Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

\_\_\_\_\_\_\_Програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЄКТ**

з \_\_Принципи конструювання програмного забезпечення\_\_

(назва дисципліни)

на тему:\_\_\_**Система управління базою данних\_\_**

\_\_**ріелторської контори(купівля-продаж житла)**\_\_\_

Виконав: студент (ка) \_\_2\_\_\_ курсу, групи\_\_\_243Б\_\_\_\_

спеціальності \_\_\_\_\_121\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр спеціальності)

\_\_\_\_Інженерія програмного забезпечення\_\_\_\_\_

(назва спеціальності)

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Бабин В.В.\_\_\_\_\_\_\_\_\_

(підпис) (прізвище, ініціали)

Керівник \_\_\_\_\_\_\_Комісарчук В.В.\_\_\_\_\_\_

(прізвище, ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| **До захисту допущено:** |  |  | **Дата захисту** «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |
| від «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |  |  | **Оцінка:** |
| Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | за національною шкалою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (словами) |
| Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (цифра) |
|  |  |  | за шкалою ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  | (літера) |

Чернівці, 2022

**ЗАВДАННЯ**

**на курсовий проєкт студента**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Миколайчук Дмитро Васильович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема проєкту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_Створення системи управління базою данних для рієлторської контори засобами мови С#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Вихідні дані до проєкту:

\_Створити програму для додадавання редагування та видалення інформації в базі данних рієлторської контори, реалзіувати пошук по базі пропозицій та попиту, реалізувати функціональну повноту роботи програми та її зрозумілість користувачу, реалізувати керування базою данних за допомогою клавіатури та створення діалого вікна допомоги користувачу при натисканні кнопки F1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити):

\_– розробка блок-схем програми;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– розробка бази даних;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– організація введення даних, пошуку і виведення даних;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

\_– блок-схеми роботи з програмою;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– скріншоти роботи з програмою.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання прийняв (ла) до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Керівник проєкту \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника)

**РЕФЕРАТ**

У курсовому проєкті розроблено управління базою даних ріелторської контори, яка працює під керуванням операційної системи Windows.

Розроблена система розрахована на ріелторів та менеджерів, що займаються нерухомістю.

Область застосування – програмне забезпечення (інформаційної системи) для недорогих комп’ютерів та ноутбуків з обмеженими обчислювальними ресурсами.

Розробка реалізована засобами середовища Visual Studio на мові C# із застосуванням WinForms. Дане середовище є зручним у використанні для швидкого та якісного створення додатків на OC Windows.

Дана розробка у майбутньому може бути розширена із додаванням нового функціоналу і видозміненою логікою обробки.

Курсовий проєкт містить: \_54\_ с., \_11\_ рис., \_8\_ табл., \_1\_ додаток, \_18\_ джерел.

ІНФОРМАЦІЙНА СИСТЕМА, НЕРУХОМІСТЬ, ПОПИТ, ПРОПОЗИЦІЯ, ОПЕРАЦІЙНА СИСТЕМА, ІНТЕРАКТИВНА ГРА, КОМП’ЮТЕР, VISUAL STUDIO.

**SUMMARY**

In the course project, the management of the database of the real estate office, which works under the control of the Windows operating system, is developed.

The developed system is designed for realtors and real estate managers.

The field of application is software (information system) for low-cost computers and laptops with limited computing resources.

The development is implemented using the Visual Studio environment in the C# language with the use of WinForms. This environment is convenient to use for quick and high-quality creation of applications on OC Windows.

This development can be expanded in the future with the addition of new functionality and modified processing logic.

The course project contains: \_54\_ pages, \_11\_ figures, \_8\_ tables, \_1\_ appendix, \_18\_ sources.

INFORMATION SYSTEM, REAL ESTATE, DEMAND, SUPPLY, OPERATING SYSTEM, INTERACTIVE GAME, COMPUTER, VISUAL STUDIO.

Зміст

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 4](#_Toc120995377)

[ВСТУП 5](#_Toc120995378)

[1 АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ 6](#_Toc120995379)

[1.1 Загальні вимоги до програми 6](#_Toc120995380)

[1.2 Призначення та область застосування 8](#_Toc120995381)

[1.3 Функціональні вимоги 8](#_Toc120995382)

[2 ОПИС ПРОГРАМИ 14](#_Toc120995383)

[2.1 Структура програми 14](#_Toc120995384)

[2.1.1 Модулі програми 14](#_Toc120995385)

[2.1.2 Алгоритми роботи програми 15](#_Toc120995386)

[2.1.3 Опис методів програми 17](#_Toc120995387)

[2.3 Програмні засоби 21](#_Toc120995388)

[2.3 Опис користувацького інтерфейсу 22](#_Toc120995389)

[ВИСНОВКИ 28](#_Toc120995390)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 29](#_Toc120995391)

[ДОДАТОК А 31](#_Toc120995392)

# **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

ADGO.NET – ActiveX Data Object для .NET

API – Application Programmine Interface

CIL – Common Intermediate Language

CLR – Common Language Runtime

IDE - комплексне програмне рішення для розробки програмного забезпечення

IL – Intermediate Language

JIT – Just In Time

UI - інтерфейсу користувача

VS - Visual Studio

WCF – Windows Communication Foundation

WF – Windows Forms

ООП - обʼєктно-орієнтоване програмування.

# **ВСТУП**

Використання автоматизованої системи управління агентствами по продажі нерухомості на сьогодні є ключовим чинником конкурентної боротьби. Інформаційна система дозволяє підвищити продуктивність праці співробітників, прискорити інформаційний обмін між підрозділами агентства, знизити витрати, контролювати в режимі реального часу всі процеси, що протікають в агентстві. Автоматизація всього спектру бізнес-процесів агентства дозволяє суттєво підвищити його конкурентоспроможність, покращити якість управління , знизити витрати, швидко обробляти великі обсяги інформації.

Комплекс завдань з автоматизації діяльність агентства нерухомість включає такі завдання: зберігання база попиту і пропозицій та здійснювати пошук необхідної інформації.

Проектування інформаційних систем – складна та тривала робота, яка потребує високого професіоналізму. У процесі створення та роботи системи потреби користувачів завжди змінюються або уточняються, що ускладнює та подовжує процес розробки таких систем.

Більшість існуючих CASE-засобів засновані на методах структурного або об'єктно-орієнтованого аналізу та проектування, що використовують специфікації у вигляді діаграм або текстів для опису зовнішніх вимог, зв’язків між моделями системи, динаміки поведінки системи та архітектури програмних засобів.

Основною метою курсового проектування є проектування, , дослідження, та оптимізація бізнес процесів агентства нерухомості, після чого буде проведено розробку програмного забезпечення для ріелторської контори.

# **1 АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ**

## **1.1 Загальні вимоги до програми**

Огляд сучасних підходів та засобів до проектування та розроблення програмного забезпечення дозволив обрати для створення власної системи ефективні технології та інструментальні засоби: IDE – Visual Studio 2019; Мова програмування – C#.

На основі виконаного аналізу предметної області можна сформулювати загальні вимоги до програми.

Розробити систему, яка автоматизує роботу ріелторської контори, дозволить вести облік пропозицій та попиту на нерухомість. Забезпечити внесення даних про попит та пропозиції.

Розробити зручний графічний інтерфейс для роботи з програмою. Основні дії та взаємодія між користувачем та системою повинні супроводжуватися відповідними повідомленнями для користувача.

Створити навігаційне меню для можливості швидкого та зручного отримання доступу до потрібної функції в системі.

Розробити супровідну документацію до створеної системи.

Курсовий проект припускає розробку додатка засобами об’єктно-орієнтованого середовища програмування .

Розробка та реалізація додатку включає 2 основні модулі:

1) модуль вводу/редагування інформації;

2) модуль пошуку інформації.

Реалізація додатка виконується з використанням технологій .NET та мови програмування c#. Результат –Windows додаток.

Вона повинна містити в собі:

**Класи додатку:**

* попит на нерухомість ;
* пропозиція на нерухомість.

**Введення:**

Вводиться і редагується: інформація про попит та пропозиції на нерухомість.

**Формування:**

* база пропозицій;
* база попиту.

**Пошук:**

* пошук по пропозиції;
* пошук по попиту.

**Результат роботи:**

* Здійснення пошуку необхідної пропозиції або попиту у базі даних.

Серед усіх функцій, які виконує система можна виділити загальні операції:

* реакція програми на вибір меню користувача;
* реакція програми на натискання кнопок в програмі;
* реакція програми на введення неправильних даних.

**Функціональні вимоги:**

* можливість додавати, редагувати та видаляти дані про пропозицію та попит;
* можливість пошуку пропозиції та попиту за різними критеріями.

**Нефункціональні вимоги:**

* для роботи програми на комп’ютері повинна бути встановлена бібліотека класів .NET Framework 4.7;
* для забезпечення роботи програми потрібно мати лише клавіатуру та мишку.

## **1.2 Призначення та область застосування**

Мета роботи полягає у розробці системи управління базою даних ріелторської контори, яка буде працювати під керуванням ОС Windows.

Реалізована версія програми «Система управління базою даних ріелторської контори» призначена для менеджерів та ріелторів, що займаються пошуком та продажом нерухомості.

Область застосування – програмне забезпечення для недорогих ноутбуків та комп’ютерів з обмеженими обчислювальними ресурсами.

## **1.3 Функціональні вимоги**

Згідно завдання на курсове проектування можна висунути функціональний список вимог до проекту (табл. 1.1).

