Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РІЕЛТОРСЬКОГО АГЕНТСТВА
REAL ESTATE AGENCY SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-118
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Джибран Е.А.
(прізвище та ініціали)

Керівник Олійник А.О.
(прізвище та ініціали)

Рецензент Гофман Є.О.
(прізвище та ініціали)

2022

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
"____" _____2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Джибран Ельміри Ахмадівни
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення ріелторського агентства. Real Estate Agency Software

керівник проєкту (роботи) Олійник Андрій Олександрович, д.т.н., професор ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від " 27 " квітня 2022 року № 102
2. Строк подання студентом проєкту (роботи) 07 червня 2022 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Олійник А.О., професор		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання “ 04 ” квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Джибран Е.А.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Олійник А.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 103 с., 34 рис., 8 табл., 4 дод., 19 джерел.

АГЕНТСТВО НЕРУХОМОСТІ, БАЗА ДАНИХ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, C#, MYSQL.

Об’єкт дослідження – процес розробки програмного забезпечення для агенцій нерухомості.

Предмет дослідження – програмні засоби для ріелторських агентств.

Метою роботи є розробка програмного забезпечення для ріелторського агентства.

Матеріали, методи та технічні засоби: мова програмування C#, система керування базами даних MySQL.

Результати. Розроблено програмне забезпечення для ріелторського агентства, що дозволяє підвищити продуктивність праці працівників агентства за рахунок скорочення часу на ведення відповідної документації та оформлення купівлі-продажу нерухомості.

Практична цінність роботи полягає у розробці програмного забезпечення для ріелторського агентства.

Висновки. Розроблено програмне забезпечення для ріелторського агентства, зокрема, виконано проєктування, конструювання та тестування програми.

Галузь використання – агентства нерухомості.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 103 pages, 34 figures, 8 tables, 4 appendixes, 19 sources.

REAL ESTATE AGENCY, DATABASE, PROGRAMMING LANGUAGE, SOFTWARE, C#, MYSQL.

The object of research is the process of software development for real estate agencies.

The subject of the research is software for real estate agencies.

The purpose of this work is to develop software for a real estate agency.

Materials, methods and technical tools: C # programming language, MySQL database management system.

Results. The software tools for a real estate agency which allows to increase the productivity of the agency's employees by reducing the time for maintaining relevant documentation and registration of real estate sales have been developed.

The practical value of the work lies in the development of software for a real estate agency.

Conclusions. The software for a real estate agency is developed, the program was designed, constructed and tested.

Scope of use – real estate agencies.

ЗМІСТ

С.

Перелік скорочень та умовних познак 8

Вступ 9

1 Аналіз предметної області 10

1.1 Поняття про ріелторську діяльність та ріелторські агентства 10

1.2 Дослідження програмних засобів для ріелторських агентств 14

1.3 Висновки за розділом 1 29

2 Матеріали і методи 30

2.1 Вибір мови програмування 30

2.2 Вибір середовища розробки для створення програмного забезпечення 31

2.3 Вибір системи керування базою даних 34

2.4 Висновки за розділом 2 35

3 Опис програми 37

3.1 Структура програмного забезпечення ріелторського агентства 37

3.2 Опис програмних компонентів програмного забезпечення ріелторського агентства 38

3.3 Основні рішення з розробки бази даних 39

3.4 Висновки за розділом 3 43

4 Експлуатація, тестування та експериментальне дослідження програми 45

4.1 Призначення й умови застосування програми 45

4.2 Характеристики програми 46

4.3 Інструкція по експлуатації програми 47

4.3.1 Звернення до програми 47

4.3.2 Вхідні й вихідні дані 47

4.3.3 Повідомлення 47

4.4 Робота з програмним забезпеченням ріелторського агентства 48

4.5 Висновки за розділом 4 54

Висновки 55
Перелік джерел посилання 58
Додаток А Технічне завдання 60
Додаток Б Опис програми 64
Додаток В Текст програми 69
Додаток Д Слайди презентації 95

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CRM – Customer Relationship Manager, система управління відносинами з клієнтами нерухомості;

SQL – structured query language, мова структурованих запитів;

БД – база даних;

ПЗ – програмне забезпечення;

ПП – програмний проєкт;

ОС – операційна система;

СКБД – система керування базами даних;

ТЗ – технічне завдання.

ВСТУП

Проблема розроблення програмного забезпечення для автоматизації діяльності малих підприємств та організацій залишається досить актуальною [1]-[5]. Існуюче програмне забезпечення для агентсв нерухомсоті є досить дорогим або вузькоспеціалізованим, або містить багато функцій, що не є корисними для більшості користувачів, що ускладнює застосування таких програмних систем на практиці [1]-[5]. Тому доцільно розробити програмне забезпечення для агентства нерухомості, що усуває зазначені недоліки [1]-[5].

Таким чином, метою роботи є розробка програмного забезпечення для ріелторського агентства.

Для досягнення поставленої мети в роботі необхідно розв’язати такі задачі:

– виконати аналіз предметної області та програмних засобів для агентств нерухомості;

– здійснити проєктування програмного забезпечення ріелторського агентства;

– створити програмне забезпечення ріелторського агентства;

– виконати тестування розробленого програмного забезпечення ріелторського агентства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття про ріелторську діяльність та ріелторські агентства

Агентства з нерухомості (ріелторські агентства) – це організації, які займаються ріелторською діяльністю, тобто забезпечують підтримку операцій з нерухомістю, об’єднують покупців і продавців та виступають їх представником у переговорах. Агенти з нерухомості, як правило, завжди отримують комісію, відсоток від ціни придбання нерухомості, тому їхній дохід залежить від їх здатності до укладання угод. В деяких державах агент з нерухомості повинен працювати або бути афілійованим з брокером з нерухомості (фізичною особою або брокерською фірмою), який має більший досвід і ліцензований вищий ступінь [1]-[5].

Нерухомість може включати майно, землю, будівлі, права на повітря над землею та права на підземне покриття під землею. Термін відноситься до реального, або фізичного, майна. Як бізнес-термін, нерухомість також відноситься до виробництва, купівлі та продажу нерухомості. Це впливає на економіку будь-якої країни, оскільки є найважливішим драйвером економічного зростання [1]-[5]. Таким чином, нерухомість – це технічно земля, а також будь-які інші суттєві споруди, які можуть бути побудовані або розміщені на ній. Нерухомістю також може бути щось, вбудоване в землю, як септик. Земля з будь-якою з цих споруд називається покращеною. Без них вона була б нерозвиненою.

Важливим елементом діяльності ріелторських агентств є робота з первинною нерухомістю, тобто тією, яка будується вперше. Будівництво нових будинків включає будівництво односімейних будинків, таунхаусів та багатоквартирних будинків. Дані про продажі нових будинків є провідним економічним показником. Щоб встановити тенденцію щодо продажів нових будинків, потрібно чотири місяці.

Ріелтори допомагають приватним особам, підприємствам та інвесторам у купівлі та продажу нерухомості. Зазвичай галузь поділяється на спеціальні сфери.

Продавці або агенти по розробці списків допомагають знайти покупців за допомогою служби кількох списків або їхніх професійних контактів. Вони оцінюють нерухомість, використовуючи списки нещодавно проданих будинків, оскільки вони схожі на ті, що зараз виставляються на продаж. Вони можуть допомогти прикрасити нерухомість, щоб вона виглядала якнайкраще для потенційних покупців. Вони допомагають у переговорах з покупцем або агентом покупця, щоб отримати найвищу можливу ціну.

Агенти з покупок надають подібні послуги для покупця житла. Вони знають місцевий ринок, тому можуть знайти нерухомість, яка відповідає найважливішим критеріям покупця. Вони також порівнюють ціни, що допомагає їм направити покупця у доступні райони. Агенти покупців ведуть переговори за них, вказуючи, чому продавець повинен погодитися на нижчу ціну.

Існує чотири види нерухомості:

– житлова нерухомість включає як новобудови, так і будинки для перепродажу. Найпоширенішою категорією є будинки на одну сім’ю, але є також квартири, кооперативи, таунхауси, дуплекси, триповерхові будинки, чотириповерхові будинки, будинки для кількох поколінь та будинки для відпочинку;

– комерційна нерухомість включає торгові центри, медичні будівлі, навчальні корпуси, готелі та офіси. Багатоквартирні будинки, навіть якщо вони використовуються як житлові будинки, часто вважаються комерційними, оскільки вони можуть використовуватися для отримання доходу;

– промислова нерухомість включає виробничі будівлі та майно та складські приміщення. Такі будинки можуть використовуватися для досліджень, виробництва, зберігання та розподілу товарів. Деякі будівлі, які розповсюджують товари, вважаються комерційною нерухомістю.

Класифікація важлива, оскільки зонування, будівництво та продаж для таких об’єктів можна обробляти по-різному;

– земля включає вільні ділянки, діючі ферми та ранчо. Підкатегорії в межах вільних земель включають незабудовану, ранню забудову або повторне використання, підрозділи та збірку ділянки.

Будь-хто, хто купує або продає будинок, інвестує в нерухомість, і це означає, що потрібно враховувати різні факторів. Деякі люди так добре інвестують у власний будинок, що хочуть купувати та продавати будинки як бізнес. Багато людей мають кілька будинків і здають їх в оренду. Інші використовують програмні сервіси, на кшталт, Airbnb або подібний сервіс як зручний спосіб здати в оренду весь будинок або його частину.

Ріелтори часто спеціалізуються на комерційній або житловій нерухомості. В обох випадках вони виконують різні обов’язки залежно від того, працюють вони на покупця чи продавця. Агенти, які працюють на продавця, радять клієнтам, як визначити ціну на нерухомість та підготувати її до продажу, включаючи поради щодо покращень в останню хвилину, які можуть підвищити ціну або заохочувати швидкі пропозиції. Агенти-продавці продають нерухомість за допомогою перерахування послуг, спілкування та реклами.

Агенти, які працюють на покупця, шукають відповідні об’єкти нерухомості, які відповідають ціновому діапазону та списку бажань покупця. Ці агенти часто переглядають історичні дані про продажі порівнянної нерухомості, щоб допомогти потенційним покупцям знайти справедливу пропозицію.

Агенти виступають посередниками для ключових сторін, які передають пропозиції, зустрічні пропозиції та інші запитання туди-сюди. Після того, як пропозиція прийнята, агенти з обох сторін зазвичай продовжують працювати, допомагаючи клієнтам з оформленням документів, пересилаючи повідомлення, консультуючи щодо перегляду та переїзду, а також часто керуючи угодою до завершення.

Традиційно агенту виплачується комісія, яка становить відсоток від ціни продажу нерухомості. Чим більше продається будинок, тим більше грошей заробляє агент. Однак традиційна структура оплати змінюється, оскільки онлайн-оголошення за допомогою програмних засобів дозволяють споживачам робити більшість покупок самостійно без допомоги посередника.

Деякі брокери стягують нижчу комісію за дорожчі будинки, а деякі виконують всю транзакцію за фіксовану плату, яка набагато нижча, ніж звичайна комісія. Інші компанії пропонують структуру ціноутворення з оплатою за послугу, яка дозволяє продавцям платити лише за певні частини процесу продажу, наприклад, додавання об’єкта до служби кількох списків.

Будь-кого, хто має базову ліцензію на роботу з нерухомістю, що може передбачати проходження певної кількості акредитованих курсів та складання іспиту, можна назвати агентом з нерухомості. Агент з нерухомості – це, по суті, продавець, кваліфікований, щоб допомогти споживачам купити або продати нерухомість. Брокер з нерухомості має додаткову освіту та підготовку. Брокери займаються технічними аспектами операцій з нерухомістю. Брокери зазвичай володіють фірмою або франшизою. Вони можуть бути індивідуальними практиками, але повинні отримати іншу вищу ліцензію, якщо хочуть найняти агентів або інших брокерів для роботи під ними. Як згадувалося раніше, агент з нерухомості, як правило, не може працювати самостійно, а повинен працювати через посередника з нерухомості.

Отже, ріелтор – це особа або підприємство, яке регулює продаж, оренду чи управління нерухомістю та іншими будівлями. Функціями ріелторських агентств є такі:

– допомога у здавачі в оренду та продажу різних активів (земля, будівлі, обладнання, підприємства);

– підготовка та виконання дослідження ринку нерухомості;

– техніко-економічне обґрунтування нерухомості;

– підготовка та подання документів про передачу права власності на нерухомість, полегшення сплати відповідних податків, сприяння видачі дозволів тощо;

– допомога у реєстрації прав на земельні ділянки;

– підготовка документів щодо ліквідації та банкрутства підприємств.

У наш час активно використовуються онлайн-сервіси для подання оголошень щодо купівлі-продажу нерухомості, де власники нерухомості мають можливість виставляти відповідні оголошення. Проте такі оголошення розташовуються на таких сайтах не більше визначеної кількості діб, після чого вони автоматично видаляються. Часто власники нерухомості не мають можливості постійно займатися процесом продажу і звертаються до ріелторських агентств. Тому такі агентства мають власні бази даних нерухомості, що продається. Отже, таку інформацію необхідно систематизовувати та обробляти, що обумовлює необхідність розробки відповідного програмного забезпечення для ріелторських агентств.

Тому дуже важливо скоротити час пошуку наявної інформації, розробивши програмні засоби агентства нерухомості, що дозволять не тільки заощадити час пошуку інформації, а і підвищити ефективність діяльності підприємства в цілому.

1.2 Дослідження програмних засобів для ріелторських агентств

У наш час активно використовуються програмні засоби для підтримки діяльності ріелторських агентств [1]-[6]. Проаналізуємо деякі з них.

Програмне забезпечення Zillow Premier Agent – це досить надійне програмне забезпечення для агентів нерухомості, яке використовує можливості понад 160 мільйонів користувачів Zillow щомісяця. На платформі Zillow можна надсилати нові списки та заявляти списки, які були завантажені в Zillow із вашого сервісу кількох списків. Програмне забезпечення Zillow Premier Agent розміщує контактну інформацію агента разом зі списками

нерухомості [7]. За додаткову плату Zillow Premier Agents також може розміщувати рекламу в списках інших агентів, вибирати, в яких поштових індексах агент хоче розміщувати рекламу, і встановлювати власний бюджет. Zillow Premier Agent страждає від поганих відгуків від некоректних агентів і клієнтів, зосередженого на продажу, а не на вирішенні проблем [7].

Zillow Premier Agent дозволяє агентам, командам і брокерам купувати медійну рекламу в мережі Zillow. В такому випадку оголошення відображатимуться потенційним клієнтам за певними поштовими індексами.

Коли відвідувачі вебсайту Zillow мають запитання щодо власності, потенційним клієнтам пропонуються агенти Zillow Premier як перший варіант. Платформа показує оголошення Zillow Premier Agent на ключових позиціях поруч із нерухомістю на продаж, а також підвищує позицію Zillow Premier Agents у верхній частині їх каталогу агентів [7].

Ціна Zillow залежить від багатьох факторів, таких як типова ціна продажу будинку, кількість фахівців з нерухомості, які змагаються за потенційних клієнтів за певним поштовим індексом, і відсоток доступних потенційних клієнтів, які агент хоче охопити. Оскільки ці фактори змінюються залежно від агента, зробити точні оцінки вартості досить важко. Zillow постійно пропонує оцінки в реальному часі потенційним агентам Zillow Premier. Zillow співпрацює з низкою брендів в індустрії нерухомості [7].

Типова ціна на місяць для агентів, які отримують прибуток від Zillow Premier Agent, починається приблизно з 1000 доларів США на місяць. За межами великих міст типова ціна на місяць для агентів, які отримують прибуток від Zillow Premier Agent, становить від 300 до 500 доларів США на місяць [7].

Програмне забезпечення Zillow Premier Agent має наступні особливості та переваги [7]:

– можливість пошуку нерухомості за критеріями;
– є пошук з використанням карти;
– можливість залишати відгуки про роботи агентів з нерухомості;

– проста навігація та зручний інтерфейс (рис. 1.1).

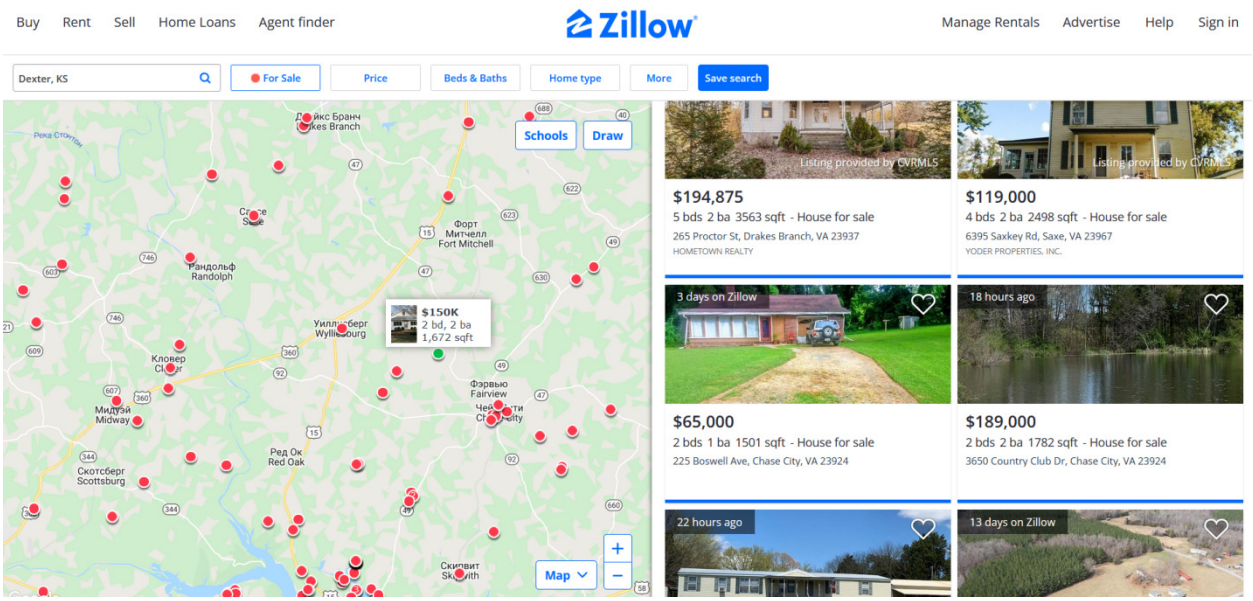


Рисунок 1.1 – Інтерфейс програмного забезпечення Zillow Premier Agent

Недоліком програмного забезпечення Zillow Premier Agent є те, що воно не є безкоштовним. Крім того, вартість цієї програми не є прозорою та встановлюється компанією з не зовсім зрозумілих критеріїв [7]. Іншим недоліком є те, що списки нерухомості є загальнодоступними, таким чином агенти з нерухомості не можуть вводити інформацію про нерухомість, яка доступна тільки у них та яку доцільно пропонувати тільки власним клієнтам [7].

Програмне забезпечення BoldLeads. В індустрії нерухомості є фундаментальна проблема: більшість агентів з нерухомості не є маркетологами. Вони не приходять на роботу з будь-якими навичками залучення потенційних клієнтів, але для успіху їм потрібен маркетинг і генерація потенційних клієнтів. BoldLeads прагне вирішити цю проблему, надаючи платформу для генерації потенційних клієнтів у сфері нерухомості, яка генерує потенційних клієнтів і допомагає вам перетворити їх у клієнтів [8].

BoldLeads генерує потенційних клієнтів у сфері нерухомості, використовуючи підхід на основі рекламного агентства. Працюючи від імені

ріелторського агентства, BoldLeads розміщує рекламу в таких місцях, як Facebook і Google, пропонуючи те, що хочуть потенційні клієнти – як правило, пропозицію безкоштовної оцінки житла для продавців або спеціалізований список будинків на продаж для покупців [8].

Коли потенційний клієнт натискає оголошення, він потрапляє на цільову сторінку, де його просять надати інформацію про потенційну нерухомість в обмін на те, що вони шукали. Після того, як вони нададуть свою контактну інформацію, вони з'являться у менеджера зі зв'язків із клієнтами BoldLead як новий контакт, внаслідок чого можна зв'язатися з ними вручну або скористатись автоматичними або консьєржними опціями [8].

BoldLeads пропонує кілька варіантів цін для агентів, команд і брокерських компаній. Більшість агентів BoldLeads вибирають пакет Professional, який розбиває вартість на дві позиції: комісія за обліковий запис, яка починається від 399 доларів США на місяць, і рекламний бюджет з мінімальними витратами 250 доларів США на місяць [8].

Після реєстрації агент з нерухомості для генерації потенційних клієнтів обирає певний географічний регіон або територію, на яких потрібно зосередитися. Крім того, враховуються території, що межують з ключовою, що означає, що населений район міста буде коштувати так само, як і сільський, малонаселений. Це створює рівні умови, на відміну від таких інструментів, як Zillow, які стягують більше для більш населених або дорогих районів. Чим більш населена область, тим більше агентів BoldLeads дозволить там рекламувати, але вони ніколи не перенасичують ринок до такої міри, що агент не отримаєте очікуваного обсягу доступу й потенційних клієнтів [8].

Після того, як було обрано регіон, звідки необхідно залучати потенційних клієнтів, наступним кроком є налаштування оголошень. На наступного етапі відбувається збір потенційних клієнтів. Тут агенту нічого не потрібно зробити, оскільки дія відбувається на стороні споживача: оголошення відображаються у Facebook, Instagram та Google перед потенційними клієнтами. Оголошення пропонують щось цінне – як правило,

безкоштовну оцінку житла або список будинків на продаж у цьому районі – щось настільки привабливе, що потенційний клієнт може вжити заходів і натиснути. Після натискання оголошення потенційний клієнт потрапляє на цільову сторінку, де йому пропонують обіцяну цінність в обмін на інформацію про себе (ім'я, номер телефону, електронну адресу, домашню адресу). Таким чином, цей крок простий у часі агента [8].

Після того, як інформація про потенційного клієнта буде надіслана до BoldLeads CRM, потенційний клієнт запускає автоматизовану серію комунікацій, щоб виконати його початковий запит. Це електронні листи, текстові повідомлення, навіть попередньо записані повідомлення голосової пошти призначені для однієї речі: розпочати розмову [8].

Перевагами програмного забезпечення BoldLeads є такі [8]:

– наявність засобів маркетингового залучення клієнтів;
– можливість роботи з інформацією про об'єкти нерухомості.

Недоліками програмного забезпечення BoldLeads є такі [8]:

– інтерфейс не дуже зручний;
– містить занадто багато надлишкової інформації.
– відсутність пошуку з використанням карти;
– відсутність можливості залишати відгуки про роботи агентів з нерухомості.

