МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
ВІДДІЛЕННЯ «ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»
ЦИКЛОВА КОМІСІЯ СПЕЦІАЛЬНОСТІ «КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи
освітньо-професійний ступінь «фаховий молодший бакалавр»
на тему:

«Розробка прикладної програми для гаражного кооперативу»

Виконав студент (ка) _4_ курсу, групи П-42
спеціальності 122 «Комп'ютерні науки»

Бондар Іван Ярославович

Керівник Устименко Ярослав Іванович

Рецензент _____

Житомир – 2024 року

РЕФЕРАТ

Записка: 47 стор., 25 рис., 1 додаток, 20 джерел.

Ключові слова: WINDOWS FORM, C#, АВТОМАТИЗАЦІЯ, ІНТЕРФЕЙС КОРИСТУВАЧА, СИСТЕМА ОБЛІКУ, РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ІНФОРМАЦІЙНА СИТЕМА, SQLite.

Дипломна робота присвячена розробці програмного забезпечення для управління гаражним кооперативом з використанням мови програмування C# та бази даних SQLite. У вступі роботи обґрунтовується актуальність теми, формулюється мета та завдання дослідження. Аналізуються сучасні методи управління гаражними кооперативами та проводиться порівняльний аналіз аналогічних програмних рішень. Детально визначаються вимоги до розроблюваної програми, обґрунтовується вибір технологій розробки. У розділі процесу розробки описується архітектура програми, реалізація функціональності та проведення тестування. У завершальному розділі формулюються висновки з результатів дослідження та висловлюються рекомендації для подальшого розвитку програмного забезпечення.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

GMS - Garage Management System (Система управління гаражем)

CoopGarage - Назва іншої програми аналогічної тематики

C# - Мова програмування

SQLite - Вбудована база даних

GUI - Графічний інтерфейс користувача

CRUD - Операції з базою даних (створення, читання, оновлення, видалення)

Вступ

У сучасному світі ефективне управління організаціями різного масштабу та призначення є невід'ємною складовою їх успішного функціонування. Однією з таких організацій є гаражні кооперативи, які об'єднують власників гаражних приміщень та забезпечують їх належну експлуатацію. Традиційні методи ведення обліку, контролю та управління в гаражних кооперативах, такі як паперові журнали та ручні розрахунки, стають все менш ефективними та застарілими, вимагаючи значних затрат часу та ресурсів.

З огляду на це, актуальним є питання автоматизації управлінських процесів у гаражних кооперативах шляхом розробки спеціалізованих програмних засобів. Використання сучасних інформаційних технологій дозволяє значно підвищити точність, швидкість і надійність обробки даних, а також забезпечити зручність у користуванні для всіх учасників кооперативу.

Актуальність теми дипломної роботи зумовлена необхідністю впровадження інноваційних технологій в управління гаражними кооперативами. Розробка прикладної програми для гаражного кооперативу дозволить автоматизувати процеси реєстрації членів кооперативу, обліку гаражів та транспортних засобів, контролю оплати внесків, а також ведення фінансової звітності. Це, у свою чергу, сприятиме підвищенню ефективності управління кооперативом, зниженню кількості помилок та мінімізації людського фактора.

Метою даної дипломної роботи є розробка прикладної програми для автоматизації управлінських процесів у гаражному кооперативі. Для досягнення цієї мети необхідно вирішити наступні завдання:

Провести аналіз вимог до програмного забезпечення для управління гаражним кооперативом.

Розробити структуру бази даних для зберігання інформації про власників, гаражі, матеріали, ціни внесків та платежі.

Реалізувати користувацький інтерфейс програми з використанням мови програмування C# та технології Windows Forms.

Забезпечити функціональність для введення, редагування та видалення даних.

Реалізувати механізми обліку та контролю фінансових операцій у кооперативі.

Впровадити засоби захисту та збереження даних, включаючи резервне копіювання та шифрування.

Провести тестування програми та оцінити її відповідність вимогам користувачів.

Об'єктом дослідження є процеси управління гаражним кооперативом. Предметом дослідження є методи та засоби автоматизації цих процесів за допомогою спеціалізованого програмного забезпечення.

Обґрунтування вибору мови програмування

Для розробки прикладної програми обрано мову програмування C# та технологію Windows Forms. Такий вибір зумовлений рядом переваг: зручність розробки графічного інтерфейсу користувача, потужні можливості для роботи з базами даних, підтримка сучасних засобів безпеки та шифрування, а також висока продуктивність і стабільність роботи.

Розробка прикладної програми для гаражного кооперативу є актуальним і важливим завданням, що сприятиме підвищенню ефективності управління кооперативом та забезпеченню високого рівня обслуговування його членів. Виконання даної дипломної роботи дозволить отримати практичний досвід розробки програмного забезпечення, що відповідає сучасним вимогам та стандартам якості.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Основною задачею розробки прикладної програми для гаражного кооперативу є створення програмного забезпечення, яке дозволить автоматизувати та оптимізувати управлінські процеси в кооперативі. Програма повинна забезпечувати ефективний облік членів кооперативу, гаражних приміщень, матеріалів, платежів, а також контроль над фінансовими потоками та збереженням інформації.

Деталізація задачі

Для досягнення основної мети необхідно вирішити наступні завдання:

1. **Розробка модуля для управління власниками:**
    - Створити функціонал для введення та збереження даних про власників (ім'я, адреса, телефонний номер, зображення власника).
    - Забезпечити можливість редагування та видалення записів про власників.
2. **Розробка модуля для управління гаражами:**
    - Реалізувати функцію додавання інформації про гаражі (номер гаража, висота, ширина, довжина, матеріал стін, кількість поверхів, зображення, характеристики).
    - Надати можливість редагування та видалення записів про гаражі.
3. **Розробка модуля для управління матеріалами:**
    - Створити можливість додавання, редагування та видалення матеріалів, з яких можуть бути побудовані гаражі.
4. **Розробка модуля для обліку цін внесків:**
    - Реалізувати функцію додавання інформації про ціни внесків (величина ціни, дата початку дії ціни).
    - Забезпечити можливість редагування та видалення записів про ціни внесків.
5. **Розробка модуля для обліку платежів:**
    - Створити систему для введення та збереження даних про платежі

(місяць, рік, сума, дата здійснення платежу).

- Забезпечити можливість редагування та видалення записів про платежі.

6. **Розробка користувацького інтерфейсу:**
    - Створити інтуїтивно зрозумілий інтерфейс користувача з використанням Windows Forms.
    - Забезпечити доступ до всіх функцій програми через зручну навігацію та інтерфейсні елементи.
7. **Тестування та валідація:**
    - Провести всебічне тестування програми для виявлення та виправлення помилок.
    - Перевірити відповідність програми вимогам користувачів та стандартам якості.

Розробка прикладної програми для гаражного кооперативу спрямована на створення інструменту, який автоматизує та оптимізує ключові процеси управління кооперативом. Виконання зазначених завдань дозволить створити функціональне, надійне та зручне у використанні програмне забезпечення, яке забезпечить ефективне управління кооперативом та задовольнить потреби його членів.

1.2. Огляд та порівняння аналогічних систем.

Для розробки прикладної програми для гаражного кооперативу корисно проаналізувати існуючі рішення на ринку. Огляд п'яти аналогічних систем дозволить визначити їхні основні функції, переваги та недоліки, що допоможе визначити оптимальні підходи для розробки власного програмного забезпечення.

Garage Organizer

Garage Organizer - це програма, призначена для обліку та управління гаражними кооперативами. Вона забезпечує управління даними про власників, гаражі, фінансові операції та надає можливість генерувати звіти.

Основні функції:

- облік власників гаражів;
- реєстрація та управління гаражними приміщеннями;
- ведення фінансових операцій та платежів;
- генерація звітів та аналітики.

Переваги:

- Інтуїтивно зрозумілий інтерфейс
- Можливість інтеграції з бухгалтерськими системами
- Регулярні оновлення та підтримка

Недоліки:

- Висока вартість ліцензії
- Обмежена кількість користувачів у базовій версії

Cooperative Garage Manager (CGM)

CGM - це веб-орієнтоване рішення для управління гаражними кооперативами. Система забезпечує доступ до даних через інтернет та дозволяє здійснювати управління кооперативом з будь-якого місця.

Основні функції:

- Управління даними про власників та гаражі
- Контроль платежів та внесків
- Нотіфікації про несплачені внески
- Аналітика та звіти

Переваги:

- Доступ з будь-якого пристрою через веб-інтерфейс
- Регулярне резервне копіювання даних
- Гнучка система налаштувань

Недоліки:

- Потребує постійного інтернет-з’єднання
- Щомісячна плата за користування

Garage Management Pro

Garage Management Pro - це програма для управління гаражними

кооперативами, яка орієнтована на локальне використання. Вона пропонує широкий набір функцій для обліку та контролю.

Основні функції:

- Реєстрація та облік гаражів
- Управління даними про власників
- Облік фінансових операцій
- Генерація детальних звітів

Переваги:

- Висока продуктивність
- Інтуїтивний користувацький інтерфейс
- Одноразова плата за ліцензію

Недоліки:

- Обмежений доступ до даних з інших пристроїв
- Відсутність підтримки хмарних сервісів

MyGarage Software

MyGarage Software - це комплексне рішення для управління гаражними кооперативами з акцентом на простоту використання та налаштування.

Основні функції:

- Управління обліковими записами власників
- Реєстрація та контроль гаражних приміщень
- Фінансовий облік та моніторинг платежів
- Створення звітів

Переваги

- Простота налаштування та використання
- Доступна ціна
- Підтримка багатьох мов

Недоліки

- Обмежена функціональність у базовій версії
- Відсутність мобільного додатку

GarageSoft

GarageSoft - це спеціалізоване ПЗ для управління гаражними кооперативами, яке пропонує широкий набір функцій для обліку та контролю.

Основні функції

- Облік власників та гаражів
- Управління фінансовими операціями
- Автоматичне нарахування внесків
- Генерація аналітичних звітів

Переваги

- Широкий набір функцій
- Підтримка різних мов
- Інтеграція з бухгалтерськими системами

Недоліки

- Висока вартість для невеликих кооперативів
- Складність налаштування для нових користувачів

Аналіз існуючих систем показав, що кожна з них має свої переваги та недоліки. Для розробки прикладної програми для гаражного кооперативу варто врахувати наступні аспекти:

Функціональність: забезпечити основні функції обліку власників, гаражів, фінансових операцій та генерування звітів.

Інтерфейс: створити інтуїтивно зрозумілий та зручний інтерфейс користувача.

Доступність: забезпечити можливість роботи як локально, так і через інтернет.

Безпека: впровадити механізми захисту даних, включаючи шифрування та резервне копіювання.

Враховуючи переваги та недоліки аналогічних систем, можна створити оптимальне рішення, що відповідатиме потребам гаражного кооперативу та забезпечить ефективне управління його діяльністю.

1.3. Вибір та обґрунтування технологій розробки

Для розробки прикладної програми для гаражного кооперативу було обрано кілька ключових технологій: мову програмування C#, платформу Windows Forms та базу даних SQLite. Кожна з цих технологій має свої переваги та особливості, що робить їх оптимальним вибором для вирішення поставлених задач.

Мова програмування C#

C# є сучасною, об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft. Вибір C# для розробки даного програмного забезпечення обумовлений наступними факторами:

– Потужність та гнучкість: C# надає широкі можливості для розробки додатків будь-якої складності, підтримуючи різноманітні програмні парадигми, такі як об'єктно-орієнтоване програмування (ООП) та функціональне програмування.

– Інтеграція з .NET: C# є основною мовою для .NET, що забезпечує велику кількість вбудованих класів і бібліотек для реалізації широкого спектру функцій, від роботи з базами даних до створення графічних інтерфейсів користувача.

– Підтримка Windows Forms: C# дозволяє легко створювати додатки з графічним інтерфейсом користувача (GUI) за допомогою Windows Forms, що робить його ідеальним вибором для розробки настільних програм.