Таблиця 1.1. – Функціональні вимоги до додатку «Система управління базою даних ріелторської контори»

|  |  |
| --- | --- |
| **Вимоги** | **Опис** |
| REQ-1 | Система повинна дозволяти користувачеві додати та редагувати попит на нерухомість |
| REQ-2 | Система повинна дозволяти користувачеві додати та редагувати інформацію про пропозиції на нерухомість |
| REQ-3 | Система повинна дозволяти користувачеві здійснювати пошук попит на нерухомість по різним критеріям |
| REQ-4 | Система повинна дозволяти користувачеві здійснювати пошук пропозиції на нерухомість по різним критеріям |

Таблиця 1.2. – Нефункціональні вимоги до додатка «Система управління базою даних ріелторської контори»

|  |  |
| --- | --- |
| **Вимоги** | **Опис** |
| REQ-5 | Додаток повинен мати простий дизайн та зручну навігації |
| REQ-6 | Поля повинні бути не порожніми, унікальними відносно вже існуючих записів |

Таблиця 1.3. – Актори та цілі додатка «Система управління базою даних ріелторської контори»

|  |  |
| --- | --- |
| **Актори** | **Цілі** |
| Користувач | Мета користувача полягає в роботі з системою. |
| Файл бази даних | Мета бази даних полягає у зберігання інформації |

Таблиця 1.4. – Опис варіантів використання додатка «Нерухомість м. Києві»

|  |  |  |
| --- | --- | --- |
| **Варіант використання** | **Ім’я** | **Опис** |
| UC1 | Вивід каталогу пропозицій | Дозволяє користувачеві вивести каталог всіх пропозицій на продаж нерухомості |
| UC2 | Додати пропозицію | Дозволяє користувачеві додати нову пропозицію |
| UC3 | Редагувати пропозицію | Дозволяє користувачеві редагувати інформацію вибраної із списку пропозиції |
| UC4 | Видалити пропозицію | Дозволяє користувачеві видалити вибрану із списку пропозицію |
| UC5 | Вивід каталогу попиту | Дозволяє користувачеві вивести каталог попиту на нерухомість |
| UC6 | Додати попит | Дозволяє користувачеві додати попит на нерухомість |
| UC7 | Редагувати попит | Дозволяє користувачеві редагувати інформацію вибраного із списку попиту |
| UC8 | Видалити попит | Дозволяє користувачеві видалити вибраний із списку попит |
| UC9 | Пошук пропозиції | Дозволяє користувачеві програми здійснювати пошук пропозиції по різним критеріям |
| UC10 | Пошук попиту | Дозволяє користувачеві програми здійснювати пошук попиту по різним критеріям |

З поставлених вимог тепер можна виставити повний опис вимог із сценаріями, що будуть вхідними даними для візуального моделювання мовою UML.

**UC1 Вивід каталогу пропозицій**

Актор: користувач.

Ціль актора: вивести інформацію про всі пропозиції про нерухомість.

Задіяний актор: база даних.

Передумова: користувач переходить по меню програми «База» –> «Пропозиції».

Післяумова: система відображає екран для виведення списку всіх пропозицій.

**UC2 Додати пропозицію**

Актор: користувач.

Ціль актора: додати нову пропозицію.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про пропозицію натискає на кнопку «Додати».

Післяумова: система додає нову пропозицію та відображає екран із списком всіх пропозицій.

**UC3 Редагувати пропозицію**

Актор: користувач.

Ціль актора: редагувати інформацію вибраної пропозиції.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку пропозицію.

Післяумова: система відображає екран для редагування інформації про вибрану пропозицію’.

**UC4 Видалити пропозицію**

Актор: користувач.

Ціль актора: видалити пропозицію із файлу бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку пропозицію.

Післяумова: система відображає екран з можливістю видалення даних про пропозицію.

**UC5 Вивід каталогу попиту**

Актор: користувач.

Ціль актора: вивести інформацію про попит на ринку нерухомості.

Задіяний актор: база даних.

Передумова: користувач переходить по меню програми «База» –> «Попит».

Післяумова: система відображає екран для виведення списку всього попиту.

**UC6 Додати попит**

Актор: користувач.

Ціль актора: додати новий попит.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про попит натискає на кнопку «Додати».

Післяумова: система додає новий попит і відображає екран із списком всього попиту.

**UC7 Редагувати попит**

Актор: користувач.

Ціль актора: редагувати вибраний із списку попит.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідний із списку попит.

Післяумова: система відображає екран для редагування інформації про вибраний попит.

**UC8 Видалити попит**

Актор: користувач.

Ціль актора: видалити вибраний попит із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідний із списку попит.

Післяумова: система відображає екран з можливістю видалення даних попиту.

**UC9 Пошук пропозиції**

Актор: користувач.

Ціль актора: здійснювати пошук пропозиції за різними критеріями.

Задіяний актор: база даних.

Передумова: користувач переходить по меню програми «Пошук» –> «Пропозицій».

Післяумова: система відображає екран з можливістю пошуку пропозиції за різними критеріями.

**UC10 Пошук попиту**

Актор: користувач.

Ціль актора: здійснювати пошук попиту за різними критеріями.

Задіяний актор: база даних.

Передумова: користувач переходить по меню програми «Пошук» –> «Попиту».

Післяумова: система відображає екран з можливістю пошуку попиту за різними критеріями.

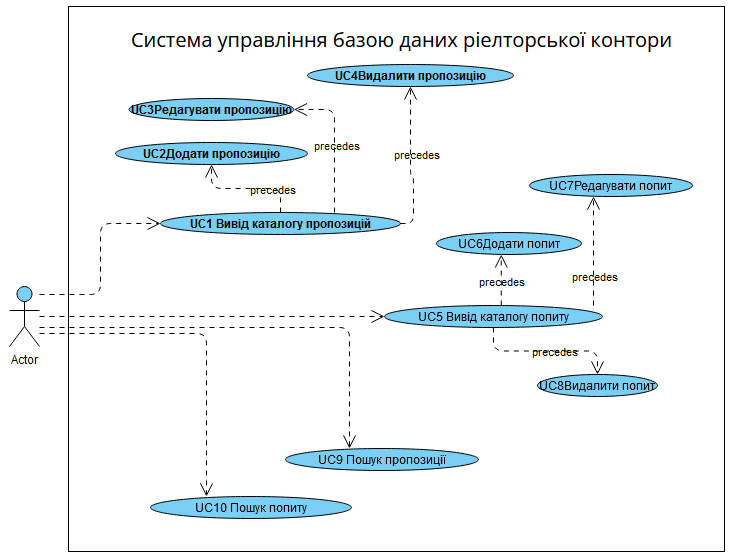


Рисунок 1.1 – Діаграма варіантів використання додатку (use case diagram)

# **2 ОПИС ПРОГРАМИ**

## **2.1 Структура програми**

## **2.1.1 Модулі програми**

Для розробки проекту було використано трирівневу архітектуру. Трирівнева архітектура (трьохланкова архітектура, англ. three-tier) — архітектурна модель програмного комплексу, що передбачає наявність у ньому трьох типів компонентів (рівнів, ланок): клієнтських додатків (з якими працюють користувачі), серверів додатків серверів баз даних (з якими працюють сервери додатків).

На рисунку 2.1 приведено структуру класів модуля.



Рисунок 2.1 – Структура класів модуля

Програма складається із одинадцяти класів, шість із яких рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс.

Клас «RealEstateOfficeMDI» становить головне вікно програми. Найважливішим елементом управління в головному вікні є розташування головного меню, за допомогою якого можна відкривати інші форми та вийти з програми.

Клас «BaseDemandForm» призначений для опрацювання даних попиту нерухомості.

Клас «UpdateBaseDemandForm» призначений для редагування даних про попит нерухомості.

Клас «BaseOffersForm» призначений для опрацювання даних пропозицій на нерухомість.

Клас «UpdateBaseOffersForm» призначений для редагування даних пропозицій на нерухомість.

Клас «BaseDemandSerchForm» призначений для пошуку попиту на нерухомість за різними критеріями.

Клас «BaseOffersSearchForm» призначений для пошуку пропозиції на нерухомість за різними критеріями.

Створення об'єктів інших команд та виклик їх методів відбувається в результаті взаємодії користувача з елементами графічного інтерфейсу програми.

## **2.1.2 Алгоритми роботи програми**

В даному підрозділі будуть приведені основні блок схеми роботи програми.

На рис. 2.2 показана блок-схема запису інформації про пропозицію на нерухомість у файл бази даних.



Рисунок 2.2 – Запис інформації про пропозицію на нерухомість у файл бази даних

На рис. 2.3 показана блок-схема додавання нового запису про пропозицію на нерухомість та виведення списку всіх пропозицій на екран.



Рисунок 2.3 – Додавання нової пропозиції

## **2.1.3 Опис методів програми**

Список методів класу BaseDemandForm та їх опис наведено в табл. 2.1.

Таблиця 2.1 – Основні методи класу «BaseDemandForm»

|  |  |  |
| --- | --- | --- |
| *№* | *Назва методу* | *Короткий опис* |
| 1 | AddBtn\_Click(object sender, EventArgs e) | Додавання попиту на нерухомість |
| 2 | ClearBtn\_Click(object sender, EventArgs e) | Очищення всіх полів для введення даних |
| 3 | ExitBtn\_Click(object sender, EventArgs e) | Закрити форму для опрацювання даних попиту |
| 4 | BaseDemandGridView \_CellClick(object sender, DataGridViewCellEventArgs e) | Виведення дочірнього вікна для редагування вибраного попиту |
| 5 | DataLoad() | Вибірка із файлу всіх даних попиту |
| 6 | LoadDataInBaseDemandGridView (List<BaseDemand> BaseDemandList) | Завантаження вибраних із файлу даних попиту у таблицю для візуального відображення |
| 7 | ClearAllControls() | Очищення полів для введення даних |
| 8 | IsDataEnteringCorrect() | Перевірка даних на введення |
| 9 | SetDataInBaseDemandList() | Встановлення редагованих даних у список для подальшого їх збереження у файл |
| 10 | SelectBaseDemand(int Number, List<BaseDemand> BaseDemandList) | Вибірка пропозиції із списку по її номеру |
| 11 | DeleteBaseDemandInfo(int Number) | Видалення інформації про пропозицію |
| 12 | SetNumber() | Встановлення номеру для кожної пропозиції |

Список методів класу UpdateBaseDemandForm та їх опис наведено в табл. 2.2.

Таблиця 2.2 – Основні методи класу «UpdateBaseDemandForm»

|  |  |  |
| --- | --- | --- |
| *№* | *Назва методу* | *Короткий опис* |
| 1 | SaveBtn\_Click(object sender, EventArgs e) | Збереження даних редагованого попиту |
| 2 | DeleteBtn\_Click(object sender, EventArgs e) | Видалення вибраного попиту із списку |
| 3 | ExitBtn\_Click(object sender, EventArgs e) | Закриття дочірньої форми для редагування попиту |
| 4 | LoadAllDate() | Завантаження вибраного попиту у поля для редагування |
| 5 | SaveData() | Збереження редагованих даних попиту у файл бази даних |
| 6 | IsDataEnteringCorrect() | Перевірка введення даних |

Список методів класу BaseDemandSerchForm та їх опис наведено в табл. 2.3.