Програмне забезпечення LionDesk – це CRM система для нерухомості, яка є досить доступною і пропонує набір корисних ріелтору функцій. Це інструмент, який допомагає агентам укладати велику кількість угод з нерухомістю [9].

LionDesk CRM коштує приблизно 21 долар на місяць. Мета LionDesk – надати агентам, командам і брокерським компаніям один простий інструмент для організації, генерації потенційних клієнтів і залучення потенційних клієнтів (рис. 1.2) [9].

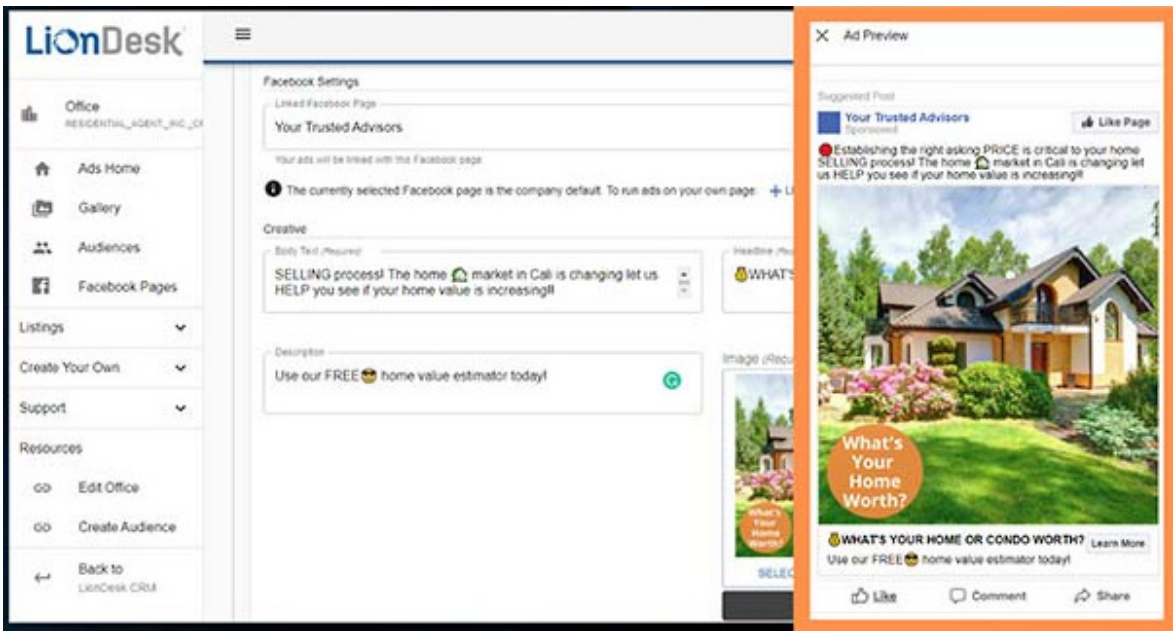


Рисунок 1.2 – Інтерфейс програмного забезпечення LionDesk [9]

LionDesk підходить для [9]:

– будь-якого агента, якому потрібно організувати контакти та підтримувати потенційних клієнтів;

– новим агентам, які будують свій бізнес;

– досвідченим агентам, які хочуть виділитися за допомогою таких технологій, як чат-боти зі штучним інтелектом та реклама у Facebook;

– невеликим брокерським компаніям та командам, які працюють та хочуть покращити свої показники.

– агентам, яким потрібна зіркова підтримка клієнтів;

– агентам, яким потрібна універсальна підписка, яка включає створення вебсайтів, цільові сторінки, швидкі дозвони та рекламу в соціальних мережах;

– брокерським агенціям, які повинні вийти на високий рівень з різними видами додаткових послуг [9].

Перевагами програмного забезпечення LionDesk є такі [9]:

– простота використання. Інтуїтивно зрозумілий інтерфейс LionDesk забезпечує доступний процес навчання, і, як наслідок, майже миттєве використання програми. Інтерфейс також містить вікно, яке пропонує короткий огляд основної роботи: поточний підрахунок вихідних і вхідних

дзвінків, текстових та електронних листів. Інформаційна панель Крім того, програма забезпечує те, що інформаційна панель легко налаштовується. Інтерфейс дуже простий, без відволікаючих кольорів і графіків, що дозволяє зосередитися на найважливішій інформації та завданнях [9];

– наявність інструментів для організації ріелторського бізнесу. На вкладці Контакти, можна швидко додати або відредагувати поточний контакт, масово завантажити контакти з іншої платформи за допомогою файлу CSV або безпосередньо імпортувати свої контакти з таких платформ, як Gmail [9]. На вкладці контакти можна отримати велику кількість інформації, зокрема електронні листи, попередні дзвінки чи текстові повідомлення, а також біографічні дані або дані про угоду, потенційних клієнтів. Існує можливість додати спеціальні поля для своїх контактів, наприклад, скільки у них дітей, ім’я собаки, улюблену команду або кредитний рейтинг [9];

– інструменти для роботи з клієнтами: масові електронні листи, текстові повідомлення, дзвінки та відео. У програмі LionDesk є можливість зателефонувати, надіслати повідомлення або надіслати електронний лист окремим контактам, або відправити масові електронні листи, тексти та відео, залежно від стратегії кампанії. Є можливість встановити правила для потенційних клієнтів з різних джерел. Наприклад, можна налаштувати LionDesk так, щоб він ініціював конкретне текстове повідомлення або відео-повідомлення та надсилав його потенційним клієнтам з реклами Facebook. Однією з корисних функцій є можливість записувати й надсилати відео безпосередньо з платформи та надсилати його вашим потенційним клієнтам електронною поштою або текстовими повідомленнями [9].

Недоліками програмного забезпечення LionDesk є такі [9]:

– містить занадто багато надлишкової інформації.

– відсутність пошуку з використанням карти;

– відсутність можливості залишати відгуки про роботи агентів з нерухомості;

– не є безкоштовним [9].

Програмне забезпечення Top Producer є програмним забезпеченням для підтримки діяльності з купівлі-продажу нерухомості [10].

Top Producer працює, щоб не тільки не відставати від мінливих потреб професіоналів у сфері нерухомості, але й передбачити наш наступний набір потреб. Їхній далекоглядний підхід проявляється в останній ітерації CRM: Top Producer (рис. 1.3) [10].

Рисунок 1.3 – Програмне забезпечення Top Producer [10]

Перевагами програмного забезпечення Top Producer є такі [10]:

– одна з найбільш інтегрованих CRM на ринку, яка може підключатися різних сторонніх платформ;

– функції керування транзакціями, які позбавлять від необхідності використання іншого інструменту;

– автоматизовані можливості зв’язку, які інтегруються з даними списку клієнтів, що робить виховання цього програмного забезпечення легшим та ефективнішим [10].

Недоліками програмного забезпечення Top Producer є такі [10]:

– мінімальний термін договору на використання програми складає 12 місяців, за кожен з яких необхідно вносити оплату;

– відсутність підтримки вебсайту ріелторського агентства в жодному з базових планів [10].

Top Producer коштує в середньому від 33 до 80 доларів на місяць. Витрати залежать від обраного тарифного плану. Інструмент реагування та керування потенційними клієнтами у Top Producer пропонує відстеження наступного рівня для агентів, які хочуть знати, звідки надходять потенційні клієнти, призначати їм спеціальні шляхи, щоб дістатися до потрібного агента (або розставляти пріоритети в правильному порядку) [10].

Інструмент аналітики у програмному забезпеченні Top Producer дозволяє підвищити ефективність потоку генерації потенційних клієнтів, над створенням якого агент з нерухомості працює, переконавшись, що правильний потенційний клієнт потрапляє до потрібної особи в потрібний час [10].

Програмне забезпечення Propertybase – це інструмент управління відносинами з клієнтами нерухомості (Customer Relationship Manager, CRM), який пропонує агентам з нерухомості основу для відстеження їх потенційних клієнтів. Він також пропонує керування транзакціями, маркетингові можливості та підключення до інших служб, таких як ваш особистий вебсайт, генерація потенційних клієнтів тощо [11].

Початкова місячна ціна для Propertybase становить 399 доларів США на місяць із 9-місячними зобов’язаннями [11].

Propertybase – це продукт, який підходить для окремих агентів, команд, а в деяких випадках і брокерських компаній (рис. 1.4). Propertybase має багато функцій, які в сукупності роблять цю програму однією з найпотужніших CRM-платформ на ринку нерухомості. Інформаційна панель Propertybase є місцем, де знаходяться події та завдання, які необхідно виконати за день, активні транзакції, а також вся необхідна інформація, яка потрібна для легкого доступу [11].

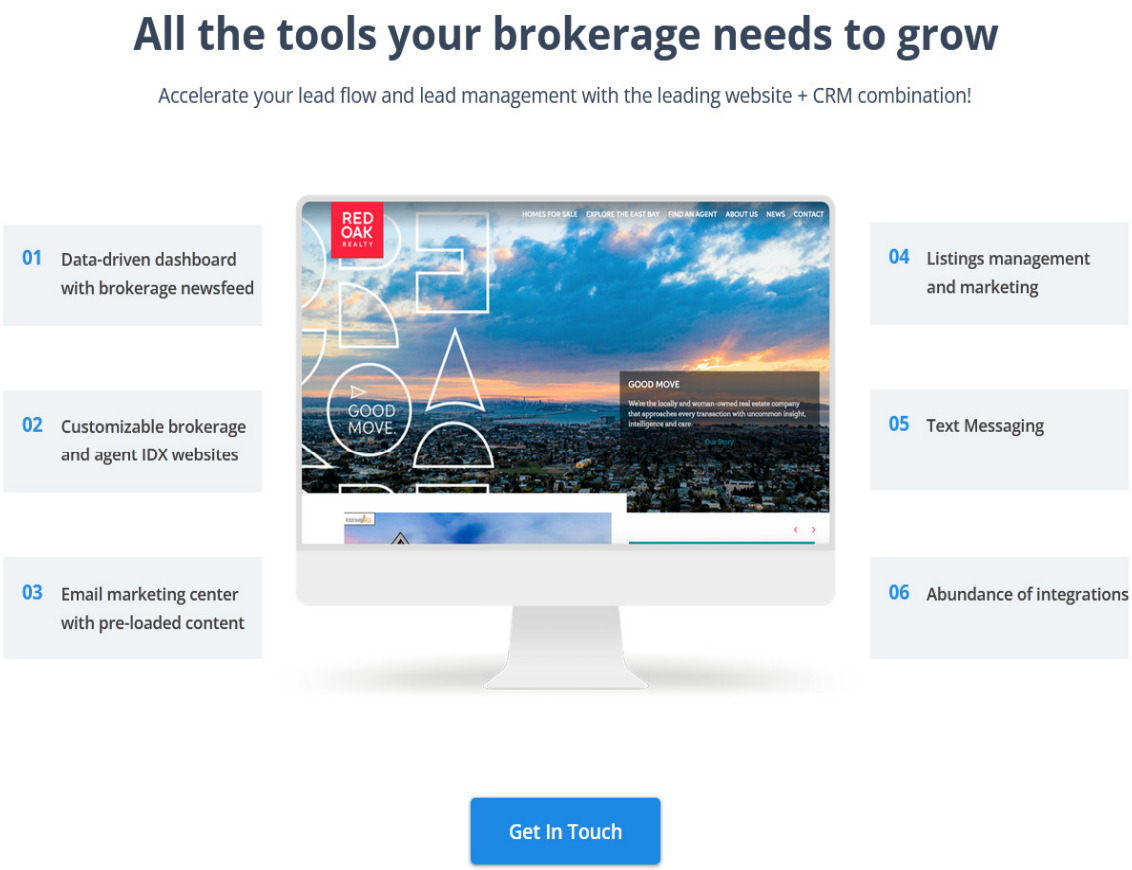


Рисунок 1.4 – Програмне забезпечення Propertybase [11]

Панель інструментів Propertybase містить важливі контакти, інформацію про поточну діяльність та іншу інформацію. Панель інструментів програми Propertybase містить найнеобхідніше і не містить зайвої інформації. Проте така панель не передбачає можливості її налаштування під конкретного користувача, що ускладнює застосування цього програмного забезпечення. Програмне забезпечпення Propertybase має ефективні класифікації, керування та визначення пріоритетів контактів користувача. Propertybase підтримує детальні профілі потенційних клієнтів, які дозволяють збирати важливу контактну інформацію, необхідну для просування потенційних клієнтів до рівня справжніх клієнтів. Усі дії потенційного клієнта автоматично заносяться в профіль клієнта, включаючи видимі дані, як-от збережені пошукові запити та вибрані властивості, а також невидимі споживачу, як-от властивості, які вони переглядають, критерії, якими вони користуються для пошуку, будь-які кліки, які вони здійснюють на навігаційній панелі інструментів, тощо [11].

Маркетингові функції Propertybase GO дозволяють зв'язуватися з контактами за допомогою електронної пошти, текстових повідомлень або телефону. Оскільки CRM під'єднано до списку клієнтів, вона також дозволяє шукати, посилатися та надсилати певні властивості потрібним людям у потрібний час. Загальна функціональність маркетингу електронної пошти також дуже суттєва, зокрема з точки зору кількості попередньо написаних електронних листів. Програма пропонує різні попередньо написані потоки, які можна використовувати для виконання різних функцій: від вітання нових потенційних клієнтів, розсилання оголошень про зниження ціни або навіть підтримання зв'язку з клієнтами після того, як продажі були закриті [11].

Програмне забезпечення Real Geeks – це програмне забезпечення у вигляді вебсайту, CRM та платформа для генерації потенційних клієнтів, яка прагне допомогти агентам з нерухомості. Замість того, щоб сидіти склавши руки і сподіватися, що потенційні клієнти стають реальними клієнтами, Real Geeks надає агенту набір інструментів, які йому потрібні, щоб підштовхувати покупців і продавців на кожному кроці шляху з купівлі-продажу нерухомості [12].

Real Geeks є комплексним рішенням для професіоналів у сфері нерухомості, коли справа доходить до управління всім ріелторським бізнесом. Набір пропозицій Real Geeks включає вебсайти з високою конверсією, інструмент управління відносинами з клієнтами (CRM), який інтегрується з їхнім вебсайтом для відстеження активності потенційних клієнтів, маркетингові інструменти, що заощаджують час, і стратегії генерації потенційних клієнтів (рис. 1.5) [12].

Вебсайти, розроблені за допомогою Real Geeks для перетворення, розміщення функціональних можливостей пошуку нерухомості, дозволяють агентам з нерухомості не тільки спілкуватися з людьми, яких вони знають, але й автоматично відстежують активність нових потенційних клієнтів на вебсайті ріелтора, полегшуючи їх перетворення з потенційних клієнтів у реальних [12].

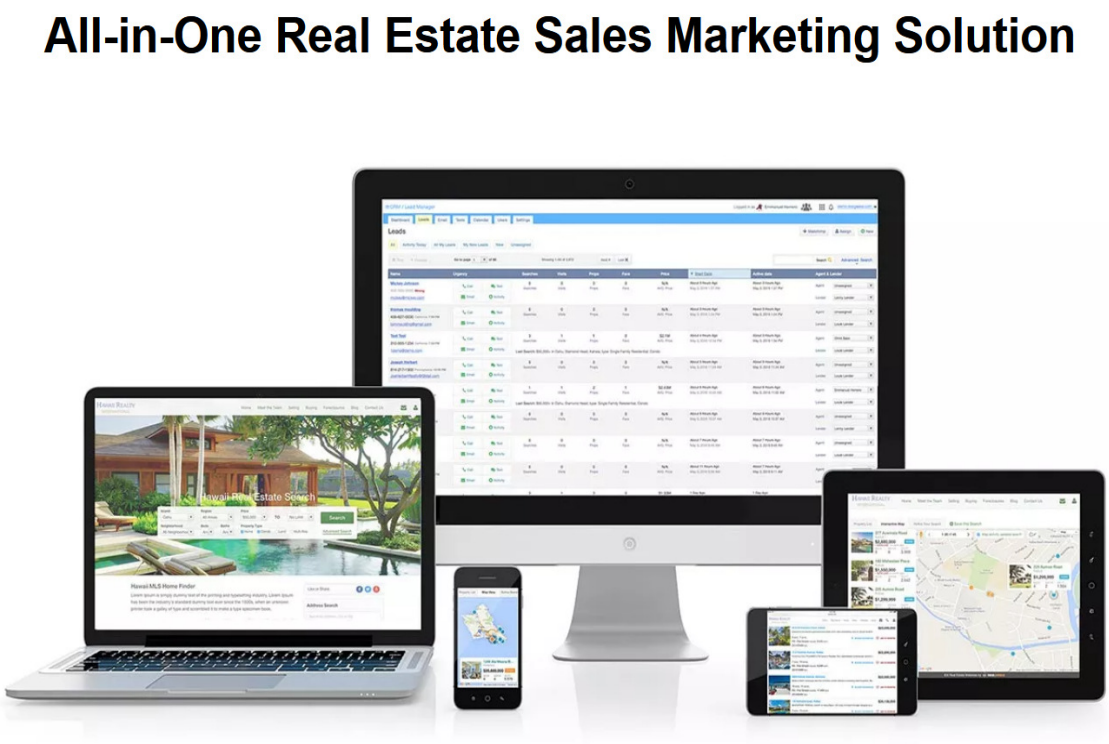


Рисунок 1.5 – Програмне забезпечення Real Geeks [12]

Ціна Real Geeks починається від 299 доларів на місяць для одного або двох користувачів плюс 25 доларів на місяць для кожного додаткового користувача. За цю ціну клієнт отримує доступ до вебсайту з підтримкою, Real Geeks CRM та бібліотеку інструкцій та посібників зі стратегії для ефективного використання інструментів [12].

Однією з переваг моделі ціноутворення Real Geeks є доступність додаткових послуг, які можна додавати в міру зростання ріелторького бізнесу, що дозволяє починати з малого та додавати функції в міру розширення діяльності агентства нерухомості [12].

На вебсайті агентства нерухомості, створеному у Real Geeks, можна спрямувати трафік за допомогою інших інструментів Real Geeks, таких як Real Leads і Facebook Marketing Platform, і зрештою, саме тут цей трафік перетворюється на потенційних клієнтів. Вебсайти Real Geeks мають усі очікувані функції, включаючи інструмент для ведення блогів, обмежені можливості налаштування та різних закликів до дії, щоб перетворити трафік сайту у потенційних клієнтів. У Real Geeks забезпечена динамічна

функціональність відображення, можливість клієнтам і потенційним клієнтам плавно переходити з настільного комп'ютера до налаштованого мобільного застосунку, і, звичайно, прямий зв'язок між вебсайтом і CRM, передаючи дії потенційних клієнтів [12].

Real Geeks також пропонує дійсно надійну пошукову оптимізацію, різні варіанти тестування для цільових сторінок, простий у використанні внутрішній інтерфейс та повністю інтегрований сайт [12].

Вебсайт Real Geeks – це місце, де потенційні клієнти взаємодіють з екосистемою Real Geeks. При використанні Real Geeks агенти з нерухомості отримують стандартну функціональність CRM, як-от користувацькі послідовності введення електронної пошти, вичерпні профілі потенційних клієнтів і параметри масової комунікації. Програмне забезпечення Real Geeks також має вбудований інтерфейс для створення повідомлень, який дозволяє спілкуватися за допомогою тексту прямо зі свого робочого столу та планувати вихід повідомлень у будь-який час. Частота відкриття та читання текстових повідомлень може бути дуже високою, що робить цю стратегію потужною, особливо для першого контакту з новими потенційними клієнтами з вебсайту агенції з нерухомості. Активність потенційних клієнтів можна використовувати для планування автоматизованих потоків зв'язку на основі того, як часто відвідувачі вебсайту переглядають певні об'єкти, зберігають пошукові запити або вибирають будинки для продажу [12].

У програмному забезпеченні Real Geeks існує базова формула генерації потенційних клієнтів у сфері нерухомості, яку використовує майже кожна стратегія: залучення потенційних клієнтів чимось цінним, і перш ніж фактично доставити це їм, програма вимагає їх контактну інформацію. Коли справа доходить до потенційних продавців житла, вони не хочуть нічого іншого, ніж оцінити вартість свого будинку. Real Geeks реалізує бажання потенційних клієнтів за допомогою настроюваних необмежених цільових сторінок оцінки житла, пропонуючи потенційним продавцям житла оцінити

вартість їхнього будинку та надаючи інформацію, необхідну для початку діалогу [12].

Користувачі Real Geeks можуть створити цільову сторінку оцінки житла для кожної великої області свого ринку, дозволяючи рекламі, яку вони створюють за допомогою служби Real Geeks або власної стратегії, бути більш націленим і, отже, отримувати більший притік клієнтів [12].

Платформа Real Geeks дає можливість використовувати потужність пошукової системи Google для відображення в релевантних результатах пошуку кожного разу, коли один із потенційних клієнтів шукає інформацію про нерухомість [12].

Перевагами програмного забезпечення Real Geeks є такі [12]:

– одна з найбільш інтегрованих CRM на ринку, яка може підключатися різних сторонніх платформ;

– функції керування транзакціями, які позбавлять від необхідності використання іншого інструменту;

– автоматизовані можливості зв’язку, які інтегруються з даними списку клієнтів, що робить виховання цього програмного забезпечення легшим та ефективнішим [12].

Недоліком програмного забезпечення Real Geeks є відсутність підтримки вебсайту ріелторського агентства в жодному з базових планів [12].

Таким чином, у наш час широко представлені програмні засоби для підтримки діяльності ріелторських агентств. Порівняльний аналіз програмного забезпечення для підтримки діяльності ріелторських агентств наведено у табл. 1.1.

З приведеної таблиці видно, що існуюче програмне забезпечення для підтримки діяльності ріелторських агентств надають різні можливості співробітникам таких агентств. Проте такі програми, як правило, є досить складними для використання, а також не підтримують можливості ведення прихованого списку нерухомості (списки нерухомості, з якою працює агентство, як правило, є загальнодоступним у наявному ПЗ), таким чином

агенти з нерухомості не можуть вводити інформацію про нерухомість, яка доступна тільки у них та яку доцільно пропонувати тільки власним клієнтам. Отже, інформацію про нерухомість, яка не повинна бути загально доступною, та є такою, що повинна пропонуватися тільки унікальним клієнтам, необхідно систематизовувати та обробляти окремо, у власних програмах, що обумовлює необхідність розробки відповідного програмного забезпечення для ріелторських агентств. Крім того, використання відомого програмного забезпечення вимагає обов'язкового внесення щомісячних платежів, що також ускладнює його використання.

