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:

«Система обліку оренди нерухомості»

Виконав здобувач освіти групи П-42
Щербина Андрій Ігорович

Керівник роботи:
Устименко Ярослав Іванович

Рецензент:
Устименко Людмила Миколаївна

Житомир – 2025

Зміст

РЕФЕРАТ

Дипломна робота на тему «Система обліку оренди нерухомості» присвячена розробці настільного додатка для автоматизації управління об’єктами нерухомості, орендарями, договорами та платежами. Обсяг роботи становить 56 сторінок, включає 30 рисунків, 1 додаток і 25 літературних джерел.

Метою роботи є створення зручного та економічно виправданого рішення для малого й середнього бізнесу. Система реалізована на платформі .NET 8 з використанням Windows Forms для інтерфейсу та SQLite для зберігання даних. Проведено аналіз аналогів, таких як Rentec Direct і Buildium, що дозволило обґрунтувати вибір локального додатка. Проєкт включає базу даних із таблицями Properties, Tenants, RentalContracts і Payments, забезпечуючи ефективне управління даними через Entity Framework Core.

Розроблено п’ять форм інтерфейсу: для навігації, управління нерухомістю, орендарями, договорами та платежами. Реалізовано функцію друку договорів із підтримкою багатосторінкового формату. Система має низькі вимоги до ресурсів (Windows 10/11, 4 ГБ ОЗП) і підтримує локальну роботу.

Робота підтверджує практичну цінність системи для оптимізації обліку оренди. Перспективи розвитку включають додавання мережевих функцій і розширеної звітності.

Ключові слова: облік оренди, нерухомість, Windows Forms, .NET 8, SQLite, автоматизація.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – Система управління базами даних.

.NET – Платформа розробки від Microsoft для створення додатків.

Windows Forms – Технологія для створення графічного інтерфейсу настільних додатків.

SQLite – Легка реляційна система управління базами даних.

EF Core – Entity Framework Core, бібліотека для об’єктно-реляційного відображення.

UI – User Interface, інтерфейс користувача.

CRUD – Create, Read, Update, Delete, основні операції з даними.

LINQ – Language Integrated Query, технологія для роботи з даними в .NET.

ID – Ідентифікатор, унікальний код для позначення запису в базі даних.

OOP – Об’єктно-орієнтоване програмування.

SQL – Structured Query Language, мова запитів до баз даних.

DbContext – Клас у Entity Framework Core для взаємодії з базою даних.

DataGridView – Компонент Windows Forms для відображення табличних даних.

API – Application Programming Interface, інтерфейс програмування додатків.

RAM – Random Access Memory, оперативна пам’ять.

SSD – Solid-State Drive, твердотільний накопичувач.

OS – Operating System, операційна система.

GUI – Graphical User Interface, графічний інтерфейс користувача.

C# – Мова програмування, що використовується в проєкті.

NuGet – Менеджер пакетів для .NET.

ВСТУП

Сучасний ринок нерухомості характеризується високою динамікою та зростаючим попитом на оренду житлових і комерційних об’єктів. Ефективне управління орендними операціями вимагає чіткого обліку договорів, платежів, нерухомих об’єктів та інформації про орендарів. Ручне ведення таких процесів є трудомістким, схильним до помилок і не відповідає вимогам сучасного бізнесу. Автоматизація обліку оренди нерухомості дозволяє оптимізувати процеси, підвищити точність даних і забезпечити зручність для всіх учасників – від орендодавців до орендарів.

Метою дипломної роботи є розробка системи обліку оренди нерухомості, яка забезпечить автоматизацію ключових процесів управління орендними операціями. Система має надавати можливості для ведення бази даних нерухомих об’єктів, орендарів, договорів оренди, а також обліку платежів із підтримкою різних методів оплати. Крім того, передбачається реалізація функцій формування звітів і друку документів, що підвищить зручність використання системи в реальних умовах.

Для реалізації проєкту обрано технологію Windows Forms на платформі .NET 8, яка забезпечує створення зручного графічного інтерфейсу для настільних додатків. Як система управління базами даних використовується SQLite – легка та ефективна СУБД, що ідеально підходить для локальних застосунків із помірним обсягом даних. Вибір цих технологій зумовлений їхньою надійністю, простотою інтеграції та достатньою продуктивністю для вирішення поставлених завдань.

Актуальність теми дипломної роботи полягає у необхідності створення доступних і функціональних інструментів для малого та середнього бізнесу в сфері нерухомості, що дозволить оптимізувати облік і підвищити ефективність управління орендними операціями. Розроблена система має потенціал для практичного застосування, адже вона враховує типові потреби орендодавців і може бути адаптована до різних сценаріїв використання.

У процесі виконання роботи планується проаналізувати предметну область, розробити структуру бази даних, спроєктувати та реалізувати інтерфейс користувача, а також забезпечити стабільну роботу всіх функціональних модулів. Особлива увага приділятиметься зручності використання, надійності збереження даних і можливості розширення системи в майбутньому.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка системи обліку оренди нерухомості спрямована на створення програмного забезпечення, яке забезпечить автоматизацію процесів управління орендними операціями. Основна задача полягає в реалізації зручного та функціонального інструменту для ведення обліку нерухомих об’єктів, орендарів, договорів оренди та платежів. Система має надавати можливості для створення, редагування та видалення записів, а також формувати звіти й друкувати документи, зокрема договори оренди. Важливим аспектом є забезпечення стабільності роботи програми, захисту даних і зручності використання для користувачів із різним рівнем технічної підготовки.

Для досягнення поставленої мети необхідно спроєктувати базу даних, яка ефективно зберігатиме інформацію про об’єкти нерухомості, орендарів, договори та платежі. Програма повинна мати інтуїтивно зрозумілий графічний інтерфейс, що дозволить швидко виконувати основні операції, такі як додавання нового договору чи облік платежу. Окрім цього, система має підтримувати різні методи оплати та забезпечувати коректне відображення фінансової інформації. Враховуючи потреби малого та середнього бізнесу в сфері нерухомості, розробка має бути економічно виправданою, з мінімальними вимогами до апаратного забезпечення та простою у розгортанні.

Вибір технологій для реалізації проєкту зумовлений необхідністю поєднання простоти розробки, надійності та продуктивності. Система створюватиметься як настільний додаток, що відповідає типовим сценаріям використання в офісному середовищі. Основними вимогами до програми є швидкий доступ до даних, захист від втрати інформації та можливість масштабування функціоналу в майбутньому. Таким чином, задача розробки охоплює аналіз предметної області, вибір оптимального технологічного стеку, проєктування архітектури системи та її практичну реалізацію з урахуванням потреб кінцевих користувачів.

1.2. Огляд та порівняння аналогічних систем.

На ринку програмного забезпечення для управління орендою нерухомості існує низка рішень, які пропонують автоматизацію обліку договорів, платежів та інших операцій, пов'язаних із нерухомістю. Ці системи варіюються за функціоналом, цільовою аудиторією та технологічною основою. Для аналізу обрано кілька популярних аналогів, які мають схожі цілі з розроблюваною системою, зокрема орієнтацію на малий та середній бізнес або індивідуальних орендодавців.

Одним із відомих рішень є Rentec Direct, платформа, яка надає інструменти для управління нерухомістю, включаючи облік орендарів, договорів і платежів. Ця система працює як вебдодаток, що забезпечує доступ із будь-якого пристрою, але потребує постійного підключення до мережі та регулярної оплати підписки. Вона підтримує інтеграцію з банківськими системами для автоматичного відстеження платежів, однак її функціонал надмірно складний для невеликих компаній, а інтерфейс може бути перевантаженим для користувачів-початківців.

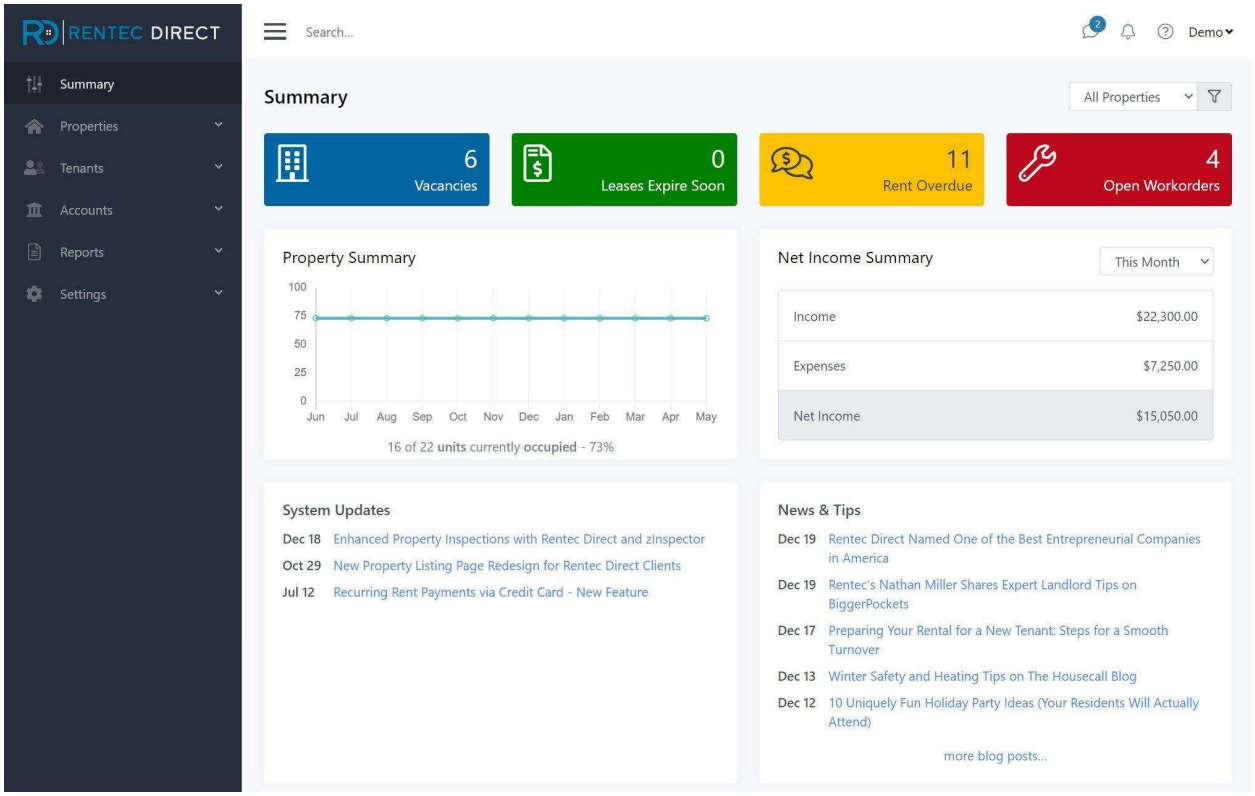


Рисунок 1.2.1 – Rentec Direct

Ще одним прикладом є Buildium, програмне забезпечення, яке також орієнтоване на управління орендою нерухомості. Buildium пропонує широкий набір функцій, таких як облік витрат, створення звітів і навіть управління технічним обслуговуванням об’єктів. Система працює в хмарі, що забезпечує гнучкість, але її висока вартість і залежність від інтернет-з’єднання роблять її менш привабливою для локального використання в невеликих організаціях. Крім того, налаштування системи вимагає певного часу та технічних навичок.

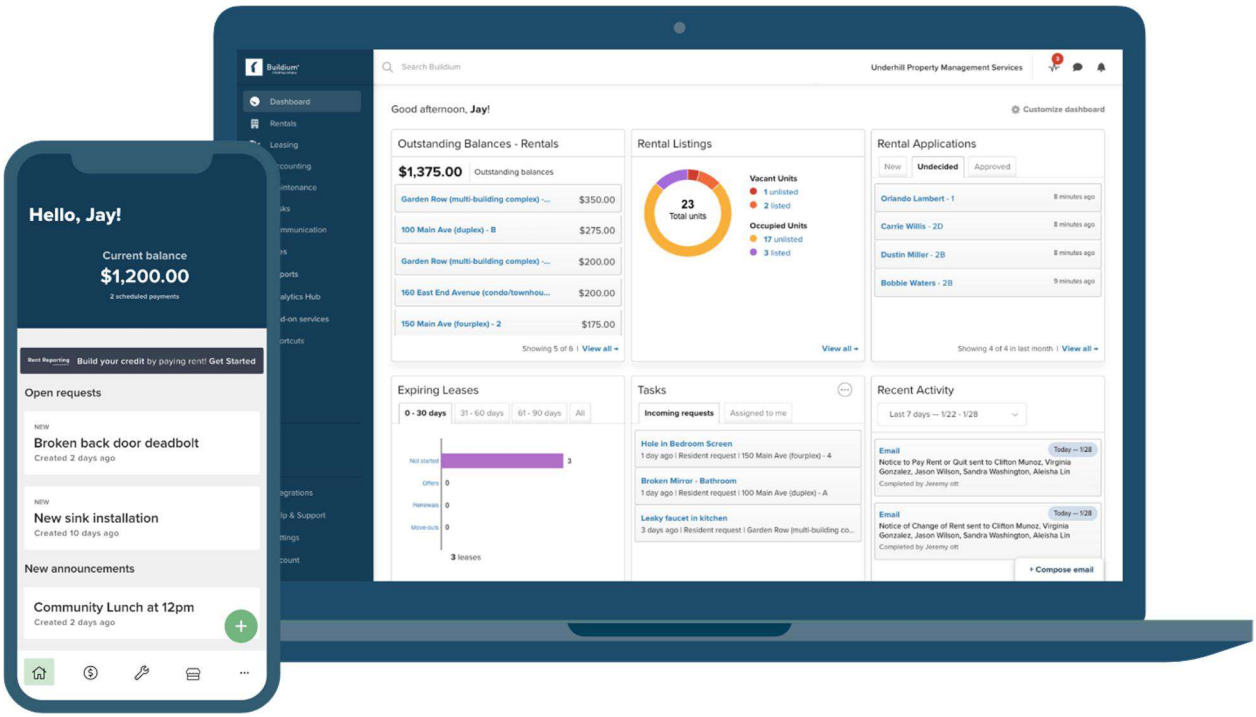


Рисунок 1.2.2 – Buildium

Для локального використання можна розглянути такі рішення, як Quicken Rental Property Manager. Ця програма призначена для настільних комп’ютерів і дозволяє вести облік орендних платежів, договорів і витрат. Вона простіша у використанні порівняно з хмарними аналогами, але її функціонал обмежений, зокрема відсутня підтримка розширених звітів чи автоматизованого друку документів. Також система не підтримує сучасні методи оплати, такі як онлайн-платежі, що знижує її актуальність.