– Безпека та управління пам'яттю: C# включає вбудовані механізми управління пам'яттю та обробки помилок, що сприяє підвищенню надійності та безпеки програмного забезпечення.

Платформа Windows Forms

Windows Forms - це бібліотека класів у .NET для створення графічних додатків на базі Windows. Вибір Windows Forms для розробки графічного інтерфейсу користувача обумовлений наступними причинами:

– Простота використання: Windows Forms надає інтуїтивно зрозумілий інтерфейс для розробки GUI, що дозволяє швидко створювати форми та

компоненти, такі як кнопки, текстові поля, таблиці тощо.

– Широкі можливості налаштування: Windows Forms дозволяє легко налаштовувати вигляд та поведінку компонентів, забезпечуючи створення зручного та функціонального інтерфейсу користувача.

– Підтримка інтеграції з базами даних: Windows Forms забезпечує засоби для легкого підключення та взаємодії з різноманітними базами даних, що є ключовою вимогою для даного проекту.

– Широке розповсюдження та підтримка: Windows Forms є добре підтримуваною технологією з великою кількістю доступної документації, прикладів та інструментів для розробки.

База даних SQLite

SQLite - це вбудована реляційна база даних, що не вимагає окремого серверного програмного забезпечення для функціонування. Вибір SQLite для зберігання даних обумовлений наступними перевагами:

– Легка інтеграція: SQLite легко інтегрується з C# та Windows Forms, забезпечуючи просту та ефективну взаємодію між програмою та базою даних.

– Автономність: SQLite є вбудованою базою даних, що дозволяє використовувати її без налаштування окремого серверного середовища, що знижує складність та витрати на розгортання.

– Висока продуктивність: SQLite забезпечує швидку обробку даних навіть при значному обсязі записів, що є важливим для забезпечення ефективності роботи програми.

– Зберігання даних у файлі: SQLite зберігає всю базу даних в одному файлі, що полегшує резервне копіювання, перенесення та управління даними.

– Безкоштовне використання: SQLite є безкоштовною для використання, що знижує загальні витрати на розробку програмного забезпечення.

Вибір технологій для розробки прикладної програми для гаражного кооперативу базується на їхній здатності забезпечити високу продуктивність, надійність, простоту використання та інтеграцію. Мова програмування C#, платформа Windows Forms та база даних SQLite разом створюють потужний

інструментарій для реалізації всіх необхідних функцій, забезпечуючи при цьому зручний та інтуїтивно зрозумілий інтерфейс користувача. Такий підхід дозволяє створити ефективне, стабільне та зручне програмне забезпечення для управління гаражним кооперативом.

Висновки до розділу 1

У процесі розробки прикладної програми для гаражного кооперативу було вирішено низку важливих завдань, спрямованих на автоматизацію управлінських процесів та підвищення ефективності діяльності кооперативу.

Автоматизація управління гаражним кооперативом є вкрай актуальною в умовах сучасного розвитку інформаційних технологій. Традиційні методи ведення обліку не задовольняють вимоги ефективності та точності, що робить розробку спеціалізованого програмного забезпечення необхідною.

Метою роботи було створення програми, яка забезпечить автоматизацію обліку та управління гаражним кооперативом. Завдання включали аналіз вимог, розробку бази даних, створення графічного інтерфейсу користувача, забезпечення функціональності для введення та редагування даних, а також забезпечення безпеки та збереження даних.

Об'єктом дослідження були процеси управління гаражним кооперативом, а предметом - методи та засоби автоматизації цих процесів. Глибоке розуміння специфіки управління кооперативом дозволило створити ефективне програмне забезпечення.

Для розробки були обрані мова програмування C#, платформа Windows Forms та база даних SQLite. Ці технології забезпечили гнучкість, продуктивність, простоту інтеграції та високу надійність програмного забезпечення.

Аналіз п'яти аналогічних систем (Garage Organizer, Cooperative Garage Manager, Garage Management Pro, MyGarage Software, GarageSoft) дозволив визначити сильні та слабкі сторони існуючих рішень та окреслити ключові вимоги до розроблюваного програмного забезпечення.

Створена програма автоматизує облік власників, гаражів, матеріалів,

платежів та цін внесків, забезпечує зручний інтерфейс користувача, високу точність обліку та контроль фінансових операцій. Програма також гарантує безпеку та конфіденційність даних.

Впровадження розробленої програми в діяльність гаражного кооперативу сприятиме підвищенню ефективності управління, зниженню кількості помилок, оптимізації витрат часу та ресурсів, а також покращенню рівня обслуговування членів кооперативу.

Таким чином, виконана робота довела свою актуальність і значущість. Розроблена прикладна програма відповідає сучасним вимогам та сприяє оптимізації управлінських процесів у гаражному кооперативі, що робить її корисним та ефективним інструментом для користувачів.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Концептуальне проектування

В процесі аналізу предметної області я виділив наступні бізнес правила для роботи гаражного кооперативу:

- Кожен гараж має бути прив'язаний до власника.
- Один власник може володіти кількома гаражами, але кожен гараж має лише одного власника.
- Кожен гараж побудований з певного матеріалу.
- Один матеріал може бути використаний для побудови кількох гаражів.
- Кожен платіж прив'язаний до конкретного гаража.
- Один гараж може мати багато платежів, але кожен платіж пов'язаний лише з одним гаражем.
- Платіж за гараж в певному місяці та році має бути унікальною, щоб уникнути дублювання платежів за один і той самий період.
- При створенні нового гаража потрібно призначити існуючого власника.
- При створенні нового гаража потрібно призначити існуючий матеріал.
- Кожен гараж має мати записи про орендні платежі, де зазначено місяць і рік оплати, суму платежу та дату платежу.

На основі цих правил я розробив концептуальну схему бази даних. Концептуальна схема бази даних дозволяє організувати та зв'язати інформацію про гаражі, власники, матеріали, платежі забезпечуючи цілісність та унікальність даних у кожній сутності. Було виділено наступні сутності та їх атрибути:

1. Сутність **Матеріали.** Атрибути**:** Назва матеріалу.
2. Сутність **Власники.** Атрибути: Ім'я, Адреса, Телефонний номер, Зображення власника.
3. Сутність **Гаражі.** Атрибути: Номер гаража, Висота гаража, Ширина гаража. Довжина гаража. Матеріал стін, з якого побудований гараж, Кількість поверхів гаража. Зображення гаража. Характеристики гаража.

4. Сутність **Ціна внеску**. Атрибути: Величина ціни, Дата, з якої ціна починає діяти.

5. Сутність **Платежі**: Атрибути: Місяць платежу, Рік платежу, Сума платежу. Дата здійснення платежу.

Використовуючи сутності я створив концептуальну схему база даних гаражного кооперативу(Рисунок 2.1.1).

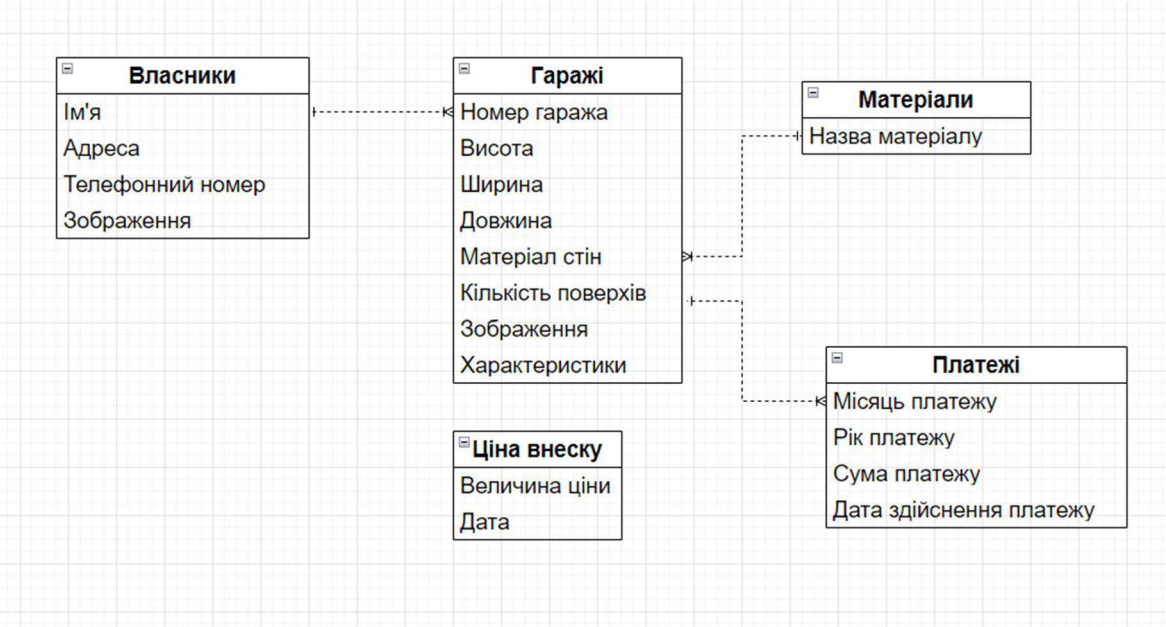


Рисунок 2.1.1 – Концептуальна схема бази даних

Сутність **Гаражі** має зв'язок "багато до одного" з сутністю **Власники**: Один власник може мати багато гаражів, але кожен гараж належить лише одному власнику.

Сутність **Гаражі** має зв'язок "багато до одного" з сутністю **Матеріали**: Один матеріал може використовуватися для багатьох гаражів, але кожен гараж побудований з одного матеріалу.

Сутність **Платежі** має зв'язок "багато до одного" з сутністю **Гаражі**: Один гараж може мати багато платежів, але кожен платіж пов'язаний лише з одним гаражем.

2.2. Логічне проектування

Логічне проектування — це етап розробки бази даних, на якому визначається структура даних, без урахування фізичних аспектів їх зберігання та доступу. На цьому етапі створюється логічна модель даних, яка включає в себе таблиці, поля (атрибути), ключі (первинні та зовнішні), обмеження та зв'язки між таблицями. При перетворенні концептуальної моделі в логічну модель було створено такі відношення:

1. Відношення Materials. Атрибути:

– id: використовується як первинний ключ. Кожне значення id є унікальним для кожного запису в таблиці.

– name: поле для зберігання назв матеріалів. Він має обмеження UNIQUE, що означає, що значення в цьому стовпці повинні бути унікальними, тобто не можуть повторюватися.

2. Відношення Owners. Атрибути:

– id: первинний ключ.

– name: унікальне поле для кожного запису. Зберігає імена власників.

– address: поле для зберігання адрес власників.

– phone: поле для зберігання телефонних номерів власників.

– image: Поле для зберігання зображень.

3. Відношення Garages. Атрибути:

– id: первинний ключ.

– number: поле, унікальне і обов'язкове для кожного запису. Зберігає номери гаражів.

– owner_id: зовнішній ключ, який посилається на стовпець id таблиці Owners.

– height, width, length: Поля для зберігання розмірів гаражів.

– material_id: зовнішній ключ, який посилається на стовпець id таблиці Materials.

– storeys: поле, яке зберігає кількість поверхів гаража.

– image: поле для зберігання зображення гаража.

– characteristics: поле для зберігання характеристик гаража.

4. Відношення Price. Атрибути:

– id: первинний ключ з автоматичним збільшенням. Унікально ідентифікує кожен запис.

– price: Поле для зберігання цін.

– from_date: Поле типу DATE, яке зберігає дату з значенням за замовчуванням, що встановлюється на поточну дату і час у локальному часовому поясі. Обов'язкове поле.

5. Відношення Payment. Атрибути:

– id: первинний ключ.

– garage_id: зовнішній ключ, який посилається на стовпець id таблиці Garages.

– month: поле, що зберігає місяць оплати.

– year: поле, що зберігає рік оплати.

– paid: поле, що зберігає суму оплати.

– payment_date: поле, що зберігає дату оплати.\

На основі відношень було створено показана логічну схему бази даних (Рисунок 2.2.1).