Таблиця 1.1 – Програмне забезпечення для підтримки діяльності ріелторських агентств

Критерій, за яким порівнюють	Zillow Premier Agent	BoldLeads	LionDesk	Top Producer	Propertybase
Можливість пошуку нерухомості на карті	+	–	–	–	–
Підтримка списку потенційних клієнтів	+	+	+	–	+
Маркетингові засоби для залучення клієнтів	–	+	–	+	+
Мобільний застосунок	+	–	+	–	–
Можливість ведення прихованого списку нерухомості	–	–	–	–	–
Безкоштовність	–	–	–	–	–

1.3 Висновки за розділом 1

Проаналізовано поняття про ріелторську діяльність та ріелторські агентства. Визначено, що ріелторські агентства – це організації, які займаються ріелторською діяльністю, тобто забезпечують підтримку операцій з нерухомістю, об’єднують покупців і продавців та виступають їх представником у переговорах. Відзначено, що часто власники нерухомості не мають можливості постійно займатися процесом продажу і звертаються до ріелторських агентств. Тому такі агентства мають власні бази даних нерухомості, що продається. Отже, таку інформацію необхідно систематизовувати та обробляти, що обумовлює необхідність розробки відповідного програмного забезпечення для ріелторських агентств.

Досліджено та проаналізовано програмні засоби для ріелторських агентств. Визначено, що існуюче програмне забезпечення для підтримки діяльності ріелторських агентств надають різні можливості співробітникам таких агентств. Проте такі програми, як правило, є досить складними для використання, а також не підтримують можливості ведення прихованого списку нерухомості (списки нерухомості, з якою працює агентство, як правило, є загальнодоступним у наявному ПЗ), таким чином агенти з нерухомості не можуть вводити інформацію про нерухомість, яка доступна тільки у них та яку доцільно пропонувати тільки власним клієнтам. Отже, інформацію про нерухомість, яка не повинна бути загально доступною, та є такою, що повинна пропонуватися тільки унікальним клієнтам, необхідно систематизовувати та обробляти окремо, у власних програмах, що обумовлює необхідність розробки відповідного програмного забезпечення для ріелторських агентств. Крім того, використання відомого програмного забезпечення вимагає обов’язкового внесення щомісячних платежів, що також ускладнює його використання.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

Для створення програмного забезпечення ріелторського агентства було обрано мову програмування C# [13]-[14].

Мова програмування C# є логічним подальшим розвитком C++ від Microsoft. Її структура та синтаксис дуже схожі на Java, але має мало спільного з Java. За допомогою C# можна писати програми на основі платформи .NET, а також ігри для ПК, XBox та інше [13]-[15].

Як і Java, ця мова є об'єктно-орієнтованою. Можливості використання програм C# на інших платформах були обмежені до 2014 року. Тепер існує незалежний від платформи варіант завдяки .NET Core, який дає змогу використовувати програми C# на кількох платформах. Таким чином, програмісти на C# не обмежені лише Windows, і не потрібно більше використовувати спеціальний фреймворк, як-от mono [13]-[15].

Якщо програміст володіє іншою мовою програмування, вивчити C# дуже легко завдяки багатьом знайомим елементам. Незалежно від того, чи пов’язано це з особистими інтересами чи пов’язано з професійною метою – вивчаючи мову програмування C# досить легко, крім того існує велика кількість спеціалізованих онлайн-спільнот, де можна обговорити та знайти відповідь на питання стосовно розробки на C# [13]-[15].

Платформа .NET являє собою платформу розробки від Microsoft, яка постійно розширюється, щоб включати майбутні функції, для розробки програмного забезпечення в середовищі Windows та для створення вебзастосунків. Платформа .NET складається з середовища виконання, яке має бути встановлено на комп’ютері (як правило, Windows або альтернатива Mono в Linux). Існують також бібліотеки або так звані простори імен для локального та віддаленого доступу до баз даних (ADO.NET, Entity Framework, WCF), а також для часто використовуваних прикладних сценаріїв, таких як запис і читання файлів XML, глобалізація, математичні операції тощо [13]-[15].

Оскільки ця мова програмування є об’єктно-орієнтованою, її можна використовувати для створення настільних програм Windows, вебпрограм ASP.NET або програм для iOS та Android, та ін. [13]-[15].

Обгрунтування вибору мови програмування для розробки програмного забезпечення ріелторського агентства наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення ріелторського агентства

Критерій порівняння мов програмування	Мова програмування		
	C#	C++	Java
Обробка баз даних	+	+–	+
Засоби створення інтерфейсу	+	+–	+
Забезпечення безпеки даних	+	+–	+
Зручність для створення програмного забезпечення ріелторського агентства	+	–	+–
Обробка винятків	+	–	+

Отже, для створення програмного забезпечення ріелторського агентства обрано мову програмування C#.

2.2 Вибір середовища розробки для створення програмного забезпечення

При розробці програмного забезпечення ріелторського агентства було прийнято рішення використовувати середовище розробки Microsoft Visual Studio [15]-[17].

Середовище розробки Microsoft Visual Studio дозволяє створювати різні програмі системи у вигляді клієнтських застосунків. Microsoft Visual Studio є

програмним продуктом з простою у використанні та ефективною системою розробки програмного забезпечення для платформи .NET із різних середовищ розробки [15]-[17].

Microsoft Visual Studio – це набір продуктів Microsoft, що включає інтегроване середовище розробки програмного забезпечення та багато інших інструментів. Microsoft Visual Studio є однією з досить ефективних програм на сьогоднішній день і дозволяє створювати різноманітне програмне забезпечення на різних мовах програмування, зокрема, і на мові C# [15]-[17].

Середовище розробки Microsoft Visual Studio дозволяє обмежене використання покажчиків, що розробники часто вважають небезпечним. У цьому випадку метод полягає в запиті блоків коду або небезпечних ключових слів для методів, які використовують цю функцію. Це ключове слово застерігає користувачів від таких кодів. Для цього потрібно конкретне завдання від компілятора параметрів за замовчуванням. Такий небезпечний код використовується для покращення можливості роботи з некерованими API, а іноді і для підвищення продуктивності окремих частин коду [15]-[17].

На відміну від екземплярів класів, екземпляри типу значення створюються не в стеку, а в стеку викликів або як частина екземпляра оголошеного об'єкта, це покращує продуктивність коду в деяких випадках. З точки зору програміста, вони схожі на класи, але з низкою обмежень: вони не можуть мати конструктор без параметрів (але вони можуть мати конструктор з параметрами), вони не можуть бути успадковані та не можуть бути успадковані від інших класи. (але інтерфейси можуть бути реалізовані) [15]-[17].

Методи за замовчуванням у Microsoft Visual Studio не є віртуальними і повинні бути оголошені явно за допомогою ключового слова virtual. Віртуальність забезпечує необхідне перекриття викликаного методу, але необхідні додаткові виклики до таблиці віртуальних методів [15]-[17].

Таким чином, типи проєктів, які підтримує Visual Studio, і доступні шаблони роблять його дуже привабливим як IDE для великих і малих команд.

Visual Studio надає різні версії залежно від потреб команди програмістів і бюджету, тому невеликі команди можуть отримати доступ до безкоштовної версії цього середовища розробки [15]-[17].

У таблиці 2.2 наведено результати обгрунтування вибору середовища розробки для створення програмного забезпечення ріелторського агентства.

Таблиця 2.2 – Обгрунтування вибору середовища розробки для створення програмного забезпечення ріелторського агентства

Критерій порівняння	Середовище розробки		
	Microsoft Visual Studio	MonoDevelop	SharpDevelop
Підтримка мови програмування C#	+	+–	+–
Зручність для створення ПЗ для ріелторського агентства	+	–	+–
Підтримка роботи з репозиторіями	+	–	+–
Робота з базами даних	+	+	+–
Швидкість створення та модифікації ПЗ	+	+–	+–

Таким чином, як фреймворк-середовище розробки для створення програмного забезпечення ріелторського агентства обрано Microsoft Visual Studio.

2.3 Вибір системи керування базою даних

Для створення програми ріелторського агентства обрано систему керування базами даних Microsoft Access [18]-[19].

Microsoft Office Access або Microsoft Access – це корпоративна система керування базами даних (СКБД) Microsoft, яка входить до пакету Microsoft Office. Ця СКБД включає в себе графічний інтерфейс користувача та інструмент розробки програмного забезпечення [18]-[19].

Ця система керування базами даних має широкий спектр функцій, таких як зв'язування запитів, підключення до зовнішніх таблиць і баз даних. Завдяки вбудованій мові VBA можна писати програми, які працюють з базами даних у Access [18]-[19].

На відміну від інших настільних СКБД, Access зберігає всі дані в одному файлі, але розділяє їх на окремі таблиці, як це має бути в пов'язаній СКБД. Система керування базами даних Microsoft Access має широкий спектр функцій, що включає повні запити, різноманітні рейтинги, збережені електронні таблиці та взаємодію з базою даних [18]-[19].

Ключовими компонентами MS Access є: електронні таблиці; конструктор екранних форм; конструктор SQL-запитів (мова SQL в MS Access не відповідає стандартам ANSI); засоби створення звіту для друку. Використовуючи сценарії VBA, у СКБД MS Access можна розробляти програми, а базу даних можна писати як з нуля, так і заповнювати дані з зовнішньої бази. MS Access є базою даних файлового сервера і використовується лише для невеликих програм [18]-[19].

При використанні СКБД Access для створення запитів використовується мова SQL (Structured Query Language), яка являє собою універсальну комп'ютерну мову, яка використовується для створення, зміни та керування даними у базах даних. Визначення стандарту мови SQL дозволяє розширити процедуру на основі мови, яку можна вважати мовою програмування [18]-[19].

У таблиці 2.3 наведено обгрунтування вибору системи керування базами даних для розробки програмного забезпечення ріелторського агентства.

Таблиця 2.3 – Порівняння систем керування базами даних для розробки програмного забезпечення ріелторського агентства

Критерій порівняння СКБД	Системи керування базами даних		
	MS Access	MySQL	PostgreSQL
Підтримка засобів безпеки даних	+	+	+–
Зручність для створення програмного забезпечення ріелторського агентства	+	–	–
Низька вартість	+	+	–
Швидкість створення та модифікації баз даних	+	+	+–
Масштабованість	+	+–	+–

Таким чином для розробки програмного забезпечення ріелторського агентства обрано СКБД Microsoft Access.

2.4 Висновки за розділом 2

Для створення програмного забезпечення ріелторського агентства було обрано мову програмування C#.

При розробці програмного забезпечення ріелторського агентства було прийнято рішення використовувати середовище Microsoft Visual Studio, що включає інтегроване середовище розробки програмного забезпечення і багато

інших інструментів тадозволяє створювати програмне забезпечення різного призначення на різних мовах програмування, зокрема, і на мові C#

Для створення програми ріелторського агентства обрано систему керування базами даних Microsoft Access.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення ріелторського агентства

Структура програмного забезпечення ріелторського агентства була розроблена відповідно до поставленого завдання та складається з окремих програмних модулів. Структурну схему програмного забезпечення ріелторського агентства наведено на рис. 3.1.

Рисунок 3.1 – Структурна схему програмного забезпечення ріелторського агентства

Головним модулем програмного забезпечення ріелторського агентства є модуль головної форми, який грає роль диспетчера, тобто визначає залежно від обраного користувачем пункту меню або команди, яку процедуру необхідно виконати. Модуль для роботи з клієнтами призначений для підтримки процесу оброблення даних про клієнтів. Аналогічно модуль для роботи з нерухомістю призначений для підтримки процесу оброблення даних про нерухомість, що може бути реалізована за допомогою ріелторського агентства. Ці модулі також реалізують й інші функції програмного забезпечення, пов’язані з обробкою даних про пропозиції на ринку нерухомості та ін.

Оброблювана інформація про роботу ріелторського агентства фіксується в базі даних програмного забезпечення, і на її основі формуються

відповідні вихідні дані. Вихідна інформація формується в результаті виконання запитів і представляється у вигляді вихідних документів, що виводяться на екран або принтер.

3.2 Опис програмних компонентів програмного забезпечення ріелторського агентства

На етапі конструювання програмного забезпечення ріелторського агентства було створено програмні компоненти, опис яких нижче:

– AddClientForm – реалізує функціонал та інтерфейс додавання нового клієнта;

– AddPhoneForm – реалізує функціонал та інтерфейс додавання нового контактного телефону клієнта;

– AddProposalForm1, AddProposalForm2, AddProposalForm3, AddProposalForm4 – реалізують функціонал та описують інтерфейс майстра додавання нової пропозиції;

– AddStreetForm – реалізує функціонал додавання вулиці в довідник;

– app.config – файл конфігурації додатку, містить основну інформацію, необхідну для коректної роботи системи;

– DeleteProposalForm – реалізує інтерфейс видалення пропозиції із вказанням причини видалення;

– EditProposalForm – повністю реалізовує функціонал та інтерфейс редагування пропозиції;

– Global – основний модуль, містить основну інформації про підключення до файлу бази даних, реалізовує деякі загальні додаткові функції відображення помилок, тощо;

– LoginForm – повністю реалізує функціонал та інтерфейс авторизації користувача в системі;

– MainForm – описує інтерфейс головного вікна, містить основні функції роботи з базою даних;

– PhonesForm – описує інтерфейс відображення довідника контактних телефонів клієнтів;

– Proposal – описує клас, який представляє сутність Predlog бази даних

– ProposalsReportForm – повністю реалізує функціонал відображення та роботи зі звітами по пропозиціям;

– SearchProposalForm – реалізує функціонал та інтерфейс фільтрації бази даних пропозиції за багатьма критеріями.

Схему взаємодії програмних компонентів наведено на рис. 3.2.

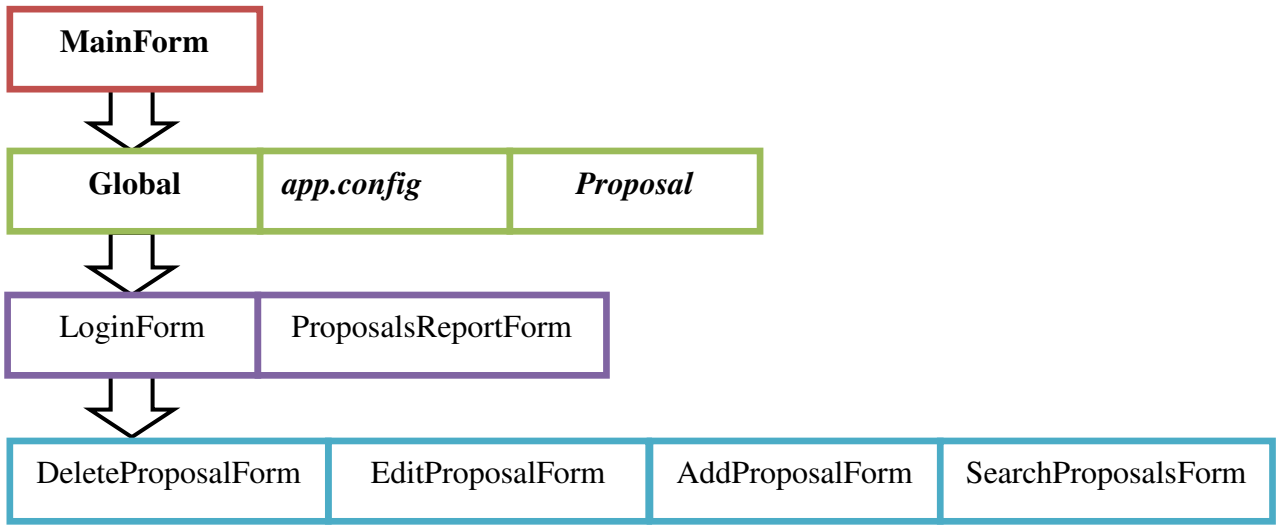


Рисунок 3.2 – Схема взаємодії програмних компонентів

3.3 Основні рішення з розробки бази даних

Для розробки програмного забезпечення ріелторського агентства використовувалася реляційна модель даних [18]-[19].

Реляційна модель відображає дані у простому табличному форматі, розділеному на рядки та стовпці, з даними на їх перетинах. Запит до таких таблиць повертає таблиці, які можуть бути предметом наступного запиту. Кожна база даних може містити кілька таблиць [18]-[19].

Аналіз конкретних об'єктів і функцій прояснює природу бази даних і дозволяє побудувати логічну схему [18]-[19]. Процес перетворення об’єктно-орієнтованої моделі бази даних у модель відношень полягає у створенні таблиці відношень для кожної об’єктної моделі. Набір окремих об'єктів діє як окрема таблиця, а властивості кожного об'єкта служать полем реляційної таблиці бази даних. Зв'язки між наборами об'єктів не змінюються, а змінюються відносини між відповідними таблицями зв'язків [18]-[19].

Як відзначено у попередньому розділі, база даних програмного забезпечення ріелторського агентства розроблена за допомогою Microsoft Access. База даних складається з декілької таблиць.

Сутність «Areas» (Id, Name) призначена для збереження інформації про райони, де можуть розташовуватися об’єкти нерухомості. В реалізації бази даних сутність представлено таблицею Areas (табл. 3.1).

Таблиця 3.1 – Структура таблиці Areas

Назва поля	Тип поля	Ключ	Опис
Id	Лічильник	Так	Ідентифікатор запису (району)
Name	Текстовий	–	Назва району

Сутність таблиці «Proposals» (Id, Street, House, Apartment, Floor, Cost, Rooms, FullSquare, HouseType, Type, YardWindows, IsolatedRooms, Redevelopment, LegalizedRedevelopment, StoreysNumber, LiveSquare, KitchenSquare, Balcony, Loggia, InteriorState, HotWater, Heating, Gas, Phone, State, PlacementDate, AreaId, TopZone, WC, Comment, Actived, DeleteReason, FlatType, ClientFIO, ClientContacts, StreetType) призначена для зберігання інформації про пропозиції продажу або аренди нерухомості. В реалізації бази даних сутність представлено таблицею Proposals (табл.3.2).

Таблиця 3.2 – Структура таблиці Proposals

Назва поля	Тип поля	Ключ	Опис
Id	Счетчик	Так	Ідентифікатор запису (код пропозиції)
Street	Текстовий	–	Вулиця
House	Текстовий	–	Номер дому
Apartment	Текстовий	–	Номер квартири
Floor	Числовий	–	Поверх
Cost	Числовий	–	Ціна
Rooms	Текстовий	–	Кількість кімнат
FullSquare	Логічний	–	Загальна площа
HouseType	Текстовий	–	Тип будівлі
YardWindows	Логічний	–	Вікна виходять у двір
IsolatedRooms	Числовий	–	Кількість ізольованих кімнат
Redevelopment	Логічний	–	Здійснювалось перепланування
LegalizedRedevelopment	Логічний	–	Перепланування легалізоване
StoreysNumber	Числовий	–	Кількість поверхів в будівлі
LiveSquare	Числовий	–	Житлова площа
KitchenSquare	Числовий	–	Площа кухні
Balcony	Текстовий	–	Кількість балконів
Loggia	Текстовий	–	Наявність лоджій
InteriorState	Текстовий	–	Стан ремонту
HotWater	Текстовий	–	Гаряча вода
Heating	Текстовий	–	Опалювання
Gas	Текстовий	–	Газ
Phone	Текстовий	–	Телефон
PlacementDate	Логічний	–	Дата розміщення
AreaId	Числовий	–	Район

Продовження таблиці 3.2

Назва поля	Тип поля	Ключ	Опис
TopZone	Числовий	–	Топозона
WC	Текстовий	–	Санвузол
Comment	Текстовий	–	Коммментар
Actived	Логічний	–	Активність запису
DeleteReason	Текстовий	–	Причина видалення пропозиції
FlatType	Текстовий	–	Тип квартири
ClientFIO	Текстовий	–	ПІБ клієнта
ClientContacts	Текстовий	–	Контактні дані клієнта
StreetType	Текстовий	–	Тиа вулиці

Сутність «Contacts» (Id, PhoneNumber). Сутність призначена для збереження інформації про всі контактні дані, з якими довелось працювати агенству. В реалізації бази даних сутність представлено таблицею Phones (табл. 3.3).

Таблиця 3.3 – Структура таблиці Contacts

Назва поля	Тип поля	Ключ	Опис
Id	Лічильник	Так	Ідентифікатор запису
PhoneNumber	Текстовий	–	Номер телефону

Сутність «Zones» (Id, AreaId, Name). Сутність призначена для збереження інформації про топозони та їх приналежність до районів. В реалізації бази даних сутність представлено таблицею Zones (табл. 3.4).

Таблиця 3.4 – Структура таблиці Zones

Назва поля	Тип поля	Ключ	Опис
Id	Лічильник	Так	Ідентифікатор запису
AreaId	Числовий	–	Район
Name	Текстовий	–	Назва вулиці

Таким чином, розроблено базу даних програмного забезпечення ріелторського агентства, що складається з пов’язаних між собою таблиць Areas (райони), Proposals (пропозиції), Contacts (контактні дані) та інші, які відображають основні об’єкти предметної області, пов’язані з ріелтерською діяльністю.

3.4 Висновки за розділом 3

Розроблено програмне забезпечення ріелторського агентства. Запропоновано структуру програмного забезпечення ріелторського агентства, яка розроблена відповідно до поставленого завдання та складається з окремих програмних модулів. Головним модулем програмного забезпечення ріелторського агентства є модуль головної форми, який грає роль диспетчера, тобто визначає залежно від обраного користувачем пункту меню або команди, яку процедуру необхідно виконати. Модуль для роботи з клієнтами призначений для підтримки процесу оброблення даних про клієнтів. Аналогічно модуль для роботи з нерухомістю призначений для підтримки процесу оброблення даних про нерухомість, що може бути реалізована за допомогою ріелторського агентства. Ці модулі також реалізують й інші функції програмного забезпечення, пов’язані з обробкою даних про пропозиції на ринку нерухомості та ін.

Надано опис основних програмних компонентів програмного забезпечення ріелторського агентства.

Розроблено базу даних програмного забезпечення ріелторського агентства, що складається з пов’язаних між собою таблиць Areas (райони), Proposals (пропозиції), Contacts (контактні дані) та інші, які відображають основні об’єкти предметної області, пов’язані з ріелтерською діяльністю.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програмне забезпечення ріелторського агентства призначено для підтримки діяльності організації, що займається операціями з нерухомістю. Програма написана на мові C#. База даних програмного забезпечення ріелторського агентства створена в Microsoft Access.

Програмне забезпечення ріелторського агентства забезпечує виконання таких функцій:

– можливість реєстрації та ідентифікації у програмі;

– підтримка операцій по роботі з даними агентства про нерухомість (додавання нових об’єктів, редагування даних про існуючі об’єкти, видалення непотрібної інформації та ін.);

– формування звітів по заданих критеріях;

– підтримка друку необхідної інформації.