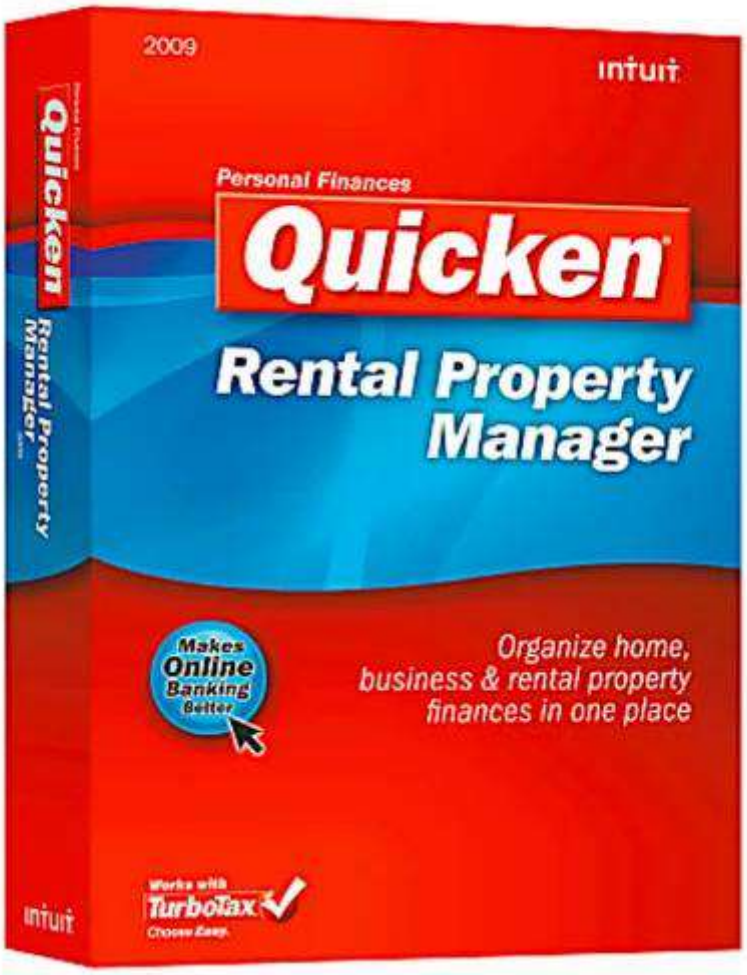


Рисунок 1.2.3 – Quicken Rental Property Manager

Порівняно з цими аналогами, розроблювана система обліку оренди нерухомості має кілька ключових особливостей. Вона створюється як настільний додаток на основі Windows Forms і .NET 8 із використанням SQLite, що забезпечує незалежність від мережевого підключення та мінімальні витрати на розгортання.

На відміну від Rentec Direct і Buildium, які орієнтовані на великі компанії з розподіленими командами, пропонована система зосереджена на потребах малого бізнесу та індивідуальних орендодавців. Порівняно з Quicken Rental Property Manager, вона пропонує ширший функціонал, зокрема підтримку різних методів оплати, друк документів і гнучке управління договорами.

Основною перевагою розроблюваної системи є її простота в установці та використанні, що не потребує додаткових витрат на хмарну інфраструктуру чи складне налаштування. Водночас, на відміну від хмарних рішень, вона має

обмеження в плані віддаленого доступу, що може бути компенсовано в майбутніх версіях шляхом додавання мережевих функцій.

Аналіз аналогів показує, що створювана система займає нішу доступного та функціонального рішення для локального використання з потенціалом для подальшого розширення.

1.3. Вибір та обґрунтування технологій розробки

Для реалізації системи обліку оренди нерухомості необхідно обрати технології, які забезпечать надійність, продуктивність і зручність розробки, а також відповідатимуть вимогам цільової аудиторії – малого та середнього бізнесу, а також індивідуальних орендодавців. Основними критеріями вибору є простота розгортання, мінімальні вимоги до апаратного забезпечення, підтримка локального використання та можливість створення інтуїтивно зрозумілого інтерфейсу. На основі цих критеріїв обрано технологічний стек, що включає Windows Forms, .NET 8 та SQLite.

Windows Forms обрано як основну технологію для створення графічного інтерфейсу користувача. Ця платформа є частиною екосистеми .NET і забезпечує швидку розробку настільних додатків із широким набором компонентів для побудови форм, таблиць і елементів керування. Windows Forms має багаторічну історію використання, що гарантує стабільність і наявність великої кількості документації та прикладів. Її перевагою є простота створення інтерфейсів, які не потребують складного налаштування, що ідеально відповідає потребам системи, орієнтованої на користувачів із базовим рівнем технічних навичок. Крім того, Windows Forms дозволяє створювати додатки, які працюють локально без залежності від мережевого підключення, що важливо для невеликих організацій із обмеженим доступом до хмарних сервісів.

Платформа .NET 8 обрана як основа для розробки завдяки її сучасним можливостям, високій продуктивності та підтримці кроссплатформених рішень. .NET 8 забезпечує ефективну роботу з пам’яттю, надійну обробку

винятків і широкі можливості для роботи з базами даних. У контексті даного проєкту .NET 8 дозволяє використовувати сучасні бібліотеки та інструменти, такі як Entity Framework Core, для спрощення взаємодії з базою даних. Ця платформа також має активну підтримку спільноти та регулярні оновлення, що гарантує актуальність технології протягом тривалого часу. Використання .NET 8 забезпечує баланс між продуктивністю та простотою розробки, що є критично важливим для створення стабільного додатка з обмеженими ресурсами.

Для управління даними обрано SQLite – легку реляційну систему управління базами даних. SQLite не потребує окремого серверного процесу, що значно спрощує розгортання та адміністрування. База даних зберігається у вигляді одного файлу, що полегшує резервне копіювання та перенесення даних. SQLite підтримує більшість стандартних SQL-запитів, що дозволяє ефективно організувати зберігання інформації про нерухомість, орендарів, договори та платежі. Завдяки низьким вимогам до ресурсів, SQLite ідеально підходить для локальних додатків, призначених для малого бізнесу. Інтеграція SQLite з .NET 8 через Entity Framework Core забезпечує зручний доступ до даних і спрощує операції створення, читання, оновлення та видалення записів.

Обґрунтування вибору цього технологічного стеку базується на його відповідності поставленим завданням. Windows Forms забезпечує швидке створення зрозумілого інтерфейсу, .NET 8 гарантує продуктивність і гнучкість розробки, а SQLite пропонує просте та ефективне рішення для зберігання даних. Такий підхід дозволяє створити систему, яка є економічно виправданою, простою в установці та використанні, а також має потенціал для подальшого розширення, наприклад, додавання підтримки мережевих функцій чи інтеграції з іншими сервісами. У порівнянні з альтернативними технологіями, такими як вебфреймворки чи хмарні бази даних, обраний стек є оптимальним для локального настільного додатка з урахуванням обмежень щодо ресурсів і потреб цільової аудиторії.

Висновки до розділу 1

Проведений аналіз предметної області та технологій розробки дозволив сформувати чітке розуміння задачі створення системи обліку оренди нерухомості. Основна мета полягає в автоматизації процесів управління нерухомими об’єктами, орендарями, договорами та платежами з акцентом на простоту використання та доступність для малого й середнього бізнесу. Визначено, що система має забезпечувати зручний інтерфейс, надійне збереження даних і функціонал для формування документів, що відповідає потребам цільової аудиторії.

Огляд аналогічних систем показав, що існуючі рішення, такі як Rentec Direct, Buildium чи Quicken Rental Property Manager, мають свої переваги, зокрема розширений функціонал або хмарну доступність, але часто є надмірно складними чи дорогими для невеликих організацій. Розроблювана система займає нішу локального настільного додатка, який поєднує простоту розгортання з достатньою функціональністю, що робить її конкурентоспроможною для локального використання з можливістю подальшого вдосконалення.

Вибір технологічного стеку, що включає Windows Forms, .NET 8 та SQLite, обґрунтовано їхньою відповідністю вимогам проєкту. Windows Forms забезпечує швидке створення інтуїтивного інтерфейсу, .NET 8 гарантує продуктивність і сучасні можливості розробки, а SQLite пропонує легке та ефективне рішення для локального зберігання даних. Такий підхід дозволяє створити економічно виправдану систему з низькими вимогами до апаратного забезпечення, яка водночас має потенціал для масштабування.

Обрані технології створюють міцну основу для реалізації функціональної та надійної системи обліку оренди нерухомості.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Розробка системи обліку оренди нерухомості потребує ретельного вибору алгоритмів для забезпечення ефективної роботи програми, швидкого доступу до даних і зручності виконання основних операцій. Основними завданнями системи є управління даними про нерухомі об’єкти, орендарів, договори та платежі, а також формування документів, таких як договори оренди. Для реалізації цих завдань обрано набір алгоритмів, які оптимізують обробку даних, враховуючи обмеження локального настільного додатка та використання SQLite як бази даних.

Для операцій із базою даних, таких як додавання, редагування, видалення та пошук записів, використано стандартні методи роботи з реляційними базами даних через Entity Framework Core. Entity Framework Core забезпечує об’єктно-реляційне відображення, що дозволяє представляти дані у вигляді об’єктів .NET, спрощуючи їх обробку. Алгоритми для цих операцій базуються на прямих SQL-запитах, які автоматично генеруються фреймворком. Наприклад, додавання нового запису про договір оренди передбачає створення об’єкта класу RentalContract, його додавання до контексту даних і збереження змін у базі. Часова складність таких операцій наближається до O(1) для вставки або оновлення одного запису, оскільки SQLite виконує ці дії з мінімальними накладними витратами для невеликих баз даних.

Пошук і фільтрація даних, наприклад, вибір усіх платежів за певним договором, реалізовано через LINQ-запити, які транслюються у відповідні SQL-запити. Для підвищення ефективності використано індексацію ключових полів, таких як ідентифікатори договорів і дати платежів, що забезпечує швидкий доступ до даних із часовою складністю O(log n) для пошуку за індексованим полем, де n – кількість записів у таблиці. Це дозволяє системі швидко обробляти запити навіть за умови зростання обсягу даних, що є важливим для масштабування.

Для відображення даних у таблицях інтерфейсу, таких як списки договорів чи платежів, використано алгоритм прив'язки даних (data binding) через DataGridView. Цей підхід мінімізує ручну обробку даних, автоматично синхронізуючи відображення з базою даних. Оновлення таблиці після зміни даних, наприклад, додавання нового платежу, виконується шляхом повторного завантаження набору даних із часовою складністю O(n), де n – кількість відображуваних записів. Для оптимізації цього процесу застосовано обмеження на кількість записів, що відображаються одночасно, що зменшує навантаження на інтерфейс.

Формування документів, зокрема друк договору оренди, передбачає генерацію текстового вмісту з подальшим його розбиттям на сторінки. Для цього використано алгоритм розбиття тексту на рядки з урахуванням ширини сторінки, який базується на вимірюванні розміру тексту за допомогою Graphics.MeasureString. Цей алгоритм має часову складність O(m), де m – кількість слів у тексті, оскільки кожне слово обробляється для визначення можливості його розміщення в поточному рядку. Для багатосторінкового друку використано ітеративний підхід, який відстежує поточну позицію в тексті та визначає необхідність створення нової сторінки, що забезпечує гнучке масштабування для документів різного обсягу.

Особливу увагу приділено алгоритму валідації введених даних, наприклад, перевірці коректності сум платежів чи дат договорів. Цей алгоритм виконує послідовну перевірку формату даних із використанням винятків для обробки помилок. Часова складність валідації становить O(1) для кожного поля, оскільки перевірка виконується за фіксованою логікою, незалежно від обсягу даних. Такий підхід гарантує надійність системи та запобігає збереженню некоректних записів.

Ефективність обраних алгоритмів оцінюється з урахуванням типового сценарію використання, де обсяг даних залишається помірним (до кількох тисяч записів). SQLite забезпечує швидке виконання запитів для таких обсягів, а Entity Framework Core мінімізує накладні витрати на доступ до даних.

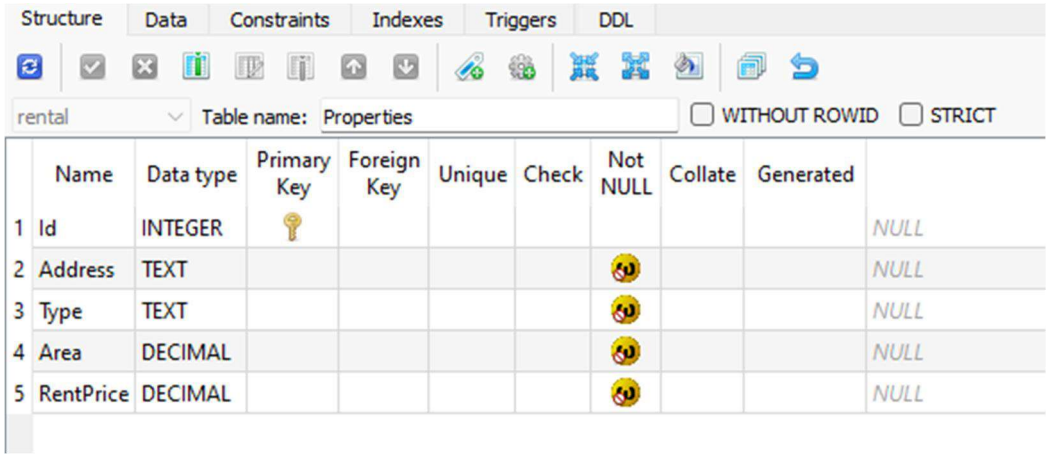
Алгоритми відображення та друку документів оптимізовані для швидкої роботи з текстовими даними, що відповідає потребам користувачів. У сукупності ці рішення забезпечують баланс між продуктивністю, простотою реалізації та зручністю використання, що є критичним для настільного додатка, призначеного для малого бізнесу.

