів на основі технологій Windows Forms, .NET 8 та SQLite, яка забезпечить зручне введення, зберігання, обробку та аналіз даних про комунальні платежі.

Завдання роботи:

1. Проаналізувати вимоги до програмного забезпечення для обліку комунальних платежів.
2. Розробити архітектуру програми з урахуванням потреб користувачів.
3. Реалізувати графічний інтерфейс користувача на основі Windows Forms.
4. Інтегрувати базу даних SQLite для зберігання інформації про послуги та платежі.
5. Забезпечити функціонал для додавання, редагування, видалення та фільтрації платежів.
6. Реалізувати автоматичне заповнення даних, таких як останні показники лічильників і тарифи.
7. Провести тестування програми та оцінити її ефективність.

Об'єкт дослідження – процес автоматизації обліку комунальних платежів.

Предмет дослідження – програмне забезпечення, розроблене на основі Windows Forms, .NET 8 та SQLite для управління даними про комунальні платежі.

Наукова новизна полягає в розробці спеціалізованого програмного рішення, яке поєднує інтуїтивно зрозумілий інтерфейс, ефективну обробку даних і підтримку автоматизації типових операцій для обліку комунальних платежів, адаптованого до потреб сучасних користувачів.

Практична цінність роботи полягає в створенні програми, яка може бути використана домогосподарствами для спрощення управління комунальними платежами, зменшення ймовірності помилок і підвищення ефективності фінансового планування.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка програми для автоматизації обліку комунальних платежів спрямована на створення зручного інструменту для управління фінансовими операціями, пов’язаними з оплатою комунальних послуг. Основна задача полягає в забезпеченні користувача функціоналом для введення, зберігання, обробки та аналізу даних про платежі, включаючи підтримку різних типів послуг, таких як фіксовані та лічильникові. Програма має бути інтуїтивно зрозумілою, ефективною та доступною для використання в домашніх умовах.

Ключовим аспектом є автоматизація рутинних операцій. Для фіксованих послуг програма повинна автоматично підставляти тарифи, визначені в базі даних, а для лічильникових — використовувати останні показники лічильників для розрахунків. Важливо забезпечити гнучкість у додаванні, редагуванні та видаленні даних про послуги та платежі, а також можливість фільтрації платежів за місяцем, роком чи типом послуги. Додатково програма має відображати загальну суму витрат, що сприяє фінансовому плануванню.

Для реалізації поставленої задачі обрано технології Windows Forms, .NET 8 та SQLite. Windows Forms забезпечує створення графічного інтерфейсу, зрозумілого для користувачів Windows, з підтримкою елементів управління, таких як таблиці, текстові поля та кнопки. Платформа .NET 8 пропонує сучасні засоби розробки, включаючи високу продуктивність, підтримку об’єктно-орієнтованого програмування та інтеграцію з базами даних. SQLite, як легка вбудована база даних, дозволяє ефективно зберігати та обробляти дані без необхідності встановлення серверного програмного забезпечення, що ідеально підходить для локального застосування.

Основні вимоги до програми включають створення бази даних із таблицями для зберігання інформації про послуги та платежі, розробку інтерфейсу для управління цими даними, забезпечення коректної обробки введених даних та їх валідації. Наприклад, для лічильникових послуг необхідно перевіряти, щоб поточні показники були більшими за попередні, а

для фіксованих — щоб сума була додатною. Програма також має підтримувати сортування платежів за різними критеріями, такими як дата чи сума, для зручності аналізу.

Задача розробки полягає в створенні програмного забезпечення, яке поєднує зручний інтерфейс, надійне зберігання даних і автоматизацію розрахунків, що відповідає потребам користувачів у сфері обліку комунальних платежів. Використання Windows Forms, .NET 8 та SQLite забезпечує оптимальний баланс між простотою, функціональністю та продуктивністю.

1.2. Огляд та порівняння аналогічних систем.

Для розробки програми автоматизації обліку комунальних платежів важливо проаналізувати існуючі рішення, щоб визначити їхні переваги, недоліки та сформувати унікальні особливості власного продукту. На ринку існує кілька програм і сервісів, які пропонують функціонал для управління комунальними платежами, включаючи як локальні, так і хмарні рішення. Нижче розглянуто три основні аналоги, їхні можливості та порівняння з розроблюваною системою.

"Домашня бухгалтерія" - популярна програма для Windows, призначена для обліку особистих фінансів. Вона дозволяє вести облік комунальних платежів, створювати категорії витрат, вводити дані про послуги та зберігати історію транзакцій. Програма підтримує імпорт даних із банківських виписок і формування звітів. Однак її інтерфейс виглядає застарілим, а функціонал для комунальних платежів обмежений: відсутня автоматична обробка показників лічильників, а для фіксованих послуг користувач мусить вручну вводити суми. Крім того, програма платна, що може бути бар’єром для окремих користувачів.

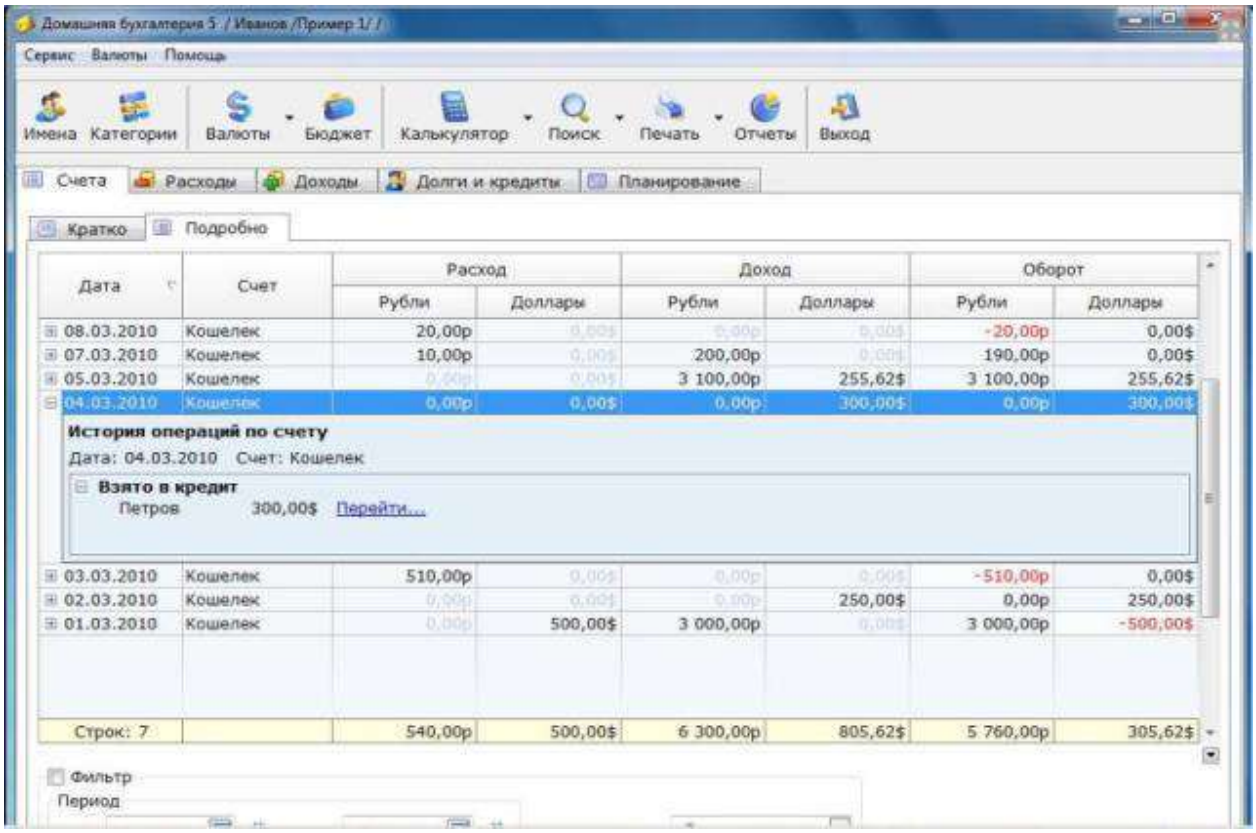

Рисунок 1.2.1 – Домашня бухгалтерія

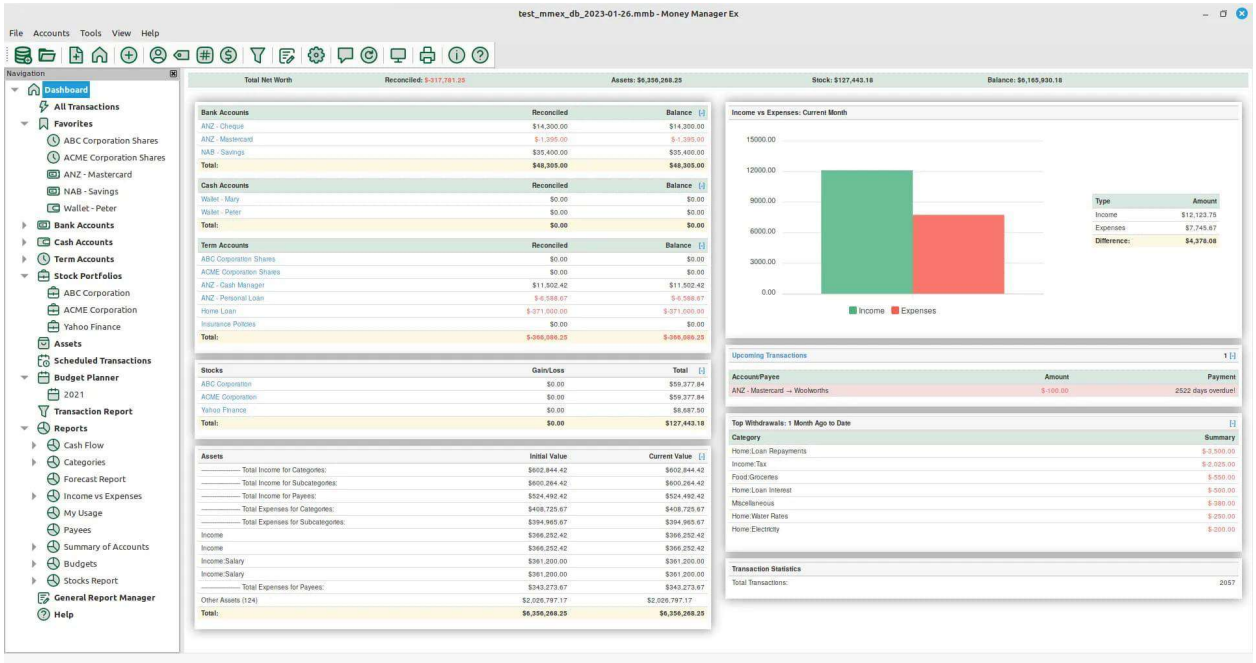

Рисунок 1.2.2 – Money Manager Ex

"Money Manager Ex" - кросплатформне програмне забезпечення з відкритим кодом. Воно пропонує гнучке управління фінансами, включаючи облік комунальних платежів, підтримку різних валют і створення графіків

витрат. Програма дозволяє зберігати дані локально або в хмарі, що забезпечує доступність. Проте "Money Manager Ex" не має спеціалізованих функцій для автоматизації комунальних платежів, таких як підстановка тарифів чи останніх показників лічильників. Налаштування програми вимагає часу, а її інтерфейс може бути складним для початківців.