Для експлуатації програмного забезпечення ріелторського агентства необхідно таке апаратне забезпечення:

- IBM-сумісний ПК з мікропроцесором Intel Pentium 1 ГHz та вище;
- оперативна пам’ять 256Мб і вище;
- вільного простору на жорсткому диску 100 Мб;
- наявність монітору;
- клавіатура та маніпулятор типу миша.

Функціональні характеристики розробленого програмного забезпечення ріелторського агентства наведено у технічному завданні (додаток А). Опис програмного забезпечення ріелторського агентства та текст програмного забезпечення ріелторського агентства наведені у додатках Б та В.

4.2 Характеристики програми

Режим роботи програмного забезпечення ріелторського агентства є цілодобовим та безпереервним. Програмне забезпечення ріелторського агентства використовує мінімамальну кількість ОЗУ.

Всі данні програмного забезпечення ріелторського агентства повинні зберігатися під управлянням реляційної СКБД.

Основними програмними компонентами є такі:

– AddClientForm – реалізує функціонал та інтерфейс додавання нового клієнта;

– AddPhoneForm – реалізує функціонал та інтерфейс додавання нового контактного телефону клієнта;

– AddProposalForm1, AddProposalForm2, AddProposalForm3, AddProposalForm4 – реалізують функціонал та описують інтерфейс майстра додавання нової пропозиції;

– AddStreetForm – реалізує функціонал додавання вулиці в довідник;

– app.config – файл конфігурації додатку, містить основну інформацію, необхідну для коректної роботи системи;

– DeleteProposalForm – реалізує інтерфейс видалення пропозиції із вказанням причини видалення;

– EditProposalForm – повністю реалізовує функціонал та інтерфейс редагування пропозиції;

– Global – основний модуль, містить основну інформації про підключення до файлу бази даних, реалізовує деякі загальні додаткові функції відображення помилок, тощо;

– LoginForm – повністю реалізує функціонал та інтерфейс авторизації користувача в системі;

– MainForm – описує інтерфейс головного вікна, містить основні функції роботи з базою даних;

– PhonesForm – описує інтерфейс відображення довідника контактних телефонів клієнтів;

– Proposal – описує клас, який представляє сутність Predlog бази даних

– ProposalsReportForm – повністю реалізує функціонал відображення та роботи зі звітами по пропозиціям;

– SearchProposalForm – реалізує функціонал та інтерфейс фільтрації бази даних пропозиції за багатьма критеріями.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для звернення до програмного забезпечення ріелторського агентства необхідно запустити відповідний файл. Програміст може змінити програму відкривши файл Agency.sln за допомогою середовища розробки MS Visual Studio або відкрити файли з розширенням .cs в текстовому редакторі.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення ріелторського агентства є дані файлу бази даних Agency.mdb.

Вихідними даними програмного забезпечення ріелторського агентства є звіти з бази даних, які можна продивитися або роздрукувати.

4.3.3 Повідомлення

Програмне забезпечення ріелторського агентства у випадку некоректного підключення до бази даних видає відповідне повідомлення.

Програма ріелторського агентства у випадку некоректного введення даних у поля вводу також видає повідомлення про помилку введення даних у некоректному форматі.

4.4 Робота з програмним забезпеченням ріелторського агентства

При запуску програми користувач має пройти авторизацію (рис. 4.1).

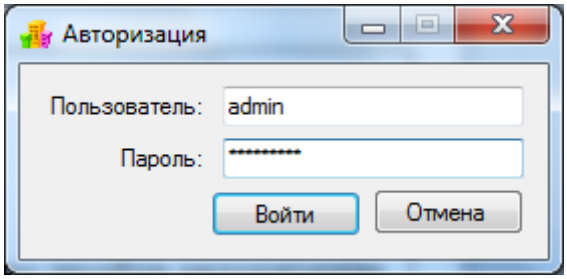


Рисунок 4.1 – Вікно авторизації

Після введення реєстраційних даних відкривається головне вікно (рис. 4.2).

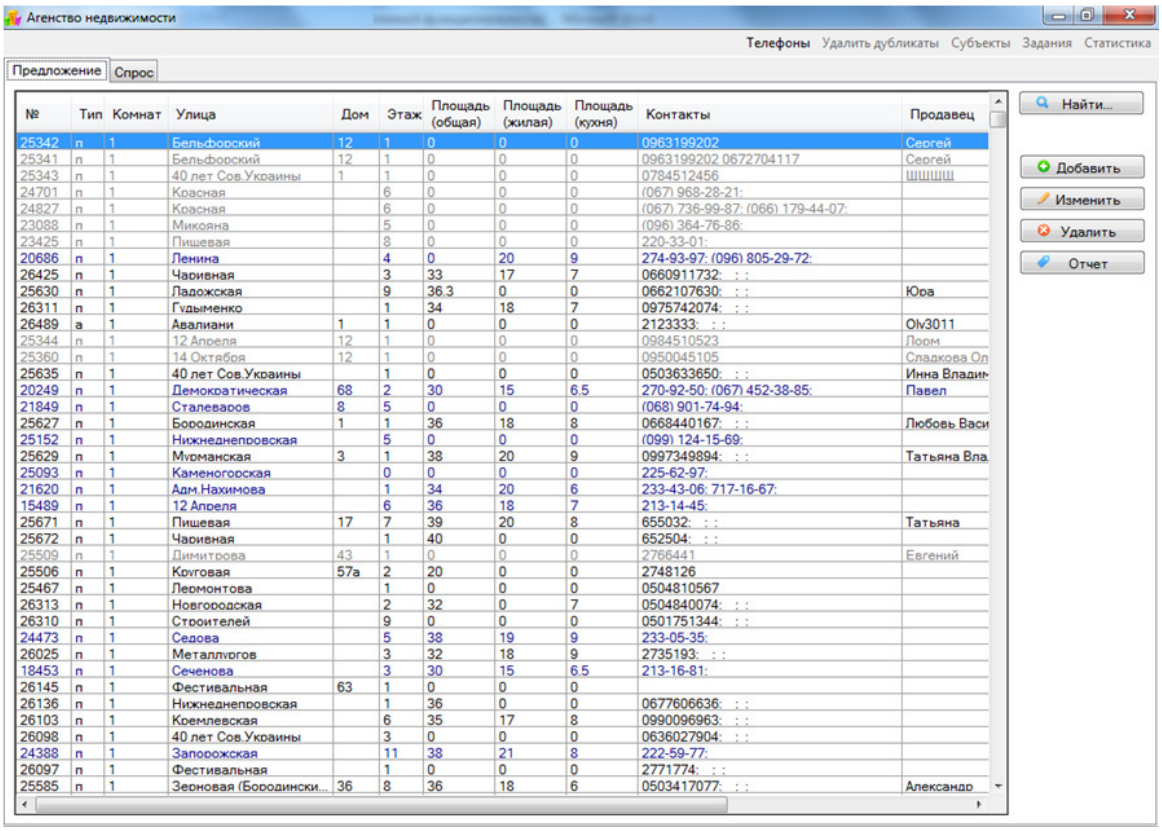


Рисунок 4.2 – Головне вікно програми

Воно складається з таких частин:

– головне меню;

– панель стану;

– панель керування;

– відображення даних.

З головного меню здійснюються переходи до додаткових вікон програми («Телефони», «Видалення дублікатів», «Статистика» тощо).

З панелі керування здійснюється управління основними функціями. Як то фільтрація, додавання, видалення та редагування пропозицій, робота зі звітами (рис. 4.3).

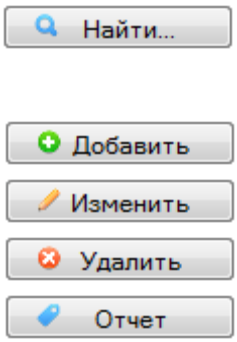


Рисунок 4.3 – Панель керування

Додавання нової пропозиції здійснюється за допомогою майстра (рис. 4.4-4.7). Тут вказуються номери телефонів клієнта, тип пропозиції, кількість кімнат на параметри розміщення об’єкту (вулиця, тип, номер дому та квартири). При відсутності вулиці в списку, користувач може самостійно додати нову в довідник (рис. 4.8). Після завершення роботи майстра на екрані відобразиться діалог редагування щойно доданої пропозиції. Це дасть змогу користувачу ввести всю необхідну додаткову інформацію.

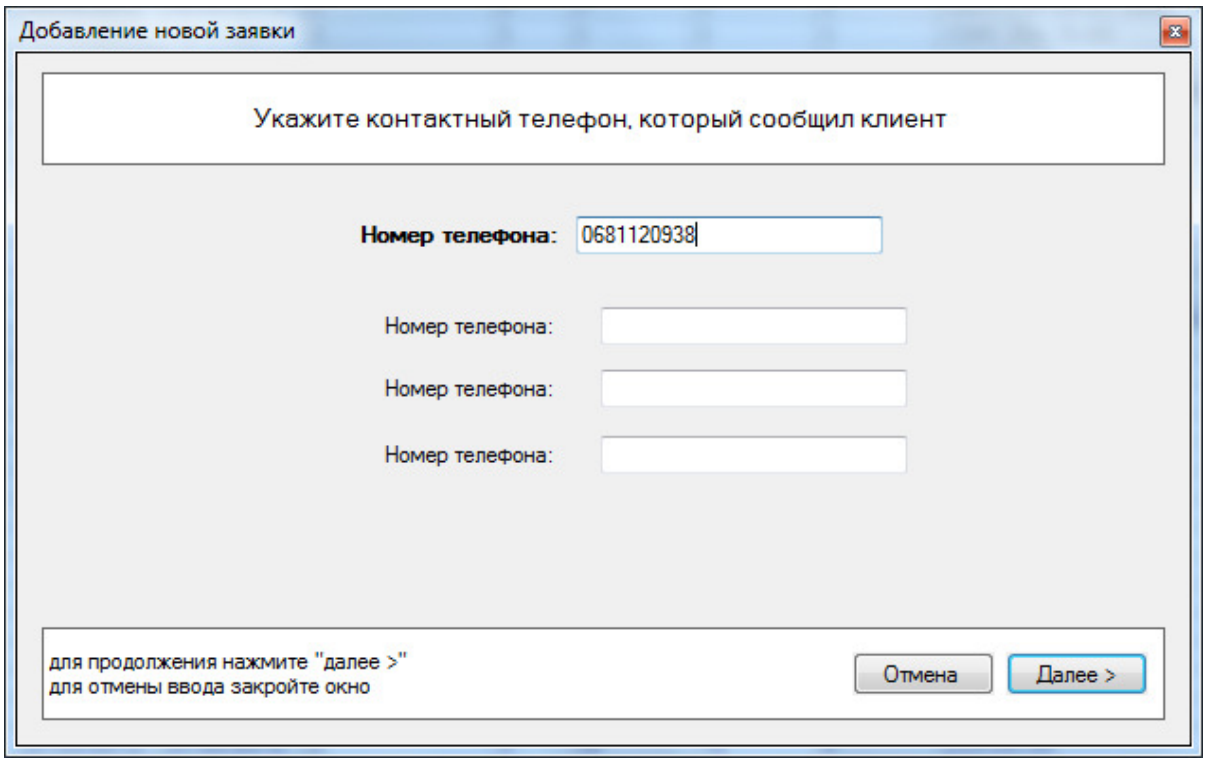


Рисунок 4.4 – Майстер додавання пропозиції, крок 1

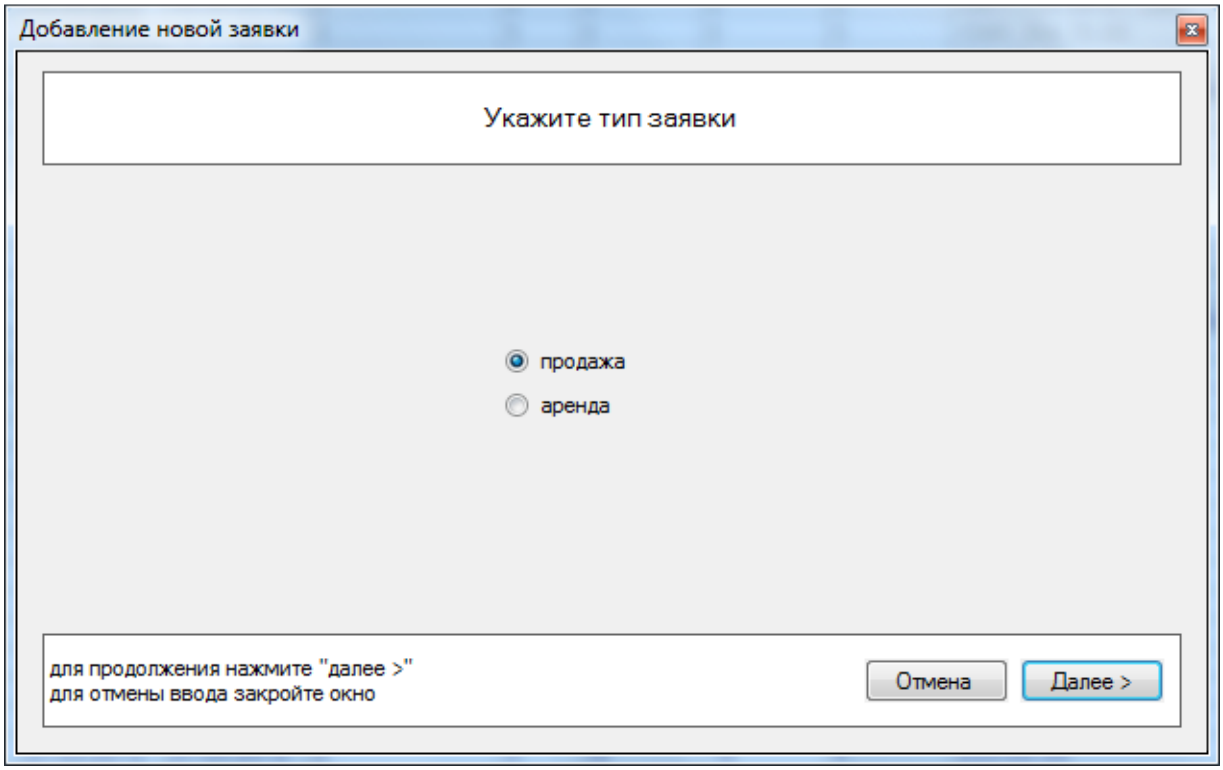


Рисунок 4.5 – Майстер додавання пропозиції, крок 2

Додавання нової пропозиції здійснюється за допомогою майстра (рис. 4.6-4.7).

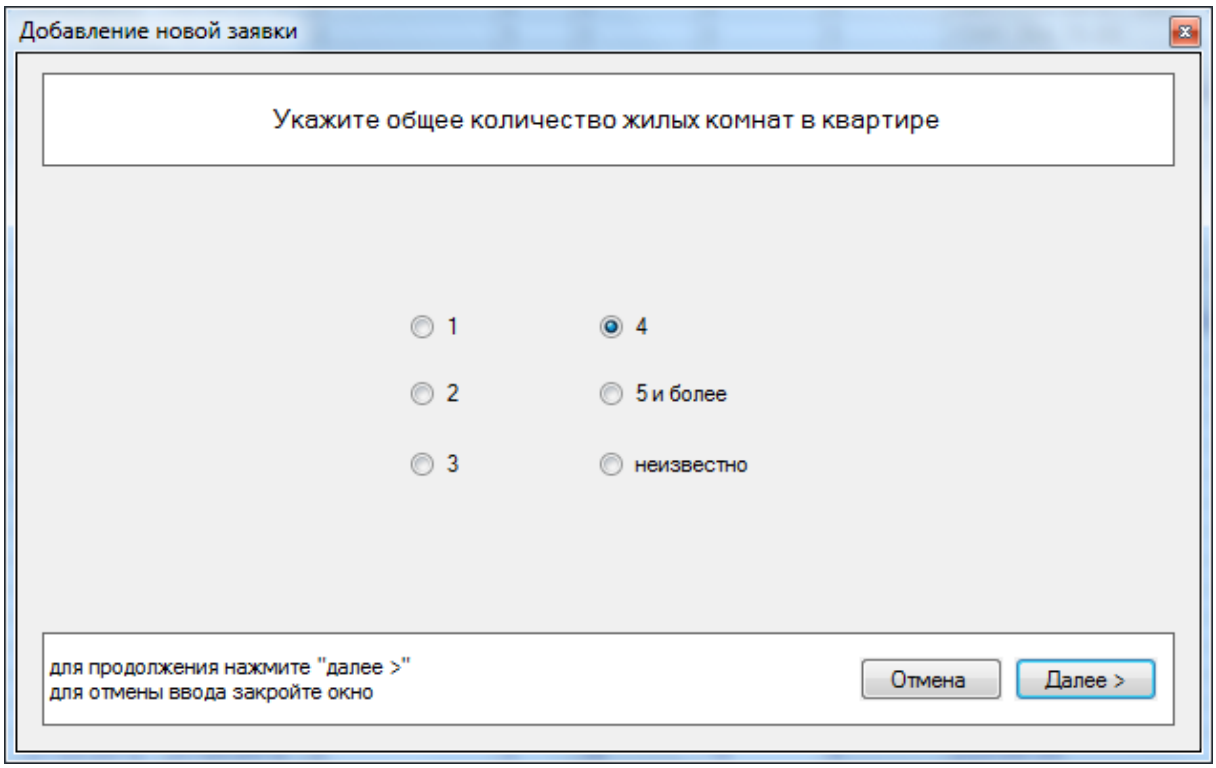


Рисунок 4.6 – Майстер додавання пропозиції, крок 3

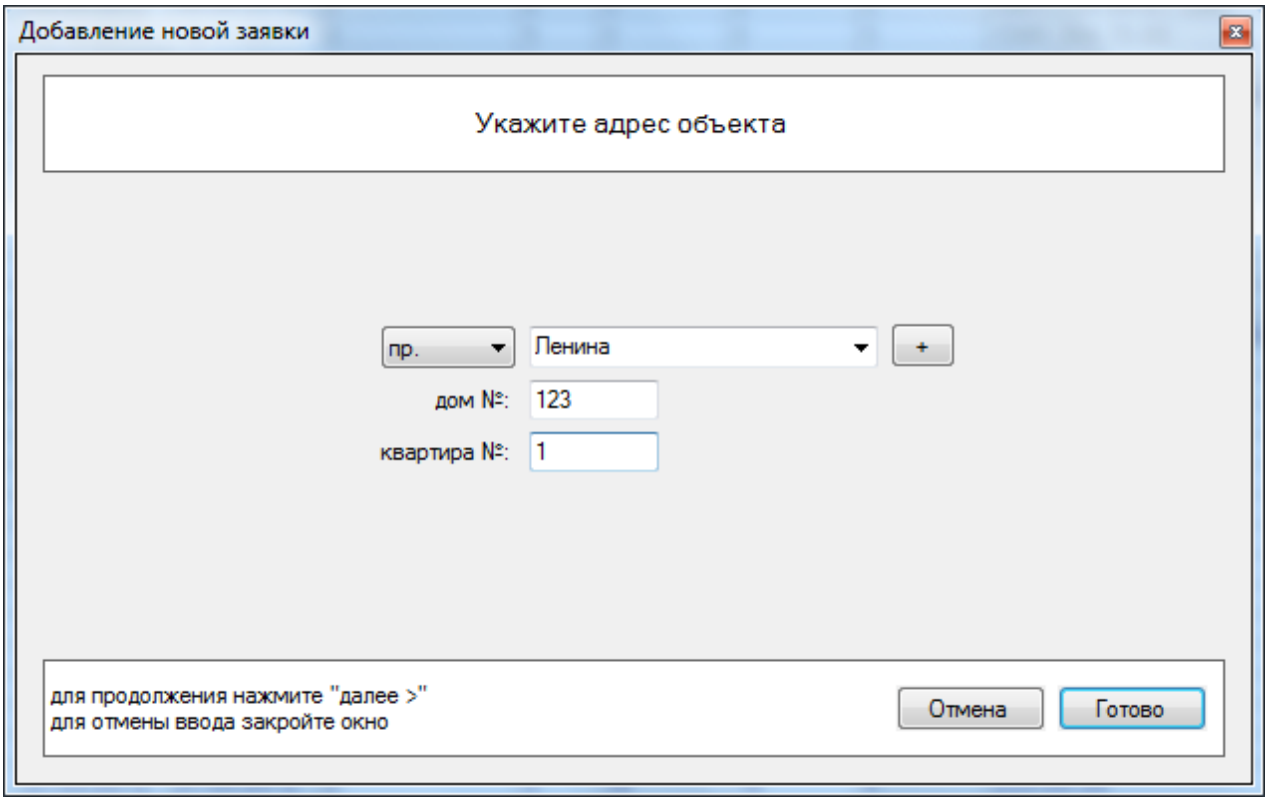


Рисунок 4.7 – Майстер додавання пропозиції, крок 4

На рисунку 4.8 наведено додавання нової вулиці в довідник.

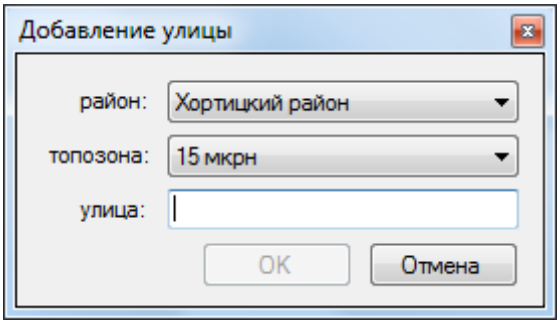


Рисунок 4.8 – Додавання нової вулиці в довідник

Для редагування запису потрібно двічі клацнути на необхідному записі в таблиці або натиснути кнопку «Змінити». При цьому відобразиться діалог редагування, в якому будуть відображені всі параметри обраної пропозиції (рис. 4.9).