2.2. База даних

Організація бази даних є ключовим елементом системи обліку оренди нерухомості, оскільки вона забезпечує надійне зберігання інформації про об’єкти нерухомості, орендарів, договори та платежі. Для реалізації бази даних обрано SQLite – легку реляційну систему управління базами даних, яка ідеально підходить для локальних настільних додатків завдяки своїй простоті, компактності та відсутності необхідності в окремому сервері. Структура бази даних спроєктована з урахуванням вимог системи, забезпечуючи ефективний доступ до даних і підтримку всіх необхідних операцій.

База даних складається з чотирьох основних таблиць, які відображають ключові сутності системи: об’єкти нерухомості, орендарів, договори оренди та платежі.

Таблиця Properties містить інформацію про нерухомі об’єкти, включаючи унікальний ідентифікатор, адресу, тип нерухомості (наприклад, квартира, офіс) і площу. Поле ідентифікатора є первинним ключем, що забезпечує унікальність кожного запису.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	Id	INTEGER								NULL
2	Address	TEXT								NULL
3	Type	TEXT								NULL
4	Area	DECIMAL								NULL
5	RentPrice	DECIMAL								NULL

Рисунок 2.2.1 – Таблиця Properties

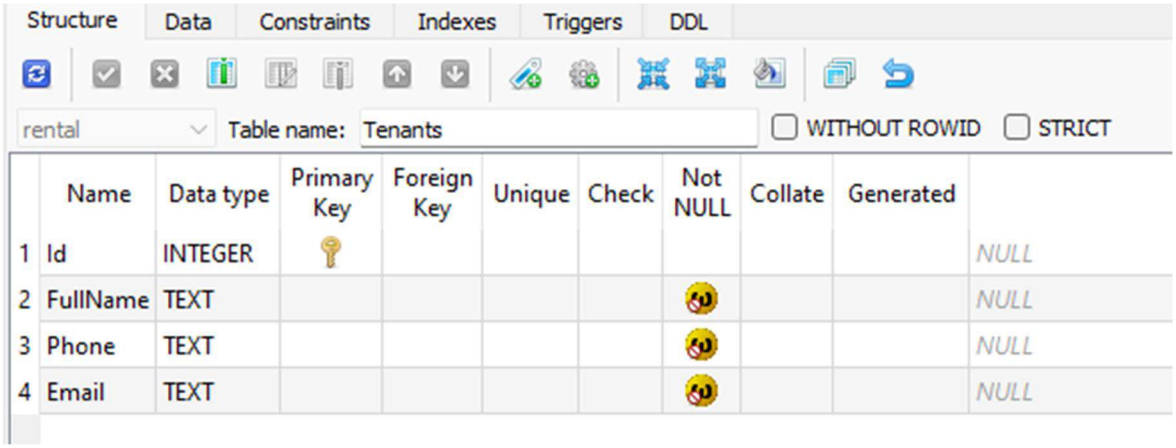


Рисунок 2.2.2 – Таблиця Tenants

Таблиця Tenants зберігає дані про орендарів, такі як унікальний ідентифікатор, повне ім’я, контактний телефон та електронна пошта, де ідентифікатор також виступає первинним ключем.

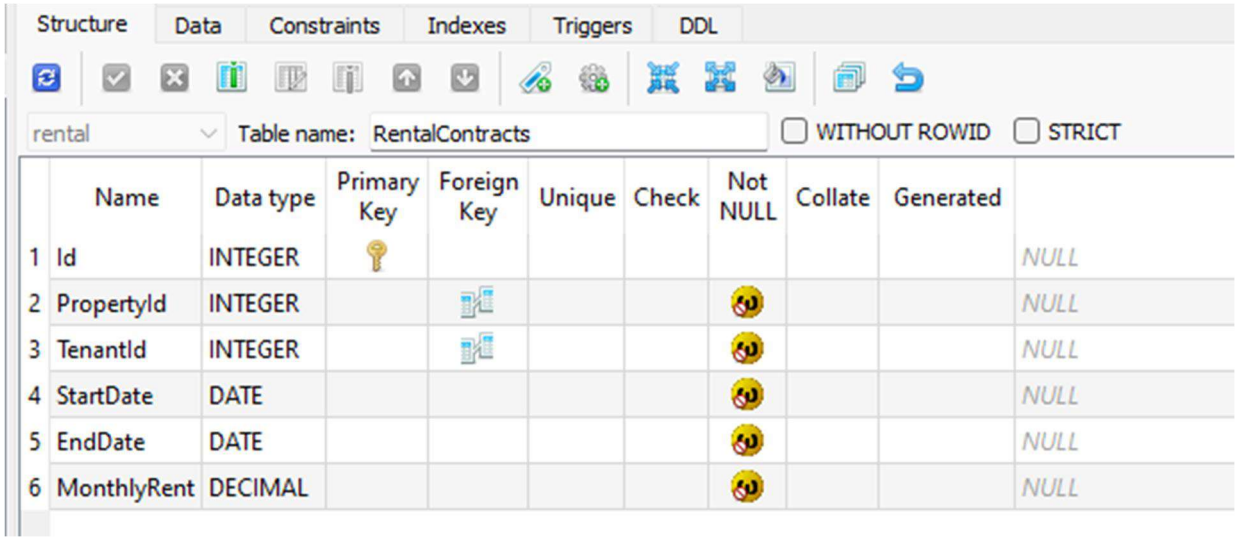
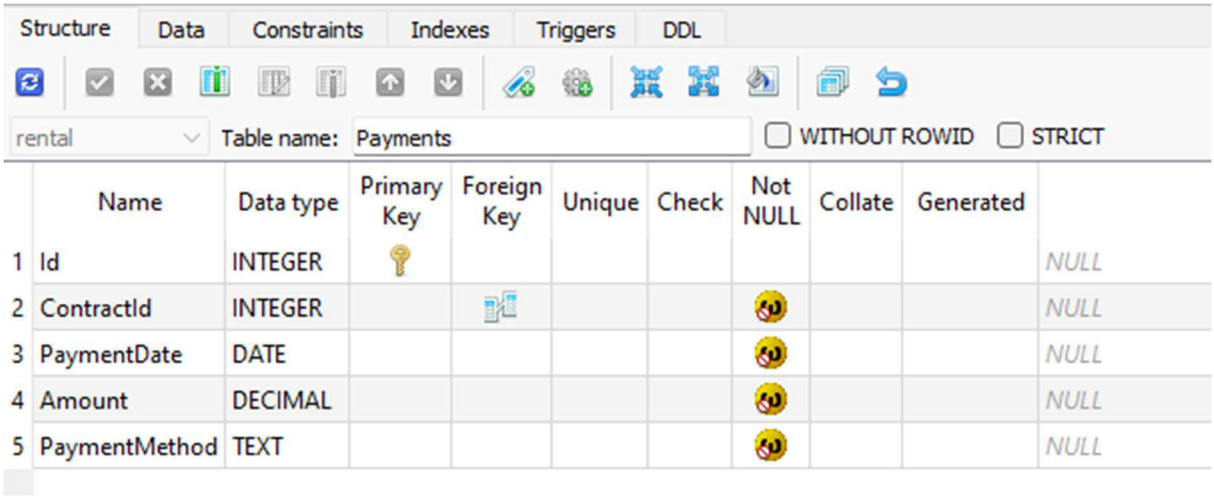


Рисунок 2.2.3 – Таблиця RentalContracts

Таблиця RentalContracts відповідає за зберігання інформації про договори оренди. Вона включає унікальний ідентифікатор договору, ідентифікатори пов’язаних об’єкта нерухомості та орендаря, дати початку та закінчення договору, а також суму місячної орендної плати. Ідентифікатор договору є первинним ключем, а ідентифікатори нерухомості та орендаря виступають зовнішніми ключами, які пов’язують цю таблицю з таблицями Properties і Tenants. Такий зв’язок забезпечує цілісність даних, дозволяючи

уникнути некоректних комбінацій. Наприклад, неможливо створити договір для неіснуючого об’єкта чи орендаря.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	Id	INTEGER								NULL
2	ContractId	INTEGER								NULL
3	PaymentDate	DATE								NULL
4	Amount	DECIMAL								NULL
5	PaymentMethod	TEXT								NULL

Рисунок 2.2.4 – Таблиця Payments

Таблиця Payments призначена для обліку платежів за договорами. Вона містить унікальний ідентифікатор платежу, ідентифікатор пов’язаного договору, дату платежу, суму та метод оплати (наприклад, готівка, картка, банківський переказ). Ідентифікатор платежу є первинним ключем, а ідентифікатор договору – зовнішнім ключем, що пов’язує платіж із відповідним договором. Це дозволяє легко відстежувати фінансові операції для кожного договору.

Для підвищення ефективності роботи бази даних використано індексацію ключових полів, зокрема ідентифікаторів і дат, що прискорює пошукові запити та фільтрацію даних. Наприклад, індекс на полі дати платежу в таблиці Payments забезпечує швидке отримання всіх платежів за певний період. Структура таблиць спроєктована таким чином, щоб уникнути надмірності даних, відповідаючи принципам нормалізації. Кожна таблиця містить лише необхідну інформацію, а зв’язки між таблицями дозволяють отримувати комплексні дані через об’єднання (JOIN).

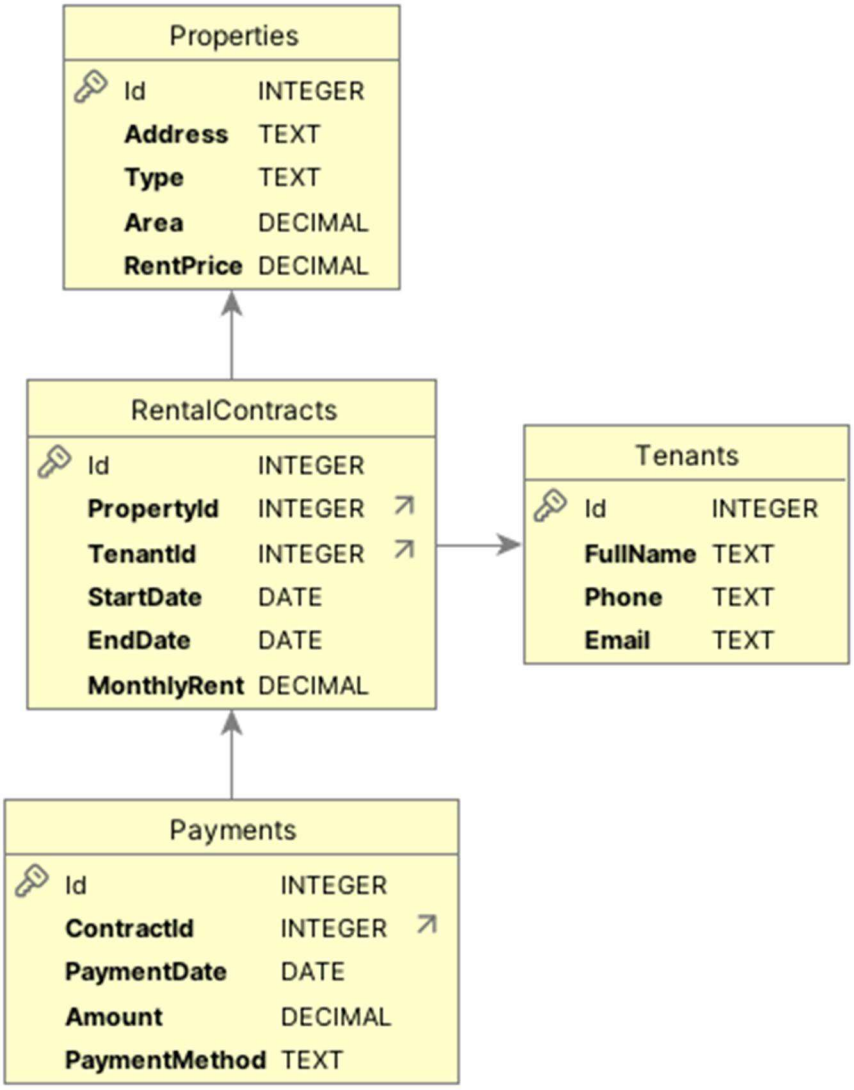


Рисунок 2.2.5 – Схема бази даних

Доступ до бази даних реалізовано через Entity Framework Core, який забезпечує об’єктно-реляційне відображення. Кожній таблиці відповідає клас у коді програми (Property, Tenant, RentalContract, Payment), а зв’язки між таблицями налаштовані через навігаційні властивості. Наприклад, клас RentalContract має властивості Property і Tenant, що дозволяють отримувати дані про об’єкт нерухомості та орендаря без додаткових запитів. Entity Framework Core автоматично генерує SQL-запити для операцій створення, читання, оновлення та видалення, що спрощує розробку та зменшує ймовірність помилок.

Для ініціалізації бази даних використано міграції Entity Framework Core, які дозволяють створювати та оновлювати схему бази даних відповідно до змін у коді. SQLite зберігає дані в одному файлі, що полегшує резервне копіювання та перенесення бази даних. Розмір бази даних залишається компактним навіть за наявності тисяч записів, що відповідає потребам малого бізнесу.

Така організація бази даних забезпечує швидкий доступ до інформації, надійність зберігання та гнучкість для подальшого розширення. Наприклад, у майбутньому можна додати таблиці для обліку витрат на утримання нерухомості чи зберігання документів, не порушуючи наявну структуру. Використання SQLite у поєднанні з Entity Framework Core дозволяє досягти балансу між простотою реалізації та функціональністю, що є критичним для настільного додатка, призначеного для локального використання.

2.3. Спроєктовані класи програми

Розробка системи обліку оренди нерухомості передбачає створення структури класів, яка відображає логіку роботи програми та забезпечує ефективну взаємодію між інтерфейсом користувача, базою даних і бізнес-логікою. Програма побудована на основі платформи .NET 8 із використанням Windows Forms для інтерфейсу та Entity Framework Core для роботи з базою даних SQLite.

Спроєктовані класи поділяються на три основні категорії:

- моделі даних;
- контекст бази даних;
- класи форм інтерфейсу.

Така організація дозволяє чітко розмежувати функціонал, забезпечуючи модульність і зручність підтримки коду.