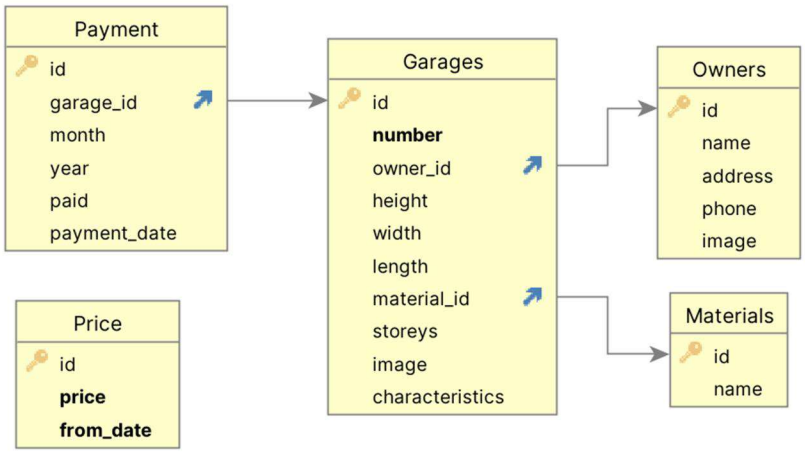


Рисунок 2.2.1 – Логічна схема бази даних

Зв'язки між таблицями:

– Таблиця Garages має зв'язок багато до одного (many-to-one) з таблицею Owners через зовнішній ключ owner_id.

– Таблиця Garages має зв'язок багато до одного (many-to-one) з таблицею Materials через зовнішній ключ material_id.

– Таблиця Payment має зв'язок багато до одного (many-to-one) з таблицею Garages через зовнішній ключ garage_id.

Для таблиці Payment задано обмеження UNIQUE (garage_id, month, year): Ця комбінація повинна бути унікальною, що означає, що для кожного гаража не може бути двох записів з однаковим місяцем і роком.

2.3. Нормалізація баз даних

Щоб перевірити схему бази даних за допомогою правил нормалізації, потрібно пройти через три основні форми нормалізації (1NF, 2NF, 3NF) і переконатися, що кожна таблиця відповідає вимогам цих форм.

Перша нормальна форма (1NF)

Перша нормальна форма вимагає, щоб всі атрибути містили тільки атомарні (нероздільні) значення, а кожне поле містило одне значення.

У всіх відношеннях Materials, Owners, Garages, Price, Payment всі поля атомарні, що відповідає першій нормальній формі (1NF).

Друга нормальна форма (2NF)

Друга нормальна форма вимагає, щоб всі атрибути, які не є частиною ключа, повністю залежали від первинного ключа, а не від частини ключа (якщо є композитний ключ).

Відношення Materials: Первинний ключ: id. Всі неключові атрибути (name) залежать від id. Відповідає 2NF.

Відношення Owners: Первинний ключ: id. Всі неключові атрибути (name, address, phone, image) залежать від id. Відповідає 2NF.

Відношення Garages: Первинний ключ: id. Всі неключові атрибути (number, owner_id, height, width, length, material_id, storeys, image, characteristics) залежать від id. Відповідає 2NF.

Відношення Price: Первинний ключ: id. Всі неключові атрибути (price, from_date) залежать від id. Відповідає 2NF.

Відношення Payment: Первинний ключ: id. Всі неключові атрибути (garage_id, month, year, paid, payment_date) залежать від id. Відповідає 2NF.

Третя нормальна форма (3NF)

Третя нормальна форма вимагає, щоб всі неключові атрибути не мали транзитивних залежностей від первинного ключа.

Відношеннях Materials, Owners, Garages, Price, Payment відсутні транзитивні залежності тому вони відповідають 3NF.

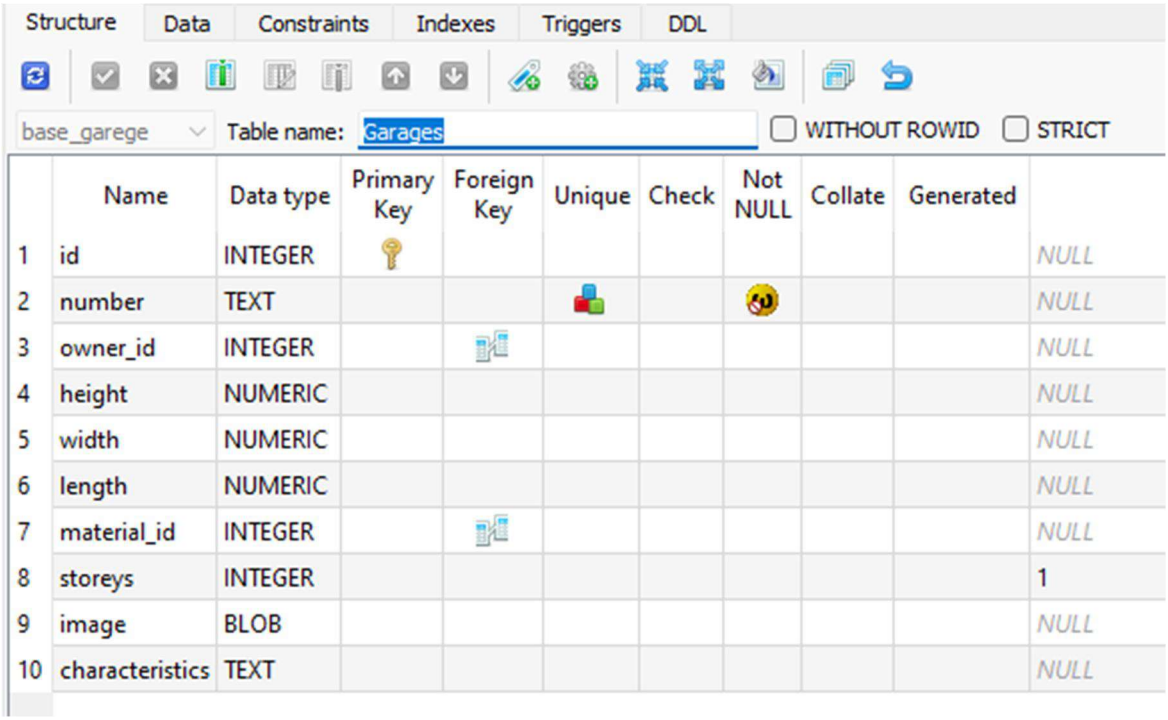
Усі відношення (Materials, Owners, Garages, Price, Payment) відповідають вимогам першої, другої і третьої нормальних форм. Схема бази даних добре нормалізована і відповідає стандартним вимогам нормалізації, що забезпечує цілісність даних та мінімізує надлишковість.

2.4. Фізичне проектування

Для фізичної реалізації було обрано СУБД SQLite. Це вбудована, безсерверна, легка використання, локальна СУБД, яка зберігає бази даних у файловій системі. Вона пропонує велику частину стандартного SQL, забезпечуючи доступ до даних через стандартний мовний інтерфейс.

Відношення були перетворені у таблиці, атрибути стали полями.

Структура таблиці Garages (Рисунок 2.4.1) дозволяє зберігати інформацію про гаражі, включаючи їхні унікальні номери, ідентифікатори власників та матеріалів, розміри, кількість поверхів, зображення та додаткові характеристики. Обмеження забезпечують цілісність даних та правильність зв'язків між таблицями.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	number	TEXT								NULL
3	owner_id	INTEGER								NULL
4	height	NUMERIC								NULL
5	width	NUMERIC								NULL
6	length	NUMERIC								NULL
7	material_id	INTEGER								NULL
8	storeys	INTEGER								1
9	image	BLOB								NULL
10	characteristics	TEXT								NULL

Рисунок 2.4.1 – Структура таблиці Garages

Обмеження, які накладаються на таблицю Garages:

– PRIMARY KEY: Стовпець id є первинним ключем, що забезпечує унікальну ідентифікацію кожного запису в таблиці.

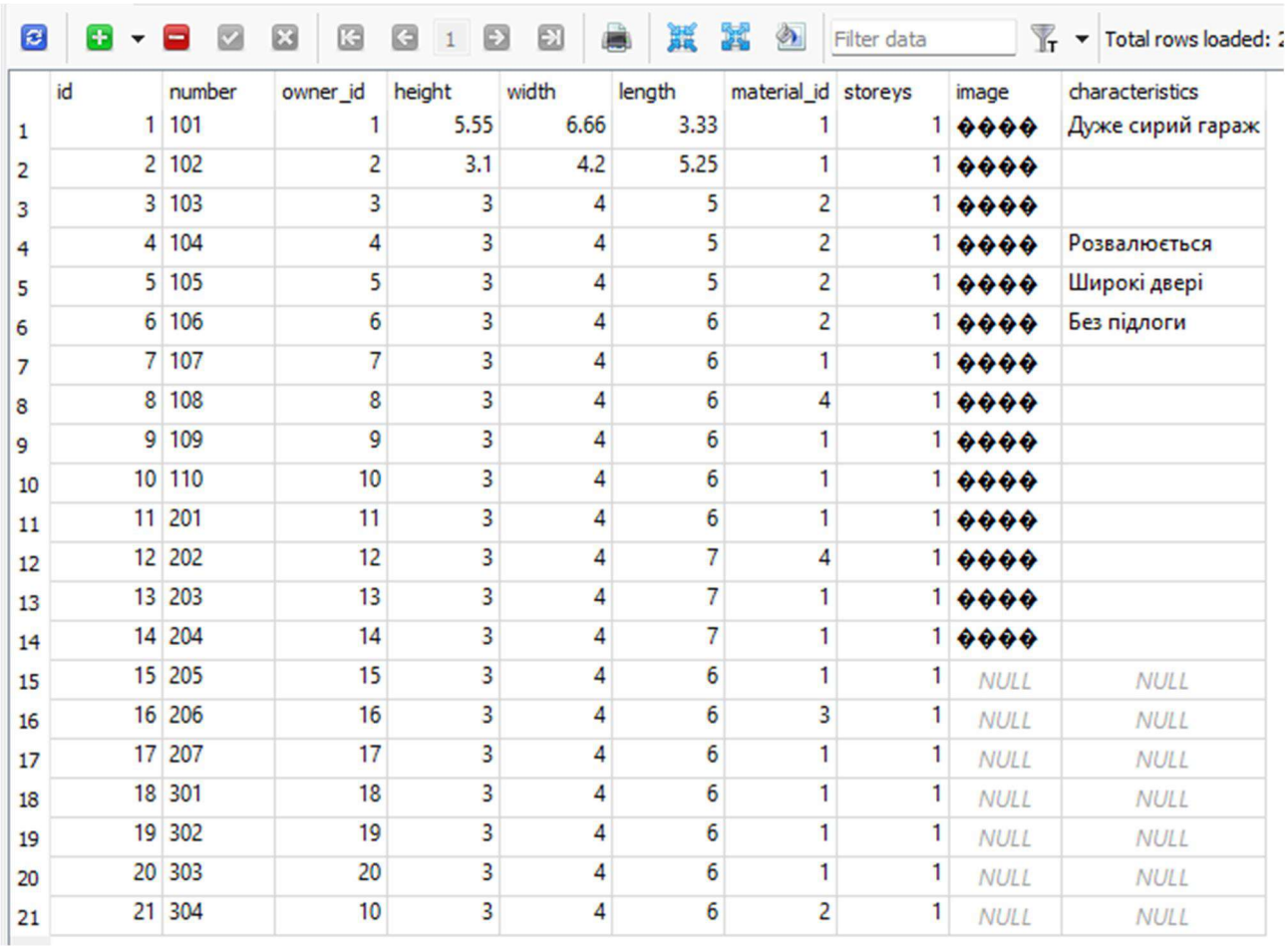
– UNIQUE: Значення в стовпці number повинні бути унікальними.

– NOT NULL: Стовпець number не може бути порожнім.