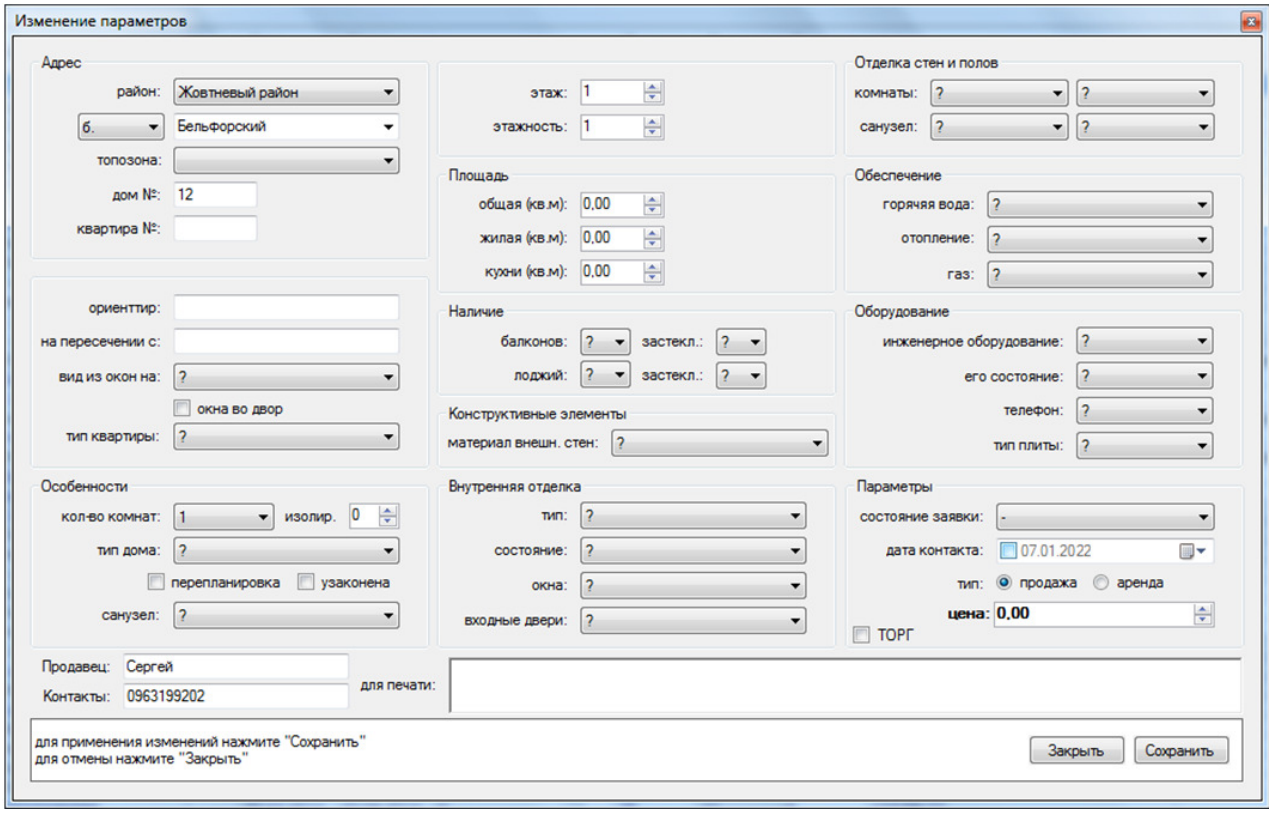


Рисунок 4.9 – Діалог редагування пропозиції

Для видалення пропозиції з системи необхідно вибрати необхідний запис в таблиці і натиснути клавішу «Delete» на клавіатурі або натиснути кнопку «Видалити» в програмі. При цьому оператор має ввести причину видалення пропозиції (рис. 4.10). Пропозиції фізично не видяляються з бази даних, а лише фіксуються як «не активні» і будь-який менеджер агентства зможе в майбутньому переглянути видалені пропозиції та причини їх видалення. Це є дуже зручним інструментом при веденні статистики роботи підприємства.

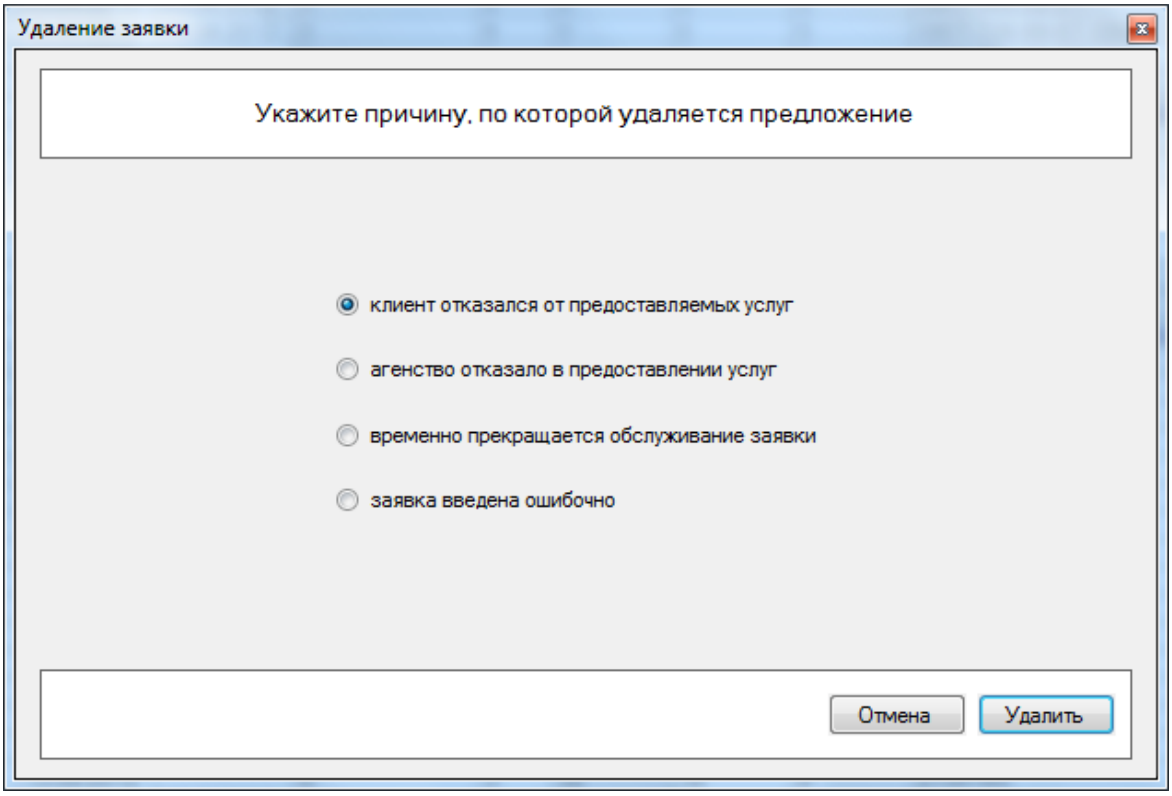


Рисунок 4.10 – Діалог видалення пропозиції

Оператор може фільтрувати всі пропозиції за багатьма критеріями. Для виклику діалогу пошуку необхідно натиснути на кнопку «Пошук» - відобразиться діалог (рис. 4.11). Тут користувач може обрати один або декілька з багатьох критеріїв пошуку. При натисненні на кнопку підтвердження результати фільтрації буде видно в таблиці пропозицій.

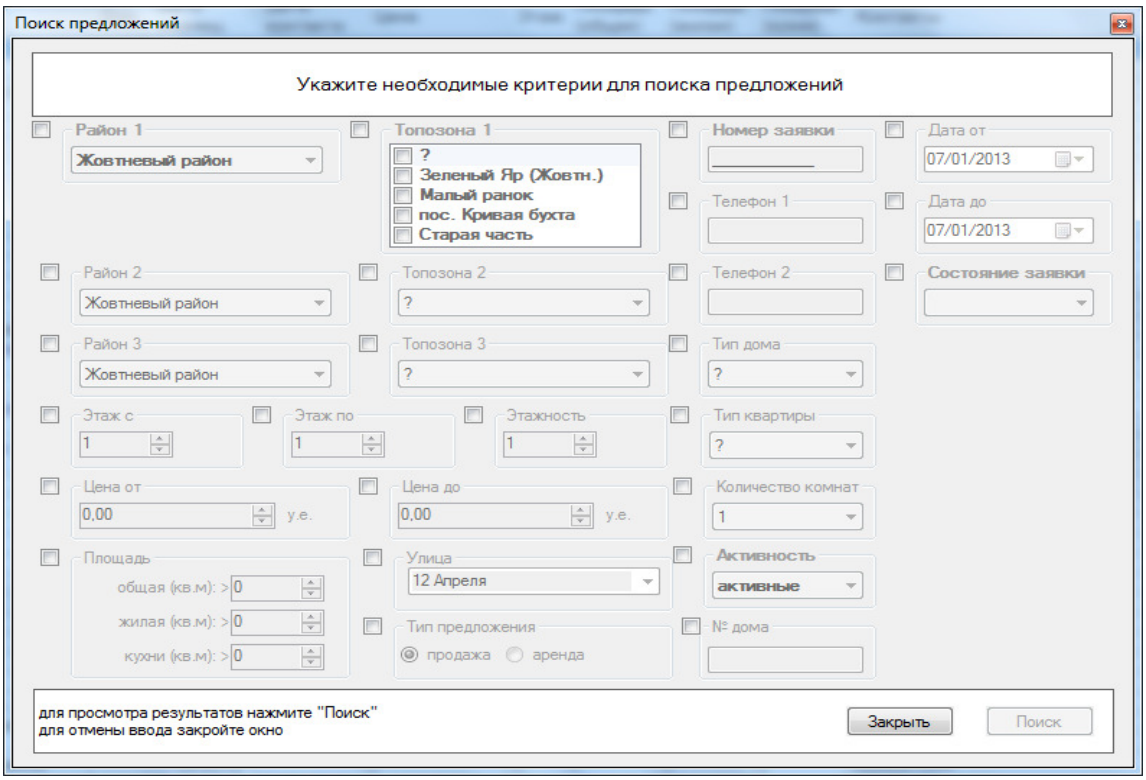


Рисунок 4.11– Діалог пошуку пропозицій

Програмне забезпечення рієлторського агентства було протестовано. Результати тестування програмного забезпечення рієлторського агентства показали, що програма працює коректно, вся інформація програмного забезпечення рієлторського агентства правильно виводиться в інтерфейсних формах та при реєстрації користувача, дані про об’єкти нерухомості коректно заносяться до бази даних.

4.5 Висновки за розділом 4

Описано програмне забезпечення рієлторського агентства. Виконано тестування розробленого програмного забезпечення рієлторського агентства. Результати тестування програмного забезпечення рієлторського агентства показали, що програма працює коректно, вся інформація програмного забезпечення рієлторського агентства правильно виводиться в інтерфейсних формах та при реєстрації користувача, дані про об’єкти нерухомості коректно заносяться до бази даних.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для агенцій нерухомості.

Проаналізовано поняття про ріелторську діяльність та ріелторські агентства. Визначено, що ріелторські агентства – це організації, які займаються ріелторською діяльністю, тобто забезпечують підтримку операцій з нерухомістю, об’єднують покупців і продавців та виступають їх представником у переговорах. Відзначено, що часто власники нерухомості не мають можливості постійно займатися процесом продажу і звертаються до ріелторських агентств. Тому такі агентства мають власні бази даних нерухомості, що продається. Отже, таку інформацію необхідно систематизовувати та обробляти, що обумовлює необхідність розробки відповідного програмного забезпечення для ріелторських агентств.

Досліджено та проаналізовано програмні засоби для ріелторських агентств. Визначено, що існуюче програмне забезпечення для підтримки діяльності ріелторських агентств надають різні можливості співробітникам таких агентств. Проте такі програми, як правило, є досить складними для використання, а також не підтримують можливості ведення прихованого списку нерухомості (списки нерухомості, з якою працює агентство, як правило, є загальнодоступним у наявному ПЗ), таким чином агенти з нерухомості не можуть вводити інформацію про нерухомість, яка доступна тільки у них та яку доцільно пропонувати тільки власним клієнтам. Отже, інформацію про нерухомість, яка не повинна бути загально доступною, та є такою, що повинна пропонуватися тільки унікальним клієнтам, необхідно систематизовувати та обробляти окремо, у власних програмах, що обумовлює необхідність розробки відповідного програмного забезпечення для ріелторських агентств. Крім того, використання відомого програмного

забезпечення вимагає обов’язкового внесення щомісячних платежів, що також ускладнює його використання.

Для створення програмного забезпечення ріелторського агентства було обрано мову програмування C#.

При розробці програмного забезпечення ріелторського агентства було прийнято рішення використовувати середовище Microsoft Visual Studio, що включає інтегроване середовище розробки програмного забезпечення і багато інших інструментів тадозволяє створювати програмне забезпечення різного призначення на різних мовах програмування, зокрема, і на мові C#.

Для створення програми ріелторського агентства обрано систему керування базами даних Microsoft Access.

Розроблено програмне забезпечення ріелторського агентства. Запропоновано структуру програмного забезпечення ріелторського агентства, яка розроблена відповідно до поставленого завдання та складається з окремих програмних модулів. Головним модулем програмного забезпечення ріелторського агентства є модуль головної форми, який грає роль диспетчера, тобто визначає залежно від обраного користувачем пункту меню або команди, яку процедуру необхідно виконати. Модуль для роботи з клієнтами призначений для підтримки процесу оброблення даних про клієнтів. Аналогічно модуль для роботи з нерухомістю призначений для підтримки процесу оброблення даних про нерухомість, що може бути реалізована за допомогою ріелторського агентства. Ці модулі також реалізують й інші функції програмного забезпечення, пов’язані з обробкою даних про пропозиції на ринку нерухомості та ін.

Надано опис основних програмних компонентів програмного забезпечення ріелторського агентства.

Розроблено базу даних програмного забезпечення ріелторського агентства, що складається з пов’язаних між собою таблиць Areas (райони), Proposals (пропозиції), Contacts (контактні дані) та інші, які відображають основні об’єкти предметної області, пов’язані з ріелтерською діяльністю.

Описано програмне забезпечення ріелторського агентства. Виконано тестування розробленого програмного забезпечення ріелторського агентства. Результати тестування програмного забезпечення ріелторського агентства показали, що програма працює коректно, вся інформація програмного забезпечення ріелторського агентства правильно виводиться в інтерфейсних формах та при реєстрації користувача, дані про об’єкти нерухомості коректно заносяться до бази даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Real Estate Agent [Electronic resource]. – Access mode: https://www.investopedia.com/terms/r/realestateagent.asp.

2. What Is Real Estate? [Electronic resource]. – Access mode: https://www.thebalance.com/real-estate-what-it-is-and-how-it-works-3305882.

3. Types of Real Estate Agencies [Electronic resource]. – Access mode: https://study.com/academy/lesson/types-of-real-estate-agencies.html.

4. What Does a Real Estate Agent Do [Electronic resource]. – Access mode: https://www.kapre.com/resources/real-estate/just-what-does-a-real-estate-agent-do/.

5. Real Estate [Electronic resource]. – Access mode: https://www.investopedia.com/terms/r/realestate.asp.

6. Best Real Estate Software & Tools for Top Agents [Electronic resource]. – Access mode: https://fitsmallbusiness.com/best-real-estate-software/.

7. Zillow Premier Agent [Electronic resource]. – Access mode: https://www.zillow.com/premier-agent.

8. BoldLeads [Electronic resource]. – Access mode : https://boldleads.com.

9. LionDesk [Electronic resource]. – Access mode : https://liondesk.com.

10. Top Producer [Electronic resource]. – Access mode: https://www.topproducer.com.

11. Propertybase [Electronic resource]. – Access mode: https://go.propertybase.com.

12. Real Geeks [Electronic resource]. – Access mode: https://www.realgeeks.com.

13. Übersicht über die Programmiersprachen [Electronic resource]. – Access mode: https://lerneprogrammieren.de/uebersicht-ueber-die-programmiersprachen/.

14. C-Sharp: Der Allrounder unter den Programmiersprachen [Electronic resource]. – Access mode: https://www.klubschule.ch/Angebote/Informatik-Neue-Medien/Programmierung/C.

15. Quintec [Electronic resource]. – Access mode: https://www.quintec.de/technologien/cnet.

16. Visual Studio [Electronic resource]. – Access mode: https://visualstudio.microsoft.com/.

17. Microsoft integrates Visual Studio with open-source Eclipse IDE [Electronic resource]. – Access mode: https://www.zdnet.com/article/microsoft-integrates-visual-studio-with-open-source-eclipse-ide/.

18. Microsoft Access [Electronic resource]. – Access mode: https://ru.bmstu.wiki/Microsoft_Access.

19. Microsoft Access [Electronic resource]. – Access mode: https://www.softwareadvice.ie/software/270144/microsoft-access.

ДОДАТОК А

Технічне завдання

Вступ

Програмне забезпечення ріелторського агентства призначено для підтримки діяльності організації, що займається операціями з нерухомістю.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення ріелторського агентства», затверджене наказом Національного університету «Запорізька політехніка» № 102 від 27 квітня 2022 р.

А.2 Призначення розробки

Програмний продукт ріелторського агентства призначений для підтримки діяльності організації, що займається операціями з нерухомістю.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення ріелторського агентства повинно задовольняти таким функціональним вимогам:

– можливість реєстрації та ідентифікації у програмі;

– підтримка операцій по роботі з даними агентства про нерухомість (додавання нових об’єктів, редагування даних про існуючі об’єкти, видалення непотрібної інформації та ін.);

– формування звітів по заданих критеріях;

– підтримка друку необхідної інформації.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програми ріелторського агентства повинен бути зручним для користувачів. Взаємодія користувача з програмою ріелторського агентства можлива за допомогою інтерфейсних форм.

А.3.3 Вимоги до надійності

Програмне забезпечення ріелторського агентства повинно обробляти критичні, проблемні та виключні ситуації та видавати повідомлення про такі ситуації.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення ріелторського агентства необхідно апаратне забезпечення з оперативною памяттю, процесором, а також засобами введення-виведення інформації. Програмне забезпечення ріелторського агентства не повинно потребувати спеціальних навичок володіння комп'ютером, режим роботи повинен бути заснований на роботі з інформацією за прикладом додатків MS Office.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення процесором, необхідні такі засоби:

- IBM-сумісний ПК з мікропроцесором Intel Pentium 1 ГHz та вище;
- оперативна пам’ять 256Мб і вище;
- вільного простору на жорсткому диску 100 Мб;
- наявність монітору;
- клавіатура та маніпулятор типу миша.

А.3.6 Вимоги до маркування і пакування

Програма ріелторського агентства може бути записана на різних носіях інформації. На пакуванні повинна бути назва програми – «Програмне забезпечення ріелторського агентства».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення ріелторського агентства є дані файлу бази даних Agency.mdb.

Вихідними даними програмного забезпечення ріелторського агентства є звіти з бази даних, які можна продивитися або роздрукувати.

ДОДАТОК Б

Опис програми

Б.1 Загальні відомості

Програмне забезпечення ріелторського агентства розроблено за допомогою мови програмування C#, база даних програмного забезпечення ріелторського агентства створена в Microsoft Access.

Б.2 Функціональне призначення

Програма призначена для підтримки діяльності організації, що займається операціями з нерухомістю.

Програмне забезпечення ріелторського агентства забезпечує виконання таких функцій:

– можливість реєстрації та ідентифікації у програмі;

– підтримка операцій по роботі з даними агентства про нерухомість (додавання нових об’єктів, редагування даних про існуючі об’єкти, видалення непотрібної інформації та ін.);

– формування звітів по заданих критеріях;

– підтримка друку необхідної інформації.

Б.3 Опис логічної структури

Структура програмного забезпечення ріелторського агентства була розроблена відповідно до поставленого завдання та складається з окремих програмних модулів.

Головним модулем програмного забезпечення ріелторського агентства є модуль головної форми, який грає роль диспетчера, тобто визначає залежно від обраного користувачем пункту меню або команди, яку процедуру необхідно виконати. Модуль для роботи з клієнтами призначений для підтримки процесу оброблення даних про клієнтів. Аналогічно модуль для роботи з нерухомістю призначений для підтримки процесу оброблення даних

про нерухомість, що може бути реалізована за допомогою ріелторського агентства. Ці модулі також реалізують й інші функції програмного забезпечення, пов’язані з обробкою даних про пропозиції на ринку нерухомості та ін.

Оброблювана інформація про роботу ріелторського агентства фіксується в базі даних програмного забезпечення, і на її основі формуються відповідні вихідні дані. Вихідна інформація формується в результаті виконання запитів і представляється у вигляді вихідних документів, що виводяться на екран або принтер.

Основними програмними компонентами є такі:

– AddClientForm – реалізує функціонал та інтерфейс додавання нового клієнта;

– AddPhoneForm – реалізує функціонал та інтерфейс додавання нового контактного телефону клієнта;

– AddProposalForm1, AddProposalForm2, AddProposalForm3, AddProposalForm4 – реалізують функціонал та описують інтерфейс майстра додавання нової пропозиції;

– AddStreetForm – реалізує функціонал додавання вулиці в довідник;

– app.config – файл конфігурації додатку, містить основну інформацію, необхідну для коректної роботи системи;

– DeleteProposalForm – реалізує інтерфейс видалення пропозиції із вказанням причини видалення;

– EditProposalForm – повністю реалізовує функціонал та інтерфейс редагування пропозиції;

– Global – основний модуль, містить основну інформації про підключення до файлу бази даних, реалізовує деякі загальні додаткові функції відображення помилок, тощо;

– LoginForm – повністю реалізує функціонал та інтерфейс авторизації користувача в системі;

– MainForm – описує інтерфейс головного вікна, містить основні функції роботи з базою даних;

– PhonesForm – описує інтерфейс відображення довідника контактних телефонів клієнтів;

– Proposal – описує клас, який представляє сутність Predlog бази даних

– ProposalsReportForm – повністю реалізує функціонал відображення та роботи зі звітами по пропозиціям;

– SearchProposalForm – реалізує функціонал та інтерфейс фільтрації бази даних пропозиції за багатьма критеріями.

Б.4 Використані технічні засоби

Для експлуатації програмного забезпечення ріелторського агентства необхідно таке апаратне забезпечення:

– IBM-сумісний ПК з мікропроцесором Intel Pentium 1 ГHz та вище;
– оперативна пам’ять 256Мб і вище;
– вільного простору на жорсткому диску 100 Мб;
– наявність монітору;
– клавіатура та маніпулятор типу миша.

Б.5 Виклик і завантаження

Для запуску програмного забезпечення ріелторського агентства необхідно запустити відповідний файл. Програміст може змінити програму відкривши файл Agency.sln за допомогою середовища розробки MS Visual Studio або відкрити файли з розширенням .cs в текстовому редакторі.

Б.6 Вхідні та вихідні дані

Вхідними даними до програмного забезпечення ріелторського агентства є дані файлу бази даних Agency.mdb.

Вихідними даними програмного забезпечення ріелторського агентства є звіти з бази даних, які можна продивитися або роздрукувати.