Для представлення даних створено набір класів моделей, які відповідають сутностям бази даних.

```
public class Property
{
    public int Id { get; set; }
    public string Address { get; set; }
    public string Type { get; set; }
    public decimal Area { get; set; }
    public decimal RentPrice { get; set; }
}
```

Рисунок 2.3.1 – Клас Property

Клас Property описує об’єкт нерухомості та містить властивості для унікального ідентифікатора, адреси, типу нерухомості (наприклад, квартира чи офіс) і площі. Цей клас є основою для зберігання інформації про об’єкти, які здаються в оренду.

```
public class Tenant
{
    public int Id { get; set; }
    public string FullName { get; set; }
    public string Phone { get; set; }
    public string Email { get; set; }
}
```

Рисунок 2.3.2 – Клас Tenant

Клас Tenant представляє орендаря і включає унікальний ідентифікатор, повне ім’я, контактний телефон і електронну пошту. Ці дані необхідні для ідентифікації орендарів і ведення їхньої контактної інформації.

```
public class RentalContract
{
    public int Id { get; set; }
    public int PropertyId { get; set; }
    public int TenantId { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public decimal MonthlyRent { get; set; }
    public Property Property { get; set; }
    public Tenant Tenant { get; set; }
}
```

Рисунок 2.3.3 – Клас RentalContract

Клас RentalContract моделює договір оренди. Він містить унікальний ідентифікатор, ідентифікатори пов'язаних об'єкта нерухомості та орендаря, дати початку й закінчення договору, а також суму місячної орендної плати. Для зручності доступу до пов'язаних даних клас включає навігаційні властивості Property і Tenant, які дозволяють отримувати інформацію про об'єкт нерухомості та орендаря без додаткових запитів до бази даних.

```
public class Payment
{
    public int Id { get; set; }
    public int ContractId { get; set; }
    public DateTime PaymentDate { get; set; }
    public decimal Amount { get; set; }
    public string PaymentMethod { get; set; }
    public RentalContract Contract { get; set; }
}
```

Рисунок 2.3.4 – Клас Payment

Клас Payment відповідає за облік платежів і має унікальний ідентифікатор, ідентифікатор пов'язаного договору, дату платежу, суму та метод оплати (наприклад, готівка, картка чи переказ). Навігаційна властивість Contract забезпечує доступ до даних про договір, до якого відноситься платіж.

```
public class DatabaseContext : DbContext
{
    public DbSet<Property> Properties { get; set; }
    public DbSet<Tenant> Tenants { get; set; }
    public DbSet<RentalContract> RentalContracts { get; set; }
    public DbSet<Payment> Payments { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlite("Data Source=rental.db");
    }
}
```

Рисунок 2.3.5 – Клас DatabaseContext

Для взаємодії з базою даних створено клас DatabaseContext, який успадковується від DbContext із Entity Framework Core. Цей клас містить властивості типу DbSet для кожної моделі (Properties, Tenants, RentalContracts, Payments), що представляють таблиці бази даних. У конструкторі

DatabaseContext налаштовується підключення до файлу бази даних SQLite. Метод OnModelCreating використовується для конфігурації зв'язків між таблицями, зокрема зовнішніх ключів між RentalContracts і таблицями Properties та Tenants, а також між Payments і RentalContracts. Це забезпечує цілісність даних і автоматичну обробку зв'язків під час виконання запитів.

Інтерфейс користувача реалізовано через класи форм Windows Forms. Клас MainForm виступає головною формою програми, надаючи доступ до основних функцій через меню або кнопки для переходу до інших форм.

```
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 b
    protected override void Dispose(bool disposing)
    {

    #region Windows Form Designer generated code

    private System.Windows.Forms.Button btnProperties;
    private System.Windows.Forms.Button btnTenants;
    private System.Windows.Forms.Button btnContracts;
    private System.Windows.Forms.Button btnPayments;

}
```

Рисунок 2.3.6 – Клас MainForm

Клас PropertyForm відповідає за управління об’єктами нерухомості, дозволяючи додавати, редагувати та видаляти записи, а також відображати їх у таблиці.

```
partial class PropertyForm
{
    /// <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
    protected override void Dispose(bool disposing)
    {

    #region Windows Form Designer generated code

    private Button btnEdit;
    private Panel panel1;
    private ComboBox comboBoxType;
}
```

Рисунок 2.3.7 – Клас PropertyForm

Аналогічно, клас TenantForm забезпечує роботу з даними орендарів.

```
partial class TenantForm
{
    /// <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
    protected override void Dispose(bool disposing)
    {

    #region Windows Form Designer generated code

    private System.Windows.Forms.TextBox txtFullName;
    private System.Windows.Forms.TextBox txtPhone;
    private System.Windows.Forms.TextBox txtEmail;
    private System.Windows.Forms.Button btnAdd;
    private System.Windows.Forms.Button btnDelete;
    private System.Windows.Forms.DataGridView dataGridViewTenants;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private Panel panel1;
    private Button btnEdit;
}
```

Рисунок 2.3.8 – Клас TenantForm

Клас ContractForm призначений для управління договорами оренди, включаючи вибір об’єкта нерухомості та орендаря, зазначення термінів і суми оренди, а також функцію друку договору з підтримкою багатосторінкового форматування.

```
partial class ContractForm
{
    /// <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;
    protected override void Dispose(bool disposing)
    {

    #region Windows Form Designer generated code

    private System.Windows.Forms.ComboBox comboBoxProperties;
    private System.Windows.Forms.ComboBox comboBoxTenants;
    private System.Windows.Forms.DateTimePicker dateTimePickerStart;
    private System.Windows.Forms.DateTimePicker dateTimePickerEnd;
    private System.Windows.Forms.TextBox txtMonthlyRent;
    private System.Windows.Forms.Button btnAdd;
    private System.Windows.Forms.Button btnDelete;
    private System.Windows.Forms.DataGridView dataGridViewContracts;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Label label5;
    private Panel panel1;
    private Button btnEdit;
    private Button btnPrint;
}
```

Рисунок 2.3.9 – Клас ContractForm

Клас PaymentForm реалізує облік платежів, дозволяючи вибирати договір, вказувати дату, суму та метод оплати, а також відображати історію платежів.

```
partial class PaymentForm
{
    /// <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;
    protected override void Dispose(bool disposing)
    {
    #region Windows Form Designer generated code

    private System.Windows.Forms.ComboBox comboBoxContracts;
    private System.Windows.Forms.DateTimePicker dateTimePickerPayment;
    private System.Windows.Forms.TextBox txtAmount;
    private System.Windows.Forms.Button btnAdd;
    private System.Windows.Forms.Button btnDelete;
    private System.Windows.Forms.DataGridView dataGridViewPayments;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label label4;
    private Panel panel1;
    private Button btnEdit;
    private ComboBox comboBoxMethod;
}
```

Рисунок 2.3.10 – Клас PaymentForm

Кожен клас форми містить методи для завантаження даних із бази даних, обробки подій інтерфейсу (наприклад, натискання кнопок) і валідації введених даних. Для відображення даних використано компонент DataGridView, який синхронізується з DbSet через прив’язку даних. Обробка винятків, таких як некоректний формат суми чи відсутність пов’язаних записів, забезпечує стабільність роботи програми. Логіка друку документів у ContractForm базується на класі PrintDocument, який формує текст договору та розбиває його на сторінки з урахуванням ширини тексту.

Структура класів забезпечує чітке розмежування обов’язків: моделі даних відповідають за структуру інформації, DatabaseContext – за доступ до бази даних, а класи форм – за взаємодію з користувачем. Використання Entity Framework Core спрощує операції з даними, дозволяючи зосередитися на

бізнес-логіці. Модульна організація програми полегшує її підтримку та розширення, наприклад, додавання нових функцій, таких як облік витрат чи генерація звітів, без значних змін у наявному коді.

2.4. Програмна реалізація

Програмна реалізація системи обліку оренди нерухомості виконана на платформі .NET 8 з використанням Windows Forms для створення графічного інтерфейсу та SQLite як бази даних. Проєкт структуровано таким чином, щоб забезпечити модульність, зручність підтримки та чітке розмежування функціональних компонентів. Основна мета реалізації полягає в створенні настільного додатка, який дозволяє ефективно управляти даними про об’єкти нерухомості, орендарів, договори оренди та платежі, надаючи користувачу інтуїтивно зрозумілий інтерфейс і надійну обробку даних.

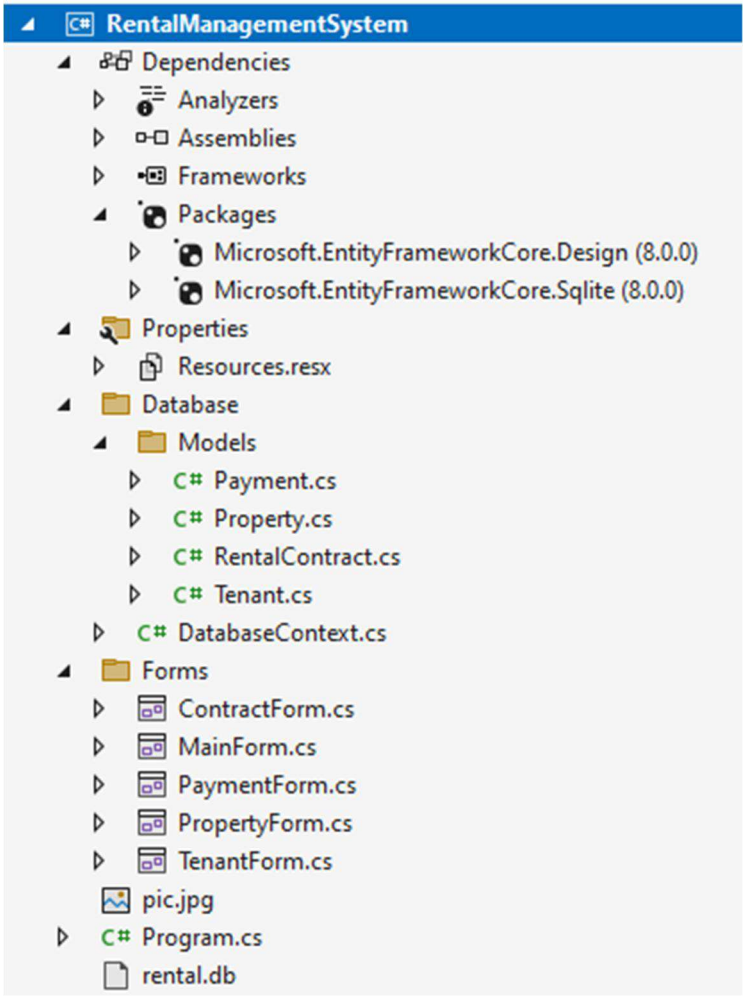


Рисунок 2.4.1 – Склад проєкту

Проєкт організовано у вигляді рішення Visual Studio, яке містить єдиний проєкт C# під назвою RentalManagementSystem. Структура проєкту поділена на кілька просторів імен для логічного групування класів за їхньою функціональністю. Основні папки та простори імен включають Database, Database.Models та Forms, що відповідають за роботу з базою даних, моделі даних і форми інтерфейсу користувача відповідно.

Папка Database містить клас DatabaseContext, який успадковується від DbContext бібліотеки Entity Framework Core. Цей клас відповідає за налаштування підключення до бази даних SQLite і визначення наборів даних (DbSet) для сутностей Properties, Tenants, RentalContracts і Payments. Конфігурація зв'язків між таблицями, таких як зовнішні ключі, виконується в методі OnModelCreating, що забезпечує цілісність даних.

Папка Database.Models включає класи моделей, які відображають структуру таблиць бази даних. Клас Property описує об'єкт нерухомості з полями для ідентифікатора, адреси, типу та площі. Клас Tenant містить дані про орендаря, включаючи ідентифікатор, ім'я, телефон і електронну пошту. Клас RentalContract моделює договір оренди з полями для ідентифікатора, посилань на нерухомість і орендаря, дат початку та закінчення, а також суми орендної плати. Клас Payment відповідає за облік платежів, зберігаючи ідентифікатор, посилання на договір, дату, суму та метод оплати. Усі моделі мають навігаційні властивості для доступу до пов'язаних даних, наприклад, RentalContract пов'язує Property і Tenant.

Папка Forms містить класи форм Windows Forms, які реалізують інтерфейс користувача. Проєкт включає п'ять основних форм: MainForm, PropertyForm, TenantForm, ContractForm і PaymentForm. Кожна форма відповідає за певний аспект управління даними та забезпечує виконання операцій створення, редагування, видалення та відображення записів.

Для інтеграції з базою даних використано NuGet-пакети Microsoft.EntityFrameworkCore.Sqlite та Microsoft.EntityFrameworkCore.Desig, які забезпечують роботу з SQLite і підтримку міграцій. Файл бази даних

(rental.db) зберігається локально в директорії програми, що спрощує розгортання. Рішення також містить файли конфігурації (appsettings.json або аналогічні), які визначають шлях до бази даних, хоча в даному випадку шлях задано безпосередньо в DatabaseContext.

MainForm є головною формою програми, яка запускається при старті та виконує роль навігаційного центру. Вона містить меню або панель із кнопками для переходу до інших форм: управління нерухомістю, орендарями, договорами та платежами. Форма не має складної логіки, її основна функція – забезпечення зручного доступу до всіх модулів системи. Інтерфейс включає заголовок програми та, за необхідності, базову статистику, наприклад, кількість активних договорів.

Рисунок 2.4.2 – MainForm

PropertyForm призначена для управління об’єктами нерухомості. Форма містить текстові поля для введення адреси, типу та площі об’єкта, а також таблицю DataGridView для відображення всіх записів. Кнопки «Додати», «Редагувати» та «Видалити» дозволяють маніпулювати даними. При виборі запису в таблиці поля форми автоматично заповнюються для

редагування. Логіка валідації перевіряє коректність введених даних, наприклад, чи є площа числовим значенням.