– FOREIGN KEY (owner_id): Посилання на стовпець id у таблиці Owners із каскадним видаленням та оновленням.

– FOREIGN KEY (material_id): Посилання на стовпець id у таблиці Materials.

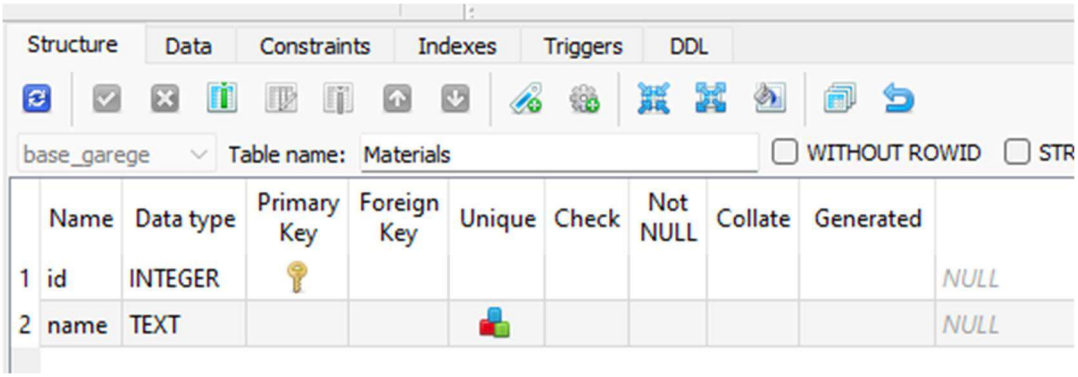
Таблиця містить інформацію, яка відображена на рисунку 2.4.2.



	id	number	owner_id	height	width	length	material_id	storeys	image	characteristics
1	1	101	1	5.55	6.66	3.33	1	1	????	Дуже сирий гараж
2	2	102	2	3.1	4.2	5.25	1	1	????	
3	3	103	3	3	4	5	2	1	????	
4	4	104	4	3	4	5	2	1	????	Розвалюється
5	5	105	5	3	4	5	2	1	????	Широкі двері
6	6	106	6	3	4	6	2	1	????	Без підлоги
7	7	107	7	3	4	6	1	1	????	
8	8	108	8	3	4	6	4	1	????	
9	9	109	9	3	4	6	1	1	????	
10	10	110	10	3	4	6	1	1	????	
11	11	201	11	3	4	6	1	1	????	
12	12	202	12	3	4	7	4	1	????	
13	13	203	13	3	4	7	1	1	????	
14	14	204	14	3	4	7	1	1	????	
15	15	205	15	3	4	6	1	1	NULL	NULL
16	16	206	16	3	4	6	3	1	NULL	NULL
17	17	207	17	3	4	6	1	1	NULL	NULL
18	18	301	18	3	4	6	1	1	NULL	NULL
19	19	302	19	3	4	6	1	1	NULL	NULL
20	20	303	20	3	4	6	1	1	NULL	NULL
21	21	304	10	3	4	6	2	1	NULL	NULL

Рисунок 2.4.2 – Дані таблиці Garages

Структура таблиці Materials забезпечує унікальну ідентифікацію кожного запису за допомогою стовпця id та гарантує, що назви матеріалів стін гаража (name) не будуть повторюватися. Структура таблиці Materials відображено на рисунку 2.4.3.



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER								NULL
2	name	TEXT								NULL

Рисунок 2.4.3 – Структура таблиці Materials

Обмеження, які накладаються на таблицю Materials:

– PRIMARY KEY: Стовпець id є первинним ключем.

– AUTOINCREMENT: Значення для стовпця id автоматично збільшується на 1 для кожного нового запису.

– UNIQUE: Значення в стовпці name повинні бути унікальними для кожного запису в таблиці.

Таблиця містить інформацію, яка відображена на рисунку 2.3.4.

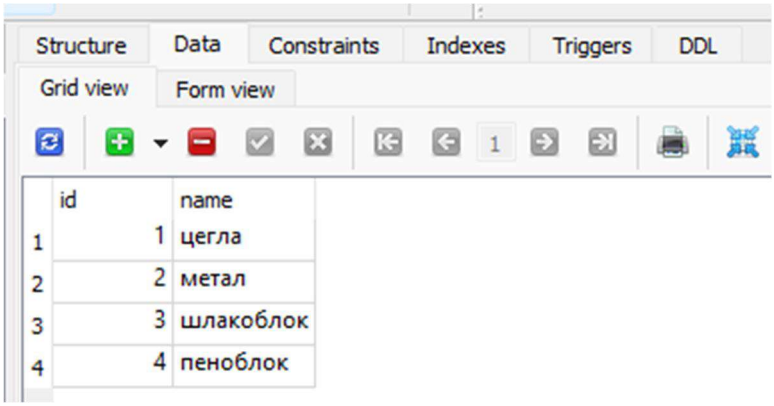


Рисунок 2.4.4 – Дані таблиці Materials

Структура таблиці Owners дозволяє зберігати інформацію про власників, включаючи їхні унікальні імена, адреси, телефонні номери та їх фото.

Структура таблиці Owners відображено на рисунку 2.4.5.

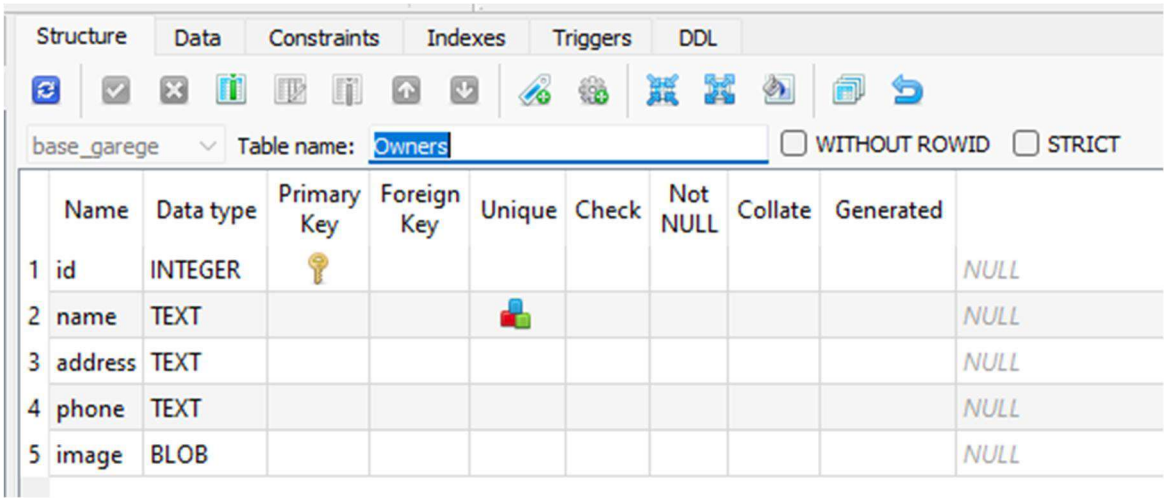
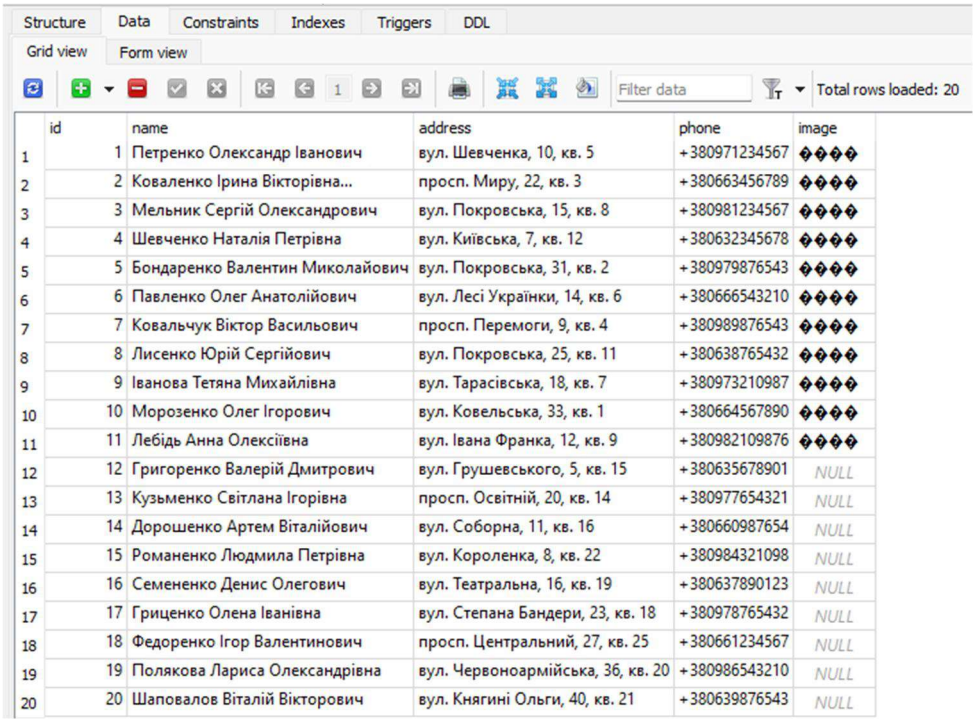


Рисунок 2.4.5 – Структура таблиці Owners

Обмеження, які накладаються на таблицю Owners :

– PRIMARY KEY: Стовпець id є первинним ключем, що забезпечує унікальну ідентифікацію кожного запису в таблиці.
– UNIQUE: Значення в стовпці name повинні бути унікальними, тобто не можуть повторюватися.

Таблиця містить інформацію, яка відображена на рисунку 2.4.6.