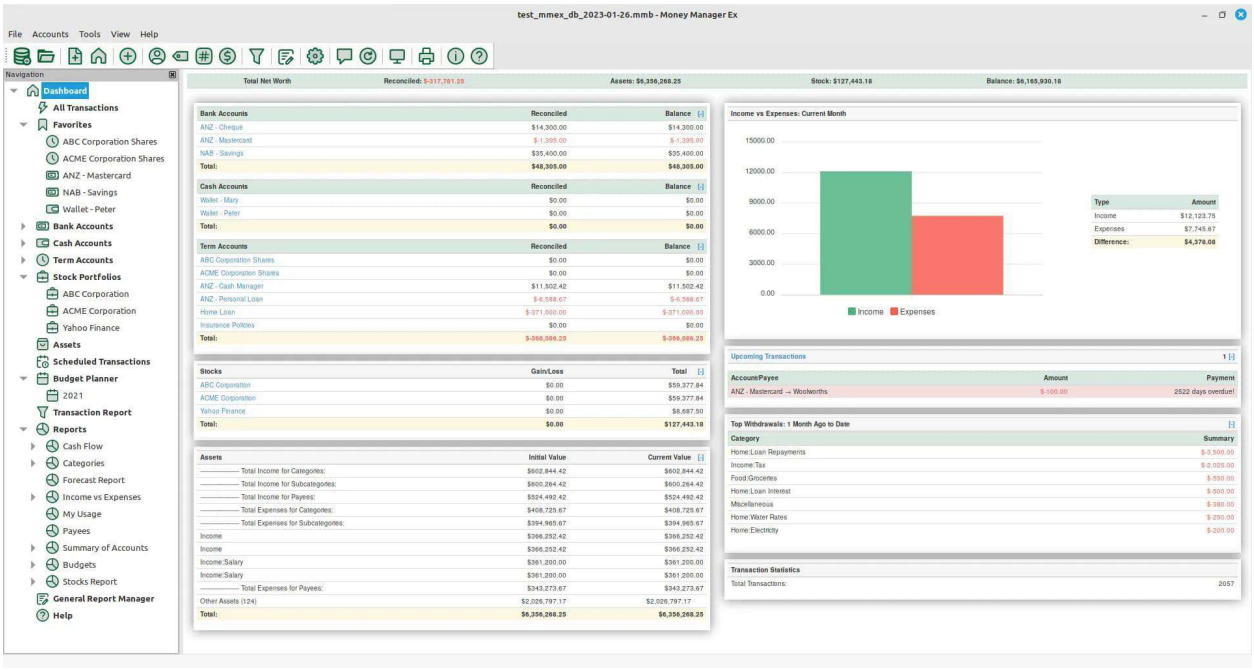

Рисунок 1.2.3 – Money Manager Ex

Онлайн-сервіс **"EasyPay"**, який функціонує як платіжна платформа з можливістю збереження історії комунальних платежів. Користувачі можуть оплачувати послуги через вебінтерфейс, отримувати нагадування про платежі та переглядати статистику. Сервіс зручний для онлайн-оплат, але не підтримує локальне зберігання даних і не пропонує функцій для детального обліку, таких як розрахунок на основі показників лічильників чи сортування платежів за різними критеріями. Залежність від інтернету та періодичні комісії за транзакції також є обмеженнями.

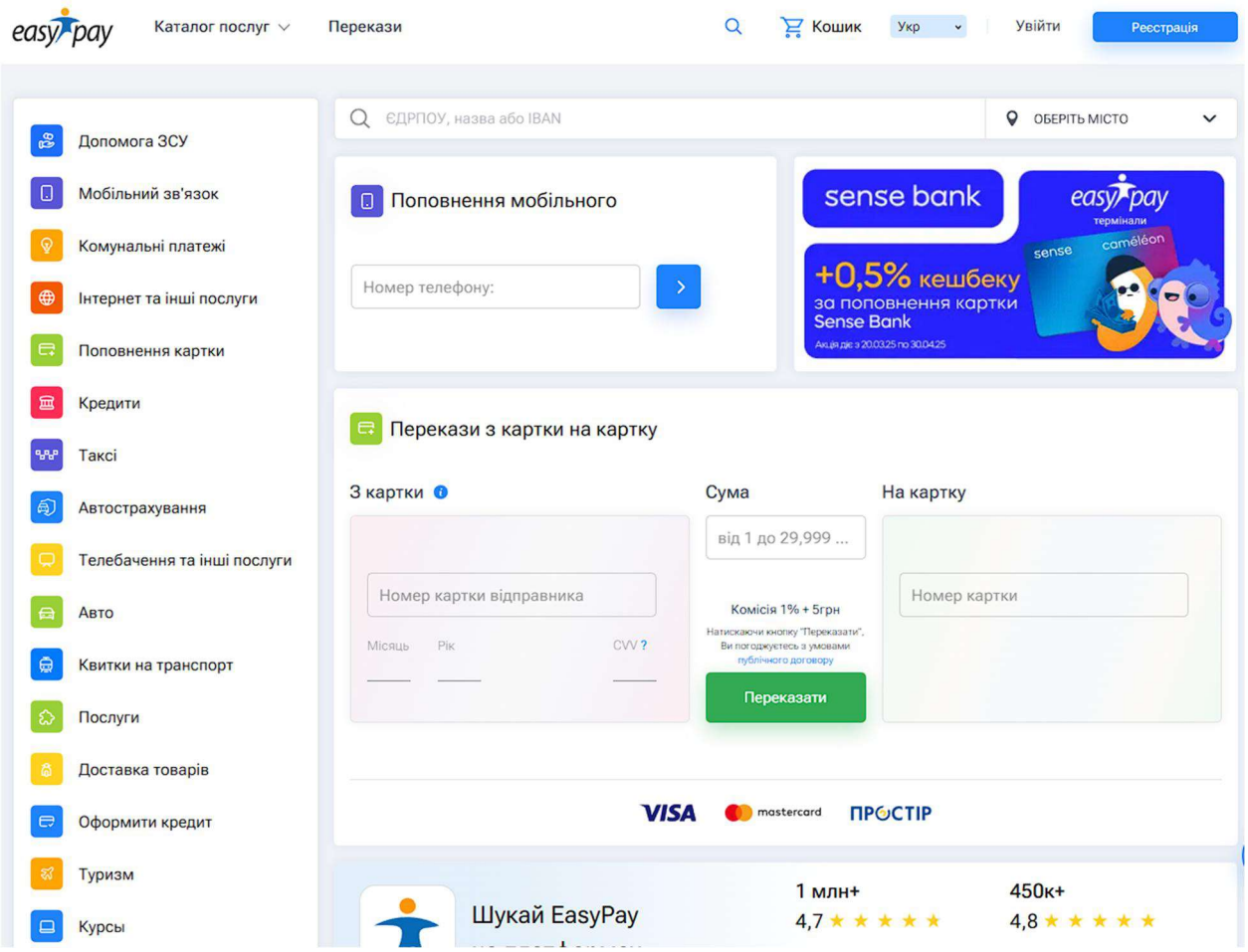

Рисунок 1.2.4 – Онлайн-сервіс "EasyPay",

Програма **HomeBank**, безкоштовне програмне забезпечення для управління особистими фінансами. HomeBank дозволяє вести облік витрат, включаючи комунальні платежі, за допомогою категорій і тегів. Вона підтримує імпорт даних із банківських виписок і створення звітів. Проте програма не має спеціалізованого функціоналу для комунальних платежів, такого як автоматичне заповнення показників лічильників чи тарифів, що ускладнює її використання для вузької задачі. HomeBank працює на платформі GTK+ і підтримує Windows, Linux та macOS, але її інтерфейс може здаватися застарілим для користувачів, звиклих до сучасних рішень.

Рисунок 1.2.5 – Програма HomeBank

Мобільний додаток **Paytm** (доступний для Android та iOS), який популярний у країнах із розвиненою системою онлайн-платежів. Paytm дозволяє оплачувати комунальні послуги, зберігати історію транзакцій і нагадувати про терміни платежів. Додаток інтегрується з постачальниками послуг, автоматично отримуючи дані про тарифи. Однак його залежність від інтернет-з'єднання та орієнтація на онлайн-оплату обмежують використання в умовах локального обліку. Крім того, Paytm не підтримує детальний аналіз показників лічильників, а його інтерфейс перевантажений додатковими функціями, що може відволікати користувача.

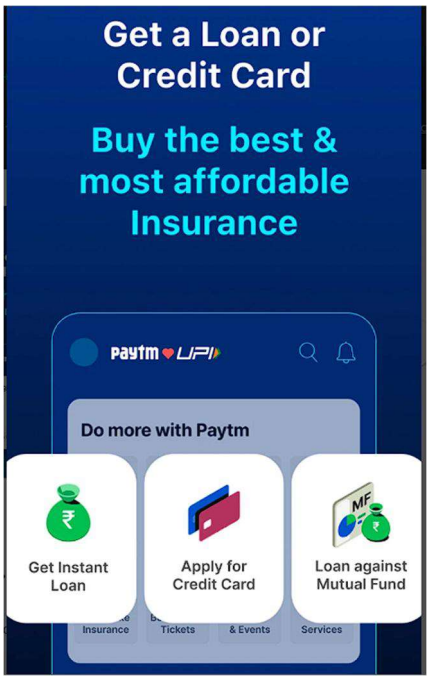

Рисунок 1.2.6 – Мобільний додаток Paytm

Microsoft Excel як універсальний інструмент для обліку. Багато користувачів створюють власні таблиці для фіксації комунальних платежів, використовуючи формули для розрахунків. Excel забезпечує гнучкість у налаштуванні, але вимагає від користувача навичок роботи з таблицями та ручного введення даних. Відсутність автоматизації, такої як інтеграція з базою даних чи автоматичне заповнення тарифів, робить його менш ефективним для регулярного використання. Excel працює на всіх платформах, але не є спеціалізованим рішенням.