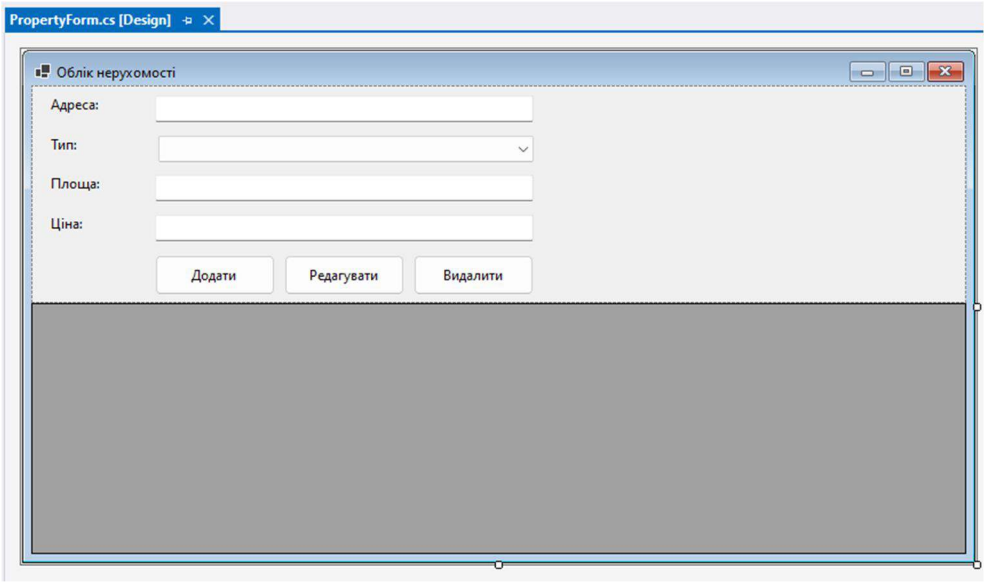


Рисунок 2.4.3 – PropertyForm

TenantForm відповідає за облік орендарів. Вона має поля для введення повного імені, телефону та електронної пошти, а також таблицю для відображення списку орендарів. Подібно до PropertyForm, форма підтримує операції додавання, редагування та видалення записів. Валідація забезпечує, що телефон і пошта мають правильний формат, хоча ці перевірки є базовими, щоб не ускладнювати використання.

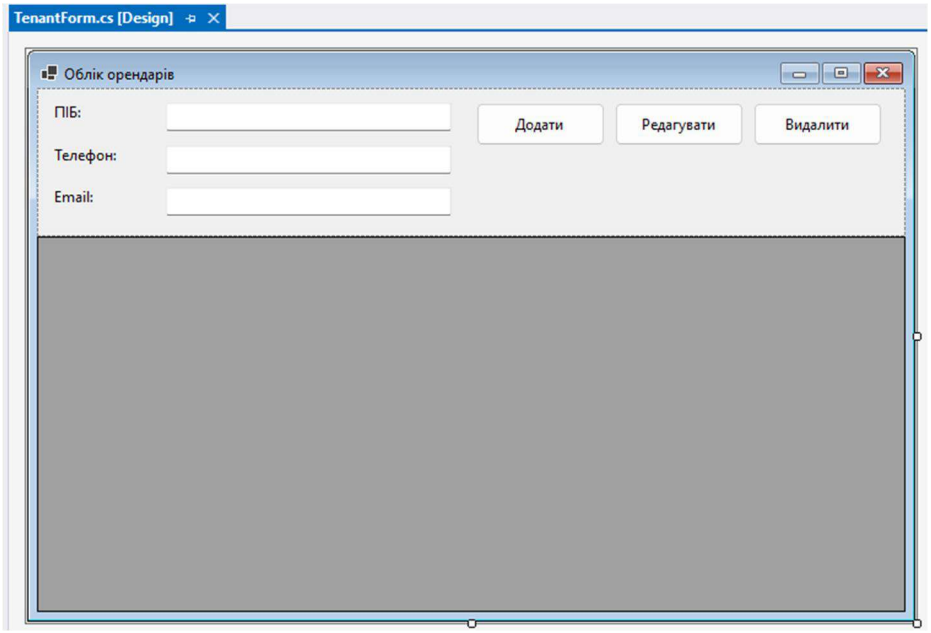


Рисунок 2.4.4 – TenantForm

ContractForm реалізує управління договорами оренди. Форма містить два випадаючих списки (ComboBox) для вибору об’єкта нерухомості та орендаря, поля для дат початку й закінчення договору, а також текстове поле для суми орендної плати. Таблиця DataGridView відображає список договорів із деталями, такими як адреса об’єкта та ім’я орендаря. Форма підтримує функцію друку договору, яка генерує документ із багатосторінковим форматуванням і переносом тексту за шириною сторінки. Логіка друку базується на класі PrintDocument, що дозволяє переглядати документ перед виведенням на принтер.

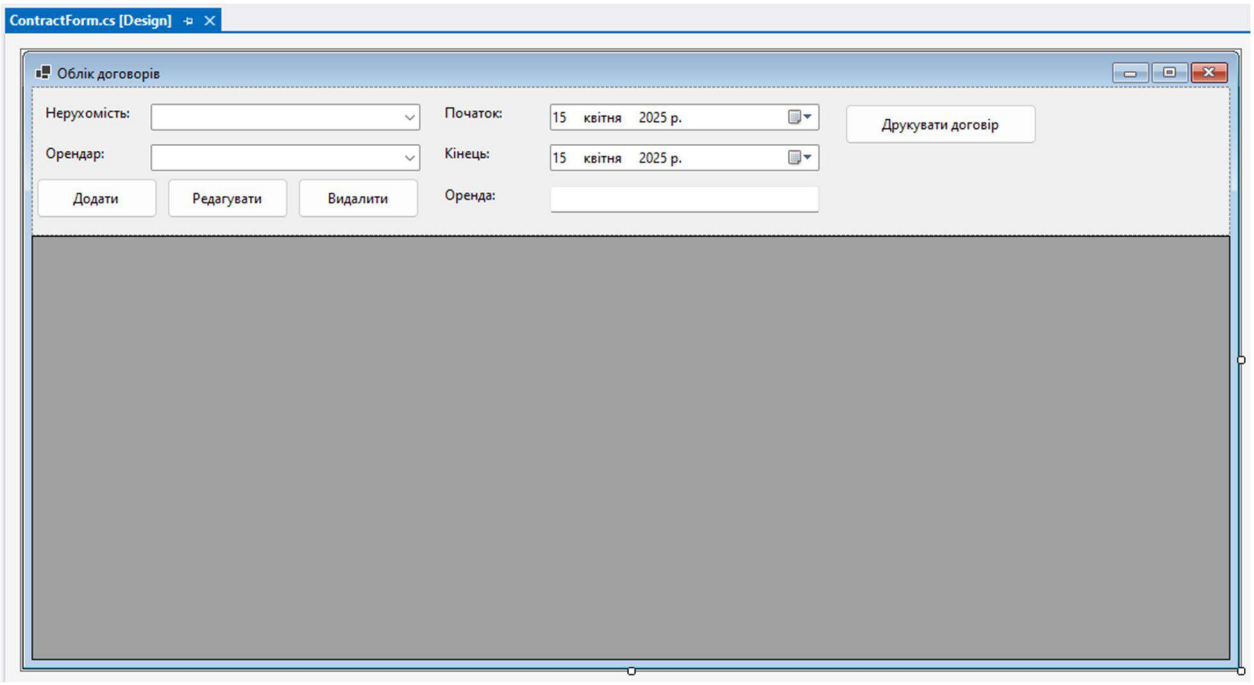


Рисунок 2.4.5 – ContractForm

PaymentForm призначена для обліку платежів. Вона включає випадаючий список для вибору договору, поле для дати платежу (DateTimePicker), текстове поле для суми та випадаючий список для методу оплати (готівка, картка, переказ тощо). Таблиця відображає історію платежів із деталями, такими як дата, сума та метод оплати. Форма забезпечує валідацію суми платежу, щоб уникнути некоректних значень, і синхронізує дані з базою після кожної операції.

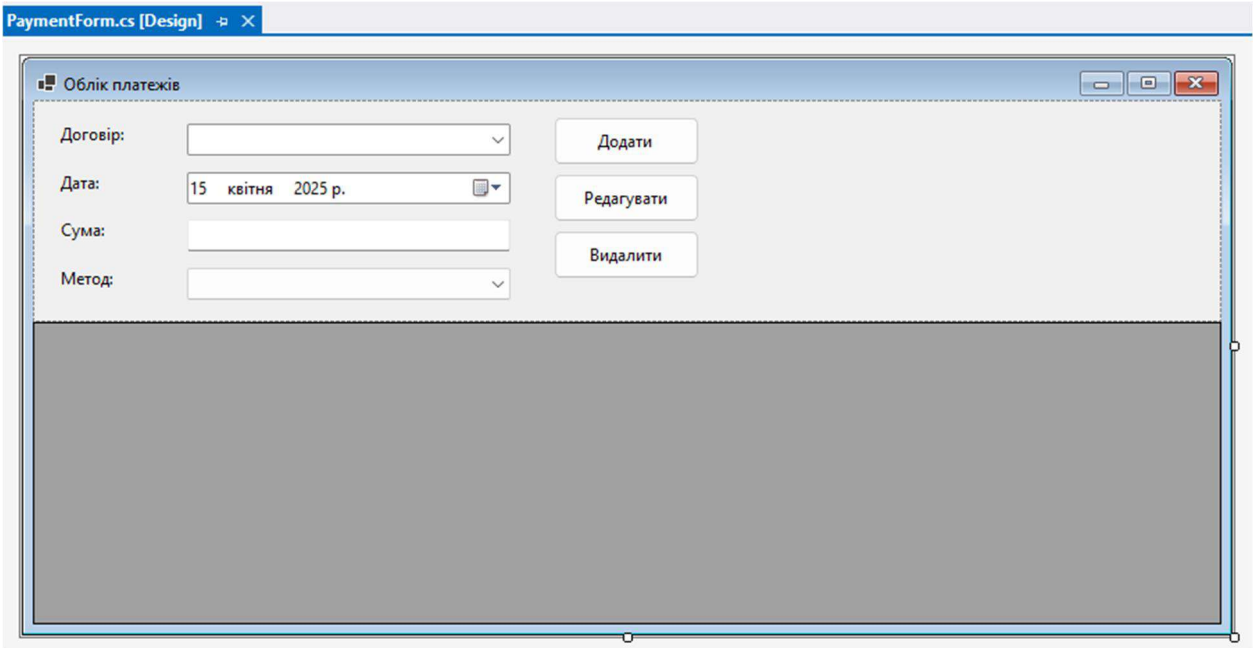


Рисунок 2.4.6 – PaymentForm

Кожна форма взаємодіє з базою даних через екземпляр DatabaseContext, який створюється при ініціалізації форми. Операції з даними виконуються через LINQ-запити, які Entity Framework Core перетворює на SQL. Для підвищення продуктивності використано ледаче завантаження (lazy loading) для навігаційних властивостей, що зменшує кількість запитів до бази даних. Обробка винятків реалізована для всіх операцій, що включають введення даних або доступ до бази, із виведенням повідомлень користувачу через MessageBox.

Інтерфейс форм побудовано з використанням стандартних компонентів Windows Forms, таких як TextBox, ComboBox, DateTimePicker і DataGridView. Для стилізації застосовано базові налаштування, що забезпечують читабельність і зручність. Логіка оновлення таблиць після зміни даних реалізована через повторне завантаження даних із DbSet, що гарантує актуальність відображення.

Система підтримує базову модульність: кожна форма є незалежним компонентом, який можна модифікувати чи розширювати без впливу на інші. Наприклад, додавання нової форми для обліку витрат потребуватиме лише створення нового класу форми та інтеграції з MainForm. Код проєкту

документується за допомогою коментарів, а структура класів відповідає принципам об’єктно-орієнтованого програмування, що полегшує подальшу розробку.

Програмна реалізація охоплює повний цикл управління орендними операціями, від введення даних до їхньої обробки та відображення. Структурований склад проєкту та продуманий дизайн форм забезпечують зручність використання, стабільність і можливість розширення функціоналу в майбутньому.

Висновки до розділу 2

У процесі проектування та розробки системи обліку оренди нерухомості виконано комплексний аналіз вимог, що дозволив сформувати ефективну структуру програми. Вибір алгоритмів для обробки даних, таких як операції з базою даних, відображення інформації та друк документів, базується на принципах оптимальності та простоти реалізації. Використання Entity Framework Core для доступу до даних і стандартних компонентів Windows Forms для інтерфейсу забезпечує швидке виконання основних операцій із часовою складністю, придатною для невеликих обсягів даних, характерних для малого бізнесу.

База даних спроєктована з урахуванням принципів нормалізації та ефективності, використовуючи SQLite як компактне рішення для локального зберігання. Структура з чотирма основними таблицями – Properties, Tenants, RentalContracts і Payments – дозволяє надійно зберігати інформацію та підтримувати зв’язки між сутностями. Застосування індексації та навігаційних властивостей підвищує швидкість пошуку та доступу до даних, що відповідає потребам системи.

Класи програми чітко розмежовують функціонал: моделі даних відповідають за структуру інформації, DatabaseContext забезпечує взаємодію з базою, а класи форм реалізують інтерфейс користувача. Така модульна організація полегшує підтримку коду та відкриває можливості для

розширення, наприклад, додавання нових функцій чи таблиць. Реалізація форм MainForm, PropertyForm, TenantForm, ContractForm і PaymentForm забезпечує повне покриття функціональних вимог, включаючи управління даними, валідацію введення та друк документів із підтримкою багатосторінкового формату.

Програмна реалізація підтвердила доцільність обраного технологічного стеку – Windows Forms, .NET 8 і SQLite. Структурований склад проєкту, із чітким поділом на простори імен і логічні компоненти, гарантує стабільність і зручність використання. Розроблена система відповідає поставленим задачам, надаючи користувачу інтуїтивний інструмент для обліку оренди нерухомості з потенціалом для подальшого вдосконалення.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для забезпечення коректної роботи системи обліку оренди нерухомості необхідно, щоб комп'ютер користувача відповідав мінімальним технічним вимогам. Оскільки програма розроблена як настільний додаток на основі Windows Forms, .NET 8 та SQLite, вона є легкою у плані ресурсів і підходить для використання на більшості сучасних персональних комп'ютерів. Нижче наведено детальний опис мінімальних вимог до апаратного та програмного забезпечення.

Програма призначена для роботи на операційних системах сімейства Microsoft Windows. Мінімально підтримуваною версією є Windows 10 (версія 1809 або новіша), хоча рекомендованою є Windows 11 для забезпечення максимальної сумісності та продуктивності. Операційна система має бути оновлена до актуального стану, щоб уникнути проблем із залежностями .NET 8.

Для запуску програми необхідно встановити .NET 8 Runtime (Desktop Runtime), оскільки додаток побудовано на цій платформі. Цей компонент забезпечує виконання коду та роботу з бібліотеками, включаючи Entity Framework Core для взаємодії з базою даних SQLite. У разі використання Visual Studio для розгортання чи модифікації програми також потрібен .NET 8 SDK, хоча він не є обов'язковим для кінцевого користувача.