Таблиця 2.3 – Основні методи класу «BaseDemandSerchForm»

|  |  |  |
| --- | --- | --- |
| *№* | *Назва методу* | *Короткий опис* |
| 1 | SearchPriceBtn\_Click(object sender, EventArgs e) | Пошук списку попиту за введеною ціною |
| 2 | AreaSearchBtn\_Click(object sender, EventArgs e) | Пошук списку попиту за введеною площею |
| 3 | RegionSearchBtn\_Click(object sender, EventArgs e) | Пошук списку попиту за введеним регіоном |
| 4 | SetNumber() | Встановлення нумерації у списку |
| 5 | LoadDataInBaseDemandGridView (List<BaseDemand> BaseDemandList) | Завантаження даних попиту у таблицю |
| 6 | IsDataEnteringCorrect() | Перевірка даних поля ціни на коректність |
| 7 | IsDataAreaCorrect() | Перевірка даних поля площа на коректність |
| 8 | IsDataRegionorrect | Перевірка даних поля район на коректність |

Список методів класу SearchBLL та їх опис наведено в табл. 2.4.

Таблиця 2.4 – Основні методи класу «SearchBLL»

|  |  |  |
| --- | --- | --- |
| *№* | *Назва методу* | *Короткий опис* |
| 1 | GetBaseOffersByNumber RroomsAndPrice (int NumberRrooms, double Price) | Повертає список пошуку пропозицій за введеною кількістю кімнат та ціною |
| 2 | GetBaseOffersByMinMaxArea(int AreaMin, int AreaMax) | Повертає список пошуку пропозицій за введеною мінімальною та максимальною площею |
| 3 | GetBaseOffersByRegion(string Region) | Повертає список пошуку пропозицій за введеним районом |
| 4 | GetBaseDemandByPrice(double Price) | Повертає список пошуку попиту за введеною ціною |
| 5 | GetBaseDemandByArea(int Area) | Повертає список пошуку попиту за введеною площею |
| 6 | GetBaseDemandByRegion(string Region) | Повертає список пошуку попиту за введеним районом |

Класи BaseOffersForm, BaseOffersSearchForm та UpdateBaseOffersForm схожі за своєю структурою на описані вище.

## **2.3 Програмні засоби**

Для побудови користувацького інтерфейсу будемо використовувати середовище розробки Visual Studio 2019 та мову програмування C#.

С# широко використовується професіоналами для розробки великих програмних продуктів завдяки наступним аспектам мови:

* Сучасна мова програмування загального призначення.
* Підтримка об’єктно-орієнтованої парадигми.
* Підтримка компонентно-орієнтованої парадигми.
* Мова легка для вивчення навчитися.
* Добре структурована.
* Дозволяє розробляти ефективні програми.
* Мова має підтримку різних комп'ютерних платформах.
* Це частина .Net Framework.

C# спроектовано таким чином, що мова відповідає традиційним мовам високого рівня, C та C++ і є об'єктно-орієнтованою мовою програмування. Мова дуже схожа на Java, має численні сильні функції програмування, які роблять його привабливим для багатьох програмістів у всьому світі.

.NET Framework - це платформа для розробників із відкритим кодом, яку можна використовувати для створення широкого кола програм. Цей безкоштовний крос-платформний фреймворк підтримує декілька мов і має великі бібліотеки коду, які спрощують створення додатків для мобільних пристроїв, робочих столів та Інтернету.

Платформа .NET була розроблена для досягнення наступних цілей:

* Сумісність;
* Підтримка різних платформам;
* Мовна незалежність;
* Бібліотека базових класів;
* Легка розробка;
* Безпека.

Для розробки користувацького інтерфейсу платформа .Net має декілька технологій, одна з яких – WinForms. Не дивлячись на те, що ця технологія досить не нова, її важко назвати застарілою. Вона надає широкий спектр різних інструментів для побудови зручного та сучасного інтерфейсу. Крім того, IDE, які підтримують C# та .Net, надають зручний інтерфейс для графічної побудови користувацького застосунку, який розробляється.

Отже, C# та платформа .Net має низку характеристик, які задовольняють вимоги щодо розробки клієнтської частини системи складського обліку. Набір готових класів у стандартній бібліотеці, лаконічний зрозумілий синтаксис мови та зручний конструктор користувацького інтерфейсу зробить розробку зручною та достатньо швидкою. Об’єктно-орієнтована парадигма дозволить спроектувати систему таким чином, що розширення функціоналу буде без накладних розходів ресурсів розробки. Платформа .Net забезпечить безпеку, ефективність програмного забезпечення, а також підтримку декількох платформ.

## **2.3 Опис користувацького інтерфейсу**

Після запуску програми буде відкрите головне вікно програми з основним меню (рис.2.4).

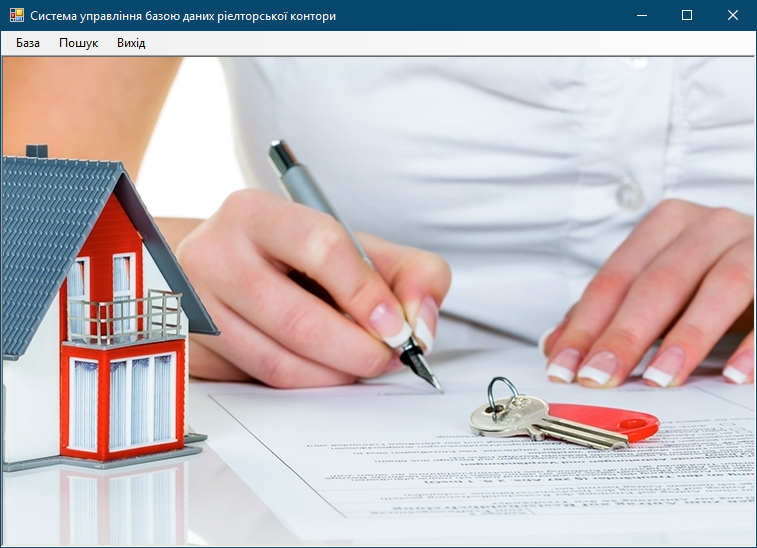


Рисунок 2.4 – Головне меню програми

Для роботи програми необхідно наповнити інформаційну систему інформацією про попит та пропозиції на ринку нерухомості.

Програма дозволяє також редагувати та видаляти введені дані про пропозиції на ринку нерухомості.

Для того, щоб додати інформацію про пропозицію необхідно перейти по меню програми «База» –> «Пропозицій», у відповідному вікні (рис. 2.5) ввести всі необхідні дані про район, номер будинку, номер квартири, площу, кількість кімнат, вартість, адресу, координати замовника та натиснути кнопку «Додати». Нова пропозиція з’явиться у правій частині екрану у списку всіх пропозицій.

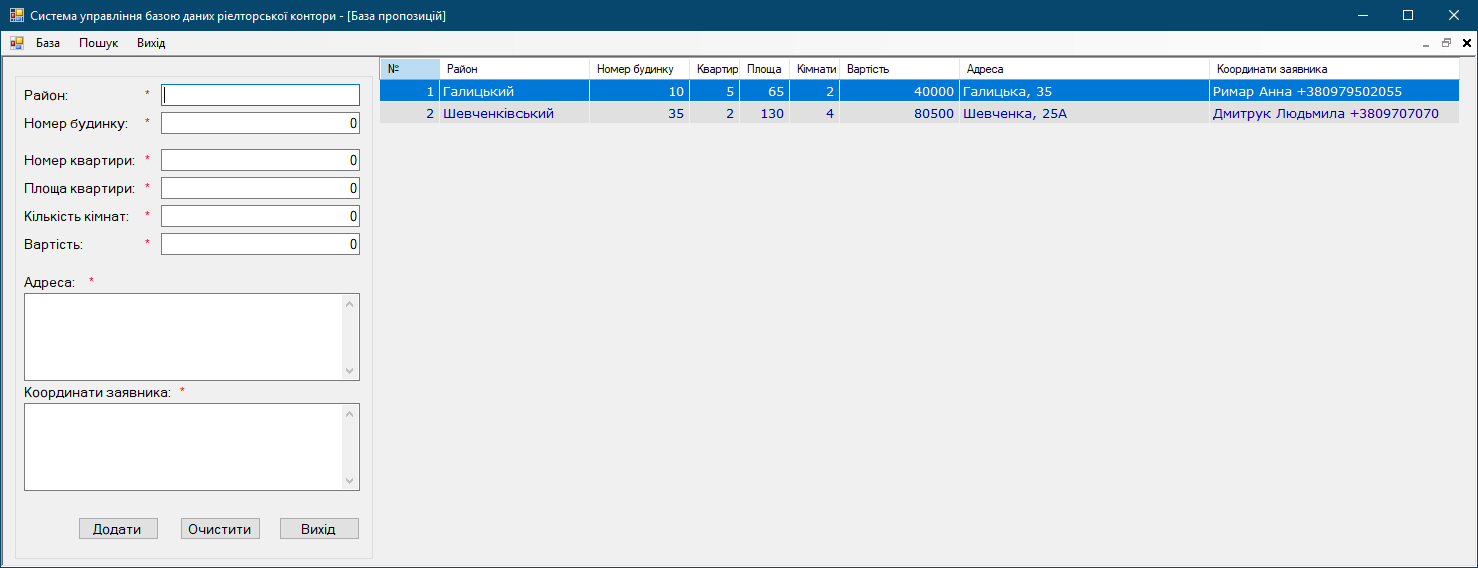


Рисунок 2.5 – Вікно для опрацювання даних про пропозицію

Також реалізована можливість редагування та видалення даних у випадку якщо це є необхідним. На рис 2.6 зображено вікно для редагування даних про пропозицію.