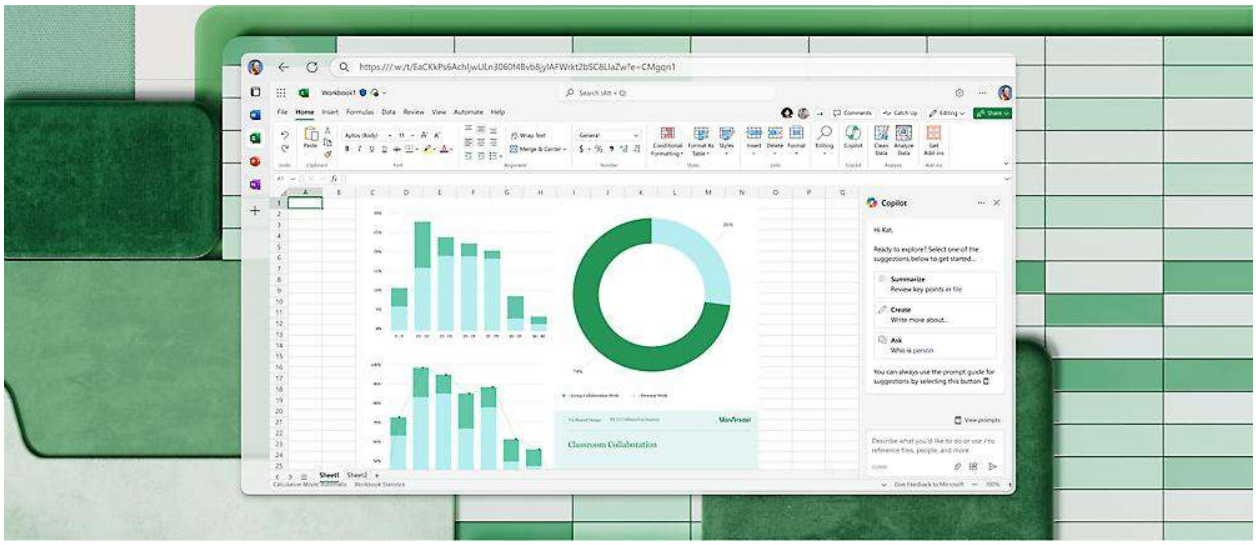

Рисунок 1.2.7 – Microsoft Excel

Порівняно з "Домашньою бухгалтерією", розроблювана система пропонує сучасніший інтерфейс і спеціалізовану автоматизацію для комунальних платежів. У порівнянні з "Money Manager Ex", вона простіша в налаштуванні та використанні, з акцентом на конкретну задачу. На відміну від "EasyPay", програма працює офлайн і не потребує додаткових витрат на транзакції. Однак, на відміну від хмарних рішень, вона не підтримує синхронізацію між пристроями, що може бути обмеженням для деяких користувачів. HomeBank підходить для загального фінансового обліку, але бракує специфічних функцій для комунальних платежів. Paytm зручний для онлайн-оплат, але не адаптований для локального управління без інтернету. Excel пропонує гнучкість, але потребує значних зусиль для автоматизації. Жодна з розглянутих систем не поєднує зручний графічний інтерфейс,

локальне зберігання даних і спеціалізований функціонал для обліку комунальних платежів із підтримкою лічильникових і фіксованих послуг.

Розроблювана програма на основі Windows Forms, .NET 8 та SQLite має переваги, які усувають недоліки аналогів. Windows Forms забезпечує інтуїтивно зрозумілий інтерфейс, схожий на стандартні Windows-додатки, що полегшує освоєння для користувачів. SQLite дозволяє зберігати дані локально без необхідності серверної інфраструктури, на відміну від Paytm. Функціонал автоматичного заповнення показників лічильників і тарифів, а також фільтрація та сортування платежів, робить програму більш спеціалізованою порівняно з HomeBank і Excel. Використання .NET 8 гарантує високу продуктивність і можливість подальшого розширення функціоналу.

Аналіз аналогів підтверджує доцільність розробки програми, яка поєднує локальне зберігання даних, автоматизацію розрахунків і зручний інтерфейс, орієнтований на облік комунальних платежів. Це рішення заповнює прогалину між універсальними фінансовими менеджерами та вузькоспеціалізованими додатками, відповідаючи потребам сучасних користувачів.

1.3. Вибір та обґрунтування технологій розробки

Для реалізації програми автоматизації обліку комунальних платежів обрано технології Windows Forms, .NET 8 та SQLite. Вибір цих інструментів ґрунтується на їх відповідності вимогам проекту, зокрема зручності інтерфейсу, ефективності обробки даних, простоті розгортання та адаптованості до потреб локального застосування.

Windows Forms є технологією для створення графічних інтерфейсів у середовищі Windows. Вона пропонує широкий набір елементів управління, таких як текстові поля, списки, таблиці та кнопки, що дозволяють швидко розробити інтуїтивно зрозумілий інтерфейс. Windows Forms ідеально підходить для настільних додатків, оскільки забезпечує знайомий для користувачів вигляд, подібний до стандартних програм Windows. Її перевагою

є простота інтеграції з .NET, підтримка подієво-орієнтованого програмування та наявність візуального дизайнера в середовищі розробки, що прискорює створення інтерфейсу. Для програми, орієнтованої на домашніх користувачів, Windows Forms забезпечує баланс між функціональністю та легкістю освоєння.

Платформа .NET 8 обрана як основа для розробки завдяки своїм сучасним можливостям і високій продуктивності. .NET 8 підтримує об’єктно-орієнтоване програмування, що спрощує створення модульного та зрозумілого коду. Вона забезпечує надійну роботу з базами даних через бібліотеки, такі як System.Data.SQLite, і пропонує інструменти для обробки винятків, валідації даних та оптимізації продуктивності. Крім того, .NET 8 є кросплатформною, що залишає можливість майбутнього портування програми на інші операційні системи, хоча поточний проект орієнтований на Windows. Висока продуктивність і підтримка сучасних стандартів C# роблять .NET 8 оптимальним вибором для створення стабільного та масштабованого додатка.

SQLite використовується як система управління базами даних через її легкість, ефективність і відсутність необхідності в серверній інфраструктурі. SQLite є вбудованою базою даних, що зберігає всі дані в одному файлі, що ідеально підходить для локальних додатків, таких як програма для обліку комунальних платежів. Вона підтримує стандартні SQL-запити, забезпечує швидкий доступ до даних і має низькі вимоги до ресурсів. SQLite також сумісна з .NET через бібліотеку System.Data.SQLite, що спрощує інтеграцію. Для зберігання інформації про послуги та платежі SQLite забезпечує достатню функціональність без ускладнення розгортання, на відміну від серверних СУБД, таких як MySQL чи PostgreSQL.

Альтернативні технології, такі як WPF для інтерфейсу чи ASP.NET для веб-додатка, були розглянуті, але відхилені. WPF, хоча й пропонує сучасніший вигляд інтерфейсу, є складнішим у реалізації та вимагає більше ресурсів, що невиправдано для настільного додатка з базовим функціоналом. ASP.NET

підходить для веб-розробки, але потребує інтернет-з’єднання та серверної інфраструктури, що суперечить вимозі локального використання. Аналогічно, інші бази даних, такі як Microsoft SQL Server, ускладнюють розгортання через необхідність встановлення серверного ПЗ.

Вибір Windows Forms, .NET 8 та SQLite обґрунтовується їхньою сумісністю, простотою використання та відповідністю вимогам проекту. Windows Forms забезпечує зручний інтерфейс, .NET 8 гарантує продуктивність і гнучкість розробки, а SQLite пропонує ефективне локальне зберігання даних. Ця комбінація дозволяє створити надійну, функціональну та доступну програму для автоматизації обліку комунальних платежів, яка відповідає потребам цільової аудиторії.

Висновки до розділу 1

У результаті аналізу та огляду технологій розробки сформовано чітке розуміння задачі створення програми для автоматизації обліку комунальних платежів. Основна мета полягає в розробці зручного, ефективного та локального програмного рішення, яке забезпечує введення, зберігання, обробку та аналіз даних про платежі з підтримкою фіксованих і лічильникових послуг. Програма має автоматизувати рутинні операції, такі як заповнення тарифів і показників лічильників, а також надавати можливості фільтрації та сортування даних.

Огляд аналогічних систем, таких як HomeBank, Paytm та Microsoft Excel, показав їхні обмеження. HomeBank не має спеціалізованих функцій для комунальних платежів, Paytm залежить від інтернету, а Excel вимагає ручного налаштування. Це підтверджує необхідність розробки програми, яка поєднує локальне зберігання, автоматизацію та зручний інтерфейс, усуваючи недоліки існуючих рішень.

Вибір технологій Windows Forms, .NET 8 та SQLite обґрунтовано їхньою відповідністю вимогам проекту. Windows Forms забезпечує інтуїтивно

зрозумілий інтерфейс, .NET 8 пропонує високу продуктивність і сучасні засоби розробки, а SQLite гарантує ефективне локальне зберігання даних без складного розгортання. Альтернативні технології, такі як WPF чи ASP.NET, виявилися менш доцільними через складність або невідповідність локальному характеру програми.

Проведений аналіз закладає основу для розробки програми, яка буде функціональною, доступною та адаптованою до потреб користувачів у сфері обліку комунальних платежів.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Розробка програми для автоматизації обліку комунальних платежів потребує ретельного вибору алгоритмів для забезпечення ефективної обробки даних, швидкості виконання операцій та зручності для користувача. Основні задачі програми включають управління даними про послуги та платежі, автоматичне заповнення показників і тарифів, фільтрацію та сортування даних. Для кожної задачі обрано відповідні алгоритми з урахуванням їхньої ефективності та відповідності вимогам локального настільного додатка.

Для управління даними про послуги та платежі використано структури даних у вигляді списків об’єктів (List<Service> та List<UtilityBill>) у поєднанні з базою даних SQLite. Операції додавання, редагування та видалення даних реалізовано через прямі SQL-запити (INSERT, UPDATE, DELETE). SQLite забезпечує ефективний доступ до даних завдяки індексації за первинними ключами, що дозволяє виконувати ці операції з часовою складністю O(1) для пошуку за ідентифікатором. Завантаження даних із таблиць (SELECT) має складність O(n), де n — кількість записів, але для невеликих обсягів даних, типових для домашнього обліку, це є прийнятним.