Що стосується апаратного забезпечення, мінімальні вимоги до процесора включають двоядерний процесор із частотою 1.6 ГГц або вище. Така конфігурація забезпечує швидке виконання операцій із базою даних і відображення інтерфейсу. Рекомендується використовувати процесор із чотирма ядрами для комфортної роботи за умов багатозадачності.

Оперативна пам'ять має становити щонайменше 4 ГБ для стабільної роботи програми. Цього достатньо для обробки бази даних із кількома тисячами записів і відображення таблиць у формах. Для великих обсягів даних

або одночасної роботи з іншими програмами рекомендується 8 ГБ оперативної пам’яті.

Для зберігання програми та бази даних потрібно не менше 500 МБ вільного місця на жорсткому диску чи SSD. Сам додаток займає незначний обсяг, а база даних SQLite зберігається у вигляді одного файлу, розмір якого залежить від кількості записів. Наприклад, база з тисячею договорів і платежів зазвичай не перевищує 10-20 МБ. Рекомендується використовувати SSD для пришвидшення операцій із базою даних.

Програма не потребує спеціальних вимог до графічної підсистеми, оскільки інтерфейс Windows Forms використовує стандартні засоби рендерингу Windows. Достатньо будь-якого графічного адаптера, сумісного з Windows 10 або 11. Для комфортного відображення інтерфейсу рекомендована роздільна здатність екрана становить 1280x720 пікселів або вище.

Оскільки система працює локально, підключення до мережі Інтернет не є обов’язковим. Однак для початкового завантаження .NET 8 Runtime або оновлень може знадобитися доступ до мережі. Програма не залежить від зовнішніх серверів, а база даних зберігається локально, що забезпечує автономність роботи.

Додаткових периферійних пристроїв не потрібно, хоча для функції друку договорів необхідний принтер, підключений до комп’ютера та налаштований у системі Windows. Програма сумісна з більшістю стандартних принтерів, що підтримують виведення текстових документів.

Система обліку оренди нерухомості має низькі вимоги до апаратного та програмного забезпечення, що робить її доступною для використання на більшості сучасних комп’ютерів. Це забезпечує простоту розгортання та можливість експлуатації навіть у невеликих організаціях із обмеженими технічними ресурсами.

3.2. Інтерфейс користувача

Інтерфейс системи обліку оренди нерухомості розроблено з використанням Windows Forms, що забезпечує простоту, інтуїтивність і зручність для користувачів із різним рівнем технічної підготовки. Програма має модульну структуру, поділену на кілька форм, кожна з яких відповідає за певний аспект управління орендними операціями: нерухомість, орендарів, договори та платежі. Основна мета інтерфейсу – забезпечити швидкий доступ до всіх функцій, чітке відображення даних і просте виконання операцій, таких як додавання, редагування чи друк документів.

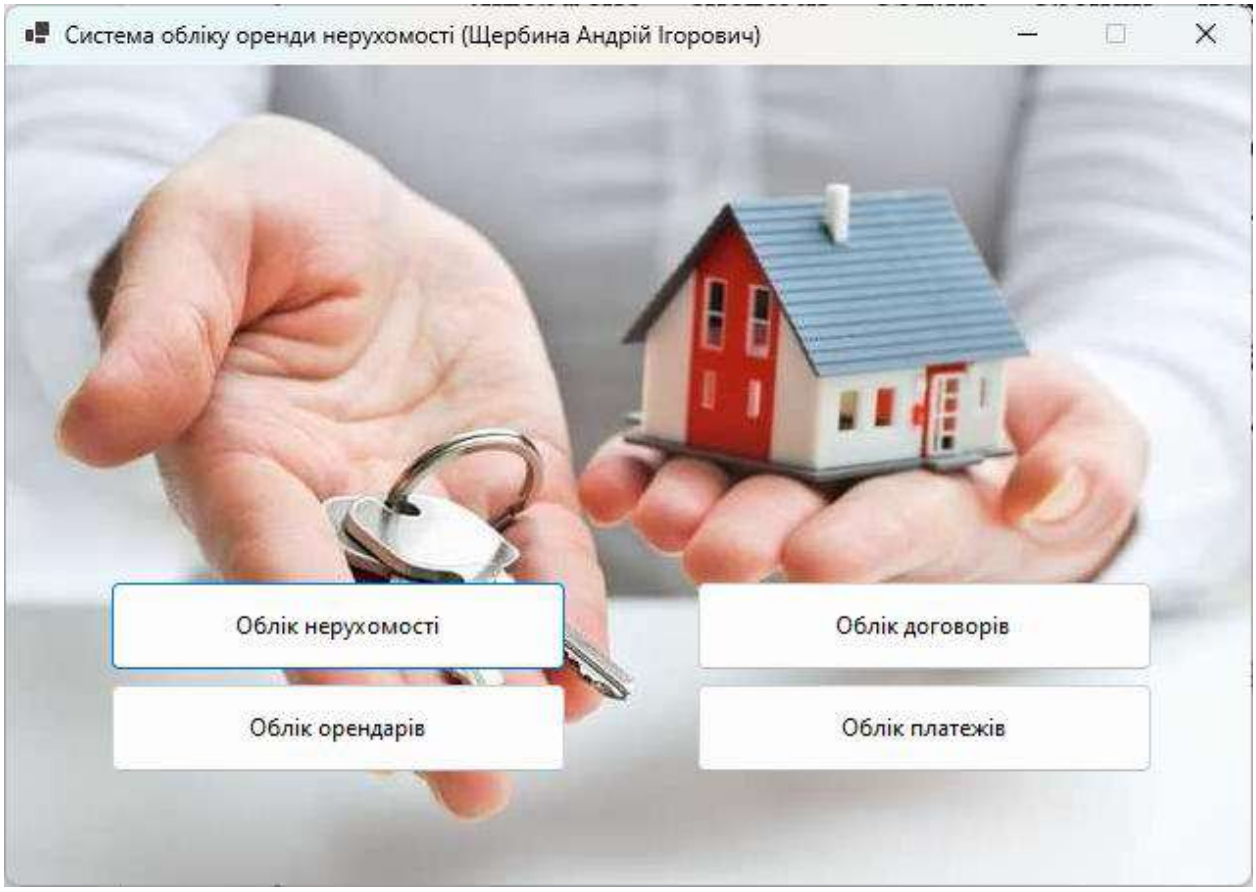


Рисунок 3.2.1 – Головна форма

При запуску програми користувач потрапляє на головну форму, яка виконує роль навігаційного центру. Інтерфейс цієї форми мінімалістичний і містить заголовок із назвою програми – «Система обліку оренди нерухомості». У верхній частині розташовано меню або панель із кнопками для переходу до інших форм: «Нерухомість», «Орендарі», «Договори» та «Платежі». Кожна

кнопка має чітке позначення, що полегшує орієнтацію. У центральній частині форми може відображатися базова інформація, наприклад, кількість активних договорів чи загальна сума платежів за місяць, хоча це залежить від конфігурації. Дизайн форми виконано в нейтральних тонах із використанням стандартної стилізації Windows Forms, що забезпечує знайомий вигляд для користувачів Windows.

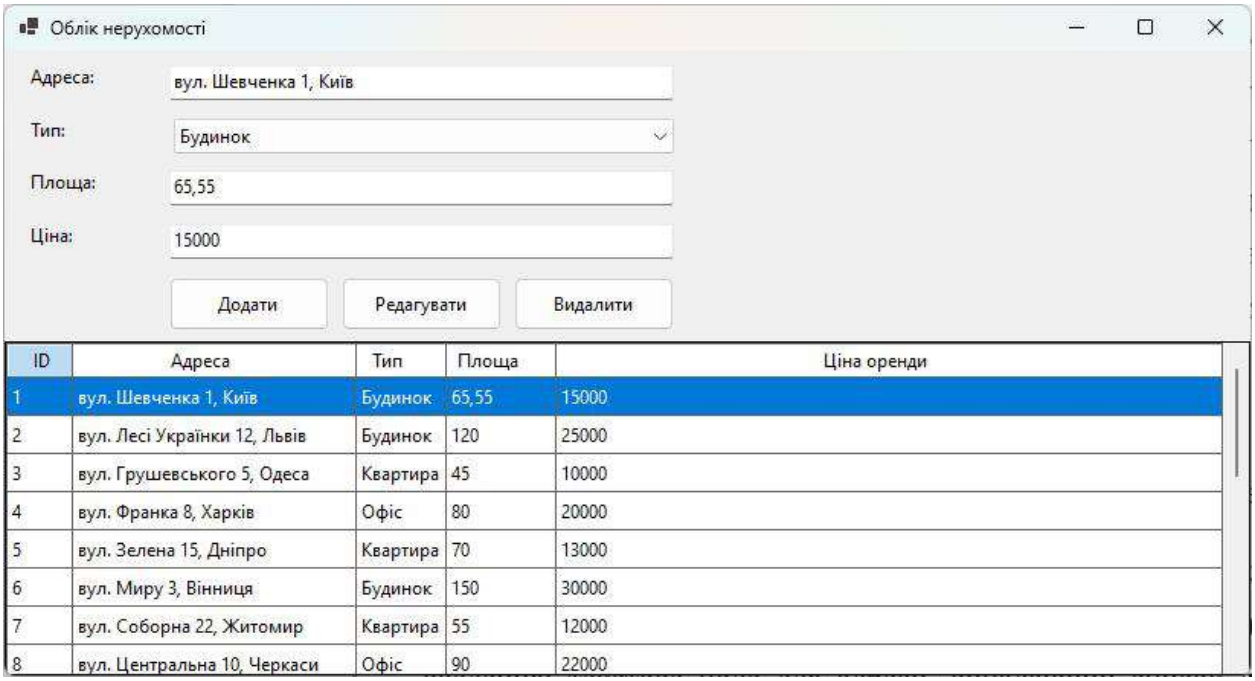
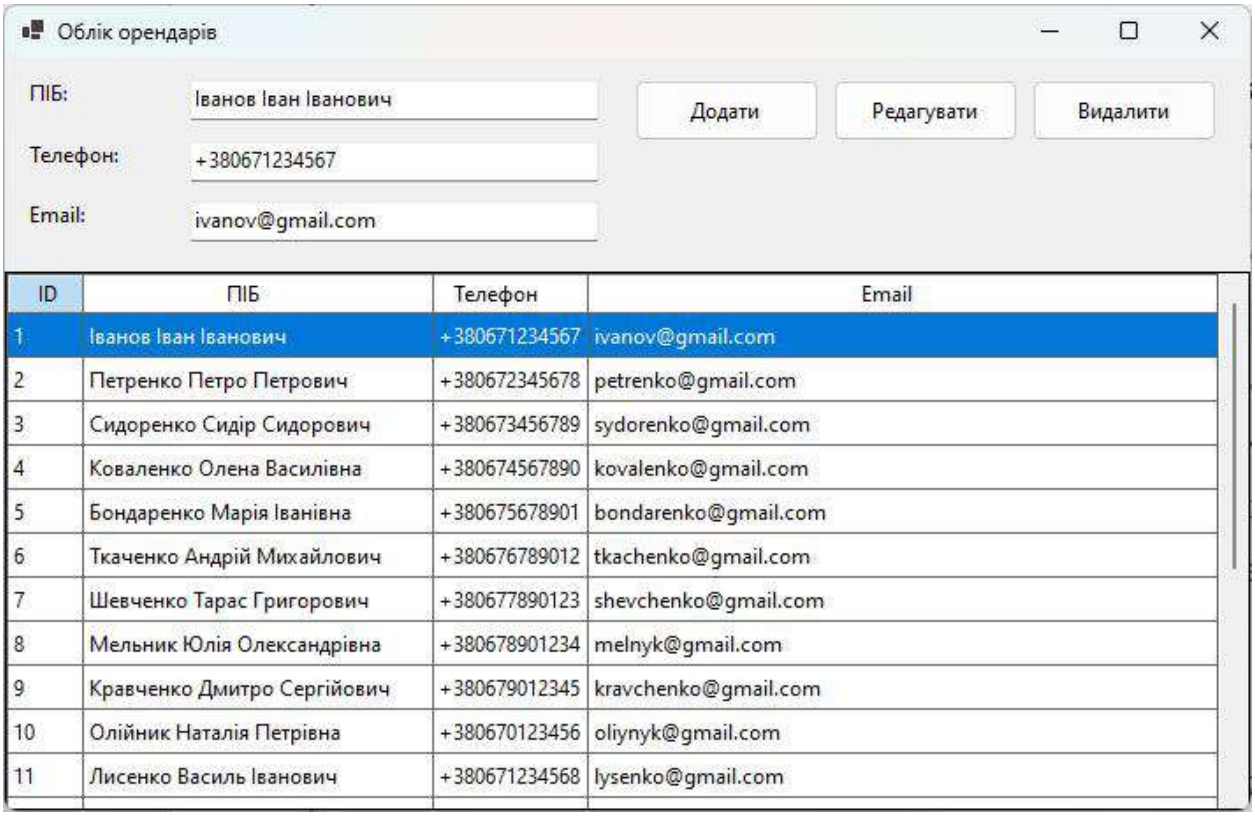


Рисунок 3.2.2 – Форма управління нерухомістю

Форма управління нерухомістю призначена для роботи з об’єктами нерухомості. У верхній частині форми розташовано панель із полями введення: текстове поле для адреси, випадаючий список для вибору типу нерухомості (наприклад, квартира, офіс, будинок) і текстове поле для вказання площі в квадратних метрах. Кожне поле супроводжується підписом, наприклад, «Адреса:» чи «Площа (кв.м):», для зрозумілості. Під полями введення розміщено три кнопки: «Додати», «Редагувати» та «Видалити», які дозволяють створювати новий об’єкт, змінювати вибраний або видаляти його з бази даних.

У нижній частині форми розташовано таблицю (DataGridView), яка відображає список усіх об’єктів нерухомості. Таблиця містить стовпці: «ID», «Адреса», «Тип» і «Площа». При виборі рядка в таблиці поля введення

автоматично заповнюються даними цього об’єкта для редагування. Таблиця підтримує сортування за стовпцями, що полегшує пошук потрібного запису. Інтерфейс форми компактний, із чітким розмежуванням зон введення та відображення даних, що мінімізує плутанину.