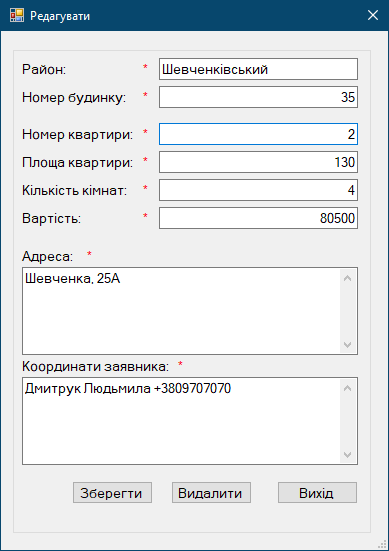


Рисунок 2.6 – Вікно для редагування даних про пропозицію

Далі необхідно заповнити базу попитом на нерухомість, для цього необхідно перейти по меню програми «База» –> «Попиту», у відповідному вікні (рис. 2.7) ввести всі необхідні дані про район, площу квартири, кількість кімнат, вартість, координати покупця та натиснути кнопку «Додати». Новий попит з’явиться у правій частині екрану у списку всього попиту.

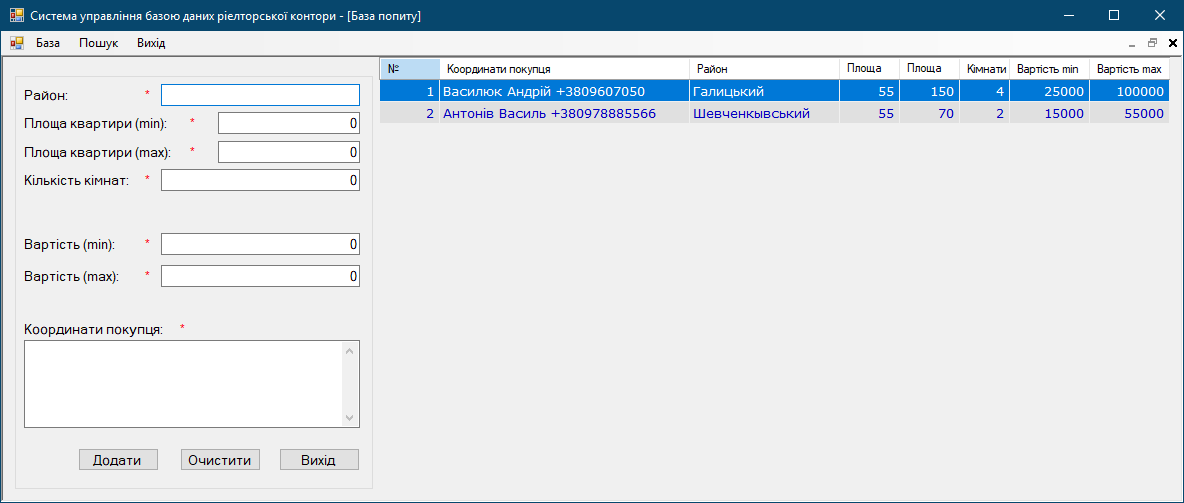


Рисунок 2.7 – Вікно для опрацювання даних про попит

Дані про попитможна відредагувати, вибравши у правій частині необхідний запис натисканням лівої кнопки мишки. Після цього з’явиться вікно для редагування даних вибраного попиту (рис. 2.8).

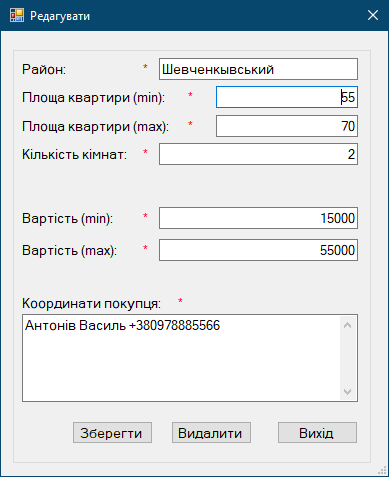


Рисунок 2.8 – Вікно для редагування даних вибраного попиту

Також у програмі реалізовано пошук інформації пропозицій та попиту на ринку нерухомості (рис. 2.9 – 2.10).

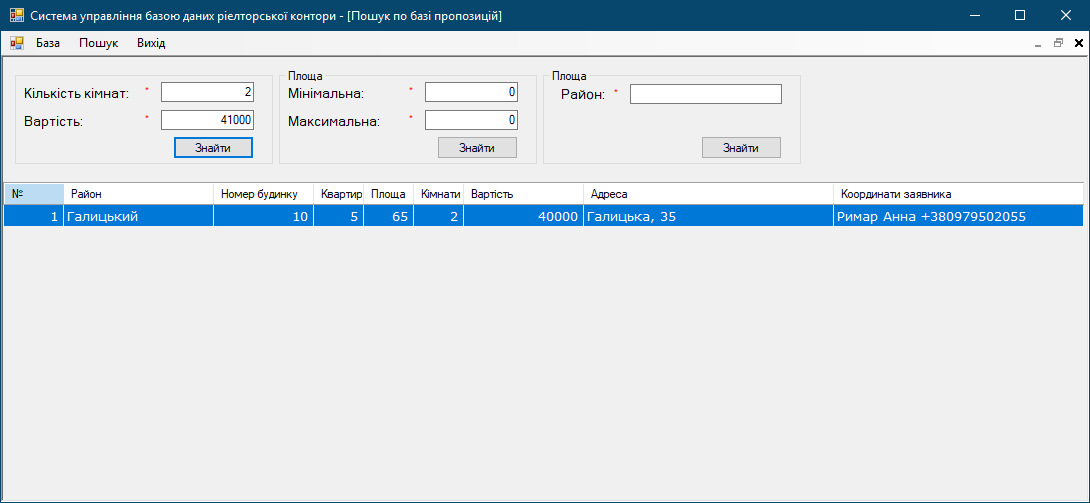


Рисунок 2.9 – Пошук пропозиції за кількістю кімнат та ціні

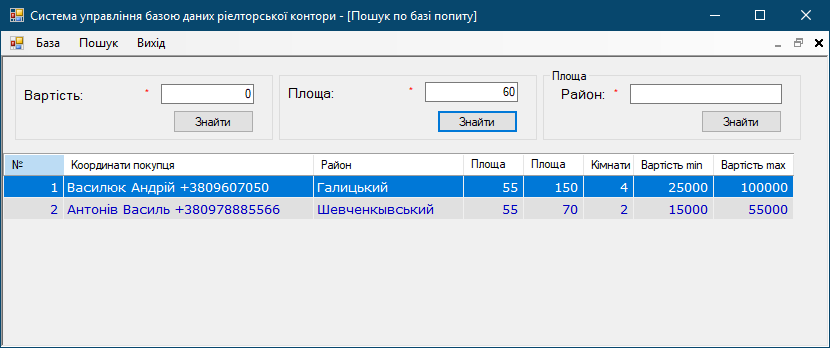


Рисунок 2.10 – Пошук попиту за площею

Також, у програмі є прив’язка дій до клавіш: F1 – допомога; Entег – згода, завершення введення; Еsс – відмова, закриття відповідного вікна програми (з відновленням екранної форми); Таb – перехід до наступного поля.

Для виходу із програми необхідно натиснути «Вихід» у головному меню.

# **ВИСНОВКИ**

Використання системи управління базою даних ріелторської контори. дозволяє підвищити продуктивність праці співробітників, прискорити інформаційний обмін між підрозділами агентства, знизити витрати, контролювати в режимі реального часу всі процеси, що протікають в агентстві. Автоматизація всього спектру бізнес-процесів агентства дозволяє суттєво підвищити його конкурентоспроможність, покращити якість управління , знизити витрати, швидко обробляти великі обсяги інформації.

В ході виконання роботи мовою С# в середовищі Visual Studio 2019 реалізовано додаток для обліку попиту та пропозицій на нерухомість для рієлторської контори. Реалізовано такі функціональні вимоги:

* можливість додавати, редагувати та видаляти дані про пропозиції для продажу нерухомості;
* можливість додавати, редагувати та видаляти дані про попит для купівлі нерухомості;
* можливість пошуку інформації за різними критеріями по попиту та пропозиції із продажу та купівлі нерухомості.

В результаті проведеної роботи вирішено актуальне технічне завдання для автоматизації бізнес процесів із продажом та купівлею нерухомості. У процесі вирішення завдання розроблено інженерну методику автоматизованої процедури обліку попиту та пропозицій нерухомості. У ході виконання роботи отримано такі основні наукові та практичні результати:

1. Розроблено гнучку систему, призначену для автоматизації бізнес процесів із обліку попиту та пропозицій нерухомості.

2. Розроблені методи в даній роботі можуть бути використані для широкого класу завдань, тому можливий подальший розвиток розробленого програмного забезпечення.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Нерухомість [Електронний ресурс] // Режим доступу: https://100realty.ua/uk/articles
2. Що таке нерухомість. Її різновиди та призначення [Електронний ресурс]. -Режим доступу: https://www.bulgariastreet.com.ua/shcho-take-neruhomist-yiyi-riznovidi-ta-priznachennya
3. Plektan [Електронний ресурс]. -Режим доступу: https://plektan.com
4. Intrum CRM [Електронний ресурс]. -Режим доступу: https://www.intrumnet.com/
5. Pro C# 7: With .NET and .NET Core 8-th edition / Andrew Troelse, Philip Japikse. –2017
6. Об'єктно-орієнтоване програмування [Електронний ресурс] Режим доступу: http://ruslan.rv.ua/python-essential/oop/oop\_basis/
7. Об'єктно-орієнтований підхід до програмування [Електронний ресурс] Режим доступу: http://www.znannya.org/?view=csharp-oop
8. Офіційна документація .NET [Електронний ресурс] Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/
9. Мова програмування C# і платформа .NET Core [Електронний ресурс] Режим доступу: <https://metanit.com/sharp/tutorial/1.1.php>
10. What's new in .NET 4.7 [Електронний ресурс] Режим доступу: <https://docs.microsoft.com/en-us/dotnet/core/dotnet-five>
11. .NET Core/5+ vs. .NET Framework for server apps [Електронний ресурс] Режим доступу: <https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server>
12. .NET 4.7 [Електронний ресурс] Режим доступу: <https://temofeev.ru/info/articles/predstavlyaem-net-5/>
13. Джон Скит, C# для професіоналів: тонкощі програмування, 608 ст. ISBN 978-5-8459-1909-0, 978-1-617-29134-0; 2014, Вільямс.
14. Литвинов О.А., Карпенко Н.В. Тестування інформаційних систем: модульне, інтеграційне, системне [Текст] – Д.: Ліра, 2016. – 283 с.
15. Литвинов О.А., Герасимов В.В., Карпенко Н.В. Об’єктно-орієнтована розробка інформаційних систем [Текст] – Д.: Ліра, 2018. – 448 с.
16. Програма для роботи з базами даних Microsoft Access: [Електронний ресурс] // Режим доступу: <https://www.microsoft.com/uk-ua/microsoft-365/access>.
17. Побудова ER-діаграм : [Електронний ресурс] // Режим доступу: <https://app.diagrams.net/>.
18. Архітектурні шаблони: [Електронний ресурс] // Режим доступу: <https://devzone.org.ua/post/nayvazhlivishi-arkhitekturni-shabloni-yaki-neobkhidno-znati>.