ДОДАТОК В

Текст програми

```
        public partial class EditProposalForm : Form
            {
                private long proposalId;
                private Boolean afterAdd;
                private Boolean savePressed = false;
                private Boolean firstLoad = true;

                public EditProposalForm(long id, Boolean afterAdd)
                {
                    InitializeComponent();
                    proposalId = id;
                    this.afterAdd = afterAdd;
                    stateComboBox.Items.AddRange(Global.statusValues);
                }

                private void EditProposalForm_Shown(object sender, EventArgs e)
                {
                    this.Cursor = Cursors.WaitCursor;
                    Global.mainForm.loadAreas(areaComboBox);
                    Global.mainForm.loadStreets(streetComboBox);
                    Global.mainForm.loadStreetTypes(streetTypeComboBox);
                    String query = String.Format("select * from Proposals where
Id = {0}", proposalId);
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    dataAdapter.Fill(proposalDataSet);
                    Global.connection.Close();
                    fillFields();
                    firstLoad = false;
                    this.Cursor = Cursors.Default;
                }

                private void fillFields()
                {
                    topZoneComboBox.SelectedIndex = 0;
                    windowsViewComboBox.SelectedIndex = 0;
                    houseTypeComboBox.SelectedIndex = 0;
                    wcComboBox.SelectedIndex = 0;
                    outerWallsMaterialComboBox.SelectedIndex = 0;
                    interiorTypeComboBox.SelectedIndex = 0;
                    interiorStateComboBox.SelectedIndex = 0;
                    interiorWindowsComboBox.SelectedIndex = 0;
                    interiorDoorsComboBox.SelectedIndex = 0;
                    facingRooms1ComboBox.SelectedIndex = 0;
                    facingRooms2ComboBox.SelectedIndex = 0;
                    facingWC1ComboBox.SelectedIndex = 0;
                    facingWC2ComboBox.SelectedIndex = 0;
                    hotWaterComboBox.SelectedIndex = 0;
                    heatingComboBox.SelectedIndex = 0;
                    gasComboBox.SelectedIndex = 0;
                    equipmentComboBox.SelectedIndex = 0;
                    equipmentStateComboBox.SelectedIndex = 0;
                    phoneComboBox.SelectedIndex = 0;
                    coockerTypeComboBox.SelectedIndex = 0;
                    stateComboBox.SelectedIndex = 0;
                    flatTypeComboBox.SelectedIndex = 0;
                    balconyComboBox.SelectedIndex = 0;
```

```
                    balconyGlazedComboBox.SelectedIndex = 0;
                    loggiaComboBox.SelectedIndex = 0;
                    loggiaGlazedComboBox.SelectedIndex = 0;

                    object[]                        fields                        =
proposalDataSet.Tables[0].Rows[0].ItemArray;
                    try
                    {

                        streetComboBox.SelectedItem = fields[1].ToString();
                        houseTextBox.Text = fields[2].ToString();
                        aptTextBox.Text = fields[3].ToString();
                        floorNumericUpDown.Value = Convert.ToDecimal(fields[4]);
                        costNumericUpDown.Value = Convert.ToDecimal(fields[5]);
                        roomsComboBox.SelectedItem = fields[6].ToString();
                        fullSquareNumericUpDown.Value                            =
Convert.ToDecimal(fields[7]);
                        houseTypeComboBox.SelectedItem = fields[9].ToString();
                        if (fields[10].ToString() == "п")
                        {
                            saleRadioButton.Checked = true;
                        }
                        else if (fields[10].ToString() == "a")
                        {
                            rentRadioButton.Checked = true;
                        }
                        intersectionTextBox.Text = fields[11].ToString();
                        windowsViewComboBox.SelectedItem                         =
fields[12].ToString();
                        yardWindowsCheckBox.Checked = (Boolean)fields[13];
                        isolatedRoomsNumericUpDown.Value                         =
Convert.ToDecimal(fields[14]);
                        redevelopmentCheckBox.Checked = (Boolean)fields[15];
                        legalizedRedevelopmentCheckBox.Checked                   =
(Boolean)fields[16];
                        storeysNumberNumericUpDown.Value                         =
Convert.ToDecimal(fields[17]);
                        liveSquareNumericUpDown.Value                            =
Convert.ToDecimal(fields[18]);
                        kitchenSquareNumericUpDown.Value                         =
Convert.ToDecimal(fields[19]);
                        balconyComboBox.SelectedItem = fields[20].ToString();
                        balconyGlazedComboBox.SelectedItem                       =
fields[21].ToString();
                        loggiaComboBox.SelectedItem = fields[22].ToString();
                        loggiaGlazedComboBox.SelectedItem                        =
fields[23].ToString();
                        outerWallsMaterialComboBox.SelectedItem                  =
fields[24].ToString();
                        interiorTypeComboBox.SelectedItem                        =
fields[25].ToString();
                        interiorStateComboBox.SelectedItem                       =
fields[26].ToString();
                        interiorWindowsComboBox.SelectedItem                     =
fields[27].ToString();
                        interiorDoorsComboBox.SelectedItem                       =
fields[28].ToString();
```

```
                        facingRooms1ComboBox.SelectedItem                       =
fields[29].ToString();
                        facingRooms2ComboBox.SelectedItem                       =
fields[30].ToString();
                        facingWC1ComboBox.SelectedItem = fields[31].ToString();
                        facingWC2ComboBox.SelectedItem = fields[32].ToString();
                        hotWaterComboBox.SelectedItem = fields[33].ToString();
                        heatingComboBox.SelectedItem = fields[34].ToString();
                        gasComboBox.SelectedItem = fields[35].ToString();
                        phoneComboBox.SelectedItem = fields[36].ToString();
                        coockerTypeComboBox.SelectedItem                        =
fields[37].ToString();
                        if (fields[38].ToString() != "")
                        {
                            stateComboBox.SelectedItem = fields[38].ToString();
                        }
                        else
                        {
                            stateComboBox.SelectedIndex = 0;
                        }
                        if (fields[40].ToString() != "")
                        {
                            contactDateDateTimePicker.Value                     =
(DateTime)fields[40];
                            contactDateDateTimePicker.Checked = true;
                        }
                        if (fields[41].ToString() != "")
                        {
                            areaComboBox.SelectedItem                           =
Global.mainForm.getAreaNameById(Convert.ToInt64(fields[41]));
                            Global.mainForm.loadZonesByArea(topZoneComboBox,
Convert.ToInt64(fields[41]));
                        }
                        else
                        {
                            areaComboBox.SelectedIndex = 0;
                            Global.mainForm.loadZonesByArea(topZoneComboBox,
Global.mainForm.getAreaIdByName(areaComboBox.SelectedItem.ToString()));
                        }

                        if (fields[42].ToString() == "0")
                        {
                            topZoneComboBox.SelectedIndex = 0;
                        }
                        else
                        {
                            topZoneComboBox.SelectedItem                        =
Global.mainForm.getZoneNameById(Convert.ToInt64(fields[42]));
                        }
                        //topZoneComboBox.SelectedItem = fields[42].ToString();
                        guideTextBox.Text = fields[43].ToString();
                        wcComboBox.SelectedItem = fields[44].ToString();
                        equipmentComboBox.SelectedItem = fields[45].ToString();
                        equipmentStateComboBox.SelectedItem                     =
fields[46].ToString();
                        commentRichTextBox.Text = fields[47].ToString();
                        flatTypeComboBox.SelectedItem = fields[48].ToString();
                        clientFIOTextBox.Text = fields[51].ToString();
```

```
                    clientContactsTextBox.Text = fields[52].ToString();
                    streetTypeComboBox.SelectedItem = fields[53].ToString();

                }
                catch (Exception ex)
                {
                    Global.showErrorMessage(ex.Message);
                }
            }

            private void closeButton_Click(object sender, EventArgs e)
            {
                this.Close();
            }

            private void saveButton_Click(object sender, EventArgs e)
            {
                savePressed = true;
                if     (MessageBox.Show("Сохранить     изменения?",     "",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                    System.Windows.Forms.DialogResult.Yes)
                {
                    Proposal p = new Proposal();
                    if (saleRadioButton.Checked) p.Type = "п";
                    else p.Type = "a";
                    p.Areaid                                                =
Global.mainForm.getAreaIdByName(areaComboBox.SelectedItem.ToString());
                    if (topZoneComboBox.SelectedIndex == 0) p.TopZone = "0";
                    else                    p.TopZone                       =
Global.mainForm.getZoneIdByNameAndArea(topZoneComboBox.SelectedItem.ToString(),
p.Areaid).ToString();
                    String contactDateStr = "";
                    if (contactDateDateTimePicker.Checked) contactDateStr =
"ContactDate = @contactDate,";
                    String stateString = "";
                    if  (stateComboBox.SelectedIndex  !=  0)  stateString  =
stateComboBox.SelectedItem.ToString();
                    String query = String.Format("update Proposals set " +
                        "Street = '{0}', " +
                        "House = '{1}', " +
                        "Apartment = '{2}', " +
                        "Floor = {3}, " +
                        "Cost = '{4}', " +
                        "Rooms = '{5}', " +
                        "FullSquare = '{6}', " +
                        "HouseType = '{7}', " +
                        "Type = '{8}', " +
                        "Intersection = '{9}', " +
                        "WindowsView = '{10}', " +
                        "YardWindows = {11}, " +
                        "IsolatedRooms = {12}, " +
                        "Redevelopment = {13}, " +
                        "LegalizedRedevelopment = {14}, " +
                        "StoreysNumber = {15}, " +
                        "LiveSquare = '{16}', " +
                        "KitchenSquare = '{17}', " +
                        "Balcony = '{18}', " +
                        "BalconyGlazed = '{19}', " +
```

```
                            "Loggia = '{20}', " +
                            "LoggiaGlazed = '{21}', " +
                            "OuterWallsMaterial = '{22}', " +
                            "InteriorType = '{23}', " +
                            "InteriorState = '{24}', " +
                            "InteriorWindows = '{25}', " +
                            "InteriorsDoors = '{26}', " +
                            "FacingRooms1 = '{27}', " +
                            "FacingRooms2 = '{28}', " +
                            "FacingWC1 = '{29}', " +
                            "FacingWC2 = '{30}', " +
                            "HotWater = '{31}', " +
                            "Heating = '{32}', " +
                            "Gas = '{33}', " +
                            "Phone = '{34}', " +
                            "CookerType = '{35}', " +
                            "{36} " +
                            "AreaId = {37}, " +
                            "TopZone = {38}, " +
                            "Guide = '{39}', " +
                            "WC = '{40}', " +
                            "Equipment = '{41}', " +
                            "EquipmentState = '{42}', Comment = '{43}', FlatType
= '{44}', StreetType = '{45}', ClientFIO = '{46}', ClientContacts = '{47}',
State = '{48}' where Id = {49}",
                            streetComboBox.SelectedItem.ToString(),
houseTextBox.Text,          aptTextBox.Text,          floorNumericUpDown.Value,
costNumericUpDown.Value,
                            roomsComboBox.SelectedItem,
fullSquareNumericUpDown.Value,     houseTypeComboBox.SelectedItem,     p.Type,
intersectionTextBox.Text, windowsViewComboBox.SelectedItem,
                            yardWindowsCheckBox.Checked,
isolatedRoomsNumericUpDown.Value,                redevelopmentCheckBox.Checked,
legalizedRedevelopmentCheckBox.Checked, storeysNumberNumericUpDown.Value,
                            liveSquareNumericUpDown.Value,
kitchenSquareNumericUpDown.Value,      balconyComboBox.SelectedItem.ToString(),
balconyGlazedComboBox.SelectedItem.ToString(),
loggiaComboBox.SelectedItem.ToString(),
                            loggiaGlazedComboBox.SelectedItem.ToString(),
outerWallsMaterialComboBox.SelectedItem,     interiorTypeComboBox.SelectedItem,
interiorStateComboBox.SelectedItem,
                            interiorWindowsComboBox.SelectedItem,
interiorDoorsComboBox.SelectedItem,          facingRooms1ComboBox.SelectedItem,
facingRooms2ComboBox.SelectedItem,
                            facingWC1ComboBox.SelectedItem,
facingWC2ComboBox.SelectedItem,       hotWaterComboBox.SelectedItem.ToString(),
heatingComboBox.SelectedItem.ToString(), gasComboBox.SelectedItem.ToString(),
                            phoneComboBox.SelectedItem.ToString(),
coockerTypeComboBox.SelectedItem, contactDateStr, p.Areaid,
                            p.TopZone,                        guideTextBox.Text,
wcComboBox.SelectedItem,                        equipmentComboBox.SelectedItem,
equipmentStateComboBox.SelectedItem, commentRichTextBox.Text,
                            flatTypeComboBox.SelectedItem,
streetTypeComboBox.SelectedItem.ToString(),              clientFIOTextBox.Text,
clientContactsTextBox.Text, stateString, proposalId);
                        OleDbCommand   command   =   new   OleDbCommand(query,
Global.connection);
                        // если было отмечено поле "дата контакта"
```

```
                        if (contactDateDateTimePicker.Checked)
                        {
                            command.Parameters.Add("@contactDate",
System.Data.OleDb.OleDbType.Date).Value =
                                contactDateDateTimePicker.Value.Date;
                        }
                        // --
                        Global.connection.Open();
                        command.ExecuteNonQuery();
                        Global.connection.Close();

                        Global.mainForm.loadProposals();
                    }
                    this.Close();
                }

                private   void   EditProposalForm_FormClosing(object   sender,
FormClosingEventArgs e)
                {
                    if (!savePressed)
                    {
                        if (afterAdd)
                        {
                            if  (MessageBox.Show("Заявка  не  будет  добавлена!
Продолжить?", "", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation) ==
                                System.Windows.Forms.DialogResult.Yes)
                            {
                                Global.mainForm.deleteProposal(proposalId, "");
                                Global.mainForm.loadProposals();
                            }
                            else
                            {
                                e.Cancel = true;
                            }
                        }
                        else
                        {
                            if (MessageBox.Show("Отменить внесенные изменения и
закрыть окно?", "", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                                System.Windows.Forms.DialogResult.No)
                            {
                                e.Cancel = true;
                            }
                        }
                    }
                }

                private  void  areaComboBox_SelectedIndexChanged(object  sender,
EventArgs e)
                {
                    if (!firstLoad)
                    {
                        Global.mainForm.loadZonesByArea(topZoneComboBox,
Global.mainForm.getAreaIdByName(areaComboBox.SelectedItem.ToString()));
                        topZoneComboBox.SelectedIndex = 0;
                    }
                }
```

```
                private    void    checkBoxTorg_CheckedChanged(object    sender,
EventArgs e)
                {
                    if (checkBoxTorg.Checked)
                    { commentRichTextBox.AppendText("ТОРГ"); }
                    else {
                     commentRichTextBox.Text=
commentRichTextBox.Text.Replace("ТОРГ","");
                    }

                }
            }

        public partial class LoginForm : Form
            {
                public LoginForm()
                {
                    InitializeComponent();
                }

                private void cancelButton_Click(object sender, EventArgs e)
                {
                    this.Close();
                }

                private void enterButton_Click(object sender, EventArgs e)
                {
                    if (userTextBox.Text == Global.user && passwordTextBox.Text
== Global.password)
                    {
                        this.DialogResult                                    =
System.Windows.Forms.DialogResult.OK;
                    }
                    else
                    {
                        Global.showErrorMessage("Не верный логин/пароль");
                    }
                }
            }

        public partial class PhonesForm : Form
            {
                public PhonesForm()
                {
                    InitializeComponent();
                }

                private void closeButton_Click(object sender, EventArgs e)
                {
                    this.Close();
                }

                private void PhonesForm_Shown(object sender, EventArgs e)
                {
                    this.Cursor = Cursors.WaitCursor;
                    phonesDataGridView.AutoGenerateColumns = true;
                    loadPhones();
                    this.Cursor = Cursors.Default;
```

```
                }

                private void loadPhones()
                {
                    phonesDataSet.Clear();
                    String query = "select Id, PhoneNumber as [Номер] from
Phones order by Id";
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    dataAdapter.Fill(phonesDataSet);
                    Global.connection.Close();
                    phonesBindingSource.DataMember = "Table";
                    phonesDataGridView.DataSource = phonesBindingSource;
                    delButton.Enabled = (phonesDataGridView.RowCount != 0);
                }

                private void deletePhone(long id)
                {
                    String query = String.Format("delete * from Phones where Id
= {0}", id);
                    OleDbCommand command = new OleDbCommand(query,
Global.connection);
                    Global.connection.Open();
                    command.ExecuteNonQuery();
                    Global.connection.Close();
                }

                private void delButton_Click(object sender, EventArgs e)
                {
                    if (MessageBox.Show("Удалить номер?", "",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                        System.Windows.Forms.DialogResult.Yes)
                    {

deletePhone(Convert.ToInt64(phonesDataGridView.CurrentRow.Cells[0].Value));
                        loadPhones();
                    }
                }

                private void addButton_Click(object sender, EventArgs e)
                {
                    AddPhoneForm addForm = new AddPhoneForm();
                    if (addForm.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
                    {
                        addPhone(addForm.phoneNumberTextBox.Text);
                        loadPhones();
                    }
                }

                private void addPhone(String phoneNumber)
                {
                    String query = String.Format("insert into
Phones(PhoneNumber) values('{0}')", phoneNumber);
                    OleDbCommand command = new OleDbCommand(query,
Global.connection);
                    Global.connection.Open();
```

```
                command.ExecuteNonQuery();
                Global.connection.Close();
            }

            private void button1_Click(object sender, EventArgs e)
            {
                if (!checkPhone(new String[] {

                    textBox1.Text

                }))
                {
                    Global.showInfoMessage("Номер НАЙДЕН в списке телефонов
агенств!");
                }
                else { Global.showInfoMessage("Номер НЕ найден в списке
телефонов агенств!"); }
            }

            private Boolean checkPhone(String[] phones)
            {
                foreach (String phone in phones)
                {
                    String query = String.Format("select * from Phones where
PhoneNumber like '%{0}%'", phone);
                    //   MessageBox.Show(query);
                    OleDbDataAdapter        dataAdapter        =        new
OleDbDataAdapter(query, Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    if (ds.Tables[0].Rows.Count > 0)
                    {
                        return false;
                    }
                }
                return true;
            }

        }

    public partial class ProposalsReportForm : Form
        {
            public ProposalsReportForm(DataSet dataSet, String filterText)
            {
                InitializeComponent();
                DataSet ds = new DataSet();
                DataTable dt = new DataTable("Table");
                dt.Columns.Add(new DataColumn("Id"));
                dt.Columns.Add(new DataColumn("Type"));
                dt.Columns.Add(new DataColumn("Rooms"));
                dt.Columns.Add(new DataColumn("Street"));
                dt.Columns.Add(new DataColumn("House"));
                dt.Columns.Add(new DataColumn("PlacementDate"));
                dt.Columns.Add(new DataColumn("ContactDate"));
                dt.Columns.Add(new DataColumn("Cost"));
                dt.Columns.Add(new DataColumn("Floor"));
```

```
                    dt.Columns.Add(new DataColumn("StoreysNumber"));
                    dt.Columns.Add(new DataColumn("FullSquare"));
                    dt.Columns.Add(new DataColumn("LiveSquare"));
                    dt.Columns.Add(new DataColumn("KitchenSquare"));
                    dt.Columns.Add(new DataColumn("Balcony"));
                    dt.Columns.Add(new DataColumn("Loggia"));
                    dt.Columns.Add(new DataColumn("Phone"));
                    dt.Columns.Add(new DataColumn("ClientId"));
                    dt.Columns.Add(new DataColumn("HouseType"));
                    dt.Columns.Add(new DataColumn("Comment"));
                    dt.Columns.Add(new DataColumn("Apartment"));
                    dt.Columns.Add(new DataColumn("InteriorType"));
                    dt.Columns.Add(new DataColumn("IsolatedRooms"));
                    dt.Columns.Add(new DataColumn("TopZone"));
                    Dictionary<long,          String>          zones          =
Global.mainForm.loadZones();
                    foreach (DataRow row in dataSet.Tables[0].Rows)
                    {
                        String phone = "-";
                        String topZone = "-";
                        if (row[15].ToString() != "?" && row[15].ToString() !=
"отсутствует" && row[15].ToString() != "")
                        {
                            phone = "+";
                        }
                        if (row[22].ToString() != "0")
                        {
                            topZone                                          =
zones[Convert.ToInt64(row[22].ToString())];
                        }
                        dt.Rows.Add(row[0],  row[1],  row[2],  row[3],  row[4],
row[5], row[6], row[7],
                            row[8], row[9], row[10], row[11], row[12], row[13],
row[14], phone, row[16],
                            row[17],  row[18],  row[19],  filterText,  row[21],
topZone);
                    }
                    ds.Tables.Add(dt);

                    proposalsBindingSource.DataSource = ds;
                    proposalsBindingSource.DataMember = "Table";
                }

                private void ProposalsReportForm_Load(object sender, EventArgs
e)
                {
                    reportViewer1.RefreshReport();
                }
            }

        public partial class MainForm : Form
            {
                private String filterText = "УСЛОВИЕ ПОИСКА: все предложения";

                public MainForm()
                {
                    InitializeComponent();
                }
```

```
                public void loadZonesByArea(ComboBox control, long areaId)
                {
                    String query = String.Format("select Name from Zones where
AreaId = {0} order by Name", areaId);
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    control.Items.Clear();
                    control.Items.Add("?");
                    foreach (DataRow row in ds.Tables[0].Rows)
                    {
                        control.Items.Add(row[0].ToString());
                    }
                }
                public void loadZonesByArea(CheckedListBox control, long areaId)
                {
                    String query = String.Format("select Name from Zones where
AreaId = {0} order by Name", areaId);
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    control.Items.Clear();
                    control.Items.Add("?");
                    foreach (DataRow row in ds.Tables[0].Rows)
                    {
                        control.Items.Add(row[0].ToString());
                    }
                }

                public String getZoneNameById(long id)
                {
                    String query = String.Format("select Name from Zones where
Id = {0}", id);
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    return ds.Tables[0].Rows[0][0].ToString();
                }

                public long getZoneIdByNameAndArea(String name, long areaId)
                {
                    String query = String.Format("select Id from Zones where
Name = '"+name+"' and AreaId = "+areaId+" and id > 0 ");
                    //MessageBox.Show(query);
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
```

```
                dataAdapter.Fill(ds);
                Global.connection.Close();
                return Convert.ToInt64(ds.Tables[0].Rows[0][0]);
            }

            public void loadAreas(ComboBox control)
            {
                String query = String.Format("select Name from Areas order
by Name");
                OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                Global.connection.Open();
                DataSet ds = new DataSet();
                dataAdapter.Fill(ds);
                Global.connection.Close();
                control.Items.Clear();
                foreach (DataRow row in ds.Tables[0].Rows)
                {
                    control.Items.Add(row[0].ToString());
                }
            }

            public long getAreaIdByName(String name)
            {
                String query = String.Format("select Id from Areas where
Name = '{0}'", name);
                OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                Global.connection.Open();
                DataSet ds = new DataSet();
                dataAdapter.Fill(ds);
                Global.connection.Close();
                return Convert.ToInt64(ds.Tables[0].Rows[0][0]);
            }
            public long getZoneByName(String name)
            {
                String query = String.Format("SELECT idzone FROM SpravZones
where street = '{0}'", name);
                OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                Global.connection.Open();
                DataSet ds = new DataSet();
                dataAdapter.Fill(ds);
                Global.connection.Close();
                return Convert.ToInt64(ds.Tables[0].Rows[0][0]);
            }
            public long getAreaIdByStreet(String streetName)
            {
                String query = String.Format("select AreaId from Proposals
where Street = '{0}' and AreaId>0", streetName);
                OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                Global.connection.Open();
                DataSet ds = new DataSet();
                dataAdapter.Fill(ds);
                Global.connection.Close();
                return Convert.ToInt64(ds.Tables[0].Rows[0][0]);
            }
```

```
                public String getAreaNameById(long id)
                {
                    String query = String.Format("select Name from Areas where
Id = {0}", id);
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    return ds.Tables[0].Rows[0][0].ToString();
                }

                public void loadStreets(ComboBox control)
                {
                    String query = String.Format("select street from SpravZones
order by street");
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    control.Items.Clear();