Автоматичне заповнення показників лічильників для послуг типу Metered реалізовано через алгоритм пошуку останнього запису в таблиці Bills за ідентифікатором послуги з сортуванням за датою (ORDER BY Date DESC LIMIT 1). Цей запит оптимізовано завдяки використанню умови ServiceId та обмеженню результату одним записом, що забезпечує швидкий доступ до даних із часовою складністю O(log m), де m — кількість платежів для конкретної послуги, за умови індексації. Для фіксованих послуг тариф отримується з об’єкта Service у пам’яті, що має складність O(1) після завантаження списку послуг.

Фільтрація платежів за місяцем, роком і типом послуги виконується через алгоритм лінійного відбору (Where) у пам’яті після завантаження всіх платежів. Це має часову складність O(n), де n — загальна кількість платежів.

Для невеликих наборів даних (сотні записів) лінійний перебір є достатньо швидким і не потребує складних структур, таких як дерева чи хеш-таблиці. Альтернативою могло бути виконання фільтрації на рівні SQL-запиту, але це ускладнило б код через динамічне формування умов, а виграш у продуктивності був би незначним для локального додатка.

Сортування платежів у таблиці (ListView) реалізовано через вбудований механізм порівняння (IComparer), який застосовується до елементів списку за вибраним стовпцем (назва послуги, сума, дата тощо). Використовується алгоритм сортування, вбудований у .NET (ListView.Sort), який базується на QuickSort із середньою часовою складністю O(n log n). Для оптимізації користувацького досвіду сортування виконується лише за вимогою (клік на заголовок стовпця), а результати зберігаються в інтерфейсі, що зменшує кількість повторних обчислень.

Для валідації введених даних (наприклад, перевірка коректності показників лічильників чи суми) використано прості умовні перевірки з часовою складністю O(1). Це включає порівняння числових значень (наприклад, поточні показники мають бути більшими за попередні) та перевірку формату введених даних через методи парсингу (decimal.TryParse). Такий підхід забезпечує швидку обробку та негайне повідомлення користувача про помилки.

Ефективність обраних алгоритмів оцінюється з огляду на масштаби даних. Для домашнього обліку кількість послуг рідко перевищує десятки, а платежів — сотні за рік. У таких умовах лінійні алгоритми (O(n)) для фільтрації та завантаження даних є достатньо швидкими, а використання SQLite забезпечує надійність і низьке споживання ресурсів. Сортування з O(n log n) виконується рідко і лише для відображення, що не впливає на загальну продуктивність.

Порівняння з альтернативними підходами, такими як використання складних структур даних (наприклад, B-дерева для фільтрації), показало, що вони ускладнили б код і не дали значного виграшу в швидкості для малих

обсягів даних. Використання асинхронних запитів до бази даних також було відхилено, оскільки затримки в SQLite мінімальні, а синхронний підхід спрощує логіку.

Обрані алгоритми забезпечують оптимальний баланс між простотою реалізації, продуктивністю та зручністю використання. Вони відповідають вимогам локального настільного додатка, мінімізуючи час обробки даних і забезпечуючи швидкий відгук інтерфейсу для користувача.

2.2. База даних

База даних є ключовим компонентом програми для автоматизації обліку комунальних платежів, забезпечуючи надійне зберігання інформації про послуги та платежі. Для реалізації використано SQLite — легку вбудовану реляційну базу даних, яка ідеально підходить для локальних настільних додатків завдяки простоті розгортання, низьким вимогам до ресурсів і підтримці стандартних SQL-запитів. Структура бази даних розроблена з урахуванням потреб програми, забезпечуючи ефективне зберігання, доступ і обробку даних.

База даних складається з двох основних таблиць: Services і Bills.

Таблиця Services призначена для зберігання інформації про комунальні послуги. Вона містить такі поля: Id (цілочисельний первинний ключ із автоінкрементом), Name (текстове поле для назви послуги, унікальне), ServiceType (текстове поле для типу послуги: "Fixed" або "Metered") і Tariff (десяткове число для зберігання тарифу). Унікальність поля Name забезпечує неможливість дублювання назв послуг, а первинний ключ Id використовується для швидкого пошуку та зв’язку з іншими таблицями.

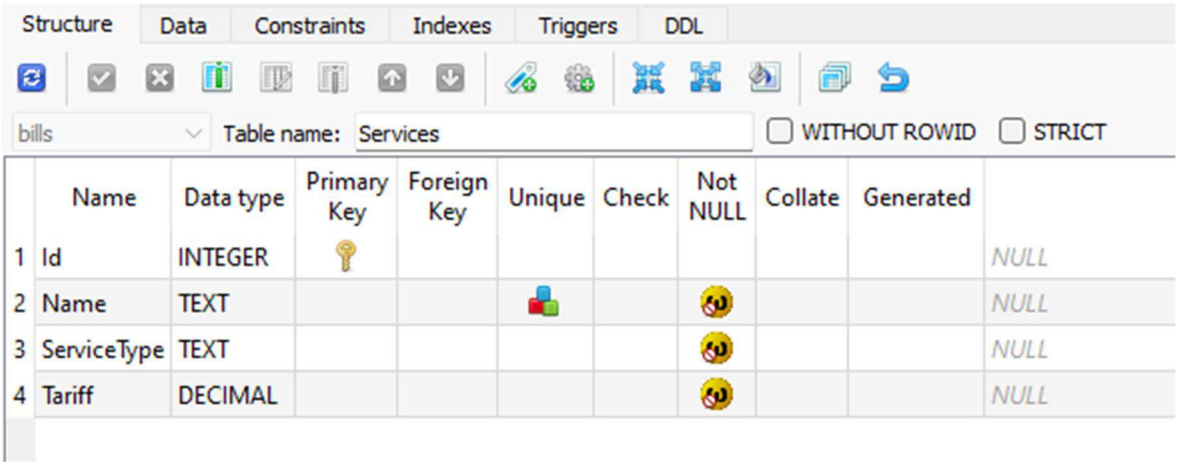

Рисунок 2.2.1 – Таблиця Services

Таблиця Bills зберігає дані про платежі та включає поля: Id (цілочисельний первинний ключ із автоінкрементом), ServiceId (цілочисельне поле, зовнішній ключ, що посилається на Id із таблиці Services), Amount (десяткове число для суми платежу), Date (текстове поле для дати у форматі "YYYY-MM-DD"), PreviousReading і CurrentReading (десяткові поля для показників лічильників, які можуть бути NULL для фіксованих послуг). Зовнішній ключ ServiceId забезпечує цілісність даних, забороняючи створення платежів для неіснуючих послуг.

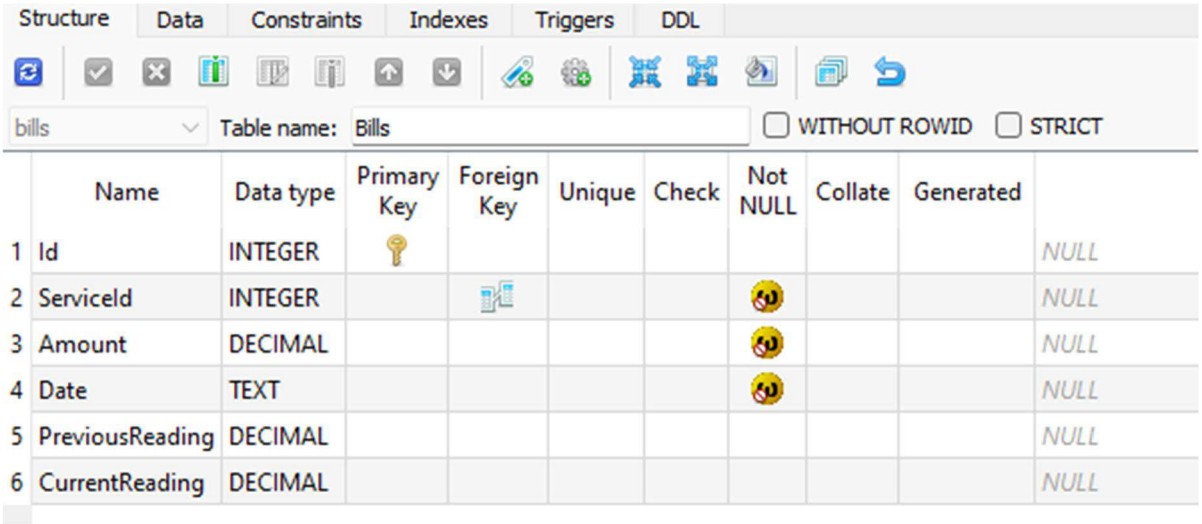

Рисунок 2.2.2 – Таблиця Bills

Структура бази даних створюється під час першого запуску програми за допомогою SQL-запитів CREATE TABLE IF NOT EXISTS. Це гарантує, що таблиці ініціалізуються автоматично, якщо файл бази даних (bills.db)

відсутній. Використання SQLite дозволяє зберігати всю базу в одному файлі, що спрощує резервне копіювання та перенесення даних між пристроями.

Для взаємодії з базою даних реалізовано набір операцій. Додавання, редагування та видалення записів у таблицях Services і Bills виконуються через SQL-запити INSERT, UPDATE і DELETE. Наприклад, для додавання послуги використовується запит INSERT INTO Services (Name, ServiceType, Tariff) VALUES (...), який повертає ідентифікатор нового запису. Завантаження даних (SELECT) застосовується для отримання списків послуг і платежів, а також для пошуку останнього показника лічильника (SELECT CurrentReading ... ORDER BY Date DESC LIMIT 1). Усі запити параметризовано для захисту від SQL-ін’єкцій і підвищення безпеки.

Ефективність бази даних забезпечується за рахунок простої структури та індексації. Первинні ключі (Id) автоматично індексуються SQLite, що гарантує швидкий пошук із часовою складністю O(1). Зовнішній ключ ServiceId у таблиці Bills оптимізує зв’язки між таблицями. Для малих обсягів даних (десятки послуг і сотні платежів) SQLite забезпечує високу швидкість виконання запитів без необхідності додаткових індексів. Для великих обсягів даних у майбутньому можна додати індекси на поле Date у таблиці Bills для прискорення фільтрації за датою.

Порівняння з альтернативними СУБД, такими як Microsoft SQL Server чи PostgreSQL, показало, що вони є надмірними для локального додатка через необхідність серверної інфраструктури та складність налаштування. SQLite, навпаки, не вимагає додаткового програмного забезпечення, що робить програму портативною та легкою для користувачів.

База даних на основі SQLite забезпечує надійне зберігання даних із простою та ефективною структурою. Таблиці Services і Bills покривають усі потреби програми, а використання SQL-запитів гарантує швидкий доступ і обробку даних.