# **ДОДАТОК А**

Лістинг програмного коду

Лістинг 1. Код класу «SearchBLL»

using RealEstateOffice.Providers;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RealEstateOffice.BLL {

class SearchBLL {

private BaseOffersProvider \_BaseOffersProvider = new BaseOffersProvider();

private List<BaseOffers> \_BaseOffersList = new List<BaseOffers>();

private BaseDemandProvider \_BaseDemandProvider = new BaseDemandProvider();

private List<BaseDemand> \_BaseDemandList = new List<BaseDemand>();

public List<BaseOffers> GetBaseOffersByNumberRroomsAndPrice(int NumberRrooms, double Price) {

List<BaseOffers> selectBaseOffers = new List<BaseOffers>();

\_BaseOffersList = \_BaseOffersProvider.GetAllData();

for (int i = 0; i < \_BaseOffersList.Count; i++) {

if (\_BaseOffersList[i].NumberRrooms == NumberRrooms && \_BaseOffersList[i].Price < Price) {

selectBaseOffers.Add(\_BaseOffersList[i]);

}

}

return selectBaseOffers;

}

public List<BaseOffers> GetBaseOffersByMinMaxArea(int AreaMin, int AreaMax) {

List<BaseOffers> selectBaseOffers = new List<BaseOffers>();

\_BaseOffersList = \_BaseOffersProvider.GetAllData();

for (int i = 0; i < \_BaseOffersList.Count; i++) {

if (\_BaseOffersList[i].ApartmentArea > AreaMin && \_BaseOffersList[i].ApartmentArea < AreaMax) {

selectBaseOffers.Add(\_BaseOffersList[i]);

}

}

return selectBaseOffers;

}

public List<BaseOffers> GetBaseOffersByRegion(string Region) {

List<BaseOffers> selectBaseOffers = new List<BaseOffers>();

\_BaseOffersList = \_BaseOffersProvider.GetAllData();

for (int i = 0; i < \_BaseOffersList.Count; i++) {

if (\_BaseOffersList[i].Region.ToLower().Contains(Region.ToLower())) {

selectBaseOffers.Add(\_BaseOffersList[i]);

}

}

return selectBaseOffers;

}

public List<BaseDemand> GetBaseDemandByPrice(double Price) {

List<BaseDemand> selectBaseDemand = new List<BaseDemand>();

\_BaseDemandList = \_BaseDemandProvider.GetAllData();

for (int i = 0; i < \_BaseDemandList.Count; i++) {

if (\_BaseDemandList[i].PriceMin < Price && \_BaseDemandList[i].PriceMax > Price) {

selectBaseDemand.Add(\_BaseDemandList[i]);

}

}

return selectBaseDemand;

}

public List<BaseDemand> GetBaseDemandByArea(int Area) {

List<BaseDemand> selectBaseDemand = new List<BaseDemand>();

\_BaseDemandList = \_BaseDemandProvider.GetAllData();

for (int i = 0; i < \_BaseDemandList.Count; i++) {

if (\_BaseDemandList[i].ApartmentAreaMin < Area && \_BaseDemandList[i].ApartmentAreaMax > Area) {

selectBaseDemand.Add(\_BaseDemandList[i]);

}

}

return selectBaseDemand;

}

public List<BaseDemand> GetBaseDemandByRegion(string Region) {

List<BaseDemand> selectBaseDemand = new List<BaseDemand>();

\_BaseDemandList = \_BaseDemandProvider.GetAllData();

for (int i = 0; i < \_BaseDemandList.Count; i++) {

if (\_BaseDemandList[i].Region.ToLower().Contains(Region.ToLower())) {

selectBaseDemand.Add(\_BaseDemandList[i]);

}

}

return selectBaseDemand;

}

}

}

Лістинг 2. Код класу «BaseDemandForm»

using RealEstateOffice.AppCode;

using RealEstateOffice.Providers;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RealEstateOffice.Forms {

public partial class BaseDemandForm : Form {

public static BaseDemand \_selectedBaseDemand = new BaseDemand();

private int \_selectedRowIndex = 0;

private Validation \_validation = new Validation();

private BaseDemandProvider \_BaseDemandProvider = new BaseDemandProvider();

public static List<BaseDemand> \_BaseDemandList = new List<BaseDemand>();

public BaseDemandForm() {

InitializeComponent();

DataLoad();

}

private void AddBtn\_Click(object sender, EventArgs e) {

if (IsDataEnteringCorrect()) {

BaseDemand BaseDemand = new BaseDemand(RegionTBox.Text, Convert.ToInt32(ApartmentAreaMinTBox.Text), Convert.ToInt32(ApartmentAreaMaxTBox.Text),

Convert.ToInt32(NumberRroomsTBox.Text), ClientCoordinatesTBox.Text, Convert.ToDouble(PriceMinTBox.Text), Convert.ToDouble(PriceMaxTBox.Text));

\_BaseDemandList.Add(BaseDemand);

\_BaseDemandProvider.SaveBaseDemand(\_BaseDemandList);

DataLoad();

ClearAllControls();

}

}

private void ClearBtn\_Click(object sender, EventArgs e) {

ClearAllControls();

}

private void ExitBtn\_Click(object sender, EventArgs e) {

this.Close();

}

private void BaseDemandGridView\_CellClick(object sender, DataGridViewCellEventArgs e) {

if (e.RowIndex >= 0) {

\_selectedRowIndex = e.RowIndex;

\_selectedBaseDemand = SelectBaseDemand(Convert.ToInt32(BaseDemandGridView[0, e.RowIndex].Value.ToString()), \_BaseDemandList);

UpdateBaseDemandForm updateBaseDemandForm = new UpdateBaseDemandForm();

updateBaseDemandForm.ShowDialog();

if (\_selectedBaseDemand.IsDelete) {

DeleteBaseDemandInfo(\_selectedBaseDemand.Number);

} else {

SetDataInBaseDemandList();

}

\_BaseDemandProvider.SaveBaseDemand(\_BaseDemandList);

DataLoad();

}

}

private void DataLoad() {

int firstRowIndex = 0;

if (BaseDemandGridView.FirstDisplayedScrollingRowIndex > 0) {

firstRowIndex = BaseDemandGridView.FirstDisplayedScrollingRowIndex;

}

try {

\_BaseDemandList.Clear();

\_BaseDemandList = \_BaseDemandProvider.GetAllData();

SetNumber();

LoadDataInBaseDemandGridView(\_BaseDemandList);

if (\_selectedRowIndex == BaseDemandGridView.Rows.Count) {

\_selectedRowIndex = BaseDemandGridView.Rows.Count - 1;

}

if (\_selectedRowIndex >= 0) {

BaseDemandGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;

BaseDemandGridView.Rows[\_selectedRowIndex].Selected = true;

}

} catch (Exception ex) {

MessageBox.Show(ex.ToString());

}

}

private void LoadDataInBaseDemandGridView(List<BaseDemand> BaseDemandList) {

BaseDemandGridView.DataSource = null;

BaseDemandGridView.Columns.Clear();

BaseDemandGridView.AutoGenerateColumns = false;

BaseDemandGridView.RowHeadersVisible = false;

BaseDemandGridView.DataSource = BaseDemandList;

DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();

numberColumn.HeaderText = "№";

numberColumn.DataPropertyName = "Number";

numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

numberColumn.Width = 60;

BaseDemandGridView.Columns.Add(numberColumn);

DataGridViewColumn ClientCoordinatesColumn = new DataGridViewTextBoxColumn();

ClientCoordinatesColumn.HeaderText = "Координати покупця";

ClientCoordinatesColumn.DataPropertyName = "ClientCoordinates";

ClientCoordinatesColumn.Width = 250;

BaseDemandGridView.Columns.Add(ClientCoordinatesColumn);

DataGridViewColumn RegionColumn = new DataGridViewTextBoxColumn();

RegionColumn.HeaderText = "Район";

RegionColumn.DataPropertyName = "Region";

RegionColumn.Width = 150;

BaseDemandGridView.Columns.Add(RegionColumn);

DataGridViewColumn ApartmentAreaMinColumn = new DataGridViewTextBoxColumn();

ApartmentAreaMinColumn.HeaderText = "Площа min";

ApartmentAreaMinColumn.DataPropertyName = "ApartmentAreaMin";

ApartmentAreaMinColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

ApartmentAreaMinColumn.Width = 60;

BaseDemandGridView.Columns.Add(ApartmentAreaMinColumn);

DataGridViewColumn ApartmentAreaMaxColumn = new DataGridViewTextBoxColumn();

ApartmentAreaMaxColumn.HeaderText = "Площа max";

ApartmentAreaMaxColumn.DataPropertyName = "ApartmentAreaMax";

ApartmentAreaMaxColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

ApartmentAreaMaxColumn.Width = 60;

BaseDemandGridView.Columns.Add(ApartmentAreaMaxColumn);

DataGridViewColumn NumberRroomsColumn = new DataGridViewTextBoxColumn();

NumberRroomsColumn.HeaderText = "Кімнати";

NumberRroomsColumn.DataPropertyName = "NumberRrooms";

NumberRroomsColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

NumberRroomsColumn.Width = 50;

BaseDemandGridView.Columns.Add(NumberRroomsColumn);

DataGridViewColumn PriceMinColumn = new DataGridViewTextBoxColumn();

PriceMinColumn.HeaderText = "Вартість min";

PriceMinColumn.DataPropertyName = "PriceMin";

PriceMinColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

PriceMinColumn.Width = 80;

BaseDemandGridView.Columns.Add(PriceMinColumn);

DataGridViewColumn PriceMaxColumn = new DataGridViewTextBoxColumn();

PriceMaxColumn.HeaderText = "Вартість max";

PriceMaxColumn.DataPropertyName = "PriceMax";

PriceMaxColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

PriceMaxColumn.Width = 80;

BaseDemandGridView.Columns.Add(PriceMaxColumn);

for (int i = 0; i < BaseDemandGridView.Columns.Count; i++) {

BaseDemandGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;

}

}

private void ClearAllControls() {

RegionTBox.Text = String.Empty;

ApartmentAreaMinTBox.Text = "0";

ApartmentAreaMaxTBox.Text = "0";

NumberRroomsTBox.Text = "0";

PriceMinTBox.Text = "0";

PriceMaxTBox.Text = "0";

ClientCoordinatesTBox.Text = String.Empty;

}

private bool IsDataEnteringCorrect() {

bool isCorrect = true;

if (\_validation.IsDataEntering(RegionTBox.Text)) {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentAreaMinTBox.Text)) {

ApartmentAreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentAreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentAreaMaxTBox.Text)) {

ApartmentAreaMaxValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentAreaMaxValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(NumberRroomsTBox.Text)) {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(PriceMinTBox.Text)) {

PriceMinValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceMinValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToDouble(PriceMaxTBox.Text)) {

PriceMaxValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceMaxValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataEntering(ClientCoordinatesTBox.Text)) {

ClientCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ClientCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

private void SetDataInBaseDemandList() {

for (int i = 0; i < \_BaseDemandList.Count; i++) {

if (\_selectedBaseDemand.Number == \_BaseDemandList[i].Number) {

\_BaseDemandList[i].Region = \_selectedBaseDemand.Region;

\_BaseDemandList[i].ApartmentAreaMin = \_selectedBaseDemand.ApartmentAreaMin;

\_BaseDemandList[i].ApartmentAreaMax = \_selectedBaseDemand.ApartmentAreaMax;

\_BaseDemandList[i].NumberRrooms = \_selectedBaseDemand.NumberRrooms;

\_BaseDemandList[i].PriceMin = \_selectedBaseDemand.PriceMin;

\_BaseDemandList[i].PriceMax = \_selectedBaseDemand.PriceMax;

\_BaseDemandList[i].ClientCoordinates = \_selectedBaseDemand.ClientCoordinates;

}

}

}

private BaseDemand SelectBaseDemand(int Number, List<BaseDemand> BaseDemandList) {

for (int i = 0; i < BaseDemandList.Count; i++) {

if (Number == BaseDemandList[i].Number) {

return BaseDemandList[i];

}

}

return (new BaseDemand());

}

private void DeleteBaseDemandInfo(int Number) {

for (int i = 0; i < \_BaseDemandList.Count; i++) {

if (Number == \_BaseDemandList[i].Number) {

\_BaseDemandList.RemoveAt(i);

}

}

}

private void SetNumber() {

for (int i = 0; i < \_BaseDemandList.Count; i++) {

\_BaseDemandList[i].Number = i + 1;

}

}

}

}

Лістинг 3. Код класу «BaseDemandSerchForm»

using RealEstateOffice.AppCode;

using RealEstateOffice.BLL;

using RealEstateOffice.Providers;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RealEstateOffice.Forms {

public partial class BaseDemandSerchForm : Form {

private Validation \_validation = new Validation();

private BaseDemandProvider \_BaseDemandProvider = new BaseDemandProvider();

private List<BaseDemand> \_BaseDemandList = new List<BaseDemand>();

private SearchBLL \_SearchBLL = new SearchBLL();

public BaseDemandSerchForm() {

InitializeComponent();

}

private void SearchPriceBtn\_Click(object sender, EventArgs e) {

if (IsDataEnteringCorrect()) {

\_BaseDemandList = \_SearchBLL.GetBaseDemandByPrice(Convert.ToDouble(PriceTBox.Text));

SetNumber();

LoadDataInBaseDemandGridView(\_BaseDemandList);

}

}

private void AreaSearchBtn\_Click(object sender, EventArgs e) {

if (IsDataAreaCorrect()) {

\_BaseDemandList = \_SearchBLL.GetBaseDemandByArea(Convert.ToInt32(AreaMinTBox.Text));

SetNumber();

LoadDataInBaseDemandGridView(\_BaseDemandList);

}

}

private void RegionSearchBtn\_Click(object sender, EventArgs e) {

if (IsDataRegionorrect()) {

\_BaseDemandList = \_SearchBLL.GetBaseDemandByRegion(RegionTBox.Text);

SetNumber();

LoadDataInBaseDemandGridView(\_BaseDemandList);

}

}

private void SetNumber() {

for (int i = 0; i < \_BaseDemandList.Count; i++) {

\_BaseDemandList[i].Number = i + 1;

}

}

private void LoadDataInBaseDemandGridView(List<BaseDemand> BaseDemandList) {

BaseDemandGridView.DataSource = null;

BaseDemandGridView.Columns.Clear();

BaseDemandGridView.AutoGenerateColumns = false;

BaseDemandGridView.RowHeadersVisible = false;

BaseDemandGridView.DataSource = BaseDemandList;

DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();

numberColumn.HeaderText = "№";

numberColumn.DataPropertyName = "Number";

numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

numberColumn.Width = 60;

BaseDemandGridView.Columns.Add(numberColumn);

DataGridViewColumn ClientCoordinatesColumn = new DataGridViewTextBoxColumn();

ClientCoordinatesColumn.HeaderText = "Координати покупця";

ClientCoordinatesColumn.DataPropertyName = "ClientCoordinates";

ClientCoordinatesColumn.Width = 250;

BaseDemandGridView.Columns.Add(ClientCoordinatesColumn);

DataGridViewColumn RegionColumn = new DataGridViewTextBoxColumn();

RegionColumn.HeaderText = "Район";

RegionColumn.DataPropertyName = "Region";

RegionColumn.Width = 150;

BaseDemandGridView.Columns.Add(RegionColumn);

DataGridViewColumn ApartmentAreaMinColumn = new DataGridViewTextBoxColumn();

ApartmentAreaMinColumn.HeaderText = "Площа min";

ApartmentAreaMinColumn.DataPropertyName = "ApartmentAreaMin";

ApartmentAreaMinColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

ApartmentAreaMinColumn.Width = 60;

BaseDemandGridView.Columns.Add(ApartmentAreaMinColumn);

DataGridViewColumn ApartmentAreaMaxColumn = new DataGridViewTextBoxColumn();

ApartmentAreaMaxColumn.HeaderText = "Площа max";

ApartmentAreaMaxColumn.DataPropertyName = "ApartmentAreaMax";

ApartmentAreaMaxColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

ApartmentAreaMaxColumn.Width = 60;

BaseDemandGridView.Columns.Add(ApartmentAreaMaxColumn);

DataGridViewColumn NumberRroomsColumn = new DataGridViewTextBoxColumn();

NumberRroomsColumn.HeaderText = "Кімнати";

NumberRroomsColumn.DataPropertyName = "NumberRrooms";

NumberRroomsColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

NumberRroomsColumn.Width = 50;

BaseDemandGridView.Columns.Add(NumberRroomsColumn);

DataGridViewColumn PriceMinColumn = new DataGridViewTextBoxColumn();

PriceMinColumn.HeaderText = "Вартість min";

PriceMinColumn.DataPropertyName = "PriceMin";

PriceMinColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

PriceMinColumn.Width = 80;

BaseDemandGridView.Columns.Add(PriceMinColumn);

DataGridViewColumn PriceMaxColumn = new DataGridViewTextBoxColumn();

PriceMaxColumn.HeaderText = "Вартість max";

PriceMaxColumn.DataPropertyName = "PriceMax";

PriceMaxColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

PriceMaxColumn.Width = 80;

BaseDemandGridView.Columns.Add(PriceMaxColumn);

for (int i = 0; i < BaseDemandGridView.Columns.Count; i++) {

BaseDemandGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;

}

}

private bool IsDataEnteringCorrect() {

bool isCorrect = true;

if (\_validation.IsDataConvertToDouble(PriceTBox.Text)) {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

private bool IsDataAreaCorrect() {

bool isCorrect = true;

if (\_validation.IsDataConvertToInt(AreaMinTBox.Text)) {

AreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

AreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

private bool IsDataRegionorrect() {

bool isCorrect = true;

if (\_validation.IsDataEntering(RegionTBox.Text)) {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

}

}

Лістинг 4. Код класу «BaseOffersForm»

using RealEstateOffice.AppCode;

using RealEstateOffice.Providers;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RealEstateOffice.Forms {

public partial class BaseOffersForm : Form {

public static BaseOffers \_selectedBaseOffers = new BaseOffers();

private int \_selectedRowIndex = 0;

private Validation \_validation = new Validation();

private BaseOffersProvider \_BaseOffersProvider = new BaseOffersProvider();

public static List<BaseOffers> \_BaseOffersList = new List<BaseOffers>();

public BaseOffersForm() {

InitializeComponent();

DataLoad();

}

private void AddBtn\_Click(object sender, EventArgs e) {

if (IsDataEnteringCorrect()) {

BaseOffers BaseOffers = new BaseOffers(RegionTBox.Text, AddressTBox.Text, Convert.ToInt32(HouseNumberTBox.Text), Convert.ToInt32(ApartmentNumberTBox.Text),

Convert.ToInt32(NumberRroomsTBox.Text), ApplicantCoordinatesTBox.Text, Convert.ToInt32(ApartmentAreaTBox.Text), Convert.ToDouble(PriceTBox.Text));

\_BaseOffersList.Add(BaseOffers);

\_BaseOffersProvider.SaveBaseOffers(\_BaseOffersList);

DataLoad();

ClearAllControls();

}

}

private void ClearBtn\_Click(object sender, EventArgs e) {

ClearAllControls();

}

private void ExitBtn\_Click(object sender, EventArgs e) {

this.Close();

}

private void BaseOffersGridView\_CellClick(object sender, DataGridViewCellEventArgs e) {

if (e.RowIndex >= 0) {

\_selectedRowIndex = e.RowIndex;