                    foreach (DataRow row in ds.Tables[0].Rows)
                    {
                        control.Items.Add(row[0].ToString());
                    }
                }

                public void loadStreetTypes(ComboBox control)
                {
                    String query = String.Format("select StreetType from
Proposals group by StreetType order by StreetType");
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    control.Items.Clear();
                    foreach (DataRow row in ds.Tables[0].Rows)
                    {
                        control.Items.Add(row[0].ToString());
                    }
                }

                private void mainTabControl_SelectedIndexChanged(object sender,
EventArgs e)
                {
                    if (mainTabControl.SelectedIndex == 1) // Спрос
                    {
                        mainTabControl.SelectedIndex = 0;
                        return;
                    }
                }
```

```
                public Dictionary<long, String> loadZones()
                {
                    String query = String.Format("select Id, Name from Zones");
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    Dictionary<long, String> zones = new Dictionary<long,
String>();
                    foreach (DataRow row in ds.Tables[0].Rows)
                    {
                        zones.Add(Convert.ToInt64(row[0].ToString()),
row[1].ToString());
                    }
                    return zones;
                }

                public void loadProposals()
                {
                    this.Cursor = Cursors.WaitCursor;
                    proposalsDataSet.Clear();
                    String query = "select Id as [№], Type as [Тип], Rooms as
[Комнат], Street as [Улица], House as [Дом], " +
                        "PlacementDate as [Дата размещ], ContactDate as [Дата
контакта], Cost as [Цена], Floor as [Этаж], " +
                        "StoreysNumber as [Этажность], FullSquare as [Площадь
(общая)], LiveSquare as [Площадь (жилая)], " +
                        "KitchenSquare as [Площадь (кухня)], Balcony as [Б],
Loggia as [Л], Phone as [Тел], " +
                        "ClientContacts as [Контакты], " +
                        "ClientFIO as [Продавец], Comment as [для печати],
Apartment as [Квартира], Actived, IsolatedRooms, TopZone as [Топозона], State
as Статус " +
                        "from Proposals order by Rooms, Cost";
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    dataAdapter.Fill(proposalsDataSet);
                    Global.connection.Close();
                    proposalsBindingSource.DataMember = "Table";
                    proposalsDataGridView.DataSource = proposalsBindingSource;
                    this.Cursor = Cursors.Default;
                }

                private void loadDemand()
                {
                    this.Cursor = Cursors.WaitCursor;
                    demandDataSet.Clear();
                    String query = "select Id as [№], ProposalId, " +
                        "(select Clients.FIO from Clients where Clients.Id =
Demand.ClientId) as [Покупатель], " +
                        "(select Clients.Contacts from Clients where Clients.Id
= Demand.ClientId) as [Контакты], ApplicationDate as [Дата подачи] " +
                        "from Demand order by Id";
                    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
```

```
                Global.connection.Open();
                dataAdapter.Fill(demandDataSet);
                Global.connection.Close();
                demandBindingSource.DataMember = "Table";
                demandDataGridView.DataSource = demandBindingSource;
                this.Cursor = Cursors.Default;
            }

            private void loadDemand(long proposalId)
            {
                this.Cursor = Cursors.WaitCursor;
                demandDataSet.Clear();
                String query = String.Format("select Id as [№], ProposalId,
" +
                    "(select Clients.FIO from Clients where Clients.Id =
Demand.ClientId) as [Покупатель], " +
                    "(select Clients.Contacts from Clients where Clients.Id
= Demand.ClientId) as [Контакты], ApplicationDate as [Дата подачи] " +
                    "from Demand where ProposalId = {0} order by Id",
proposalId);
                OleDbDataAdapter dataAdapter = new OleDbDataAdapter(query,
Global.connection);
                Global.connection.Open();
                dataAdapter.Fill(demandDataSet);
                Global.connection.Close();
                demandBindingSource.DataMember = "Table";
                demandDataGridView.DataSource = demandBindingSource;
                this.Cursor = Cursors.Default;
            }

            private void checkTables()
            {
                Global.connection.Open();
                DataTable schema = Global.connection.GetOleDbSchemaTable(
                    OleDbSchemaGuid.Tables, new object[] { null, null, null,
"TABLE" });
                Global.connection.Close();
                List<String> tables = new List<string>();
                foreach (DataRow row in schema.Rows)
                {
                    tables.Add((String)row["TABLE_NAME"]);
                }
                Boolean exists = false;
                foreach (String table in tables)
                {
                    if (table == "Phones")
                    {
                        exists = true;
                        break;
                    }
                }
                if (!exists)
                {
                    String[] columns = new String[] {
                        "[Id] COUNTER PRIMARY KEY",
                        "[PhoneNumber] Text (20)"
                    };
```

```
                        String query = String.Concat("CREATE TABLE Phones (",
String.Join(",", columns), ")");
                        Global.connection.Open();
                        OleDbCommand command = new OleDbCommand(query,
Global.connection);
                        command.ExecuteNonQuery();
                        Global.connection.Close();
                    }
                }

                private void editProposalButton_Click(object sender, EventArgs
e)
                {
                    editProposal(getCurrentProposalId(), false);
                }

                private void editProposal(long id, Boolean afterAdd)
                {
                    EditProposalForm eForm = new EditProposalForm(id, afterAdd);

                    eForm.ShowDialog();
                }

                public void deleteProposal(long id, String reason)
                {
                    String query = String.Format("update Proposals set Actived =
false, DeleteReason = '{0}' where Id = {1}", reason, id);
                    OleDbCommand command = new OleDbCommand(query,
Global.connection);
                    Global.connection.Open();
                    command.ExecuteNonQuery();
                    Global.connection.Close();
                }

                private void delProposalButton_Click(object sender, EventArgs e)
                {
                    deleteProposal();
                }

                private void deleteProposal()
                {
                    DeleteProposalForm dForm = new DeleteProposalForm();
                    if (dForm.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
                    {
                        String reason = "";
                        if (dForm.radioButton1.Checked) reason =
dForm.radioButton1.Text;
                        if (dForm.radioButton2.Checked) reason =
dForm.radioButton2.Text;
                        if (dForm.radioButton3.Checked) reason =
dForm.radioButton3.Text;
                        if (dForm.radioButton4.Checked) reason =
dForm.radioButton4.Text;
                        deleteProposal(getCurrentProposalId(), reason);
                        loadProposals();
                    }
                }
```

```
                private long getCurrentProposalId()
                {
                    return
Convert.ToInt64(proposalsDataGridView.CurrentRow.Cells[0].Value);
                }

                private long getLastProposalId()
                {
                    String query = String.Format("select top 1 Id from Proposals
order by Id desc");
                    OleDbDataAdapter  dataAdapter  =  new  OleDbDataAdapter(query,
Global.connection);
                    Global.connection.Open();
                    DataSet ds = new DataSet();
                    dataAdapter.Fill(ds);
                    Global.connection.Close();
                    return Convert.ToInt64(ds.Tables[0].Rows[0][0]);
                }

                private    void    proposalsDataGridView_MouseDoubleClick(object
sender, MouseEventArgs e)
                {
                    editProposal(getCurrentProposalId(), false);
                }
                private bool otchet2 = false;
                private    void    searchProposalsButton_Click(object    sender,
EventArgs e)
                {
                    SearchProposalForm sForm = new SearchProposalForm();
                    if                  (sForm.ShowDialog()                  ==
System.Windows.Forms.DialogResult.OK)
                    {
                        this.Cursor = Cursors.WaitCursor;
                        List<String> filters = new List<string>();
                        List<String> filtersText = new List<string>();
                        if (sForm.byTypeGroupBox.Enabled)
                        {
                            String type = "";
                            if (sForm.saleRadioButton.Checked) type = "п";
                            else type = "a";
                            filters.Add("Type = '" + type + "'");
                            filtersText.Add("тип предложения - " + type);
                        }
                        if            (sForm.byAreaGroupBox.Enabled            ||
sForm.byAreaGroupBox2.Enabled || sForm.byAreaGroupBox3.Enabled)
                        {
                            List<String> filters_tmp = new List<string>();
                            String text = "район - ";
                            if (sForm.byAreaGroupBox.Enabled)
                            {
                                filters_tmp.Add("AreaId       =       "       +
Global.mainForm.getAreaIdByName(

sForm.areaComboBox.SelectedItem.ToString()));
                                text += sForm.areaComboBox.SelectedItem + ", ";
                            }
                            if (sForm.byAreaGroupBox2.Enabled)
```

```
                            {
                                filters_tmp.Add("AreaId       =       "       +
Global.mainForm.getAreaIdByName(

sForm.areaComboBox2.SelectedItem.ToString()));
                                text += sForm.areaComboBox2.SelectedItem + ", ";
                            }
                            if (sForm.byAreaGroupBox3.Enabled)
                            {
                                filters_tmp.Add("AreaId       =       "       +
Global.mainForm.getAreaIdByName(

sForm.areaComboBox3.SelectedItem.ToString()));
                                text += sForm.areaComboBox3.SelectedItem + ", ";
                            }
                            String filter = "(" + filters_tmp[0];
                            for (int o = 1; o < filters_tmp.Count; o++)
                            {
                                filter += " or " + filters_tmp[o];
                            }
                            filter += ")";
                            filters.Add(filter);
                            filtersText.Add(text);
                        }
                        if (sForm.byRoomsGroupBox.Enabled)
                        {
                            filters.Add("Rooms          =          '"          +
sForm.roomsComboBox.SelectedItem + "'");
                            filtersText.Add("кол-во     комнат     -     "     +
sForm.roomsComboBox.SelectedItem);
                        }
                        if (sForm.byFloorFromGroupBox.Enabled)
                        {
                            filters.Add("Floor          >=          "          +
Convert.ToString(sForm.floorFromNumericUpDown.Value));
                            filtersText.Add("этаж       с       -       "       +
sForm.floorFromNumericUpDown.Value);
                        }
                        if (sForm.byFloorToGroupBox.Enabled)
                        {
                            filters.Add("Floor          <=          "          +
Convert.ToString(sForm.floorToNumericUpDown.Value));
                            filtersText.Add("этаж       по       -       "       +
sForm.floorToNumericUpDown.Value);
                        }
                        if (sForm.byStoreysGroupBox.Enabled)
                        {
                            filters.Add("StoreysNumber        =        "        +
Convert.ToString(sForm.storeysNumericUpDown.Value));
                            filtersText.Add("этажность        -        "        +
sForm.storeysNumericUpDown.Value);
                        }
                        if (sForm.byCostFromGroupBox.Enabled)
                        {
                            filters.Add("Cost           >=          "          +
sForm.costFromNumericUpDown.Value.ToString().Replace(",",".."));
                            filtersText.Add("цена       от       -       "       +
sForm.costFromNumericUpDown.Value);
```

```
                        }
                        if (sForm.byCostToGroupBox.Enabled)
                        {
                            filters.Add("Cost          <=          "          +
sForm.costToNumericUpDown.Value.ToString().Replace(",", "."));
                            filtersText.Add("цена      до      -      "      +
sForm.costToNumericUpDown.Value);
                        }
                        if (sForm.byFlatTypeGroupBox.Enabled)
                        {
                            filters.Add("FlatType         =         '"         +
sForm.flatTypeComboBox.SelectedItem.ToString() + "'");
                            filtersText.Add("тип     квартиры     -     "     +
sForm.flatTypeComboBox.SelectedItem.ToString());
                        }
                        if (sForm.byStreetGroupBox.Enabled)
                        {
                            filters.Add("Street          =          '"          +
sForm.streetComboBox.SelectedItem.ToString() + "'");
                            filtersText.Add("улица         -         "         +
sForm.flatTypeComboBox.SelectedItem.ToString());
                        }
                        if (sForm.byHouseTypeGroupBox.Enabled)
                        {
                            filters.Add("HouseType        =         '"         +
sForm.houseTypeComboBox.SelectedItem.ToString() + "'");
                            filtersText.Add("тип      дома      -      "      +
sForm.houseTypeComboBox.SelectedItem.ToString());
                        }
                        if (sForm.groupBoxHouse.Enabled)
                        {
                            filters.Add("House = '" + sForm.houseTextBox.Text +
"'");
                            filtersText.Add("№      дома      -      "      +
sForm.houseTextBox.Text);
                        }
                        if (sForm.byPhone1GroupBox.Enabled)
                        {
                            filters.Add("ClientContacts     like     '%"     +
sForm.phone1TextBox.Text + "%'");
                            filtersText.Add("телефон      1      -      "      +
sForm.phone1TextBox.Text);
                        }
                        if (sForm.byPhone2GroupBox.Enabled)
                        {
                            filters.Add("ClientContacts     like     '%"     +
sForm.phone2TextBox.Text + "%'");
                            filtersText.Add("телефон      2      -      "      +
sForm.phone2TextBox.Text);
                        }
                        if (sForm.byIdGroupBox.Enabled)
                        {
                            filters.Add("Id = " + sForm.byIdMaskedTextBox.Text);
                            filtersText.Add("номер     заявки     -     "     +
sForm.byIdMaskedTextBox.Text);
                        }
                        if (sForm.byActivedGroupBox.Enabled)
                        {
```

```
                            if (sForm.activedComboBox.SelectedIndex == 0)
                            {
                                filters.Add("Actived = true");
                                filtersText.Add("активность  предложения  -  "  +
sForm.activedComboBox.SelectedItem.ToString());
                            }
                            else if (sForm.activedComboBox.SelectedIndex == 1)
                            {
                                filters.Add("Actived = false");
                                filtersText.Add("активность  предложения  -  "  +
sForm.activedComboBox.SelectedItem.ToString());
                            }
                        }
                        if (sForm.bySquaregroupBox.Enabled)
                        {
                            filters.Add("FullSquare     >"     +Convert.ToInt32(
sForm.fullSquareNumericUpDown.Text) );
                            filtersText.Add("общая     площадь     -     "     +
sForm.fullSquareNumericUpDown.Text);
                            filters.Add("LiveSquare     >"   +   Convert.ToInt32(
sForm.liveSquareNumericUpDown.Text ));
                            filtersText.Add("жилая     площадь     -     "     +
sForm.liveSquareNumericUpDown.Text);
                            filters.Add("KitchenSquare    >"  +  Convert.ToInt32(
sForm.kitchenSquareNumericUpDown.Text));
                            filtersText.Add("кухни     площадь     -     "     +
sForm.kitchenSquareNumericUpDown.Text);
                        }
                        if            (sForm.byZoneGroupBox.Enabled            ||
sForm.byZoneGroupBox2.Enabled || sForm.byZoneGroupBox3.Enabled)
                        {
                            if   (sForm.zoneComboBox.SelectedIndex   !=   0   ||
sForm.zoneComboBox2.SelectedIndex  != 0  ||  sForm.zoneComboBox3.SelectedIndex  !=
0)
                            {
                                List<String> filters_tmp = new List<string>();
                                String text = "топозона - ";
                                if  (sForm.zoneComboBox.SelectedIndices.Count  !=
0)
                                {
                                    for     (int     i     =     0;     i     <
sForm.zoneComboBox.CheckedItems.Count; i++)
                                    {

                                        filters_tmp.Add("TopZone     =     "     +
Global.mainForm.getZoneIdByNameAndArea(sForm.zoneComboBox.CheckedItems[i].ToStr
ing(),
                                            Global.mainForm.getAreaIdByName(

sForm.areaComboBox.SelectedItem.ToString())));
                                        text                                    +=
sForm.zoneComboBox.CheckedItems[i].ToString() + ", ";
                                    }
                                }
                                if (sForm.zoneComboBox2.SelectedIndex != 0)
                                {
```

```
                                    filters_tmp.Add("TopZone      =      "      +
Global.mainForm.getZoneIdByNameAndArea(sForm.zoneComboBox2.SelectedItem.ToStrin
g(),
                                        Global.mainForm.getAreaIdByName(

sForm.areaComboBox2.SelectedItem.ToString())));
                                    text                                        +=
sForm.zoneComboBox2.SelectedItem.ToString() + ", ";
                                }
                                if (sForm.zoneComboBox3.SelectedIndex != 0)
                                {
                                    filters_tmp.Add("TopZone      =      "      +
Global.mainForm.getZoneIdByNameAndArea(sForm.zoneComboBox3.SelectedItem.ToStrin
g(),
                                        Global.mainForm.getAreaIdByName(

sForm.areaComboBox3.SelectedItem.ToString())));
                                    text                                        +=
sForm.zoneComboBox3.SelectedItem.ToString() + ", ";
                                }
                                String filter = "(" + filters_tmp[0];
                                for (int o = 1; o < filters_tmp.Count; o++)
                                {
                                    filter += " or " + filters_tmp[o];
                                }
                                filter += ")";
                                filters.Add(filter);
                                filtersText.Add(text);
                            }
                            else
                            {
                                otchet2 = true;
                            }
                        }
                        // поиск по дате
                        if (sForm.byDateFromGroupBox.Enabled)
                        {
                            filters.Add("PlacementDate >= @byDateFrom");
                            filtersText.Add("дата      от      -      "      +
sForm.dateFromDateTimePicker.Value.ToString());
                        }
                        if (sForm.byDateToGroupBox.Enabled)
                        {
                            filters.Add("PlacementDate <= @byDateTo");
                            filtersText.Add("дата      до      -      "      +
sForm.dateToDateTimePicker.Value.ToString("dd.MM.yyyy"));
                        }
                        // --
                        if (sForm.byStateGroupBox.Enabled)
                        {
                            String stateSearchText = "";
                            if (sForm.stateComboBox.SelectedItem != null)
                            {
                                if (sForm.stateComboBox.SelectedIndex != 0)
                                {
                                    stateSearchText                             =
sForm.stateComboBox.SelectedItem.ToString();
                                }
```

```
                                filters.Add("State = '" + stateSearchText +
"'");
                                filtersText.Add("состояние заявки - " +
stateSearchText);
                            }
                        }
                        String query = "select Id as [№], Type as [Тип], Rooms
as [Комнат], Street as [Улица], House as [Дом], " +
                        "PlacementDate as [Дата размещ], ContactDate as [Дата
контакта], Cost as [Цена], Floor as [Этаж], " +
                        "StoreysNumber as [Этажность], FullSquare as [Площадь
(общая)], LiveSquare as [Площадь (жилая)], " +
                        "KitchenSquare as [Площадь (кухня)], Balcony as [Б],
Loggia as [Л], Phone as [Тел], " +
                        "ClientContacts as [Контакты], " +
                        "ClientFIO as [Продавец], Comment as [для печати],
Apartment as [Квартира], Actived, IsolatedRooms, TopZone as [Топозона] from
Proposals";
                        if (filters.Count > 0)
                        {
                            query += " where " + filters[0];
                            filterText = "УСЛОВИЕ ПОИСКА: " + filtersText[0] +
"; ";
                            for (int i = 1; i < filters.Count; i++)
                            {
                                query += " and " + filters[i];
                                filterText += filtersText[i] + "; ";
                            }
                        }
                        query += " order by Rooms, Cost";
                        proposalsDataSet.Clear();
                        OleDbDataAdapter dataAdapter = new
OleDbDataAdapter(query, Global.connection);
                        // если был выбран фильтр по дате
                        if (sForm.byDateFromGroupBox.Enabled)
                        {

dataAdapter.SelectCommand.Parameters.Add("@byDateFrom",
System.Data.OleDb.OleDbType.Date).Value =
                                sForm.dateFromDateTimePicker.Value;
                        }
                        if (sForm.byDateToGroupBox.Enabled)
                        {

dataAdapter.SelectCommand.Parameters.Add("@byDateTo",
System.Data.OleDb.OleDbType.Date).Value =
                                sForm.dateToDateTimePicker.Value;
                        }
                        // --
                        Global.connection.Open();
                        dataAdapter.Fill(proposalsDataSet);
                        Global.connection.Close();
                        proposalsDataGridView.AutoGenerateColumns = true;
                        proposalsBindingSource.DataMember = "Table";
                        proposalsDataGridView.DataSource =
proposalsBindingSource;
                        showAllButton.Visible = true;
                        redrawRows();
```

```
                this.Cursor = Cursors.Default;
            }
        }

        private void showAllButton_Click(object sender, EventArgs e)
        {
            loadProposals();
            showAllButton.Visible = false;
            filterText = "УСЛОВИЕ ПОИСКА: все предложения";
            redrawRows();
        }

        private    void    reportProposalsButton_Click(object    sender,
EventArgs e)
        {
            if (otchet2)
            {

                ProposalsReportForm2         rForm2         =         new
ProposalsReportForm2(proposalsDataSet, filterText);
                rForm2.ShowDialog();
            }
            else
            {


                ProposalsReportForm         rForm         =         new
ProposalsReportForm(proposalsDataSet, filterText);
                rForm.ShowDialog();
            }
        }

        private   void   proposalsDataGridView_KeyDown(object   sender,
KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Delete)
            {
                deleteProposal();
            }
        }

        private void deleteDemand()
        {
            if       (MessageBox.Show("Удалить       заявку?",       "",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                System.Windows.Forms.DialogResult.Yes)
            {
                deleteDemand(getCurrentDemandId());
                loadProposals();
            }
        }

        public void deleteDemand(long id)
        {
            String query = String.Format("delete * from Demand where Id
= {0}", id);
            OleDbCommand    command    =    new    OleDbCommand(query,
Global.connection);
```

```
                Global.connection.Open();
                command.ExecuteNonQuery();
                Global.connection.Close();
            }

            private void searchDemandButton_Click(object sender, EventArgs
e)
            {
                SearchDemandForm sForm = new SearchDemandForm();
                if (sForm.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
                {
                    this.Cursor = Cursors.WaitCursor;
                    List<String> filters = new List<string>();
                    if (sForm.byFIOGroupBox.Enabled)
                    {
                        filters.Add(
                            "(select Clients.FIO from Clients where
Demand.ClientId = Clients.Id) like '%" +
                            sForm.clientFIOTextBox.Text + "%'");
                    }
                    if (sForm.byContactGroupBox.Enabled)
                    {
                        filters.Add(
                            "(select Clients.Contacts from Clients where
Demand.ClientId = Clients.Id) like '%" +
                            sForm.clientContactTextBox.Text + "%'");
                    }
                    if (sForm.byDateFromGroupBox.Enabled)
                    {
                        filters.Add("ApplicationDate >= #" +
sForm.dateFromDateTimePicker.Value.ToString("dd:MM:yyyy") + "#");
                    }
                    if (sForm.byDateToGroupBox.Enabled)
                    {
                        filters.Add("ApplicationDate <= #" +
sForm.dateToDateTimePicker.Value.ToString("dd:MM:yyyy") + "#");
                    }

                    String query = "select Id as [заявка №], ProposalId, " +
                    "(select Clients.FIO from Clients where Clients.Id =
Demand.ClientId) as [Покупатель], " +
                    "(select Clients.Contacts from Clients where Clients.Id
= Demand.ClientId) as [Контакты], ApplicationDate as [Дата подачи] " +
                    "from Demand";
                    if (filters.Count > 0)
                    {
                        query += " where " + filters[0];
                        for (int i = 1; i < filters.Count; i++)
                        {
                            query += " and " + filters[i];
                        }
                    }
                    query += " order by Id";
                    demandDataSet.Clear();
                    OleDbDataAdapter dataAdapter = new
OleDbDataAdapter(query, Global.connection);
                    Global.connection.Open();
```

```
                    dataAdapter.Fill(demandDataSet);
                    Global.connection.Close();
                    demandDataGridView.AutoGenerateColumns = true;
                    demandBindingSource.DataMember = "Table";
                    demandDataGridView.DataSource = demandBindingSource;
                    showAllDemandButton.Visible = true;
                    this.Cursor = Cursors.Default;
                }
            }

            private void toolStripButton3_Click(object sender, EventArgs e)
            {
                removeDuplicates();
                Global.showInfoMessage("Реорганизация ядра базы данных
произошла успешно.\nИзменения вступят в силу после повторного запуска
программы");
                Application.Exit();
            }

            public void removeDuplicates()
            {
                String query = String.Format("update Proposals set Comment =
" +
                    " Left([Comment], Len([Comment]) / 2) " +
                    " where Id <= 25338");
                OleDbCommand command = new OleDbCommand(query,
Global.connection);
                Global.connection.Open();
                command.ExecuteNonQuery();
                Global.connection.Close();
            }
        }
```