Structure | Data | Constraints | Indexes | Triggers | DDL

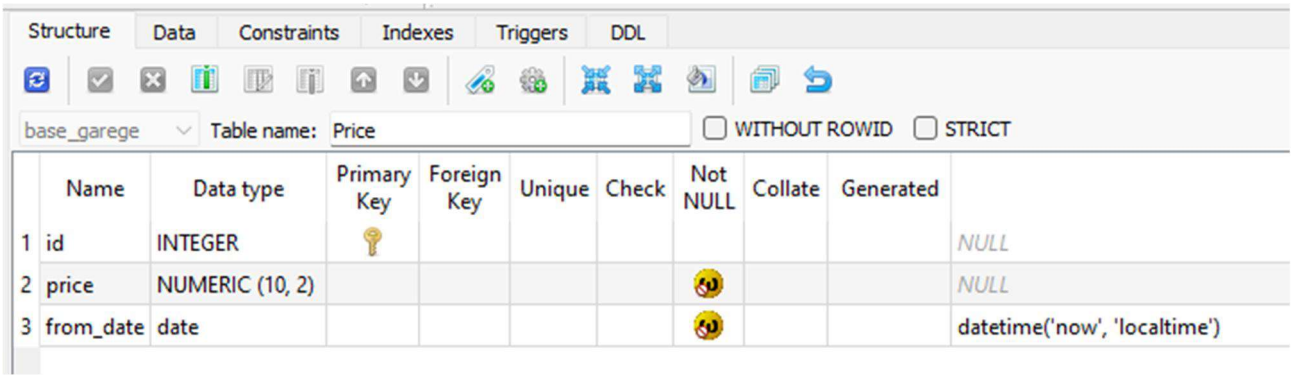
Grid view | Form view

Filter data | Total rows loaded: 20

	id	name	address	phone	image
1	1	Петренко Олександр Іванович	вул. Шевченка, 10, кв. 5	+380971234567	◆◆◆◆
2	2	Коваленко Ірина Вікторівна...	просп. Миру, 22, кв. 3	+380663456789	◆◆◆◆
3	3	Мельник Сергій Олександрович	вул. Покровська, 15, кв. 8	+380981234567	◆◆◆◆
4	4	Шевченко Наталія Петрівна	вул. Київська, 7, кв. 12	+380632345678	◆◆◆◆
5	5	Бондаренко Валентин Миколайович	вул. Покровська, 31, кв. 2	+380979876543	◆◆◆◆
6	6	Павленко Олег Анатолійович	вул. Лесі Українки, 14, кв. 6	+380666543210	◆◆◆◆
7	7	Ковальчук Віктор Васильович	просп. Перемоги, 9, кв. 4	+380989876543	◆◆◆◆
8	8	Лисенко Юрій Сергійович	вул. Покровська, 25, кв. 11	+380638765432	◆◆◆◆
9	9	Іванова Тетяна Михайлівна	вул. Тарасівська, 18, кв. 7	+380973210987	◆◆◆◆
10	10	Морозенко Олег Ігорович	вул. Ковельська, 33, кв. 1	+380664567890	◆◆◆◆
11	11	Лебідь Анна Олексіївна	вул. Івана Франка, 12, кв. 9	+380982109876	◆◆◆◆
12	12	Григоренко Валерій Дмитрович	вул. Грушевського, 5, кв. 15	+380635678901	NULL
13	13	Кузьменко Світлана Ігорівна	просп. Освітній, 20, кв. 14	+380977654321	NULL
14	14	Дорошенко Артем Віталійович	вул. Соборна, 11, кв. 16	+380660987654	NULL
15	15	Романенко Людмила Петрівна	вул. Короленка, 8, кв. 22	+380984321098	NULL
16	16	Семененко Денис Олегович	вул. Театральна, 16, кв. 19	+380637890123	NULL
17	17	Гриценко Олена Іванівна	вул. Степана Бандери, 23, кв. 18	+380978765432	NULL
18	18	Федоренко Ігор Валентинович	просп. Центральний, 27, кв. 25	+380661234567	NULL
19	19	Полякова Лариса Олександрівна	вул. Червоноармійська, 36, кв. 20	+380986543210	NULL
20	20	Шаповалов Віталій Вікторович	вул. Княгині Ольги, 40, кв. 21	+380639876543	NULL

Рисунок 2.4.6 – Дані таблиці Owners

Таблиця Price таблиці дозволяє зберігати інформацію про ціни разом із датою їх введення в дію. Обмеження забезпечують цілісність даних та правильність зберігання цінових значень. Структура таблиці Price відображено на рисунку 2.4.7.



Structure | Data | Constraints | Indexes | Triggers | DDL

base_garege | Table name: Price | WITHOUT ROWID | STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	id	INTEGER	🔑							NULL
2	price	NUMERIC (10, 2)					🔒			NULL
3	from_date	date					🔒			datetime('now', 'localtime')

Рисунок 2.4.7 – Структура таблиці Price

Обмеження, які накладаються на таблицю Price:

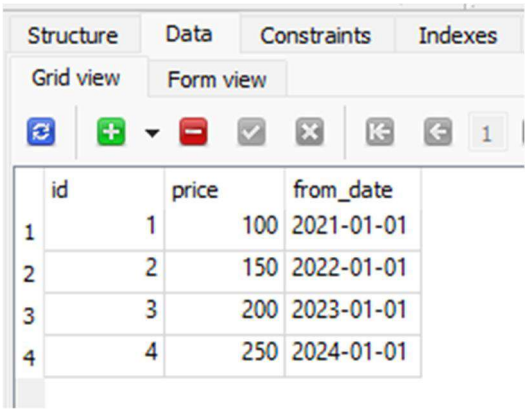
PRIMARY KEY: Стовпець id є первинним ключем, що забезпечує унікальну ідентифікацію кожного запису в таблиці.

AUTOINCREMENT: Значення для стовпця id автоматично збільшується на 1 для кожного нового запису.

NOT NULL: Стовпці price і from_date не можуть бути порожніми.

DEFAULT: Стовпець from_date має значення за замовчуванням, яке встановлюється на поточну дату і час в локальному часовому поясі.

Таблиця містить інформацію, яка відображена на рисунку 2.4.8.



	id	price	from_date
1	1	100	2021-01-01
2	2	150	2022-01-01
3	3	200	2023-01-01
4	4	250	2024-01-01

Рисунок 2.4.8 – Дані таблиці Price

Таблиця Payment дозволяє зберігати інформацію про платежі за гаражі, включаючи ідентифікатор гаража, місяць, рік, суму оплати та дату оплати. Обмеження забезпечують цілісність даних та унікальність записів для кожного гаража за певний місяць і рік. Структура таблиці Payment відображено на рисунку 2.4.9.

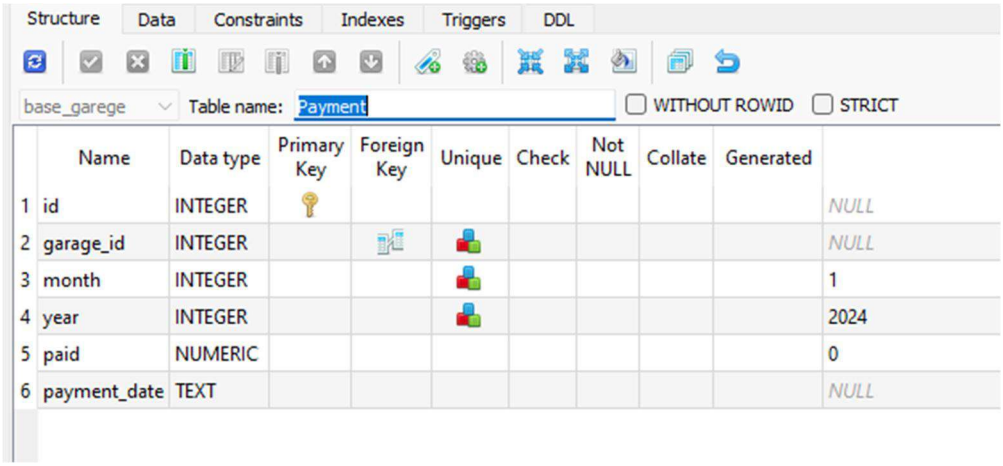


Рисунок 2.4.9 – Структура таблиці Payment

Обмеження, які накладаються на таблицю Payment:

- PRIMARY KEY: Стовпець id є первинним ключем, що забезпечує унікальну ідентифікацію кожного запису в таблиці.
- FOREIGN KEY (garage_id): Посилання на стовпець id у таблиці Garages з обмеженнями ON DELETE NO ACTION та ON UPDATE NO ACTION, що означає відсутність автоматичних дій при видаленні або оновленні пов'язаних записів у таблиці Garages.
- DEFAULT: Значення за замовчуванням для стовпців month, year, і paid встановлюються відповідно в 1, 2024, і 0.
- UNIQUE (garage_id, month, year): Ця комбінація стовпців повинна бути унікальною, тобто для кожного гаража не може бути двох записів з однаковим місяцем і роком.

Таблиця містить інформацію, яка відображена на рисунку 2.4.10.

Structure | Data | Constraints | Indexes | Triggers | DDL

Grid view | Form view

Filter data

	id	garage_id	month	year	paid	payment_da
1	1	1	3	2024	250	2024-03-21
2	2	1	2	2024	250	2024-02-15
3	3	2	2	2024	250	2024-05-05
4	4	2	3	2024	250	2024-05-01
5	5	2	1	2024	250	2024-05-15
6	6	2	4	2024	250	2024-05-01
7	7	1	1	2024	250	2024-01-03
8	8	3	1	2024	250	2024-01-25
9	9	4	1	2024	250	NULL
10	10	5	1	2024	250	NULL
11	11	6	1	2024	250	NULL
12	12	7	1	2024	250	NULL
13	13	8	1	2024	250	NULL
14	14	9	1	2024	250	NULL
15	15	10	1	2024	250	NULL
16	16	11	1	2024	250	NULL
17	17	12	1	2024	250	NULL
18	18	13	1	2024	250	NULL
19	19	14	1	2024	250	NULL
20	20	15	1	2024	250	NULL
21	21	16	1	2024	250	NULL
22	22	17	1	2024	250	NULL
23	23	18	1	2024	250	NULL
24	24	19	1	2024	250	NULL
25	25	20	1	2024	250	NULL
26	26	21	1	2024	250	NULL
27	27	3	2	2024	250	2024-02-16
28	28	4	2	2024	250	NULL
29	29	5	2	2024	250	NULL

Рисунок 2.4.10 – Дані таблиці Payment

Результуюча схема бази даних зображена на 2.4.11.

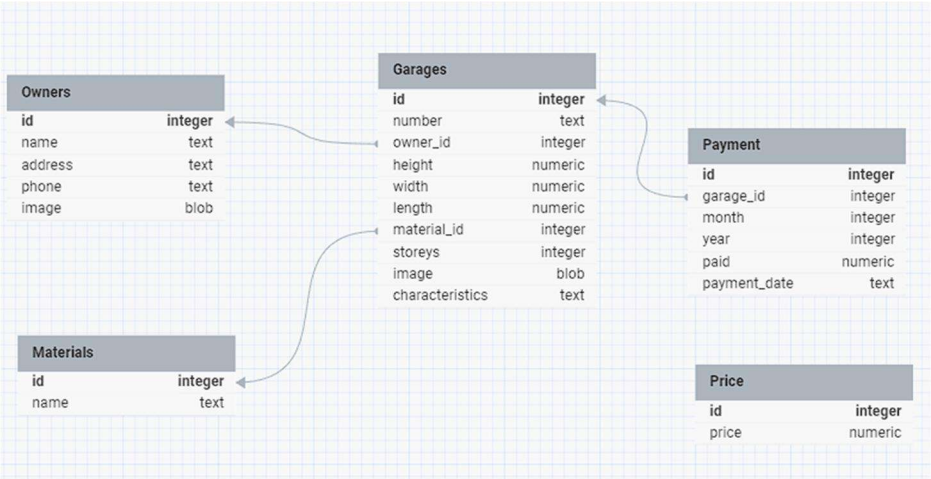


Рисунок 2.4.11 – Схема бази даних гаражного кооперативу

2.5. Структура проєкту

Структура проекту відображена на рисунку 2.5.1. Проект містить сім форм для роботи з даними гаражного кооперативу.

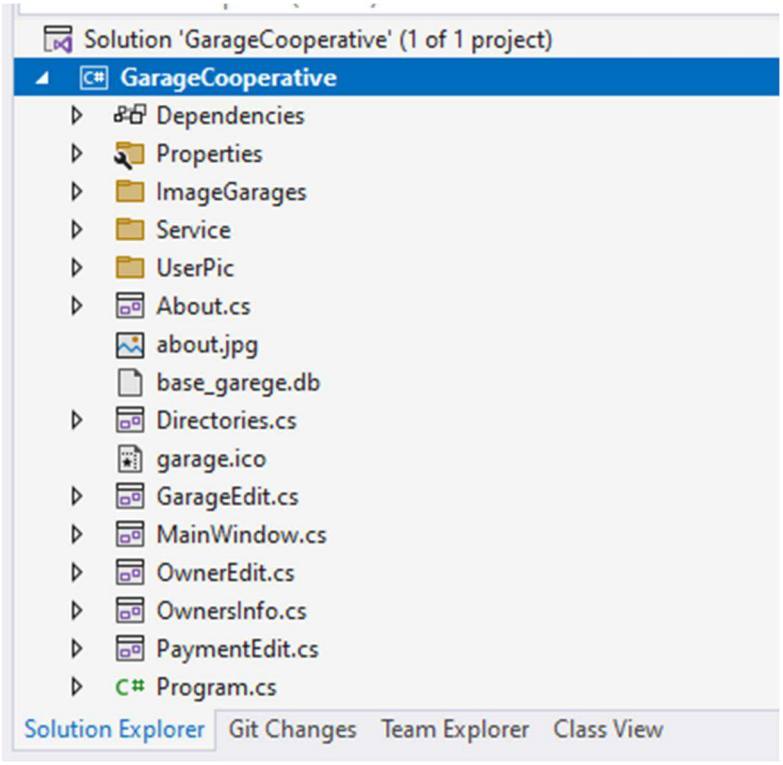


Рисунок 2.5.1 – Структура проекту GarageCooperative

Кожна форма побудована на ООП з використанням класу. Були розроблені даткові класи для роботи із зображенням та базою даних. Діаграма класі відображена на рисунку 2.5.2.