2.3. Спроєктовані класи програми

Розробка програми для автоматизації обліку комунальних платежів базується на об'єктно-орієнтованому підході, що забезпечує модульність, зрозумілість і легкість підтримки коду. Для реалізації функціоналу спроєктовано набір класів, які відображають основні сутності програми, їхні взаємозв'язки та поведінку. Усі класи розроблено в середовищі .NET 8 з використанням C# і інтегровано з інтерфейсом Windows Forms та базою даних SQLite.

Основним класом для представлення комунальної послуги є Service. Цей клас містить властивості Id (цілочисельний ідентифікатор), Name (назва послуги), ServiceType (перерахування для типу послуги: Fixed або Metered) і Tariff (десяткове число для тарифу). Перерахування ServiceType визначено окремо для чіткого розмежування типів послуг. Клас Service використовується для зберігання даних про послуги, отриманих із таблиці Services бази даних, і є основою для взаємодії з інтерфейсом та обробки платежів.

```
public enum ServiceType
{
    Fixed,
    Metered
}

public class Service
{

    public int Id { get; set; }
    17 references
    public string? Name { get; set; }
    11 references
    public ServiceType ServiceType { get; set; }
    11 references
    public decimal Tariff { get; set; }
}
```

Рисунок 2.3.1 – Клас Service

Клас UtilityBill описує платіж за комунальну послугу. Він включає властивості Id (ідентифікатор платежу), ServiceId (посилання на ідентифікатор

послуги), ServiceName (назва послуги для відображення), Amount (сума платежу), Date (дата платежу), PreviousReading і CurrentReading (необов'язкові десяткові значення для показників лічильників, які застосовуються лише для послуг типу Metered). Цей клас відображає записи з таблиці Bills і використовується для збереження та відображення інформації про платежі.

```
public class UtilityBill
{
    public int Id { get; set; }
    public int ServiceId { get; set; }
    public string? ServiceName { get; set; }
    public decimal Amount { get; set; }
    public DateTime Date { get; set; }
    public decimal? PreviousReading { get; set; }
    public decimal? CurrentReading { get; set; }
}
```

Рисунок 2.3.2 – Клас UtilityBill

Для реалізації сортування платежів у таблиці інтерфейсу створено клас ListViewItemComparer, який реалізує інтерфейс IComparer. Він містить властивості SortColumn (індекс стовпця для сортування) і Order (порядок сортування: висхідний або низхідний). Метод Compare порівнює елементи ListView за значеннями в заданому стовпці, підтримуючи сортування за текстовими (наприклад, назва послуги), числовими (сума, показники) або датними (дата платежу) полями. Цей клас забезпечує гнучке сортування даних у таблиці без необхідності додаткових обчислень.

```
3 references
public class ListViewItemComparer : IComparer
{
    4 references
    public int SortColumn { get; set; }
    5 references
    public SortOrder Order { get; set; }
    1 reference
    public ListViewItemComparer()...
    0 references
    public int Compare(object x, object y)...
}
```

Рисунок 2.3.3 – Клас ListViewItemComparer

Клас MainForm, успадкований від Form, є центральним компонентом програми, що відповідає за інтерфейс і логіку взаємодії з користувачем. Він містить поля для зберігання списків services (List<Service>) і bills (List<UtilityBill>), а також екземпляр ListViewItemComparer для сортування. Основні методи включають InitializeDatabase (створення таблиць бази даних), LoadServices і LoadBills (завантаження даних із SQLite), SaveBill і DeleteBill (збереження та видалення платежів), UpdateInputFields (оновлення полів введення залежно від типу послуги) і UpdateBillList (фільтрація та відображення платежів). Події, такі як cmbService_SelectedIndexChanged чи btnAddBill_Click, обробляються для реалізації інтерактивності.

```
3 references
partial class MainForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;
    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...
    Windows Form Designer generated code
    private TextBox txtAmount;
    private Label lblPreviousReading;
    private TextBox txtPreviousReading;
    private Label lblCurrentReading;
    private TextBox txtCurrentReading;
    private ComboBox cmbService;
    private ListView lstBills;
    private Label lblTotalAmount;
    private Label lblAmount;
    private Label lblService;
    private Button btnAddService;
    private Button btnAddBill;
    private Label lblDate;
    private DateTimePicker dtpDate;
    private Label lblFilterMonth;
    private ComboBox cmbFilterMonth;
    private Label lblFilterYear;
    private NumericUpDown nudFilterYear;
    private Label lblFilterService;
    private ComboBox cmbFilterService;
    private Button btnApplyFilter;
    private Button btnDeleteBill;
    private GroupBox groupBox1;
    private Panel panel1;
}
```

Рисунок 2.3.4 – Клас MainForm

Клас ServiceForm, також успадкований від Form, відповідає за управління послугами. Він містить поле services (List<Service>), змінну currentService для редагування та логічний прапорець isEditing. Методи LoadServices, AddService, UpdateService і DeleteService забезпечують взаємодію з таблицею Services у базі даних. Метод UpdateServiceGrid оновлює таблицю інтерфейсу, а ClearInputFields скидає поля введення. Обробники подій, такі як btnAdd_Click чи btnSave_Click, реалізують логіку додавання, редагування та видалення послуг.

```
3 references
partial class ServiceForm
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;
    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...
    Windows Form Designer generated code
    private Label lblNewService;
    private TextBox txtName;
    private Label lblServiceType;
    private ComboBox cmbServiceType;
    private Label lblTariff;
    private TextBox txtTariff;
    private DataGridView dgvServices;
    private Button btnAdd;
    private Button btnEdit;
    private Button btnDelete;
    private Button btnSave;
    private Button btnCancel;
}
```

Рисунок 2.3.5 – Клас ServiceForm

Взаємозв’язки між класами організовано таким чином, що MainForm і ServiceForm взаємодіють із базою даних через об’єкти Service і UtilityBill. MainForm відкриває ServiceForm для управління послугами, після чого оновлює власні списки. ListViewItemComparer використовується виключно в MainForm для сортування таблиці платежів. Усі класи спроєктовано з урахуванням принципу єдиної відповідальності, що полегшує їх тестування та модифікацію.

Структура класів забезпечує чітке розділення функціоналу: Service і UtilityBill відповідають за дані, MainForm і ServiceForm — за інтерфейс і логіку, а ListViewItemComparer — за допоміжну функцію сортування. Використання об’єктно-орієнтованого підходу дозволяє легко розширювати програму, наприклад, додаванням нових типів послуг чи функцій аналітики, зберігаючи код організованим і зрозумілим.

2.4. Програмна реалізація

Програмна реалізація програми для автоматизації обліку комунальних платежів виконана в середовищі .NET 8 з використанням технології Windows Forms і бази даних SQLite. Проект структуровано для забезпечення модульності, зрозумілості коду та зручності підтримки. У цьому розділі описано склад проекту, включаючи файли, класи та форми, а також їх функціональне призначення.

Склад проекту

Проект створено в Visual Studio 2022 і має наступну структуру:

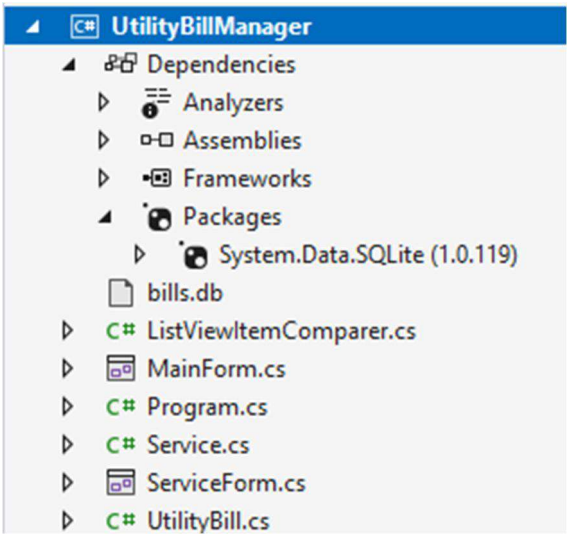

Рисунок 2.4.1 – Проект

UtilityBillManager **основний простір імен**, який містить усі класи та форми програми.

Файли класів:

Service.cs: Визначає клас Service для представлення комунальної послуги з властивостями Id, Name, ServiceType (перерахування ServiceType з значеннями Fixed і Metered) та Tariff. Перерахування ServiceType також визначено в цьому файлі.

UtilityBill.cs: Містить клас UtilityBill, який описує платіж із властивостями Id, ServiceId, ServiceName, Amount, Date, PreviousReading і CurrentReading.

ListViewItemComparer.cs: Реалізує клас ListViewItemComparer, що успадковує IComparer для сортування елементів ListView за різними стовпцями (назва послуги, сума, дата тощо).

Файли форм:

MainForm.cs, MainForm.Designer.cs: Містять код головної форми та її дизайнерське представлення.

ServiceForm.cs, ServiceForm.Designer.cs: Містять код форми для управління послугами та її дизайнерське представлення.

Зовнішня бібліотека: System.Data.SQLite — підключена через NuGet для роботи з базою даних SQLite. Файл бази даних (bills.db) створюється автоматично в директорії програми.

Інші файли:

Program.cs: Точка входу програми, ініціалізує та запускає MainForm.

Конфігураційні файли Visual Studio (.csproj, .sln) для управління залежностями та налаштуваннями проекту.

Проект компілюється в єдиний виконуваний файл (.exe), який разом із бібліотеками SQLite і файлом бази даних забезпечує повну функціональність програми.

Програма містить дві основні форми, реалізовані як класи, успадковані від Form: MainForm і ServiceForm. Кожна форма має графічний інтерфейс, створений за допомогою Windows Forms, і пов’язану логіку.