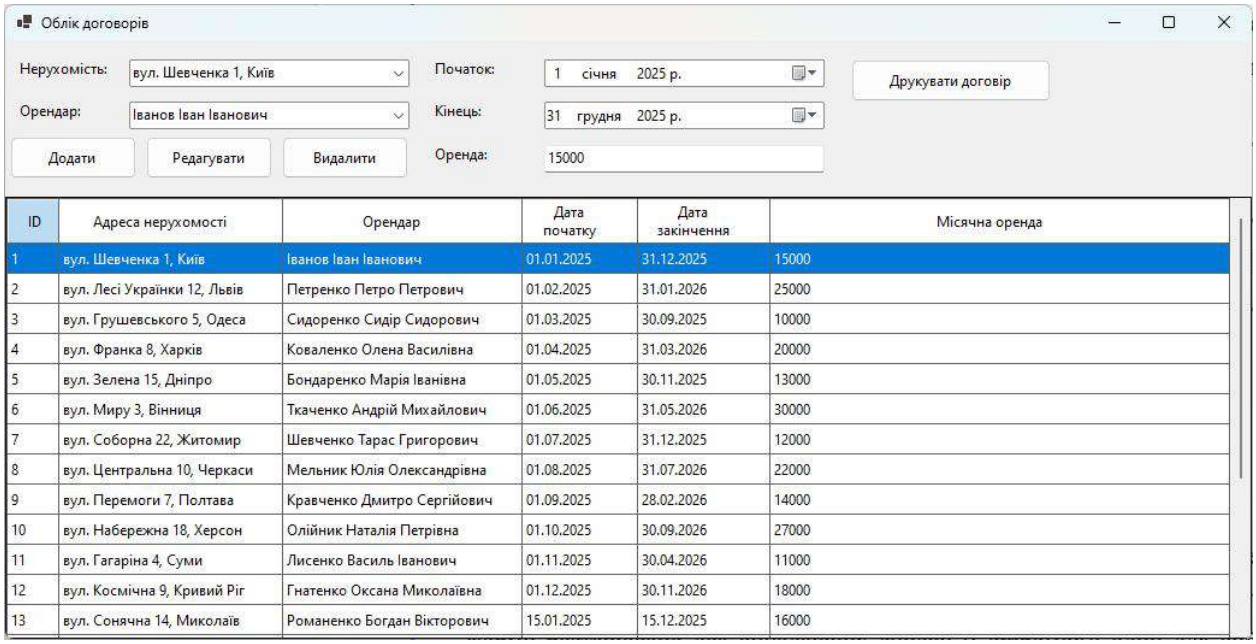


ID	ПІБ	Телефон	Email
1	Іванов Іван Іванович	+380671234567	ivanov@gmail.com
2	Петренко Петро Петрович	+380672345678	petrenko@gmail.com
3	Сидоренко Сидір Сидорович	+380673456789	sydorenko@gmail.com
4	Коваленко Олена Василівна	+380674567890	kovalenko@gmail.com
5	Бондаренко Марія Іванівна	+380675678901	bondarenko@gmail.com
6	Ткаченко Андрій Михайлович	+380676789012	tkachenko@gmail.com
7	Шевченко Тарас Григорович	+380677890123	shevchenko@gmail.com
8	Мельник Юлія Олександрівна	+380678901234	melnyk@gmail.com
9	Кравченко Дмитро Сергійович	+380679012345	kravchenko@gmail.com
10	Олійник Наталія Петрівна	+380670123456	oliynyk@gmail.com
11	Лисенко Василь Іванович	+380671234568	lysenko@gmail.com

Рисунок 3.2.3 – Форма управління орендарями

Форма управління орендарями має подібну структуру. У верхній частині розташовані текстові поля для введення повного імені, контактного телефону та електронної пошти, кожне з відповідним підписом. Наприклад, поле телефону позначено як «Телефон:», а пошта – як «Електронна пошта:». Під полями введення знаходяться кнопки «Додати», «Редагувати» та «Видалити» для виконання основних операцій.

Таблиця в нижній частині форми відображає список орендарів із стовпцями: «ID», «Повне ім’я», «Телефон» і «Електронна пошта». Як і в PropertyForm, вибір рядка в таблиці заповнює поля введення для редагування, а кнопки дозволяють керувати даними. Інтерфейс форми забезпечує просте введення даних, а розташування елементів відповідає стандартним принципам Windows Forms, що робить його передбачуваним для користувача.



ID	Адреса нерухомості	Орендар	Дата початку	Дата закінчення	Місячна оренда
1	вул. Шевченка 1, Київ	Іванов Іван Іванович	01.01.2025	31.12.2025	15000
2	вул. Лесі Українки 12, Львів	Петренко Петро Петрович	01.02.2025	31.01.2026	25000
3	вул. Грушевського 5, Одеса	Сидоренко Сидір Сидорович	01.03.2025	30.09.2025	10000
4	вул. Франка 8, Харків	Коваленко Олена Василівна	01.04.2025	31.03.2026	20000
5	вул. Зелена 15, Дніпро	Бондаренко Марія Іванівна	01.05.2025	30.11.2025	13000
6	вул. Миру 3, Вінниця	Ткаченко Андрій Михайлович	01.06.2025	31.05.2026	30000
7	вул. Соборна 22, Житомир	Шевченко Тарас Григорович	01.07.2025	31.12.2025	12000
8	вул. Центральна 10, Черкаси	Мельник Юлія Олександрівна	01.08.2025	31.07.2026	22000
9	вул. Перемоги 7, Полтава	Кравченко Дмитро Сергійович	01.09.2025	28.02.2026	14000
10	вул. Набережна 18, Херсон	Олійник Наталія Петрівна	01.10.2025	30.09.2026	27000
11	вул. Гагаріна 4, Суми	Лисенко Василь Іванович	01.11.2025	30.04.2026	11000
12	вул. Космічна 9, Кривий Ріг	Гнатенко Оксана Миколаївна	01.12.2025	30.11.2026	18000
13	вул. Сонячна 14, Миколаїв	Романенко Богдан Вікторович	15.01.2025	15.12.2025	16000

Рисунок 3.2.4 – Форма управління договорами

Форма управління договорами є центральною частиною системи, оскільки договори пов’язують нерухомість і орендарів. У верхній частині форми розташовано два випадаючих списки (ComboBox): один для вибору об’єкта нерухомості, другий – для орендаря. Ці списки заповнюються даними з бази, показуючи адреси об’єктів і імена орендарів. Далі йдуть два елементи DateTimePicker для вказання дат початку та закінчення договору, а також текстове поле для введення суми місячної орендної плати з підписом «Місячна оренда (грн):».

Під полями введення розміщено чотири кнопки: «Додати», «Редагувати», «Видалити» та «Друк». Кнопка «Друк» відкриває діалог попереднього перегляду, де користувач може переглянути договір перед виведенням на принтер. Договір автоматично форматується з урахуванням ширини сторінки та розбивається на кілька сторінок, якщо текст не вміщується. Таблиця в нижній частині форми відображає договори зі стовпцями: «ID», «Адреса нерухомості», «Орендар», «Дата початку», «Дата закінчення» і «Місячна оренда». Вибір рядка заповнює поля для редагування, а таблиця підтримує сортування для зручності пошуку.

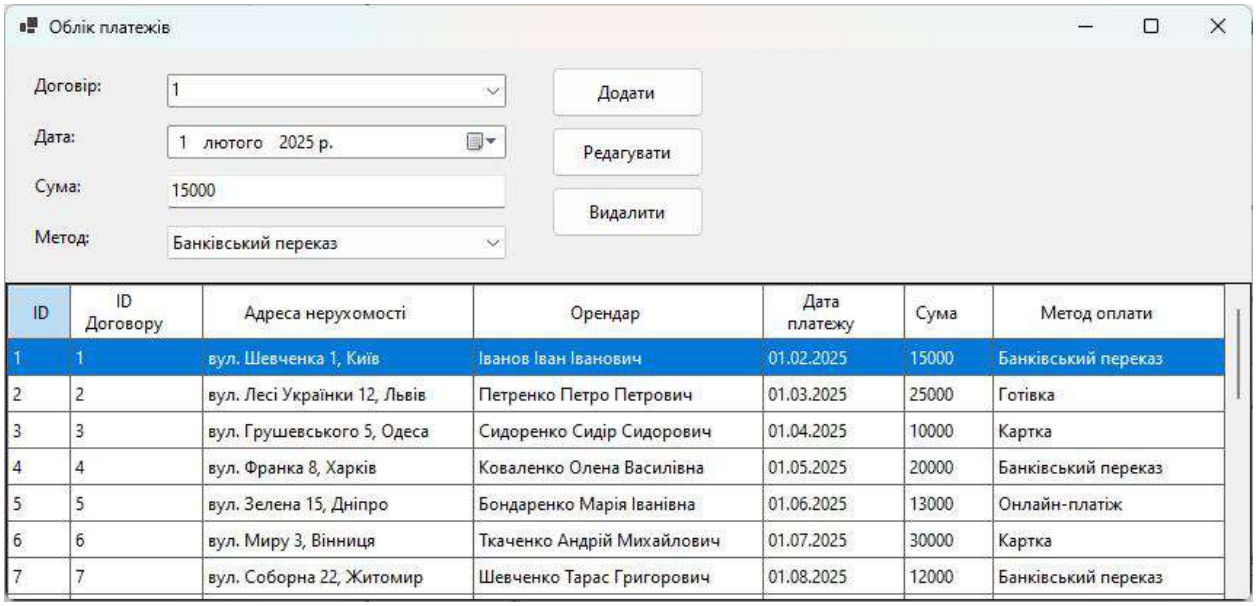


Рисунок 3.2.5 – Форма управління платежами

Форма управління платежами дозволяє вести облік фінансових операцій. У верхній частині розташовано випадаючий список для вибору договору, який показує номери договорів і пов’язані з ними адреси чи імена орендарів. Далі йде елемент DateTimePicker для вибору дати платежу, текстове поле для введення суми з підписом «Сума (грн):» і випадаючий список для методу оплати (готівка, картка, банківський переказ).

Кнопки «Додати», «Редагувати» та «Видалити» розміщені під полями введення і дозволяють керувати записами про платежі. Таблиця в нижній частині форми відображає історію платежів із стовпцями: «ID», «Договір», «Дата», «Сума» і «Метод оплати». Як і в інших формах, вибір рядка заповнює поля для редагування, а валідація даних (наприклад, перевірка суми на числове значення) запобігає помилкам.

Інтерфейс усіх форм виконано в єдиному стилі, використовуючи стандартні кольори та шрифти Windows Forms (наприклад, шрифт Segoe UI, розмір 9). Елементи керування розташовані логічно: поля введення – у верхній частині, кнопки – під ними, таблиця – внизу. Це забезпечує послідовність і передбачуваність взаємодії. Усі форми підтримують обробку помилок із виведенням сповіщень через MessageBox, наприклад, про некоректний формат суми чи відсутність вибраного запису.

Таблиці DataGridView дозволяють сортувати дані, а їхній розмір адаптується до розміру вікна форми. Випадаючі списки та поля введення мають обмеження для запобігання некоректним даним, наприклад, дата закінчення договору не може бути раніше дати початку. Функція друку в ContractForm включає попередній перегляд, що дозволяє користувачу перевірити вигляд документа перед виведенням.

Інтерфейс програми є простим і функціональним, орієнтованим на швидке виконання типових операцій. Модульна структура форм забезпечує чітке розмежування функцій, а єдиний стиль створює зручне середовище для роботи з даними оренди нерухомості.

3.3. Інструкція для роботи з програмою

Система обліку оренди нерухомості розроблена для автоматизації управління об’єктами нерухомості, орендарями, договорами та платежами. Програма має інтуїтивно зрозумілий інтерфейс, побудований на основі Windows Forms, і працює локально, не потребуючи підключення до мережі. Ця інструкція описує основні кроки для роботи з програмою, включаючи запуск, введення даних, управління записами та використання функції друку.

Встановлення та запуск програми

Перед початком роботи переконайтеся, що на комп’ютері встановлено операційну систему Windows 10 (версія 1809 або новіша) або Windows 11, а також .NET 8 Desktop Runtime. Якщо .NET 8 не встановлено, завантажте його з офіційного сайту Microsoft і виконайте інсталяцію. Програма поставляється у вигляді виконуваного файлу (RentalManagementSystem.exe) і файлу бази даних SQLite (rental.db), які мають бути розміщені в одній директорії.

Для запуску програми двічі клацніть на файлі RentalManagementSystem.exe. При першому запуску програма автоматично перевірить наявність бази даних і, якщо потрібно, ініціалізує її. У разі виникнення помилки, наприклад, відсутності файлу бази даних, з’явиться

повідомлення з описом проблеми. Після успішного запуску відкриється головна форма програми.

Навігація по програмі

Головна форма є центральною точкою доступу до всіх функцій. Вона містить меню або панель із чотирма кнопками: «Нерухомість», «Орендарі», «Договори» та «Платежі». Клацання на кожну кнопку відкриває відповідну форму для роботи з даними. Наприклад, вибір кнопки «Договори» відкриває форму управління договорами оренди. Щоб повернутися до головної форми, закрийте поточну форму або скористайтеся кнопкою «Назад», якщо вона передбачена.

Управління об’єктами нерухомості

Для роботи з об’єктами нерухомості відкрийте форму «Нерухомість». У верхній частині форми розташовано поля для введення адреси, типу нерухомості (випадаючий список із варіантами, наприклад, «Квартира», «Офіс») і площі. Щоб додати новий об’єкт, виконайте наступні дії:

1. Заповніть поле «Адреса», наприклад, «м. Київ, вул. Шевченка, 10».
2. Виберіть тип нерухомості зі списку.
3. Вкажіть площу у квадратних метрах, наприклад, «50».
4. Натисніть кнопку «Додати».

Новий запис з’явиться в таблиці в нижній частині форми, яка показує всі об’єкти з їхніми ідентифікаторами, адресами, типами та площею. Для редагування об’єкта клацніть на рядок у таблиці – поля автоматично заповняться даними. Внесіть зміни та натисніть «Редагувати». Щоб видалити об’єкт, виберіть його в таблиці та натисніть «Видалити». Програма попросить підтвердження, щоб уникнути випадкового видалення.

Управління орендарями

Форма «Орендарі» дозволяє додавати, редагувати та видаляти дані про орендарів. У верхній частині форми є поля для імені, телефону та електронної пошти. Щоб додати нового орендаря:

1. Введіть повне ім’я, наприклад, «Петренко Іван Іванович».