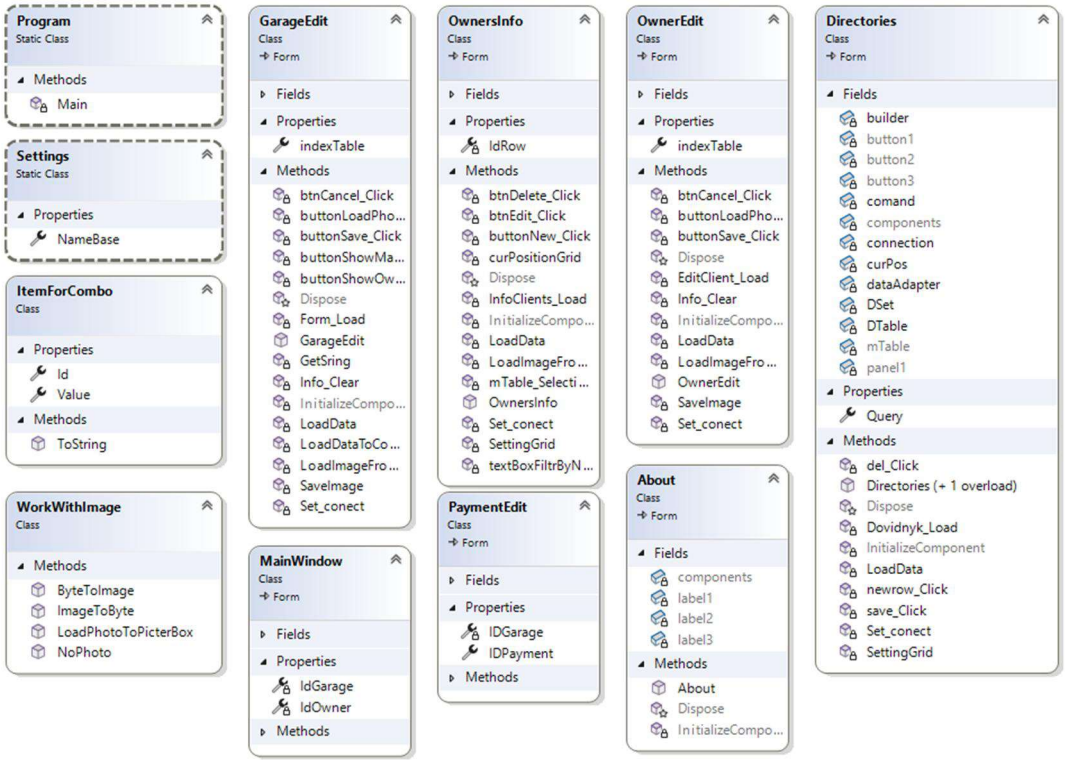


Рисунок 2.5.2 – Діаграма класів проекту GarageCooperative

2.6. Тестування програмного продукту

Над програмною розробкою для роботи гаражного кооперативу було проведено ряд тестів:

– **Додавання власника**

Введення всіх необхідних даних (ім'я, адреса, телефонний номер, зображення). Перевірка, чи новий власник з'являється в списку власників після збереження. Перевірка, чи всі дані зберігаються правильно в базі даних.

– **Редагування власника**

Вибір існуючого власника зі списку. Зміна одного або кількох атрибутів (наприклад, адреси або телефонного номера). Перевірка, чи зміни зберігаються після натискання кнопки збереження. Перевірка оновлених даних у базі даних.

– **Видалення власника**

Вибір власника зі списку. Видалення обраного власника. Перевірка, чи власник видаляється зі списку та з бази даних.

– **Додавання гаража:**

Введення всіх необхідних даних (номер гаража, висота, ширина, довжина, матеріал стін, кількість поверхів, зображення, характеристики). Перевірка, чи новий гараж з'являється в списку гаражів після збереження. Перевірка, чи всі дані зберігаються правильно в базі даних.

– **Редагування гаража**

Вибір існуючого гаража зі списку. Зміна одного або кількох атрибутів (наприклад, висоти або матеріалу стін). Перевірка, чи зміни зберігаються після натискання кнопки збереження. Перевірка оновлених даних у базі даних.

– **Видалення гаража**

Вибір гаража зі списку. Видалення обраного гаража. Перевірка, чи гараж видаляється зі списку та з бази даних.

– **Додавання платежу**

Введення всіх необхідних даних (місяць, рік, сума платежу, дата здійснення). Перевірка, чи новий платіж з'являється в списку платежів після збереження. Перевірка, чи всі дані зберігаються правильно в базі даних.

– **Редагування платежу**

Вибір існуючого платежу зі списку. Зміна одного або кількох атрибутів (наприклад, суми або дати платежу). Перевірка, чи зміни зберігаються після натискання кнопки збереження. Перевірка оновлених даних у базі даних.

– **Видалення платежу**

Вибір платежу зі списку. Видалення обраного платежу. Перевірка, чи платіж видаляється зі списку та з бази даних.

– **Додавання ціни внеску**

Введення всіх необхідних даних (величина ціни, дата початку дії). Перевірка, чи нова ціна з'являється в списку цін після збереження. Перевірка, чи всі дані зберігаються правильно в базі даних.

– **Редагування ціни внеску**

Вибір існуючої ціни зі списку. Зміна одного або кількох атрибутів (наприклад, величини або дати початку дії). Перевірка, чи зміни зберігаються після натискання кнопки збереження. Перевірка оновлених даних у базі даних.

– **Видалення ціни внеску**

Вибір ціни зі списку. Видалення обраної ціни. Перевірка, чи ціна видаляється зі списку та з бази даних.

– **Додавання матеріалу**

Введення назви матеріалу. Перевірка, чи новий матеріал з'являється в списку матеріалів після збереження. Перевірка, чи всі дані зберігаються правильно в базі даних.

– **Редагування матеріалу**

Вибір існуючого матеріалу зі списку. Зміна назви матеріалу. Перевірка, чи зміни зберігаються після натискання кнопки збереження. Перевірка оновлених даних у базі даних.

– **Видалення матеріалу**

Вибір матеріалу зі списку. Видалення обраного матеріалу. Перевірка, чи матеріал видаляється зі списку та з бази даних.

– **Перевірка зв'язків між таблицями:**

Перевірка, чи правильно відображаються дані при виборі власника гаража. Перевірка, чи коректно відображаються платежі, пов'язані з конкретним гаражем або власником.

– **Пошук та фільтрація:**

Тестування функцій пошуку та фільтрації даних за різними критеріями. Перевірка коректності результатів пошуку та фільтрації.

– **Обробка помилок та валідація даних**

Перевірка обробки помилок при введенні некоректних даних. Валідація обов'язкових полів та обмеження на введення даних.

Ці тести допомогли переконатися в якості та надійності розробленої системи для гаражного кооперативу, а також впевнитися, що вона відповідає всім вимогам та очікуванням користувачів.

Висновки до розділу 2

Висновки до дипломної роботи про розробку програми для гаражного кооперативу мають на меті узагальнити отримані результати дослідження та визначити їх важливість і практичну цінність. У результаті проведеного аналізу була виявлена актуальність розробки програмного забезпечення для гаражного кооперативу, оскільки існуючі методи управління можуть бути неефективними та часомірними.

Основною метою дослідження було створення функціональної та зручної в управлінні програми для гаражного кооперативу, що дозволить оптимізувати процеси управління та підвищити ефективність роботи кооперативу.

Для досягнення поставленої мети були сформульовані завдання, які включали в себе аналіз вимог користувачів, розробку архітектури програми, імплементацію функціоналу та тестування програмного забезпечення.

Програма для гаражного кооперативу була успішно розроблена та пройшла процес тестування, що підтвердило її працездатність та ефективність.

Застосування розробленої програми може значно полегшити управління гаражним кооперативом, зменшити витрати часу та зусиль на адміністративні процедури та забезпечити більшу точність обліку даних.

У майбутньому можна розглядати можливість розширення функціоналу програми, враховуючи нові потреби та вимоги користувачів, а також інтеграцію з іншими інформаційними системами для підвищення її функціональності.

В цілому, розробка програмного забезпечення для гаражного кооперативу є актуальною та перспективною задачею, яка може значно сприяти оптимізації управління такими об'єднаннями та полегшити їхню діяльність.

РОЗДІЛ 3. КЕРІВНИЦТВО ВИКОРИСТАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1. Мінімальні вимоги до системи

Для забезпечення коректної та ефективної роботи прикладної програми для гаражного кооперативу необхідно врахувати мінімальні вимоги до системи, на якій вона буде працювати. Ці вимоги включають як апаратні, так і програмні компоненти.

Апаратні вимоги

- **Процесор**: Двоядерний процесор з тактовою частотою не менше 1.6 ГГц.
- **Оперативна пам'ять**: Мінімум 4 ГБ оперативної пам'яті.
- **Жорсткий диск**: Мінімум 500 МБ вільного місця на жорсткому диску для встановлення програми та зберігання бази даних.
- **Монітор**: Роздільна здатність екрану не менше 1024x768 пікселів.
- **Відеокарта**: Інтегрована або дискретна відеокарта з підтримкою DirectX 9.0 або вище.
- **Інші пристрої**: Миша та клавіатура для зручного введення даних.

Програмні вимоги

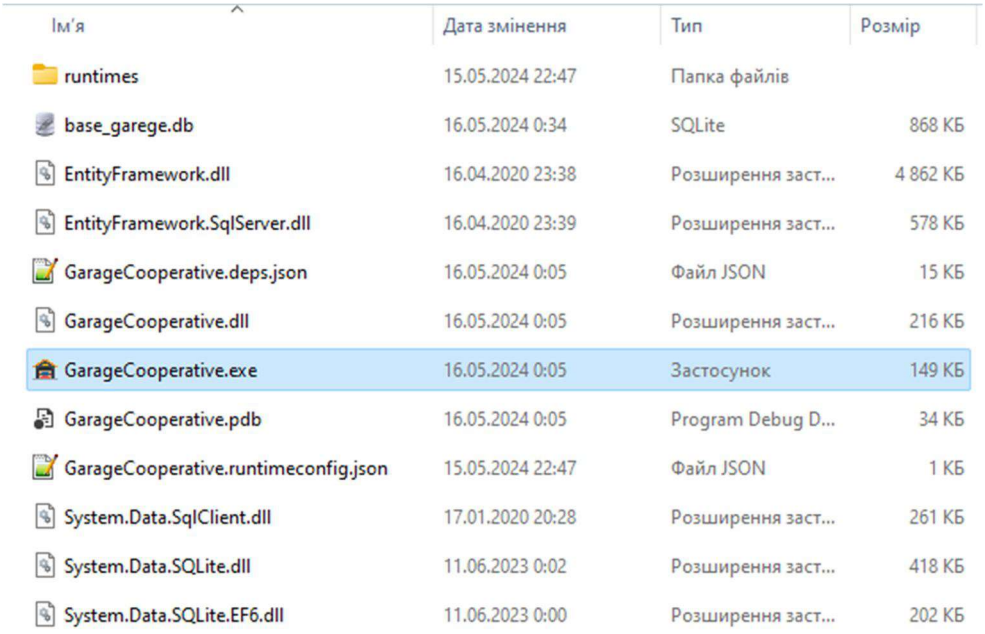
- **Операційна система**: Windows 7 SP1 або вище (рекомендовано Windows 10).
- **.NET 8** або вище.
- **База даних**: SQLite, включена у програму (не потребує додаткового встановлення).
- **Браузер**: Будь-який сучасний браузер для доступу до документації та підтримки (не є критичним для роботи програми).
- **Антивірусне ПЗ**: Рекомендовано використовувати актуальне антивірусне програмне забезпечення для забезпечення безпеки системи.