MainForm є головною формою програми, яка відповідає за облік платежів і взаємодію з користувачем. Її інтерфейс зображено на малюнку(Рисунок 2.4.2)

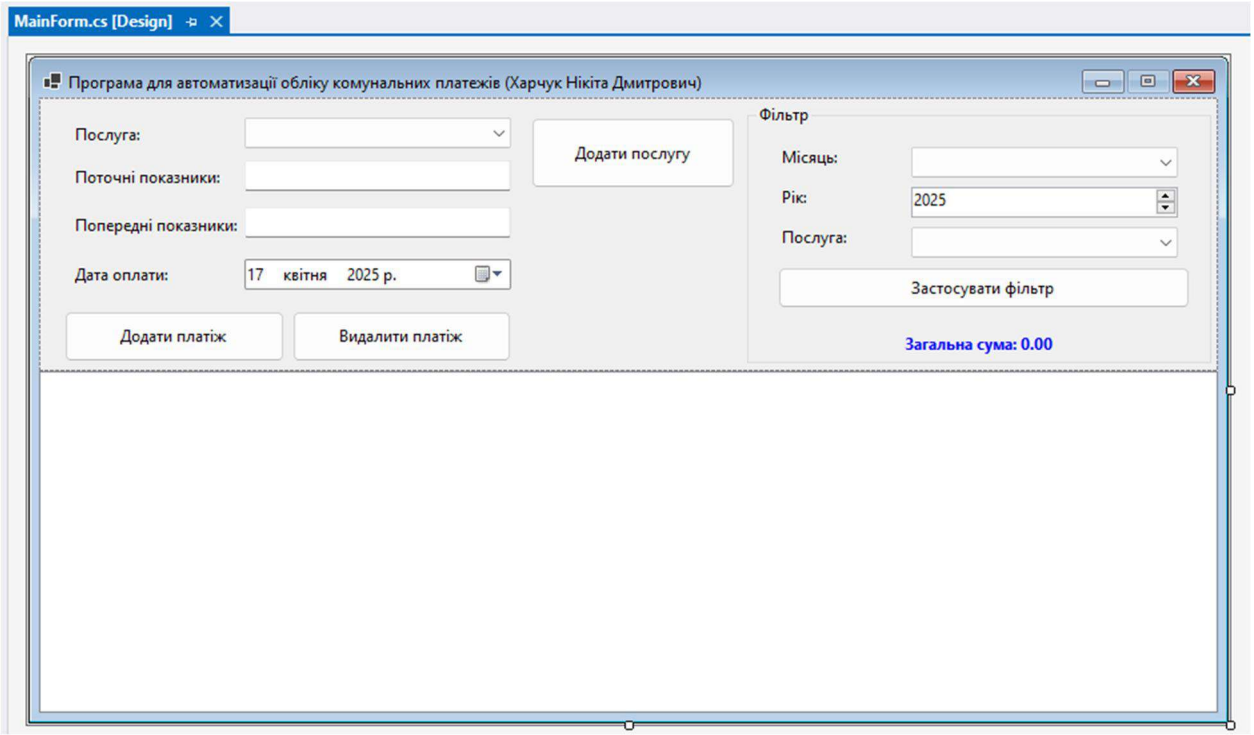

Рисунок 2.4.2 – MainForm

Логіка MainForm реалізована через методи:

- InitializeDatabase: Створює таблиці Services і Bills.
- LoadServices, LoadBills: Завантажують дані з бази.
- UpdateInputFields: Оновлює видимість полів залежно від типу послуги, підставляючи тариф або останні показники.
- UpdateBillList: Фільтрує та відображає платежі.
- SaveBill, DeleteBill: Зберігають і видаляють платежі.
- Обробники подій (cmbService_SelectedIndexChanged, btnAddBill_Click, lstBills_ColumnClick тощо) забезпечують інтерактивність.

Форма **ServiceForm** відповідає за управління списком послуг. Її інтерфейс зображено на малюнку (Рисунок 2.4.3)

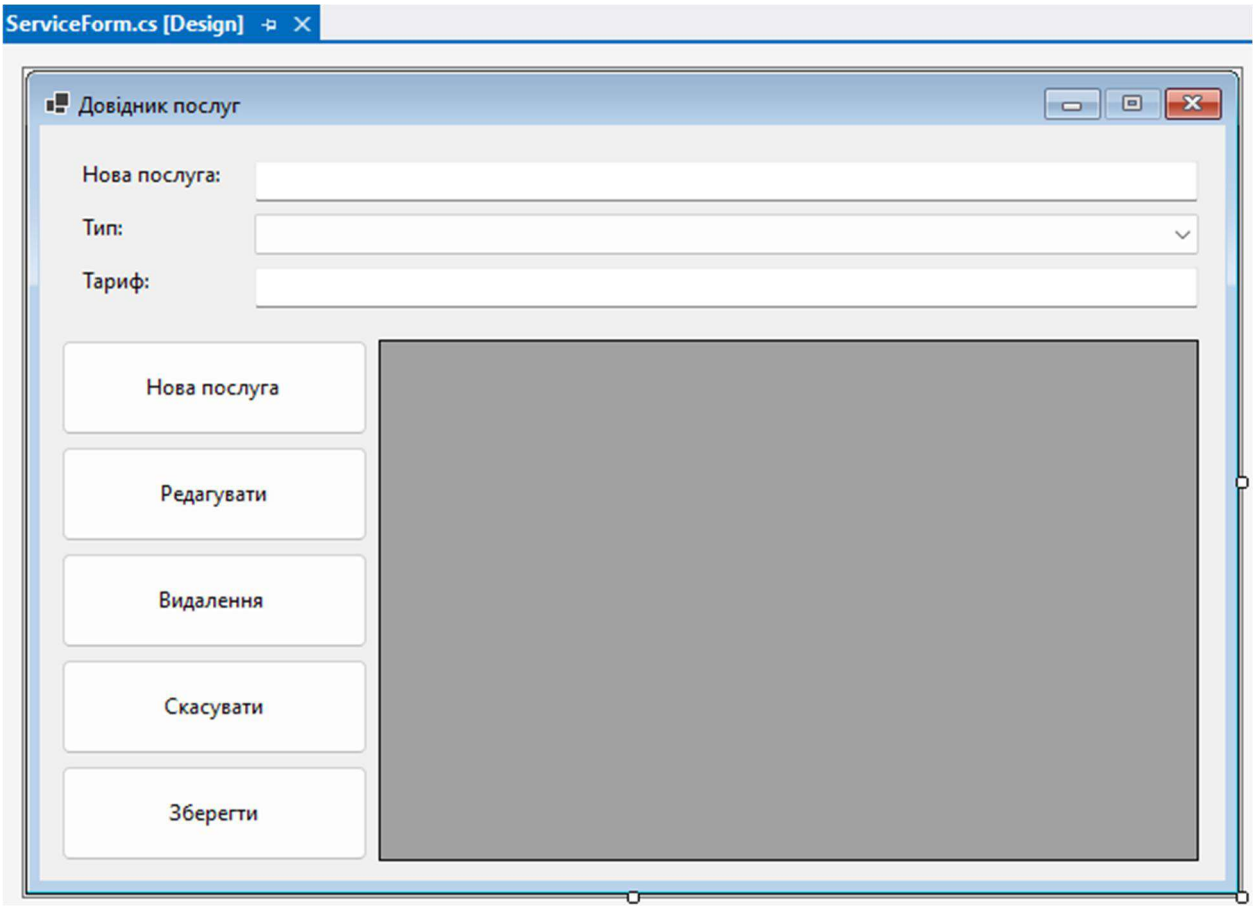

Рисунок 2.4.3 – ServiceForm

Логіка ServiceForm включає методи:

- LoadServices: Завантажує послуги з бази.
- UpdateServiceGrid: Оновлює таблицю dgvServices.
- AddService, UpdateService, DeleteService: Виконують операції з базою.
- ClearInputFields: Скидає поля введення.
- Обробники подій (btnAdd_Click, btnSave_Click тощо) керують взаємодією з користувачем.

Форми взаємодіють через діалогове вікно: MainForm відкриває ServiceForm за допомогою ShowDialog, оновлюючи списки після закриття. Дані зберігаються в SQLite (bills.db), а їх обробка виконується через параметризовані SQL-запити для безпеки. Інтерфейс адаптується до типу послуги: MainForm динамічно показує/ховає поля для лічильників або суми, а ServiceForm забезпечує зручне редагування послуг.

Висновки до розділу 2

У процесі проектування та розробки програми для автоматизації обліку комунальних платежів виконано комплексний аналіз і реалізацію всіх необхідних компонентів. Обрані алгоритми, такі як лінійна фільтрація, сортування QuickSort і прямі SQL-запити, забезпечують ефективну обробку даних для невеликих обсягів, типових для домашнього обліку. Їхня простота реалізації та низька обчислювальна складність (O(n) для фільтрації, O(n log n) для сортування, O(1) для пошуку за ключем) відповідають вимогам локального настільного додатка, гарантуючи швидкий відгук інтерфейсу.

База даних на основі SQLite із таблицями Services і Bills забезпечує надійне зберігання даних із підтримкою зв'язків через зовнішні ключі та швидким доступом завдяки індексації. Проста структура бази та використання параметризованих запитів сприяють безпеці й ефективності, а локальний характер SQLite усуває потребу в серверній інфраструктурі, що робить програму портативною.

Спроєктовані класи (Service, UtilityBill, ListViewItemComparer, MainForm, ServiceForm) відображають об'єктно-орієнтований підхід, забезпечуючи модульність і чітке розділення відповідальностей. Клас Service представляє послуги, UtilityBill — платежі, а ListViewItemComparer підтримує гнучке сортування. Форми MainForm і ServiceForm реалізують інтерфейс і логіку, адаптуючись до потреб користувача, наприклад, через динамічне відображення полів залежно від типу послуги.

Програмна реалізація в середовищі .NET 8 з використанням Windows Forms і SQLite забезпечує зручний інтерфейс, інтеграцію з базою даних і стабільну роботу. Проект структуровано з окремими файлами для класів і форм, що полегшує підтримку та розширення. Форма MainForm відповідає за облік платежів із підтримкою фільтрації та сортування, а ServiceForm — за управління послугами, що разом покриває всі функціональні вимоги.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для коректної роботи програми автоматизації обліку комунальних платежів необхідно забезпечити відповідність апаратного та програмного забезпечення мінімальним системним вимогам. Програма розроблена як локальний настільний додаток на основі .NET 8, Windows Forms і SQLite, що робить її легкою та невимогливою до ресурсів. Нижче наведено вимоги до системи для забезпечення стабільної роботи.

Програма має низькі вимоги до апаратного забезпечення, оскільки працює з невеликими обсягами даних і не потребує значних обчислювальних ресурсів. Рекомендується комп’ютер із процесором із частотою від 1 ГГц, що є типовим для сучасних систем. Обсяг оперативної пам’яті має становити щонайменше 2 ГБ, хоча для комфортної роботи достатньо й 1 ГБ, оскільки .NET 8 і SQLite ефективно оптимізують використання пам’яті. Для зберігання програми, бібліотек і файлу бази даних (bills.db) потрібно близько 100 МБ вільного місця на диску, що робить її придатною навіть для систем із обмеженим обсягом пам’яті. Монітор із роздільною здатністю від 1024x768 пікселів забезпечує комфортне відображення інтерфейсу Windows Forms.