\_selectedBaseOffers = SelectBaseOffers(Convert.ToInt32(BaseOffersGridView[0, e.RowIndex].Value.ToString()), \_BaseOffersList);

UpdateBaseOffersForm updateBaseOffersForm = new UpdateBaseOffersForm();

updateBaseOffersForm.ShowDialog();

if (\_selectedBaseOffers.IsDelete) {

DeleteBaseOffersInfo(\_selectedBaseOffers.Number);

} else {

SetDataInBaseOffersList();

}

\_BaseOffersProvider.SaveBaseOffers(\_BaseOffersList);

DataLoad();

}

}

private void DataLoad() {

int firstRowIndex = 0;

if (BaseOffersGridView.FirstDisplayedScrollingRowIndex > 0) {

firstRowIndex = BaseOffersGridView.FirstDisplayedScrollingRowIndex;

}

try {

\_BaseOffersList.Clear();

\_BaseOffersList = \_BaseOffersProvider.GetAllData();

SetNumber();

LoadDataInBaseOffersGridView(\_BaseOffersList);

if (\_selectedRowIndex == BaseOffersGridView.Rows.Count) {

\_selectedRowIndex = BaseOffersGridView.Rows.Count - 1;

}

if (\_selectedRowIndex >= 0) {

BaseOffersGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;

BaseOffersGridView.Rows[\_selectedRowIndex].Selected = true;

}

} catch (Exception ex) {

MessageBox.Show(ex.ToString());

}

}

private void LoadDataInBaseOffersGridView(List<BaseOffers> BaseOffersList) {

BaseOffersGridView.DataSource = null;

BaseOffersGridView.Columns.Clear();

BaseOffersGridView.AutoGenerateColumns = false;

BaseOffersGridView.RowHeadersVisible = false;

BaseOffersGridView.DataSource = BaseOffersList;

DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();

numberColumn.HeaderText = "№";

numberColumn.DataPropertyName = "Number";

numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

numberColumn.Width = 60;

BaseOffersGridView.Columns.Add(numberColumn);

DataGridViewColumn RegionColumn = new DataGridViewTextBoxColumn();

RegionColumn.HeaderText = "Район";

RegionColumn.DataPropertyName = "Region";

RegionColumn.Width = 150;

BaseOffersGridView.Columns.Add(RegionColumn);

DataGridViewColumn HouseNumberColumn = new DataGridViewTextBoxColumn();

HouseNumberColumn.HeaderText = "Номер будинку";

HouseNumberColumn.DataPropertyName = "HouseNumber";

HouseNumberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

HouseNumberColumn.Width = 100;

BaseOffersGridView.Columns.Add(HouseNumberColumn);

DataGridViewColumn ApartmentNumberColumn = new DataGridViewTextBoxColumn();

ApartmentNumberColumn.HeaderText = "Квартира";

ApartmentNumberColumn.DataPropertyName = "ApartmentNumber";

ApartmentNumberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

ApartmentNumberColumn.Width = 50;

BaseOffersGridView.Columns.Add(ApartmentNumberColumn);

DataGridViewColumn ApartmentAreaColumn = new DataGridViewTextBoxColumn();

ApartmentAreaColumn.HeaderText = "Площа";

ApartmentAreaColumn.DataPropertyName = "ApartmentArea";

ApartmentAreaColumn.Width = 50;

ApartmentAreaColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

BaseOffersGridView.Columns.Add(ApartmentAreaColumn);

DataGridViewColumn NumberRroomsColumn = new DataGridViewTextBoxColumn();

NumberRroomsColumn.HeaderText = "Кімнати";

NumberRroomsColumn.DataPropertyName = "NumberRrooms";

NumberRroomsColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

NumberRroomsColumn.Width = 50;

BaseOffersGridView.Columns.Add(NumberRroomsColumn);

DataGridViewColumn PriceColumn = new DataGridViewTextBoxColumn();

PriceColumn.HeaderText = "Вартість";

PriceColumn.DataPropertyName = "Price";

PriceColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

PriceColumn.Width = 120;

BaseOffersGridView.Columns.Add(PriceColumn);

DataGridViewColumn AddressColumn = new DataGridViewTextBoxColumn();

AddressColumn.HeaderText = "Адреса";

AddressColumn.DataPropertyName = "Address";

AddressColumn.Width = 250;

BaseOffersGridView.Columns.Add(AddressColumn);

DataGridViewColumn ApplicantCoordinatesColumn = new DataGridViewTextBoxColumn();

ApplicantCoordinatesColumn.HeaderText = "Координати заявника";

ApplicantCoordinatesColumn.DataPropertyName = "ApplicantCoordinates";

ApplicantCoordinatesColumn.Width = 250;

BaseOffersGridView.Columns.Add(ApplicantCoordinatesColumn);

for (int i = 0; i < BaseOffersGridView.Columns.Count; i++) {

BaseOffersGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;

}

}

private void ClearAllControls() {

RegionTBox.Text = String.Empty;

HouseNumberTBox.Text = "0";

ApartmentNumberTBox.Text = "0";

ApartmentAreaTBox.Text = "0";

NumberRroomsTBox.Text = "0";

PriceTBox.Text = "0";

AddressTBox.Text = String.Empty;

ApplicantCoordinatesTBox.Text = String.Empty;

}

private bool IsDataEnteringCorrect() {

bool isCorrect = true;

if (\_validation.IsDataEntering(RegionTBox.Text)) {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(HouseNumberTBox.Text)) {

HouseNumberValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

HouseNumberValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentNumberTBox.Text)) {

ApartmentNumberValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentNumberValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentAreaTBox.Text)) {

ApartmentAreaValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentAreaValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(NumberRroomsTBox.Text)) {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataEntering(AddressTBox.Text)) {

AddressValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

AddressValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataEntering(ApplicantCoordinatesTBox.Text)) {

ApplicantCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApplicantCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToDouble(PriceTBox.Text)) {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

private void SetDataInBaseOffersList() {

for (int i = 0; i < \_BaseOffersList.Count; i++) {

if (\_selectedBaseOffers.Number == \_BaseOffersList[i].Number) {

\_BaseOffersList[i].Region = \_selectedBaseOffers.Region;

\_BaseOffersList[i].Address = \_selectedBaseOffers.Address;

\_BaseOffersList[i].HouseNumber = \_selectedBaseOffers.HouseNumber;

\_BaseOffersList[i].ApartmentNumber = \_selectedBaseOffers.ApartmentNumber;

\_BaseOffersList[i].ApartmentArea = \_selectedBaseOffers.ApartmentArea;

\_BaseOffersList[i].NumberRrooms = \_selectedBaseOffers.NumberRrooms;

\_BaseOffersList[i].ApplicantCoordinates = \_selectedBaseOffers.ApplicantCoordinates;

\_BaseOffersList[i].Price = \_selectedBaseOffers.Price;

}

}

}

private BaseOffers SelectBaseOffers(int Number, List<BaseOffers> BaseOffersList) {

for (int i = 0; i < BaseOffersList.Count; i++) {

if (Number == BaseOffersList[i].Number) {

return BaseOffersList[i];

}

}

return (new BaseOffers());

}

private void DeleteBaseOffersInfo(int Number) {

for (int i = 0; i < \_BaseOffersList.Count; i++) {

if (Number == \_BaseOffersList[i].Number) {

\_BaseOffersList.RemoveAt(i);

}

}

}

private void SetNumber() {

for (int i = 0; i < \_BaseOffersList.Count; i++) {

\_BaseOffersList[i].Number = i + 1;

}

}

}

}

Лістинг 5. Код класу «BaseOffersSearchForm»

using RealEstateOffice.AppCode;

using RealEstateOffice.BLL;

using RealEstateOffice.Providers;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RealEstateOffice.Forms {

public partial class BaseOffersSearchForm : Form {

private Validation \_validation = new Validation();

private BaseOffersProvider \_BaseOffersProvider = new BaseOffersProvider();

private List<BaseOffers> \_BaseOffersList = new List<BaseOffers>();

private SearchBLL \_SearchBLL = new SearchBLL();

public BaseOffersSearchForm() {

InitializeComponent();

}

private void SearchBtn\_Click(object sender, EventArgs e) {

if (IsDataEnteringCorrect()) {

\_BaseOffersList = \_SearchBLL.GetBaseOffersByNumberRroomsAndPrice(Convert.ToInt32(NumberRroomsTBox.Text), Convert.ToDouble(PriceTBox.Text));

SetNumber();

LoadDataInBaseOffersGridView(\_BaseOffersList);

}

}

private void AreaSearchBtn\_Click(object sender, EventArgs e) {

if (IsDataAreaCorrect()) {

\_BaseOffersList = \_SearchBLL.GetBaseOffersByMinMaxArea(Convert.ToInt32(AreaMinTBox.Text), Convert.ToInt32(AreaMaxTBox.Text));

SetNumber();

LoadDataInBaseOffersGridView(\_BaseOffersList);

}

}

private void RegionSearchBtn\_Click(object sender, EventArgs e) {

if (IsDataRegionorrect()) {

\_BaseOffersList = \_SearchBLL.GetBaseOffersByRegion(RegionTBox.Text);

SetNumber();

LoadDataInBaseOffersGridView(\_BaseOffersList);

}

}

private void SetNumber() {

for (int i = 0; i < \_BaseOffersList.Count; i++) {

\_BaseOffersList[i].Number = i + 1;

}

}

private void LoadDataInBaseOffersGridView(List<BaseOffers> BaseOffersList) {

BaseOffersGridView.DataSource = null;

BaseOffersGridView.Columns.Clear();

BaseOffersGridView.AutoGenerateColumns = false;

BaseOffersGridView.RowHeadersVisible = false;

BaseOffersGridView.DataSource = BaseOffersList;

DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();

numberColumn.HeaderText = "№";

numberColumn.DataPropertyName = "Number";

numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

numberColumn.Width = 60;

BaseOffersGridView.Columns.Add(numberColumn);

DataGridViewColumn RegionColumn = new DataGridViewTextBoxColumn();

RegionColumn.HeaderText = "Район";

RegionColumn.DataPropertyName = "Region";

RegionColumn.Width = 150;