– **Резервне копіювання**: Наявність механізму резервного копіювання для збереження копій бази даних на випадок апаратних або програмних збоїв.

Забезпечення вказаних мінімальних вимог до системи гарантує стабільну роботу прикладної програми для гаражного кооперативу, дозволяючи користувачам ефективно управляти обліком власників, гаражів, матеріалів та фінансових операцій. Виконання цих вимог також сприятиме захисту даних та забезпечить зручність використання програми.

3.2. Інтерфейс користувача

Програма запускається з папки через файл GarageCooperative.exe. Для роботи програми потрібні додаткові файли. Перелік файлів зображено на рисунку 3.2.1.



Ім'я	Дата змінення	Тип	Розмір
runtimes	15.05.2024 22:47	Папка файлів	
base_garege.db	16.05.2024 0:34	SQLite	868 КБ
EntityFramework.dll	16.04.2020 23:38	Розширення заст...	4 862 КБ
EntityFramework.SqlServer.dll	16.04.2020 23:39	Розширення заст...	578 КБ
GarageCooperative.deps.json	16.05.2024 0:05	Файл JSON	15 КБ
GarageCooperative.dll	16.05.2024 0:05	Розширення заст...	216 КБ
GarageCooperative.exe	16.05.2024 0:05	Застосунок	149 КБ
GarageCooperative.pdb	16.05.2024 0:05	Program Debug D...	34 КБ
GarageCooperative.runtimeconfig.json	15.05.2024 22:47	Файл JSON	1 КБ
System.Data.SqlClient.dll	17.01.2020 20:28	Розширення заст...	261 КБ
System.Data.SQLite.dll	11.06.2023 0:02	Розширення заст...	418 КБ
System.Data.SQLite.EF6.dll	11.06.2023 0:00	Розширення заст...	202 КБ

Рисунок 3.2.1 – Файл запуску програми GarageCooperative

При успішному запуску програми ми бачимо вікно програми «Гаражний кооператив» показане на рисунку 3.2.2.

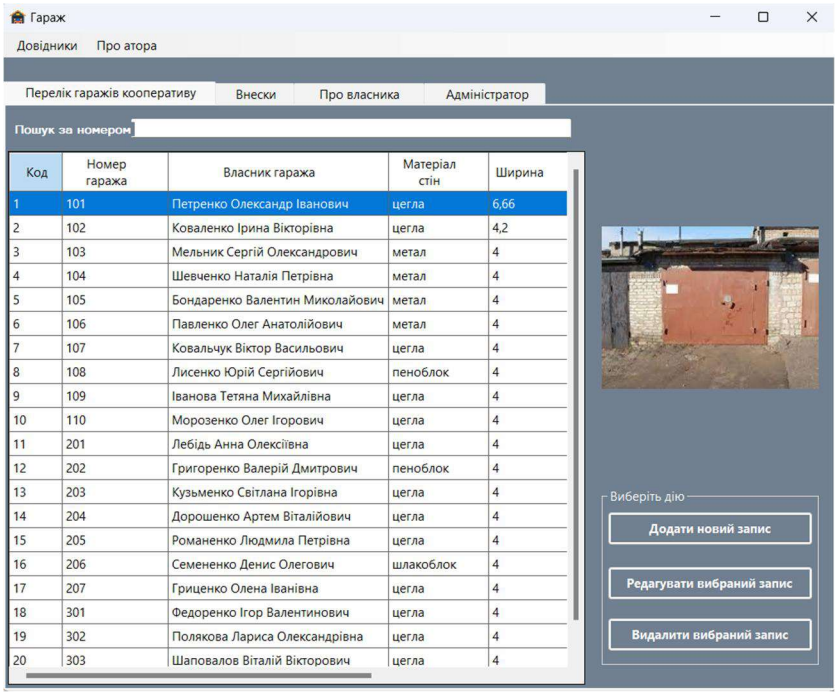


Рисунок 3.2.2 – Інтерфейс програми

На головній формі програми відкрита закладка «Перелік гаражів кооперативу». Тут можна створити або редагувати інформацію про гараж.

Роботу із внесками користувач може здійснити на вкладці «Внески» показаній на рисунку 3.2.3. Можливо видалити оплату, відредагувати оплату, додати нову оплату.

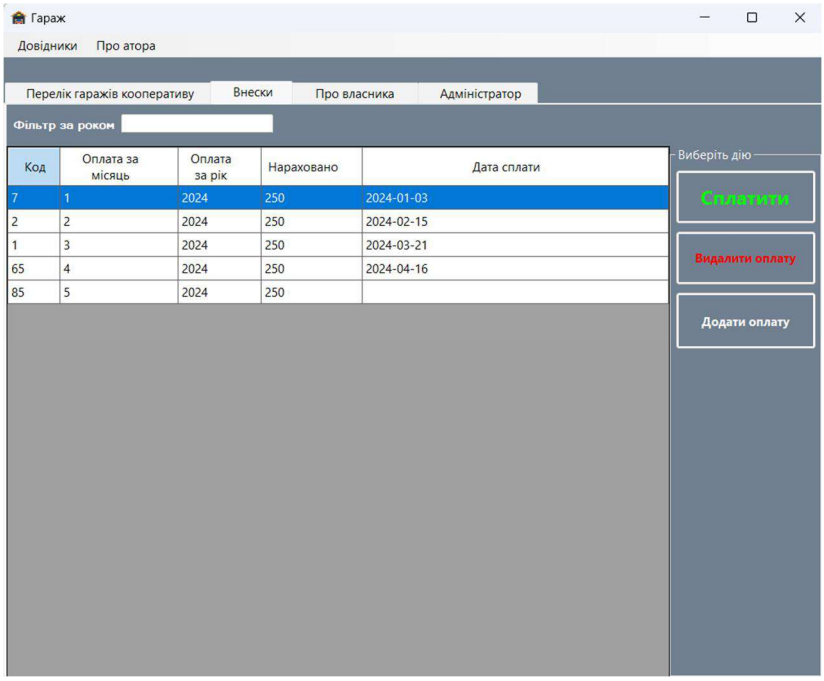


Рисунок 3.2.3 – Вкладка «Внески»

Наступна вкладка програми має назву «Про власника» показана на рисунку 3.2.4. Дана вкладка надає можливість лише переглядати інформацію про власника вибраного гаража.

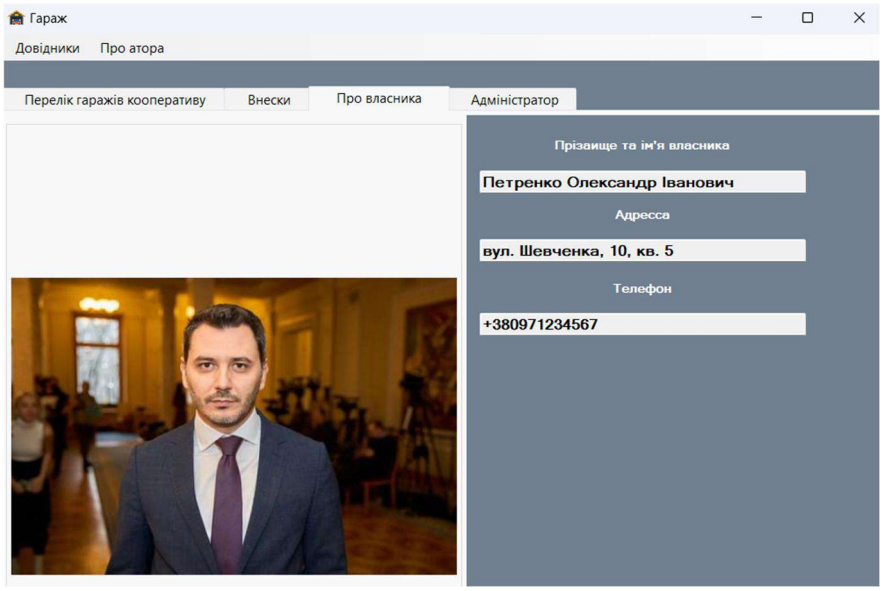


Рисунок 3.2.4 – Вкладка «Про власника»

Наступна вкладка має назву «Адміністратор», тут можна переглянути статистику боржників та створити оплату внесків на новий період. Якщо період вже є такий то він не додається в базу даних. Вигляд форми показано на рисунку 3.2.4.

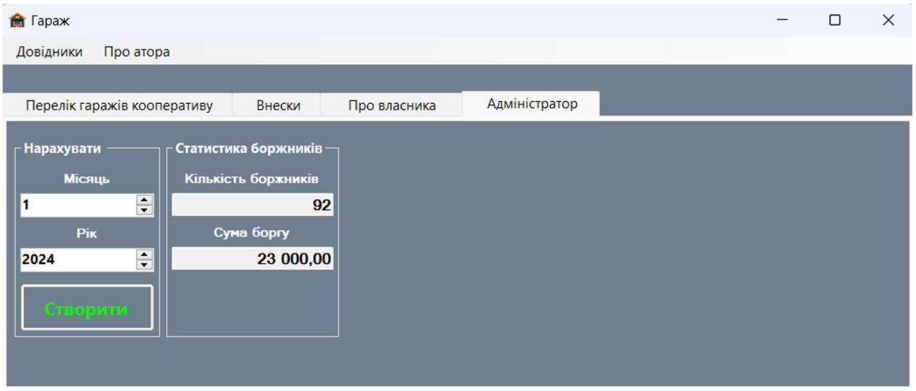


Рисунок 3.2.5 – Вкладка «Адміністратор»

В програмі наявні довідники для роботи «Гаражного кооперативу» що показані на рисунку 3.2.6. Через меню можна відкрити довідник та переглянути інформацію про матеріал стін гаража(), інформацію про власників() та вартість місячного внеску().

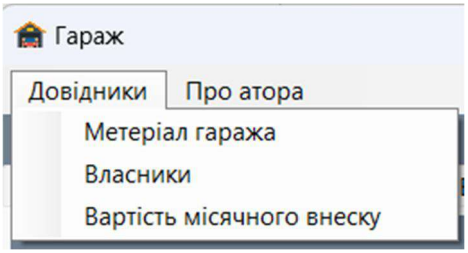


Рисунок 3.2.6 – Меню «Довідники»

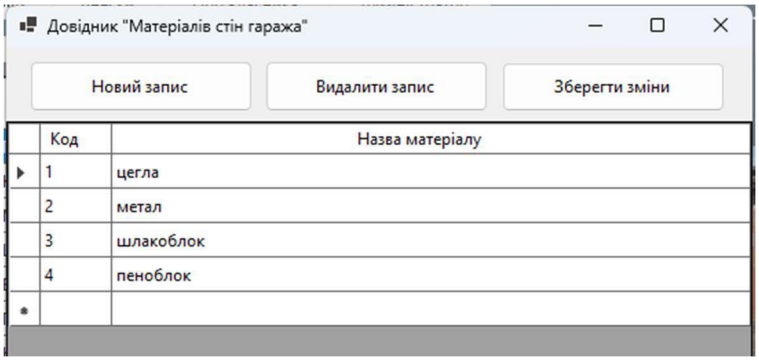


Рисунок 3.2.7 – Довідник «Матеріал стін гаража»