Програма призначена для роботи в операційній системі Windows, сумісній із .NET 8. Мінімально підтримується Windows 10 (версія 1809 або новіша), але для оптимальної продуктивності рекомендується Windows 11. Перед встановленням необхідно переконатися, що на комп’ютері встановлено .NET 8 Runtime (Desktop Runtime), який забезпечує виконання програми. Цей компонент можна завантажити з офіційного сайту Microsoft, якщо він відсутній. SQLite не вимагає окремого встановлення, оскільки бібліотека System.Data.SQLite вбудована в програму та розгортається разом із нею.

Для взаємодії з інтерфейсом необхідні стандартні пристрої введення — клавіатура та миша. Інтернет-з’єднання не потрібне, оскільки програма працює локально, а база даних зберігається на пристрої користувача. Для резервного копіювання або перенесення даних рекомендується

використовувати зовнішні носії, такі як USB-накопичувачі, хоча це не є обов’язковим.

Для забезпечення максимальної стабільності рекомендується періодично оновлювати Windows і .NET Runtime до останніх версій, щоб уникнути проблем сумісності. Якщо користувач планує працювати з великою кількістю платежів (тисячі записів), бажано мати 4 ГБ оперативної пам’яті та сучасніший процесор (наприклад, Intel Core i3 або еквівалент) для швидшої обробки даних. Використання антивірусного програмного забезпечення не впливає на роботу програми, але слід уникати блокування файлу bills.db.

Ці мінімальні вимоги роблять програму доступною для використання на більшості сучасних і навіть застарілих комп’ютерів із Windows. Простота розгортання, відсутність потреби в серверній інфраструктурі та низьке споживання ресурсів забезпечують широку сумісність і зручність для домашніх користувачів, які прагнуть автоматизувати облік комунальних платежів.

3.2. Інтерфейс користувача

Інтерфейс програми для автоматизації обліку комунальних платежів розроблено з використанням Windows Forms, що забезпечує зручний і знайомий вигляд для користувачів Windows. Програма складається з двох основних вікон: головного вікна для обліку платежів і вікна для управління послугами. Інтерфейс є інтуїтивно зрозумілим, із логічним розташуванням елементів і адаптивним відображенням залежно від типу послуги.

Головне вікно поділено на три основні зони: введення даних, фільтрація та відображення платежів.

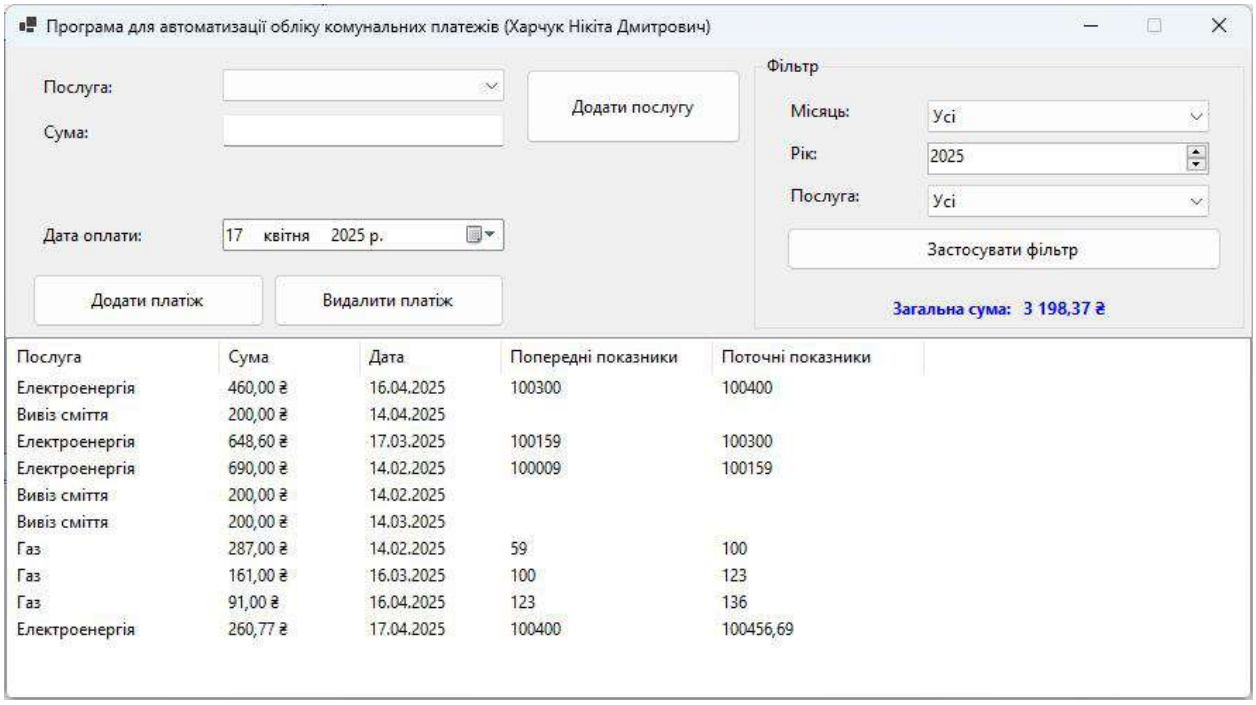

Рисунок 3.2.1 – Головне вікно

Зона введення даних

"Послуга:" випадаючий список дозволяють вибрати послугу (наприклад, "Електрика", "Вода").

Для фіксованих послуг відображаються "Сума:", автоматично заповнене тарифом із бази даних.

Для лічильникових послуг показуються "Попередні показники:", "Поточні показники:" та відповідні поля, де поле попередніх показників заповнюється останніми даними.

"Дата оплати:" дозволяють вказати дату платежу.

Кнопка "Додати послугу" відкриває вікно для управління послугами.

Кнопки "Додати платіж" і "Видалити платіж" додають або видаляють платіж.

Зона фільтрації

"Місяць:" це випадаючий список дозволяють вибрати місяць ("Усі", "Січень" тощо).

"Рік:" це числове поле задають рік (від 2000 до 2100).

"Послуга:" це випадаючий список фільтрують за конкретною послугою або "Усі".

Кнопка "Застосувати фільтр" оновлює таблицю відповідно до вибраних параметрів.

"Загальна сума:" показує суму відфільтрованих платежів.

Зона відображення:

Таблиця платежів відображає дані зі стовпцями: "Послуга", "Сума", "Дата", "Попередні показники", "Поточні показники". Клік на заголовок стовпця сортує таблицю за відповідним параметром.

Елементи введення адаптуються до типу послуги: для фіксованих послуг поля лічильників ховаються, а для лічильникових — поле суми. Розташування елементів, таких як "Дата оплати:", елемент вибору дати, кнопки "Додати платіж" і "Видалити платіж", змінюється для компактності залежно від типу послуги.

Вікно управління послугами

Вікно для управління послугами має дві зони: введення даних і відображення списку.

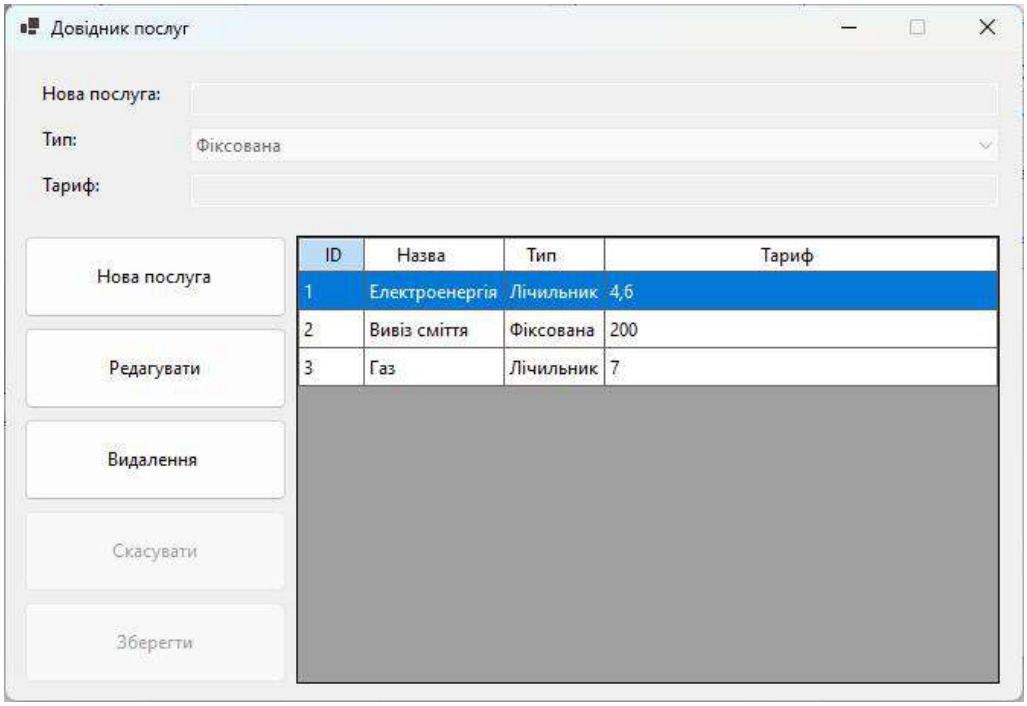

Рисунок 3.2.2 – Довідник послуг

Зона введення:

"Нова послуга:" - поле для введення назви послуги.

"Тип:" - випадаючий список для вибору типу ("Фіксована" або "Лічильник").

"Тариф:" -це поле для введення тарифу.

Зона відображення та управління (у нижній частині):

Таблиця послуг показує список із стовпцями: "ID", "Назва", "Тип", "Тариф".

Кнопки "Нова послуга", "Редагувати", "Видалення" дозволяють додавати, змінювати або видаляти послугу.

Кнопки "Зберегти" і "Скасувати" зберігають зміни або скидають поля.

Інтерфейс обох вікон виконано в стандартному стилі Windows із чіткими мітками, що полегшує навігацію для користувачів із мінімальним досвідом.

3.3. Інструкція для роботи з програмою

Встановлення

1. Переконайтеся, що на комп’ютері встановлено Windows 10 (версія 1809 або новіша) або Windows 11.
2. Встановіть .NET 8 Desktop Runtime із сайту Microsoft, якщо він відсутній.
3. Завантажте виконуваний файл програми та бібліотеки з дистрибутива.
4. Розмістіть файли в будь-якій директорії та запустіть програму, двічі клікнувши на .exe файл. Файл бази даних створюється автоматично.

Робота з головним вікном

1. **Запуск програми**:

Після запуску відкривається головне вікно. Таблиця платежів відображає всі наявні записи, а мітка "Загальна сума:" показує їхню загальну суму.

2. **Додавання платежу**:

У випадаючому списку біля мітки "Послуга:" виберіть потрібну послугу.