2. Вкажіть телефон, наприклад, «+380671234567».
3. Введіть електронну пошту, наприклад, «ivan@example.com».
4. Натисніть «Додати».

Запис додасться до таблиці, яка відображає ідентифікатор, ім’я, телефон і пошту всіх орендарів. Для редагування виберіть рядок у таблиці, змініть дані в полях і натисніть «Редагувати». Видалення виконується через вибір рядка та натискання кнопки «Видалити» з подальшим підтвердженням.

Управління договорами оренди

Форма «Договори» призначена для створення та управління договорами. Вона містить випадаючі списки для вибору об’єкта нерухомості та орендаря, поля для дат і суми оренди. Щоб створити новий договір:

1. У списку «Нерухомість» виберіть об’єкт, наприклад, «м. Київ, вул. Шевченка, 10».
2. У списку «Орендар» виберіть ім’я, наприклад, «Петренко Іван Іванович».
3. Вкажіть дату початку договору за допомогою календаря (DateTimePicker).
4. Вкажіть дату закінчення договору.
5. Введіть суму місячної оренди, наприклад, «10000».
6. Натисніть «Додати».

Договір з’явиться в таблиці, яка показує ідентифікатор, адресу об’єкта, ім’я орендаря, дати та суму. Для редагування виберіть договір у таблиці, змініть дані та натисніть «Редагувати». Для видалення натисніть «Видалити» та підтвердіть дію.

Форма також дозволяє друкувати договір. Виберіть договір у таблиці та натисніть «Друк». Відкриється вікно попереднього перегляду, де можна перевірити вигляд документа. Договір автоматично форматується з урахуванням ширини сторінки та розбивається на кілька сторінок, якщо текст довгий. У вікні перегляду натисніть кнопку друку, щоб вивести документ на принтер, або закрийте вікно, якщо друк не потрібен.

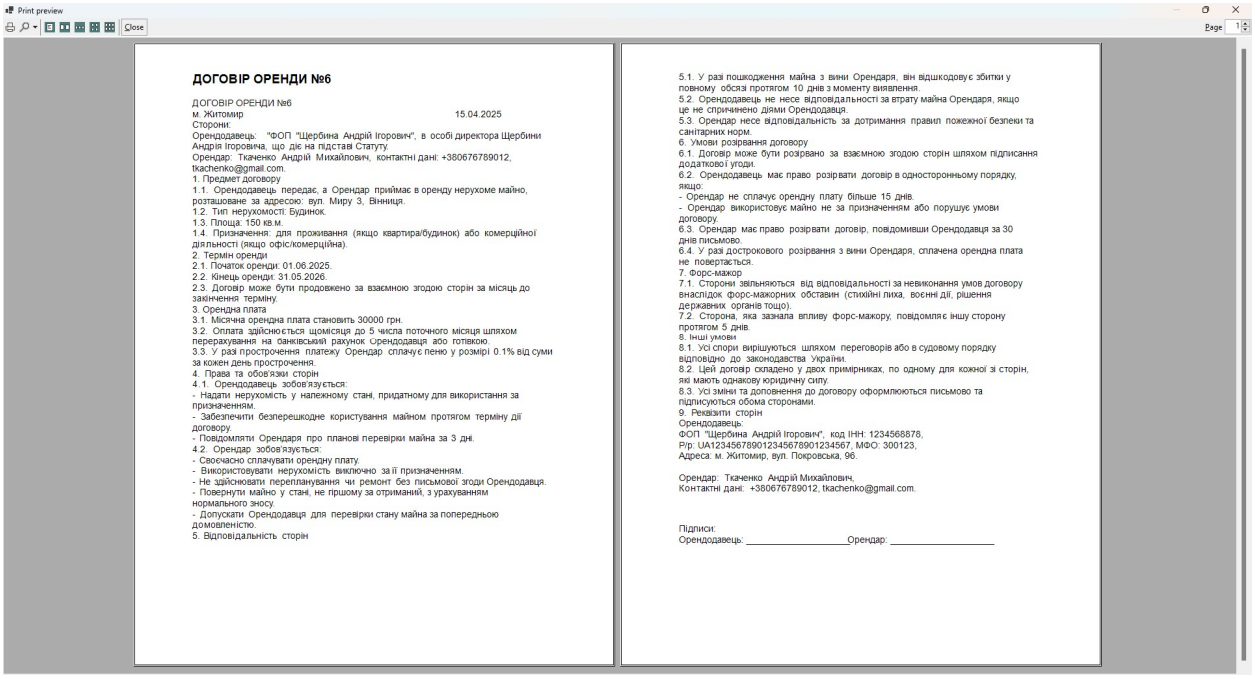


Рисунок 3.3.1 – Вікно попереднього перегляду договору оренди

Управління платежами

Форма «Платежі» використовується для обліку фінансових операцій. Вона містить список договорів, поле для дати, суми та методу оплати. Щоб додати платіж:

1. Виберіть договір зі списку, наприклад, «Договір №1 – вул. Шевченка, 10».
2. Вкажіть дату платежу за допомогою календаря.
3. Введіть суму, наприклад, «10000».
4. Виберіть метод оплати зі списку: «Готівка», «Картка» або «Переказ».
5. Натисніть «Додати».

Платіж додасться до таблиці, яка відображає ідентифікатор, договір, дату, суму та метод оплати. Для редагування виберіть платіж, змініть дані та натисніть «Редагувати». Видалення виконується через кнопку «Видалити» з підтвердженням.

Програма автоматично перевіряє введені дані. Наприклад, якщо сума оренди чи платежу не є числом, з’явиться повідомлення про помилку, наприклад, «Введіть коректне числове значення». Якщо спробувати видалити договір, який має пов’язані платежі, програма попередить про неможливість

дії. У разі проблем із базою даних, таких як відсутність файлу rental.db, відобразиться повідомлення з описом проблеми та рекомендаціями.

Для виходу з програми закрийте головну форму, натиснувши хрестик у правому верхньому куті вікна. Усі зміни в базі даних зберігаються автоматично після виконання операцій (додавання, редагування, видалення). Рекомендується періодично створювати резервну копію файлу rental.db, щоб уникнути втрати даних у разі збою.

Ця інструкція охоплює основні сценарії роботи з програмою. Інтерфейс розроблено так, щоб мінімізувати складність: усі дії виконуються через кілька клацань, а таблиці забезпечують швидкий огляд даних. Якщо потрібна додаткова допомога, зверніться до адміністратора системи або документації.

Висновки до розділу 3

Розроблена система обліку оренди нерухомості має низькі вимоги до апаратного та програмного забезпечення, що робить її доступною для використання на більшості сучасних комп’ютерів із операційною системою Windows 10 або 11. Необхідність встановлення .NET 8 Desktop Runtime і мінімальні апаратні ресурси, такі як 4 ГБ оперативної пам’яті та 500 МБ дискового простору, забезпечують простоту розгортання навіть у невеликих організаціях із обмеженим технічним оснащенням. Локальна робота програми без залежності від мережі Інтернет підвищує її надійність і автономність.

Інтерфейс користувача, побудований на основі Windows Forms, є інтуїтивно зрозумілим і логічно структурованим. Модульна організація форм – MainForm, PropertyForm, TenantForm, ContractForm і PaymentForm – забезпечує чітке розмежування функцій, дозволяючи користувачам швидко орієнтуватися в програмі. Елементи керування, такі як текстові поля, випадаючі списки, календарі та таблиці, розташовані передбачувано, що полегшує введення та перегляд даних. Функція друку договорів із підтримкою

багатосторінкового формату додає практичної цінності, відповідаючи потребам документообігу.

Інструкція для роботи з програмою охоплює всі основні сценарії використання: від запуску та навігації до управління об’єктами нерухомості, орендарями, договорами й платежами. Описано процеси додавання, редагування, видалення записів і друку документів, а також обробку помилок, що забезпечує захист від некоректних дій. Програма розроблена з урахуванням потреб користувачів із базовим рівнем технічних навичок, що підтверджується простотою виконання операцій і чіткими повідомленнями про помилки.

ВИСНОВКИ

У процесі виконання дипломної роботи на тему «Система обліку оренди нерухомості» було розроблено настільний додаток, який забезпечує автоматизацію управління об’єктами нерухомості, орендарями, договорами оренди та платежами. Реалізація проєкту базується на технологіях Windows Forms, .NET 8 та SQLite, що дозволило створити зручне, надійне та економічно виправдане рішення для малого та середнього бізнесу.

На етапі аналізу предметної області визначено ключові вимоги до системи, зокрема необхідність інтуїтивного інтерфейсу, швидкого доступу до даних і підтримки друку документів. Огляд аналогічних рішень показав, що більшість із них, таких як Rentec Direct чи Buildium, орієнтовані на хмарні технології та великі компанії, тоді як розроблена система займає нішу локальних додатків із мінімальними вимогами до ресурсів. Вибір Windows Forms, .NET 8 і SQLite обґрунтовано їхньою простотою розгортання, продуктивністю та відповідністю задачам проєкту.

Проектування системи включало розробку структури бази даних із чотирма основними таблицями – Properties, Tenants, RentalContracts і Payments, що забезпечують ефективне зберігання та обробку даних. Використання Entity Framework Core спростило взаємодію з базою, а індексація ключових полів підвищила швидкість пошуку. Алгоритми для операцій із даними, відображення таблиць і друку документів оптимізовані для роботи з помірними обсягами інформації, характерними для цільової аудиторії.

Програмна реалізація охоплює модульну структуру з п’ятьма формами: MainForm, PropertyForm, TenantForm, ContractForm і PaymentForm. Кожна форма відповідає за окремий аспект управління, забезпечуючи чітке розмежування функцій. Інтерфейс користувача розроблено з акцентом на простоту, із логічним розташуванням елементів керування та обробкою помилок. Функція друку договорів із підтримкою багатосторінкового формату і переносом тексту відповідає практичним потребам документообігу.

Керівництво користувача детально описує мінімальні системні вимоги, інтерфейс і процес роботи з програмою. Низькі вимоги до апаратного забезпечення – Windows 10/11, 4 ГБ оперативної пам’яті, 500 МБ дискового простору – роблять систему доступною для широкого кола користувачів. Інструкція охоплює всі основні операції, від додавання записів до друку документів, що полегшує освоєння програми навіть для користувачів із базовими технічними навичками.

Розроблена система успішно виконує поставлені задачі, забезпечуючи автоматизацію обліку оренди нерухомості з високим рівнем зручності та стабільності. Її модульна архітектура дозволяє легко розширювати функціонал, наприклад, додаванням модулів для обліку витрат чи генерації звітів. Практична цінність системи полягає в її здатності оптимізувати рутинні процеси для орендодавців, зменшуючи час і зусилля на ведення обліку.

Подальший розвиток проєкту може включати інтеграцію з мережевими функціями для віддаленого доступу, підтримку експорту даних у формати PDF чи Excel, а також розширення звітності для аналізу фінансових показників. Ці вдосконалення підвищать гнучкість системи та її привабливість для ширшої аудиторії. На даному етапі розроблена система є завершеним рішенням, готовим до використання в реальних умовах малого та середнього бізнесу.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Андерсон П. Програмування на C# для початківців. Київ : Діалектика, 2020. 512 с.
2. Бойко В. В. Інформаційні системи і технології в економіці : навч. посіб. Київ : Знання, 2018. 320 с.
3. Ватсон К. C# 8.0 і .NET Core 3.0 : сучасне програмування і розробка додатків. Харків : Фабула, 2021. 784 с.
4. Войтович О. П. Розробка баз даних із використанням SQLite : навч. посіб. Львів : Магнолія, 2019. 240 с.
5. Гаврилов М. В. Інформатика та інформаційні технології : підручник. Київ : Либідь, 2017. 432 с.
6. Гнатів І. М. Основи програмування на платформі .NET : навч. посіб. Тернопіль : Навчальна книга – Богдан, 2020. 288 с.
7. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 47 с.
8. Єрмоленко Т. О. Проектування інформаційних систем : навч. посіб. Одеса : ОНУ ім. І. І. Мечникова, 2019. 360 с.
9. Жуковський О. І. Технології розробки програмного забезпечення : підручник. Київ : КНЕУ, 2018. 400 с.
10. Зубков С. В. Windows Forms для розробки настільних додатків. Дніпро : Баланс-Клуб, 2020. 256 с.
11. Іванова Н. М. Управління нерухомістю : навч. посіб. Харків : ХНУ ім. В. Н. Каразіна, 2017. 280 с.
12. Коваленко А. С. Бази даних і системи управління базами даних : навч. посіб. Київ : Фенікс, 2019. 336 с.
13. Козловський В. О. Програмування інтерфейсів користувача в .NET. Львів : ЛНУ ім. Івана Франка, 2021. 304 с.
14. Корнійчук В. І. Автоматизація бізнес-процесів : навч. посіб. Київ : Центр учбової літератури, 2018. 352 с.

15. Ладика В. І. Економіка нерухомості : монографія. Суми : Університетська книга, 2019. 304 с.
16. Левицький О. М. Основи розробки баз даних SQLite : навч. посіб. Івано-Франківськ : Прикарпатський НУ, 2020. 200 с.
17. Мак-Дональд М. C# і .NET: створення додатків для Windows. Київ : ДіаСофт, 2020. 672 с.
18. Олійник Ю. В. Технології управління нерухомістю : навч. посіб. Дніпро : НМетАУ, 2018. 264 с.
19. Петренко О. П. Інформаційні системи в управлінні : підручник. Київ : Видавництво Ліра-К, 2019. 416 с.
20. Пономаренко Л. А. Проектування баз даних : навч. посіб. Харків : ХНЕУ ім. С. Кузнеця, 2017. 328 с.
21. Сидоренко В. М. Програмування на C# із використанням .NET Core. Одеса : Одеська політехніка, 2021. 384 с.
22. Степаненко П. Д. Автоматизовані системи управління : навч. посіб. Київ : НАУ, 2018. 296 с.
23. Ткачук М. В. Розробка програмного забезпечення : навч. посіб. Львів : Видавництво Львівської політехніки, 2019. 344 с.
24. Фрідман А. Основи об’єктно-орієнтованого програмування на C#. Київ : Видавнича група BHV, 2020. 464 с.
25. Шевченко О. О. Сучасні інформаційні технології в економіці : навч. посіб. Дніпро : ДДТУ, 2018. 272 с.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace RentalManagementSystem
{
    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()
        {
            btnProperties = new Button();
```