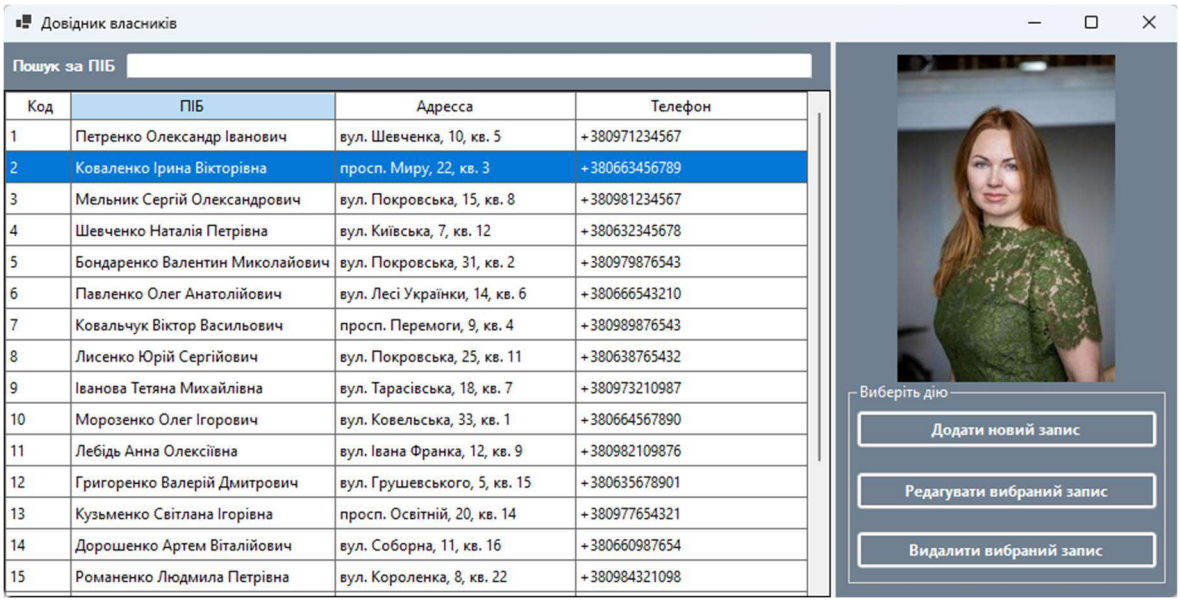


Рисунок 3.2.8 – Довідник «Власник гаража»

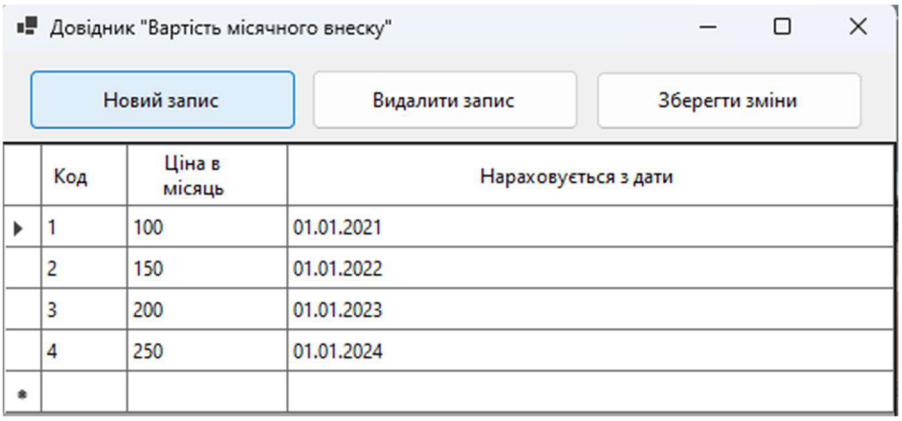


Рисунок 3.2.9 – Довідник «Матеріал стін гаража»

Форма з інформацією про розробника має вигляд представлений на рисунку 3.2.10.

Рисунок 3.2.10 – Вікно про розробника «Гаражний кооператив»

Висновки до розділу 3

Програма для гаражного кооперативу є важливим інструментом для керування та організації різних аспектів управління гаражами та їх власниками. Програма призначена для використання в будь-якому гаражному кооперативі, незалежно від його розміру чи складності. Вона може бути налаштована під конкретні потреби та вимоги кожного кооперативу. Програма надає широкий спектр функцій, включаючи управління власниками гаражів, додавання та редагування інформації про гаражі, визначення цін внесків, облік платежів та інше. Вона дозволяє зручно вести облік та контролювати фінансові операції кооперативу.

Програма має інтуїтивно зрозумілий і легкий у використанні інтерфейс. Це дозволяє користувачам легко орієнтуватися та швидко знаходити необхідні функції без додаткового навчання.

Програма побудована з урахуванням модульної архітектури, що дозволяє легко розширювати та змінювати її функціональність. Це дозволяє внесення змін та додавання нових функцій без значних зусиль.

Програма працює ефективно та швидко, забезпечуючи оперативну обробку даних та відповідь на користувальницькі запити.

ВИСНОВКИ

У цій дипломній роботі була розглянута тема розробки прикладної програми для гаражного кооперативу. Було проведено аналіз актуальності теми, визначені основні цілі та завдання розробки програми, а також обґрунтовано вибір технологій та мов програмування.

У ході роботи було розроблено програму, яка включає в себе модулі управління власниками гаражів, гаражами, матеріалами, цінами внесків та платежами. Програма має зручний інтерфейс, що дозволяє з легкістю виконувати різноманітні операції з керування кооперативом.

Аналіз та порівняння схожих програм дозволили виокремити переваги та недоліки розробленої програми, а також виявити можливості для її подальшого вдосконалення. Загалом, програма для гаражного кооперативу є надійним інструментом для автоматизації рутинних процесів управління гаражами та фінансами. Вона допомагає ефективно керувати різними аспектами діяльності кооперативу та забезпечує оптимальну роботу всієї системи

Отже, дипломна робота виявилася успішною у досягненні своєї мети - розробки програмного забезпечення для гаражного кооперативу. Результати роботи можуть бути корисними для організацій, що мають потребу у зручному та ефективному інструменті для управління своєю діяльністю.

Перелік літературних джерел

1. Закон України "Про кооперативи". URL: https://zakon.rada.gov.ua/laws/show/1576-14 (дата звернення: [01.06.2024]).
2. Національна спілка кооперативів України. URL: https://nscu.org.ua/ (дата звернення: [01.06.2024]).
3. "Кооперативне життя" - онлайн-журнал про кооперативний рух в Україні. URL: https://kooperatyvne-zhyttya.com.ua/ (дата звернення: [01.06.2024]).
4. Інформаційно-аналітичний портал "Кооперативна Україна". URL: https://cooperative.in.ua/ (дата звернення: [01.06.2024]).
5. Сайт "Український кооперативний рух". URL: https://koop.org.ua/ (дата звернення: [01.06.2024]).
6. Міністерство регіонального розвитку, будівництва та ЖКГ України. URL: https://minregion.gov.ua/ (дата звернення: [01.06.2024]).
7. Головне управління статистики у Волинській області. URL: http://volynstat.gov.ua/ (дата звернення: [01.06.2024]).
8. Асоціація громадських об'єднань "Наші гроші". URL: https://www.nashi-groshi.org/ (дата звернення: [01.06.2024]).
9. Міністерство економічного розвитку, торгівлі та сільського господарства. URL: https://www.me.gov.ua/ (дата звернення: [01.06.2024]).
10. Сайт "Житло в Україні". URL: https://zhytlo.gov.ua/ (дата звернення: [01.06.2024]).
11. Інформаційно-аналітичний портал "Консалтингова група ACE". URL: https://www.agrupa.com.ua/ (дата звернення: [01.06.2024]).
12. Український інститут соціальних досліджень ім. Олександра Ярошa. URL: https://ukrainske.sociology.org.ua/ (дата звернення: [01.06.2024]).
13. Асоціація регіональних розвитку України. URL: http://arr.org.ua/ (дата звернення: [01.06.2024]).
14. Інформаційно-аналітичний портал "Громадське місто". URL: https://www.hrom-misto.org/ (дата звернення: [01.06.2024]).

15. Український союз кооператорів. URL: http://ukrcoopunion.com/ (дата звернення: [01.06.2024]).
16. Агентство регіонального розвитку. URL: https://arr.org.ua/ (дата звернення: [01.06.2024).
17. Проект "СмартСіті". URL: http://smartcity.gov.ua/ (дата звернення: [01.06.2024]).
18. Спілка адвокатів України. URL: https://www.uaa.org.ua/ (дата звернення: [01.06.2024]).
19. Українська асоціація консультантів з управління. URL: https://uacm.org.ua/ (дата звернення: [01.06.2024]).
20. Інститут з аналізу місцевої політики. URL: http://localpolitics.org.ua/ (дата звернення: [01.06.2024]).

ДОДАТКИ

Додаток А

Програмний код модулів

```
namespace GarageCooperative {
   partial class Directories//<T>  where T : class     {
      /// <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()       {
         System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Directories));
         mTable = new System.Windows.Forms.DataGridView();
         panel1 = new System.Windows.Forms.Panel();
         button3 = new System.Windows.Forms.Button();
         button1 = new System.Windows.Forms.Button();
         button2 = new System.Windows.Forms.Button();
```

```
        ((System.ComponentModel.ISupportInitialize)mTable).BeginInit();
        panel1.SuspendLayout();
        SuspendLayout();            //
        // mTable            //
        mTable.ColumnHeadersHeightSizeMode = System.
Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
        mTable.Dock = System.Windows.Forms.DockStyle.Fill;
        mTable.Location = new System.Drawing.Point(0, 72);
        mTable.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
        mTable.Name = "mTable";
        mTable.RowHeadersWidth = 51;
        mTable.Size = new System.Drawing.Size(619, 483);
        mTable.TabIndex = 2;            //
        // panel1            //
        panel1.Controls.Add(button3);
        panel1.Controls.Add(button1);
        panel1.Controls.Add(button2);
        panel1.Dock = System.Windows.Forms.DockStyle.Top;
        panel1.Location = new System.Drawing.Point(0, 0);
        panel1.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
        panel1.Name = "panel1";
        panel1.Size = new System.Drawing.Size(619, 72);
        panel1.TabIndex = 6;            //
        // button3            //
        button3.Anchor        =        System.Windows.Forms.AnchorStyles.Top        |
System.Windows.Forms.AnchorStyles.Bottom                                        |
System.Windows.Forms.AnchorStyles.Left;
        button3.Location = new System.Drawing.Point(19, 14);
        button3.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
        button3.Name = "button3";
```

```
            button3.Size = new System.Drawing.Size(185, 49);
            button3.TabIndex = 11;
            button3.Text = "Новий запис";
            button3.UseVisualStyleBackColor = true;
            button3.Click += newrow_Click;          //
            // button1          //
            button1.Anchor      =      System.Windows.Forms.AnchorStyles.Top      |
System.Windows.Forms.AnchorStyles.Bottom                                       |
System.Windows.Forms.AnchorStyles.Left;
            button1.Location = new System.Drawing.Point(411, 15);
            button1.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
            button1.Name = "button1";
            button1.Size = new System.Drawing.Size(185, 48);
            button1.TabIndex = 10;
            button1.Text = "Зберегти зміни";
            button1.UseVisualStyleBackColor = true;
            button1.Click += save_Click;          //
            // button2          //
            button2.Anchor      =      System.Windows.Forms.AnchorStyles.Top      |
System.Windows.Forms.AnchorStyles.Bottom                                       |
System.Windows.Forms.AnchorStyles.Left;
            button2.Location = new System.Drawing.Point(215, 14);
            button2.Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
            button2.Name = "button2";
            button2.Size = new System.Drawing.Size(185, 49);
            button2.TabIndex = 9;
            button2.Text = "Видалити запис";
            button2.UseVisualStyleBackColor = true;
            button2.Click += del_Click;          //
            // Directories          //
```

```
            AutoScaleDimensions = new System.Drawing.SizeF(8F, 20F);
            AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            ClientSize = new System.Drawing.Size(619, 555);
            Controls.Add(mTable);
            Controls.Add(panel1);
           //  Icon = (System.Drawing.Icon)resources.GetObject("$this.Icon");
            Margin = new System.Windows.Forms.Padding(4, 5, 4, 5);
            MinimumSize = new System.Drawing.Size(634, 590);
            Name = "Directories";
            StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            Text = "Довідник";
            Load += Dovidnyk_Load;
            ((System.ComponentModel.ISupportInitialize)mTable).EndInit();
            panel1.ResumeLayout(false);
            ResumeLayout(false);        }
        #endregion
        private System.Windows.Forms.DataGridView mTable;
        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
    }
}}}}
```