Для фіксованих послуг поле біля "Сума:" автоматично заповнюється тарифом. За потреби відредагуйте значення.

Для лічильникових послуг поле біля "Попередні показники:" заповнюється останніми даними (якщо є). Введіть нові показники в поле біля "Поточні показники:".

У елементі вибору біля "Дата оплати:" вкажіть дату платежу.

Натисніть кнопку "Додати платіж". У разі помилки (наприклад, некоректні показники) з’явиться повідомлення. Після успішного додавання платіж відобразиться в таблиці платежів.

3. **Фільтрація платежів**:

У списку біля "Місяць:" виберіть місяць або "Усі".

У полі біля "Рік:" задайте рік.

У списку біля "Послуга:" виберіть послугу або "Усі".

Натисніть "Застосувати фільтр". Таблиця платежів оновиться, а "Загальна сума:" покаже суму відфільтрованих записів.

4. **Сортування**:

Клікніть на заголовок стовпця в таблиці платежів (наприклад, "Сума" або "Дата") для сортування за зростанням. Повторний клік змінить порядок на спадання.

5. **Видалення платежу**:

Виділіть платіж у таблиці платежів, клікнувши на рядок.

Натисніть "Видалити платіж". Підтвердіть дію у діалоговому вікні. Платіж зникне з таблиці.

6. **Управління послугами**:

Натисніть "Додати послугу", щоб відкрити вікно управління послугами.

Робота з вікном управління послугами

1. **Перегляд послуг**:

При відкритті вікна таблиця послуг показує всі записи з їхніми ідентифікаторами, назвами, типами та тарифами.

2. **Додавання послуги**:

Натисніть "Нова послуга". Поля для назви, типу та тарифу стануть активними.

Введіть назву в поле біля "Нова послуга:", виберіть тип ("Фіксована" або "Лічильник") у списку біля "Тип:", вкажіть тариф у полі біля "Тариф:".

Натисніть "Зберегти". Якщо дані коректні, послуга додасться до таблиці. У разі помилки (наприклад, порожня назва) з’явиться повідомлення.

3. **Редагування послуги**:

Виділіть послугу в таблиці послуг, клікнувши на рядок.

Натисніть "Редагувати". Поля заповняться поточними даними.

Відредагуйте потрібні поля та натисніть "Зберегти". Зміни збережуться.

4. **Видалення послуги**:

Виділіть послугу в таблиці послуг.

Натисніть "Видалення". Підтвердіть дію у діалоговому вікні. Якщо послуга не використовується в платежах, вона видалиться.

5. **Скасування**:

Натисніть "Скасувати", щоб скинути поля та повернутися до початкового стану.

Завершення роботи

Закрийте програму, натиснувши хрестик у верхньому правому куті. Дані автоматично зберігаються в базі.

Для резервного копіювання скопіюйте файл бази даних у безпечне місце.

Інтерфейс спроєктовано для простоти, а автоматичне заповнення тарифів і показників зменшує кількість ручних операцій. У разі помилок програма видає зрозумілі повідомлення, що допомагає швидко виправити проблему.

Висновки до розділу 3

Розділ містить детальне керівництво для користування програмою автоматизації обліку комунальних платежів, що забезпечує легке освоєння її функціоналу. Визначено мінімальні системні вимоги, які включають Windows 10 (1809 або новіша), .NET 8 Desktop Runtime, процесор від 1 ГГц, 2 ГБ оперативної пам’яті та 100 МБ дискового простору. Ці вимоги роблять програму доступною для більшості сучасних і навіть застарілих комп’ютерів, а локальний характер SQLite усуває потребу в серверній інфраструктурі чи інтернет-з’єднанні.

Опис інтерфейсу користувача підкреслює його інтуїтивність і адаптивність. Головне вікно дозволяє зручно вводити платежі, фільтрувати та сортувати дані, а вікно управління послугами забезпечує просте додавання, редагування та видалення послуг. Динамічне відображення полів залежно від типу послуги (фіксована чи лічильникова) мінімізує складність для користувача. Використання стандартного стилю Windows Forms із чіткими мітками сприяє швидкому освоєнню програми навіть для користувачів із базовим досвідом.

Інструкція для роботи з програмою детально описує всі етапи: від встановлення до виконання основних операцій, таких як додавання платежів, фільтрація, сортування та управління послугами. Автоматичне заповнення тарифів і показників лічильників спрощує введення даних, а зрозумілі повідомлення про помилки допомагають швидко виправляти проблеми. Інструкція орієнтована на кінцевого користувача, використовуючи підписи елементів інтерфейсу замість технічних назв, що підвищує її доступність.

ВИСНОВКИ

Дипломна робота присвячена розробці програми для автоматизації обліку комунальних платежів із використанням технологій Windows Forms, .NET 8 та SQLite. У процесі виконання роботи досягнуто поставленої мети — створено функціональний, зручний і ефективний інструмент для управління даними про комунальні платежі, який відповідає потребам домашніх користувачів.

У першому розділі сформульовано задачу розробки, проведено огляд аналогічних систем (HomeBank, Paytm, Microsoft Excel), що підтвердив необхідність спеціалізованого рішення з локальним зберіганням і автоматизацією. Обґрунтовано вибір Windows Forms для інтуїтивного інтерфейсу, .NET 8 для продуктивності та SQLite для легкої бази даних, що забезпечують оптимальний баланс між простотою, функціональністю та доступністю.

Другий розділ охоплює проектування та розробку програми. Обрано ефективні алгоритми (лінійна фільтрація, QuickSort для сортування, прямі SQL-запити) з часовою складністю, придатною для невеликих обсягів даних. Розроблено базу даних із таблицями Services і Bills, що забезпечують надійне зберігання та швидкий доступ до даних. Спроєктовано класи (Service, UtilityBill, MainForm, ServiceForm, ListViewItemComparer) за об’єктно-орієнтованим підходом, що сприяє модульності та розширюваності. Програмна реалізація включає дві форми: головне вікно для обліку платежів із підтримкою фільтрації та сортування, і вікно управління послугами для додавання, редагування та видалення даних.

Третій розділ містить керівництво користувача, де описано мінімальні системні вимоги (Windows 10, 2 ГБ ОЗП, 100 МБ на диску), що роблять програму доступною для широкого кола комп’ютерів. Інтерфейс програми є інтуїтивним, із динамічним відображенням полів залежно від типу послуги та чіткими мітками. Інструкція детально пояснює встановлення, введення платежів, фільтрацію, сортування та управління послугами, використовуючи

підписи елементів для зрозумілості. Автоматизація заповнення тарифів і показників лічильників спрощує роботу, а повідомлення про помилки допомагають користувачу.

Розроблена програма успішно виконує поставлені завдання: автоматизує облік комунальних платежів, зменшує ймовірність помилок і економить час користувача. Вона перевершує аналоги за рахунок спеціалізації, локального зберігання даних і простоти використання. Практична цінність полягає в можливості використання програми домогосподарствами для фінансового планування та контролю витрат.

Подальший розвиток програми може включати додавання функцій аналітики (графіки витрат, прогнози), підтримку експорту даних у формати CSV або PDF, а також експортування на інші платформи за допомогою .NET MAUI. Ці вдосконалення можуть розширити аудиторію та функціонал програми, зберігаючи її простоту та ефективність.

Дипломна робота демонструє успішну реалізацію програмного рішення, яке відповідає сучасним вимогам до автоматизації побутових задач і має значний потенціал для практичного застосування.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Бойко В. І. Інформаційні системи та технології в економіці: навч. посібник. – К.: Центр учбової літератури, 2018. – 432 с.
2. Гриценко О. М. Бази даних: теорія та практика. – Х.: Фоліо, 2019. – 288 с.
3. Державний стандарт України ДСТУ 8302:2015 "Інформація та документація. Бібліографічний запис. Бібліографічний опис. Загальні положення та правила складання". – К.: Держспоживстандарт України, 2015. – 47 с.
4. Захарченко О. В. Автоматизація облікових процесів: монографія. – Львів: Львівська політехніка, 2020. – 210 с.
5. Іваненко П. Р. Розробка програмного забезпечення для бухгалтерського обліку. – К.: КНЕУ, 2017. – 156 с.
6. Ковальчук С. М. Програмні засоби обробки даних. – Вінниця: ВНТУ, 2019. – 320 с.
7. Лисенко Т. О. Управління базами даних: підручник. – Дніпро: Дніпровський університет, 2021. – 415 с.
8. Мельник О. В. Інформаційні системи в управлінні комунальними послугами. – Одеса: ОНЕУ, 2018. – 278 с.
9. Назаренко І. С. Web-орієнтовані інформаційні системи. – К.: Академвидав, 2020. – 342 с.
10. Олійник А. А. Програмування на мові Python: практичний курс. – Х.: Вид. група "Основа", 2021. – 480 с.
11. Петренко В. С. Автоматизовані системи обліку в ЖКГ. – К.: КПІ, 2019. – 189 с.
12. Романюк М. П. Розробка програмного забезпечення: метод. вказівки. – Львів: НУ "Львівська політехніка", 2020. – 134 с.
13. Савчук Л. В. Економіка комунального господарства. – К.: Ліра-К, 2017. – 305 с.

14. Ткаченко О. М. Інформаційні технології в бухгалтерському обліку. – К.: ЦУЛ, 2018. – 224 с.
15. Федоренко Р. В. SQL для початківців. – Х.: Видавництво "Штрих", 2021. – 176 с.
16. Шевченко К. Л. Управління ІТ-проектами. – Д.: Пороги, 2020. – 412 с.
17. Яковлєв С. П. Програмні інструменти для автоматизації бізнес-процесів. – К.: Наукова думка, 2019. – 267 с.
18. Bennett J. Database Systems: A Practical Approach to Design, Implementation, and Management. – Pearson, 2019. – 1440 p.
19. Fowler M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2018. – 560 p.
20. Hoffer J. A. Modern Database Management. – Pearson, 2020. – 672 p.
21. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. – Prentice Hall, 2017. – 736 p.
22. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. – Prentice Hall, 2018. – 464 p.
23. Sommerville I. Software Engineering. – Pearson, 2021. – 816 p.
24. Tanenbaum A. S. Modern Operating Systems. – Pearson, 2022. – 1136 p.
25. Viescas J. L. SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL. – Addison-Wesley, 2018. – 672 p.
26. Офіційна документація Django [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/
27. Документація PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace UtilityBillManager
{
   partial class MainForm
   {
      /// <summary>
      ///  Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;
      /// <summary>
      ///  Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }
      #region Windows Form Designer generated code

      /// <summary>
      ///  Required method for Designer support - do not modify
      ///  the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
```