BaseOffersGridView.Columns.Add(RegionColumn);

DataGridViewColumn HouseNumberColumn = new DataGridViewTextBoxColumn();

HouseNumberColumn.HeaderText = "Номер будинку";

HouseNumberColumn.DataPropertyName = "HouseNumber";

HouseNumberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

HouseNumberColumn.Width = 100;

BaseOffersGridView.Columns.Add(HouseNumberColumn);

DataGridViewColumn ApartmentNumberColumn = new DataGridViewTextBoxColumn();

ApartmentNumberColumn.HeaderText = "Квартира";

ApartmentNumberColumn.DataPropertyName = "ApartmentNumber";

ApartmentNumberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

ApartmentNumberColumn.Width = 50;

BaseOffersGridView.Columns.Add(ApartmentNumberColumn);

DataGridViewColumn ApartmentAreaColumn = new DataGridViewTextBoxColumn();

ApartmentAreaColumn.HeaderText = "Площа";

ApartmentAreaColumn.DataPropertyName = "ApartmentArea";

ApartmentAreaColumn.Width = 50;

ApartmentAreaColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

BaseOffersGridView.Columns.Add(ApartmentAreaColumn);

DataGridViewColumn NumberRroomsColumn = new DataGridViewTextBoxColumn();

NumberRroomsColumn.HeaderText = "Кімнати";

NumberRroomsColumn.DataPropertyName = "NumberRrooms";

NumberRroomsColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

NumberRroomsColumn.Width = 50;

BaseOffersGridView.Columns.Add(NumberRroomsColumn);

DataGridViewColumn PriceColumn = new DataGridViewTextBoxColumn();

PriceColumn.HeaderText = "Вартість";

PriceColumn.DataPropertyName = "Price";

PriceColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

PriceColumn.Width = 120;

BaseOffersGridView.Columns.Add(PriceColumn);

DataGridViewColumn AddressColumn = new DataGridViewTextBoxColumn();

AddressColumn.HeaderText = "Адреса";

AddressColumn.DataPropertyName = "Address";

AddressColumn.Width = 250;

BaseOffersGridView.Columns.Add(AddressColumn);

DataGridViewColumn ApplicantCoordinatesColumn = new DataGridViewTextBoxColumn();

ApplicantCoordinatesColumn.HeaderText = "Координати заявника";

ApplicantCoordinatesColumn.DataPropertyName = "ApplicantCoordinates";

ApplicantCoordinatesColumn.Width = 250;

BaseOffersGridView.Columns.Add(ApplicantCoordinatesColumn);

for (int i = 0; i < BaseOffersGridView.Columns.Count; i++) {

BaseOffersGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;

}

}

private bool IsDataEnteringCorrect() {

bool isCorrect = true;

if (\_validation.IsDataConvertToInt(NumberRroomsTBox.Text)) {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToDouble(PriceTBox.Text)) {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

private bool IsDataAreaCorrect() {

bool isCorrect = true;

if (\_validation.IsDataConvertToInt(AreaMinTBox.Text)) {

AreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

AreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(AreaMaxTBox.Text)) {

AreaMaxValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

AreaMaxValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

private bool IsDataRegionorrect() {

bool isCorrect = true;

if (\_validation.IsDataEntering(RegionTBox.Text)) {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

}

}

Лістинг 6. Код класу «UpdateBaseDemandForm»

using RealEstateOffice.AppCode;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RealEstateOffice.Forms {

public partial class UpdateBaseDemandForm : Form {

private Validation \_validation = new Validation();

BaseDemand \_BaseDemand = new BaseDemand();

public UpdateBaseDemandForm() {

InitializeComponent();

LoadAllDate();

}

private void SaveBtn\_Click(object sender, EventArgs e) {

if (IsDataEnteringCorrect()) {

SaveData();

this.Close();

}

}

private void DeleteBtn\_Click(object sender, EventArgs e) {

if (MessageBox.Show("Ви дійсно бажаєте видалити цей елемент?", "Видалити", MessageBoxButtons.YesNo) == DialogResult.Yes) {

BaseDemandForm.\_selectedBaseDemand.IsDelete = true; ;

this.Close();

}

}

private void ExitBtn\_Click(object sender, EventArgs e) {

this.Close();

}

private void LoadAllDate() {

RegionTBox.Text = BaseDemandForm.\_selectedBaseDemand.Region;

ApartmentAreaMinTBox.Text = BaseDemandForm.\_selectedBaseDemand.ApartmentAreaMin.ToString();

ApartmentAreaMaxTBox.Text = BaseDemandForm.\_selectedBaseDemand.ApartmentAreaMax.ToString();

NumberRroomsTBox.Text = BaseDemandForm.\_selectedBaseDemand.NumberRrooms.ToString();

PriceMinTBox.Text = BaseDemandForm.\_selectedBaseDemand.PriceMin.ToString();

PriceMaxTBox.Text = BaseDemandForm.\_selectedBaseDemand.PriceMax.ToString();

ClientCoordinatesTBox.Text = BaseDemandForm.\_selectedBaseDemand.ClientCoordinates;

}

private void SaveData() {

BaseDemandForm.\_selectedBaseDemand.Region = RegionTBox.Text;

BaseDemandForm.\_selectedBaseDemand.ApartmentAreaMin = Convert.ToInt32(ApartmentAreaMinTBox.Text);

BaseDemandForm.\_selectedBaseDemand.ApartmentAreaMax = Convert.ToInt32(ApartmentAreaMaxTBox.Text);

BaseDemandForm.\_selectedBaseDemand.NumberRrooms = Convert.ToInt32(NumberRroomsTBox.Text);

BaseDemandForm.\_selectedBaseDemand.PriceMin = Convert.ToDouble(PriceMinTBox.Text);

BaseDemandForm.\_selectedBaseDemand.PriceMax = Convert.ToDouble(PriceMaxTBox.Text);

BaseDemandForm.\_selectedBaseDemand.ClientCoordinates = ClientCoordinatesTBox.Text;

}

private bool IsDataEnteringCorrect() {

bool isCorrect = true;

if (\_validation.IsDataEntering(RegionTBox.Text)) {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentAreaMinTBox.Text)) {

ApartmentAreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentAreaMinValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentAreaMaxTBox.Text)) {

ApartmentAreaMaxValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentAreaMaxValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(NumberRroomsTBox.Text)) {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(PriceMinTBox.Text)) {

PriceMinValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceMinValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToDouble(PriceMaxTBox.Text)) {

PriceMaxValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceMaxValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataEntering(ClientCoordinatesTBox.Text)) {

ClientCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ClientCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

}

}

Лістинг 7. Код класу «UpdateBaseOffersForm»

using RealEstateOffice.AppCode;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RealEstateOffice.Forms {

public partial class UpdateBaseOffersForm : Form {

private Validation \_validation = new Validation();

BaseOffers \_BaseOffers = new BaseOffers();

public UpdateBaseOffersForm() {

InitializeComponent();

LoadAllDate();

}

private void SaveBtn\_Click(object sender, EventArgs e) {

if (IsDataEnteringCorrect()) {

SaveData();

this.Close();

}

}

private void DeleteBtn\_Click(object sender, EventArgs e) {

if (MessageBox.Show("Ви дійсно бажаєте видалити цей елемент?", "Видалити", MessageBoxButtons.YesNo) == DialogResult.Yes) {

BaseOffersForm.\_selectedBaseOffers.IsDelete = true; ;

this.Close();

}

}

private void ExitBtn\_Click(object sender, EventArgs e) {

this.Close();

}

private void LoadAllDate() {

RegionTBox.Text = BaseOffersForm.\_selectedBaseOffers.Region;

HouseNumberTBox.Text = BaseOffersForm.\_selectedBaseOffers.HouseNumber.ToString();

ApartmentNumberTBox.Text = BaseOffersForm.\_selectedBaseOffers.ApartmentNumber.ToString();

ApartmentAreaTBox.Text = BaseOffersForm.\_selectedBaseOffers.ApartmentArea.ToString();

NumberRroomsTBox.Text = BaseOffersForm.\_selectedBaseOffers.NumberRrooms.ToString();

PriceTBox.Text = BaseOffersForm.\_selectedBaseOffers.Price.ToString();

AddressTBox.Text = BaseOffersForm.\_selectedBaseOffers.Address;

ApplicantCoordinatesTBox.Text = BaseOffersForm.\_selectedBaseOffers.ApplicantCoordinates;

}

private void SaveData() {

BaseOffersForm.\_selectedBaseOffers.Region = RegionTBox.Text;

BaseOffersForm.\_selectedBaseOffers.HouseNumber = Convert.ToInt32(HouseNumberTBox.Text);

BaseOffersForm.\_selectedBaseOffers.ApartmentNumber = Convert.ToInt32(ApartmentNumberTBox.Text);

BaseOffersForm.\_selectedBaseOffers.ApartmentArea = Convert.ToInt32(ApartmentAreaTBox.Text);

BaseOffersForm.\_selectedBaseOffers.NumberRrooms = Convert.ToInt32(NumberRroomsTBox.Text);

BaseOffersForm.\_selectedBaseOffers.Price = Convert.ToDouble(PriceTBox.Text);

BaseOffersForm.\_selectedBaseOffers.Address = AddressTBox.Text;

BaseOffersForm.\_selectedBaseOffers.ApplicantCoordinates = ApplicantCoordinatesTBox.Text;

}

private bool IsDataEnteringCorrect() {

bool isCorrect = true;

if (\_validation.IsDataEntering(RegionTBox.Text)) {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

RegionValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(HouseNumberTBox.Text)) {

HouseNumberValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

HouseNumberValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentNumberTBox.Text)) {

ApartmentNumberValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentNumberValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(ApartmentAreaTBox.Text)) {

ApartmentAreaValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApartmentAreaValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToInt(NumberRroomsTBox.Text)) {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

NumberRroomsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataEntering(AddressTBox.Text)) {

AddressValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

AddressValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataEntering(ApplicantCoordinatesTBox.Text)) {

ApplicantCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

ApplicantCoordinatesValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

if (\_validation.IsDataConvertToDouble(PriceTBox.Text)) {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;

} else {

PriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;

isCorrect = false;

}

return isCorrect;

}

}

}