ДОДАТОК Д

Слайди презентації

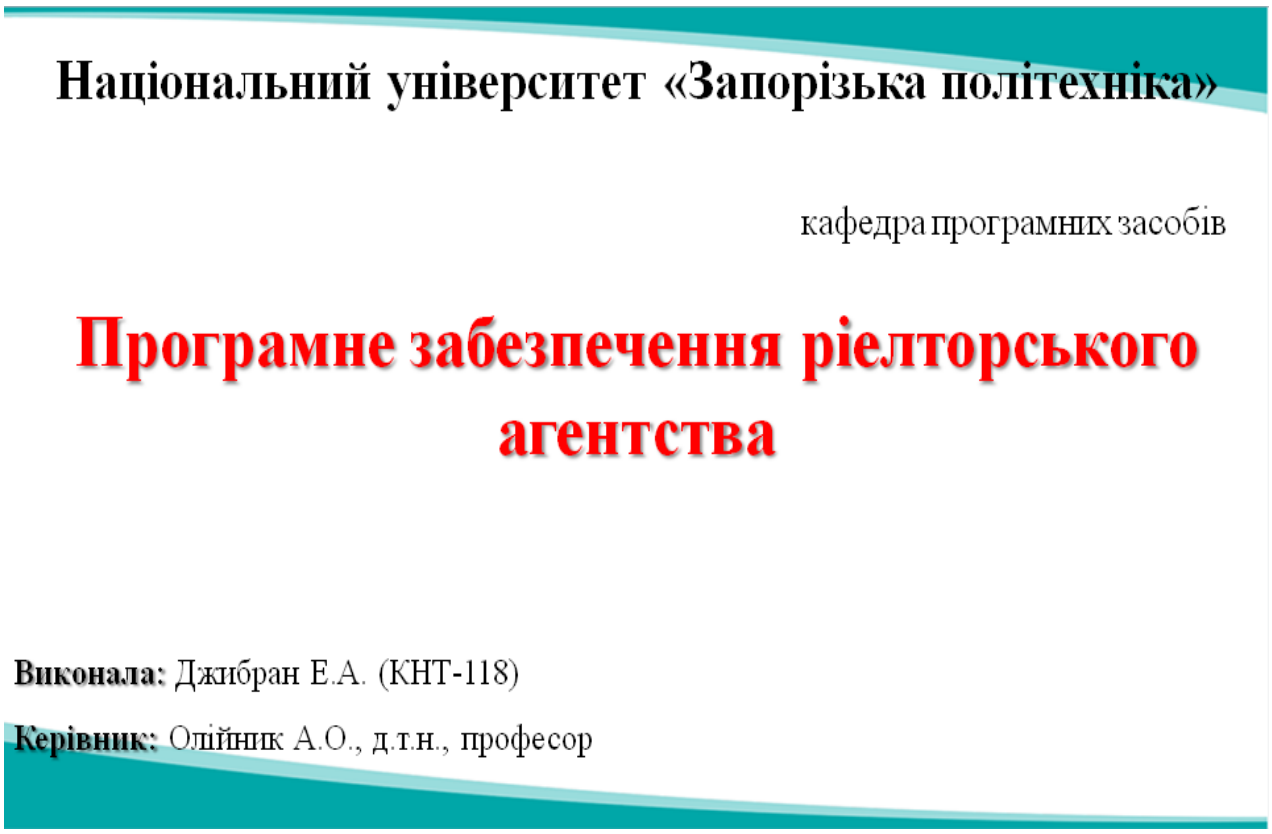


Рисунок Д.1 – Слайд 1

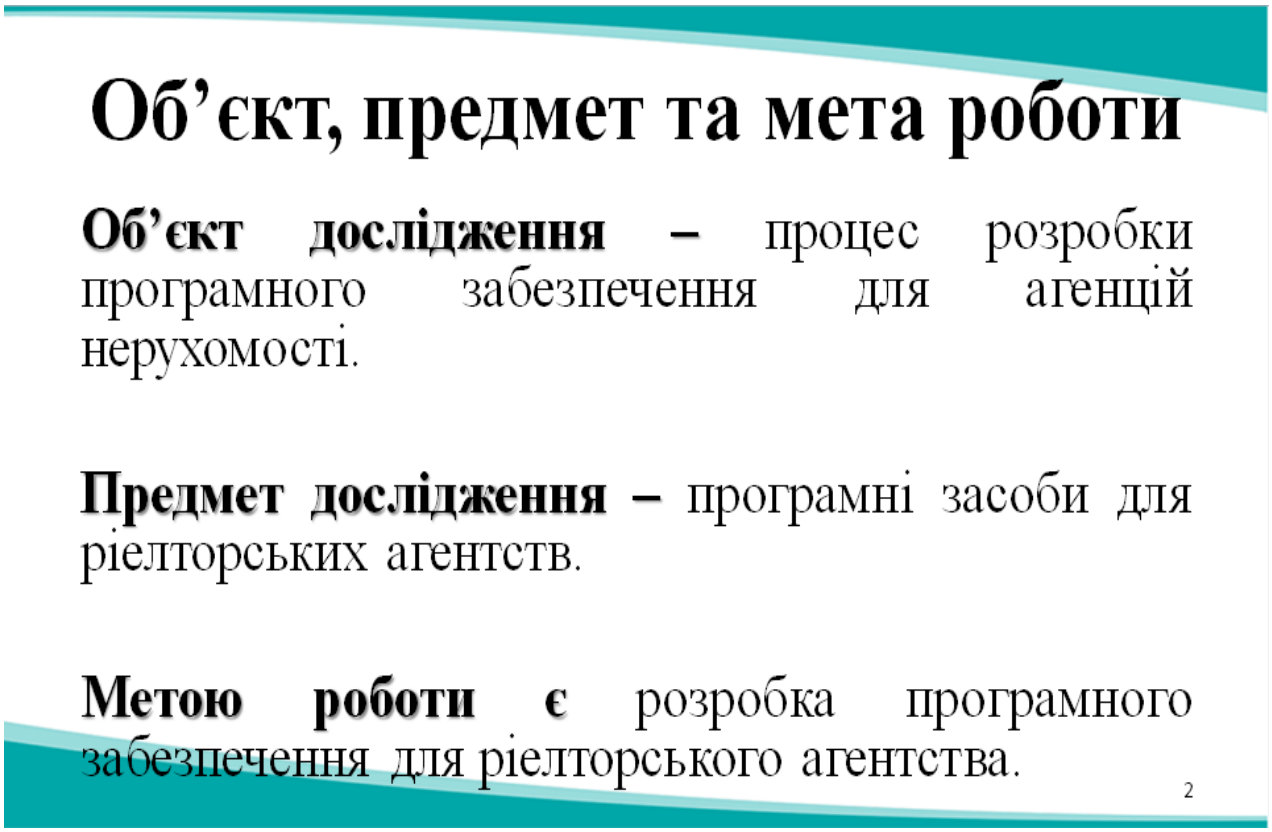


Рисунок Д.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети в роботі необхідно розв'язати такі задачі:

– виконати аналіз предметної області та програмних засобів для агентств нерухомості;

– здійснити проєктування програмного забезпечення ріелторського агентства;

– створити програмне забезпечення ріелторського агентства;

– виконати тестування розробленого програмного забезпечення ріелторського агентства.

3

Рисунок Д.3 – Слайд 3

Порівняння існуючих аналогів

Критерій, за яким порівнюють	Zillow Premier Agent	BoldLeads	LionDesk	Top Producer	Propertybase
Можливість пошуку нерухомості на карті	+	–	–	–	–
Підтримка списку потенційних клієнтів	+	+	+	–	+
Маркетингові засоби для залучення клієнтів	–	+	–	+	+
Мобільний застосунок	+	–	+	–	–
Можливість ведення прихованого списку нерухомості	–	–	–	–	–
Безкоштовність	–	–	–	–	–

4

Рисунок Д.4 – Слайд 4

Функціональні вимоги до розробки

Програмне забезпечення ріелторського агентства повинно задовольняти таким функціональним вимогам:

– можливість реєстрації та ідентифікації у програмі;

– підтримка операцій по роботі з даними агентства про нерухомість (додавання нових об'єктів, редагування даних про існуючі об'єкти, видалення непотрібної інформації та ін.);

– формування звітів по заданих критеріях;

– підтримка друку необхідної інформації.

5

Рисунок Д.5 – Слайд 5

Обґрунтування вибору мови програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	C++	Java
Обробка баз даних	+	+–	+
Засоби створення інтерфейсу	+	+–	+
Забезпечення безпеки даних	+	+–	+
Зручність для створення програмного забезпечення ріелторського агентства	+	–	+–
Обробка винятків	+	–	+

6

Рисунок Д.6 – Слайд 6

Обґрунтування вибору середовища розробки

Критерій порівняння	Середовище розробки		
	Microsoft Visual Studio	MonoDevelop	SharpDevelop
Підтримка мови програмування C#	+	+–	+–
Зручність для створення ПЗ для ріелторського агентства	+	–	+–
Підтримка роботи з репозиторіями	+	–	+–
Робота з базами даних	+	+	+–
Швидкість створення та модифікації ПЗ	+	+–	+–

7

Рисунок Д.7 – Слайд 7

Обґрунтування вибору системи керування базами даних

Критерій порівняння СКБД	Системи керування базами даних		
	MS Access	MySQL	PostgreSQL
Підтримка засобів безпеки даних	+	+	+–
Зручність для створення програмного забезпечення ріелторського агентства	+	–	–
Низька вартість	+	+	–
Швидкість створення та модифікації баз даних	+	+	+–
Масштабованість	+	+–	+–

8

Рисунок Д.8 – Слайд 8

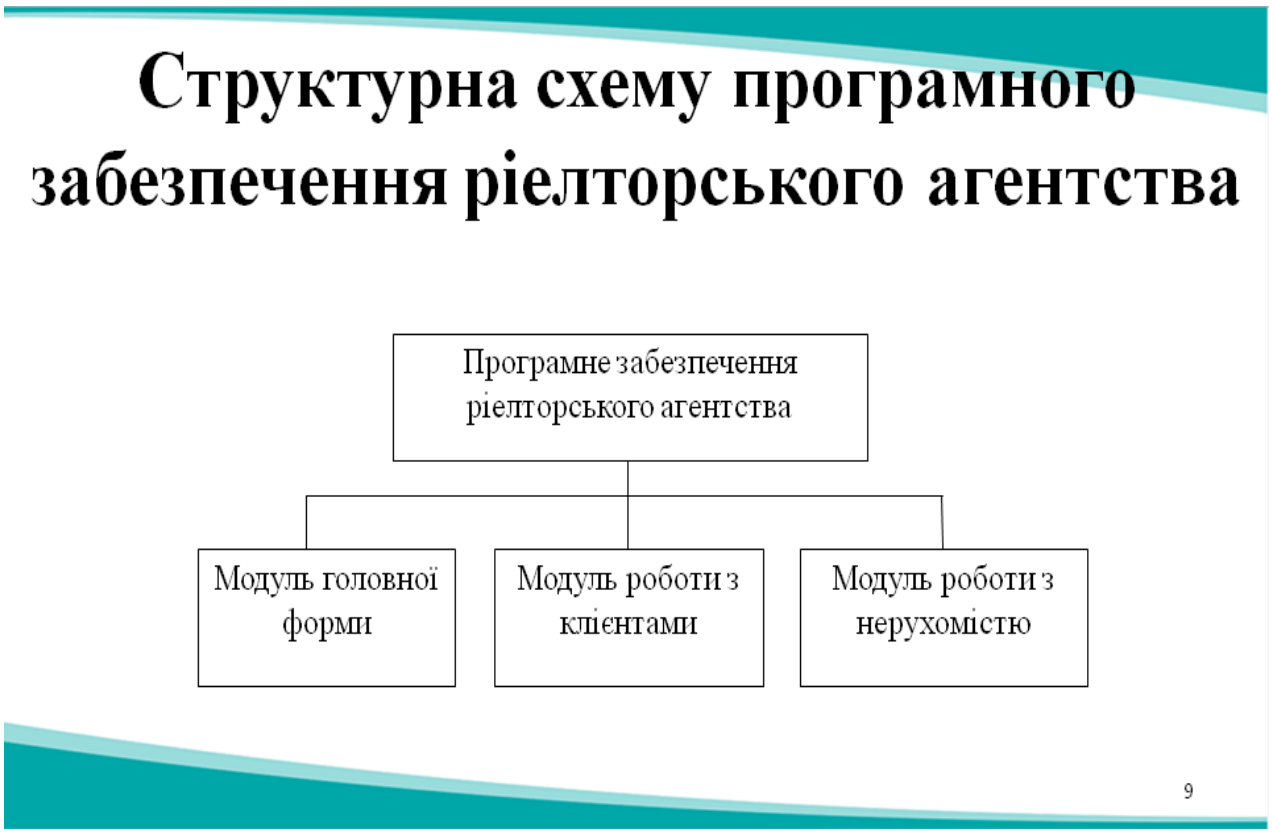


Рисунок Д.9 – Слайд 9

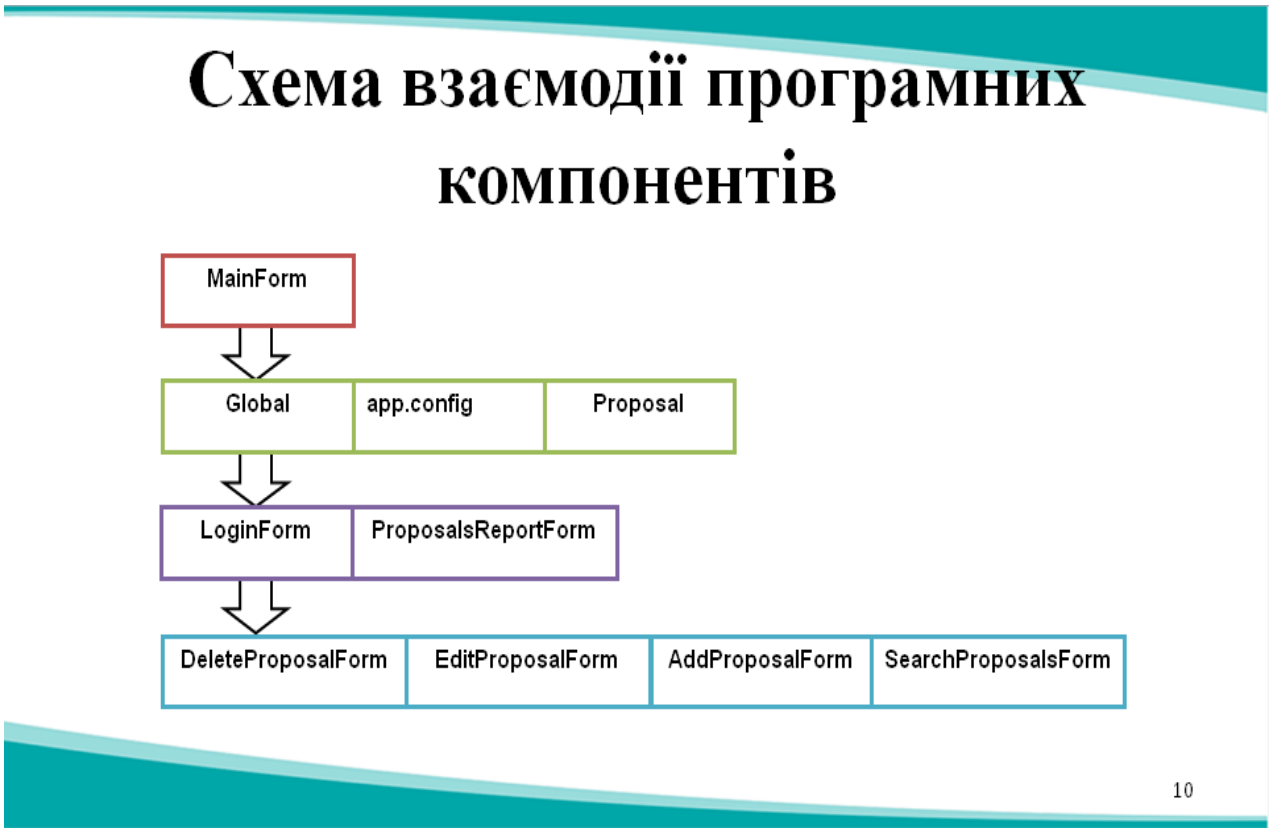


Рисунок Д.10 – Слайд 10

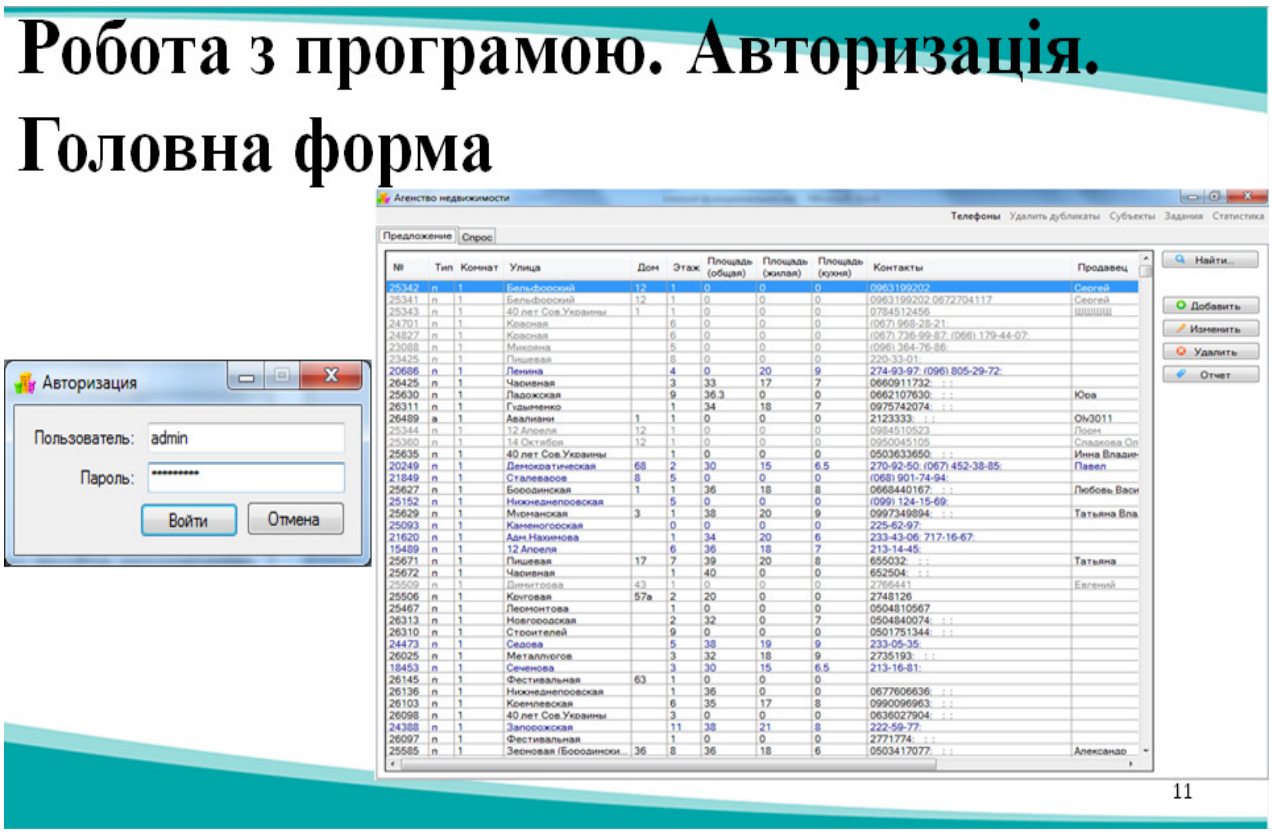


Рисунок Д.11 – Слайд 11

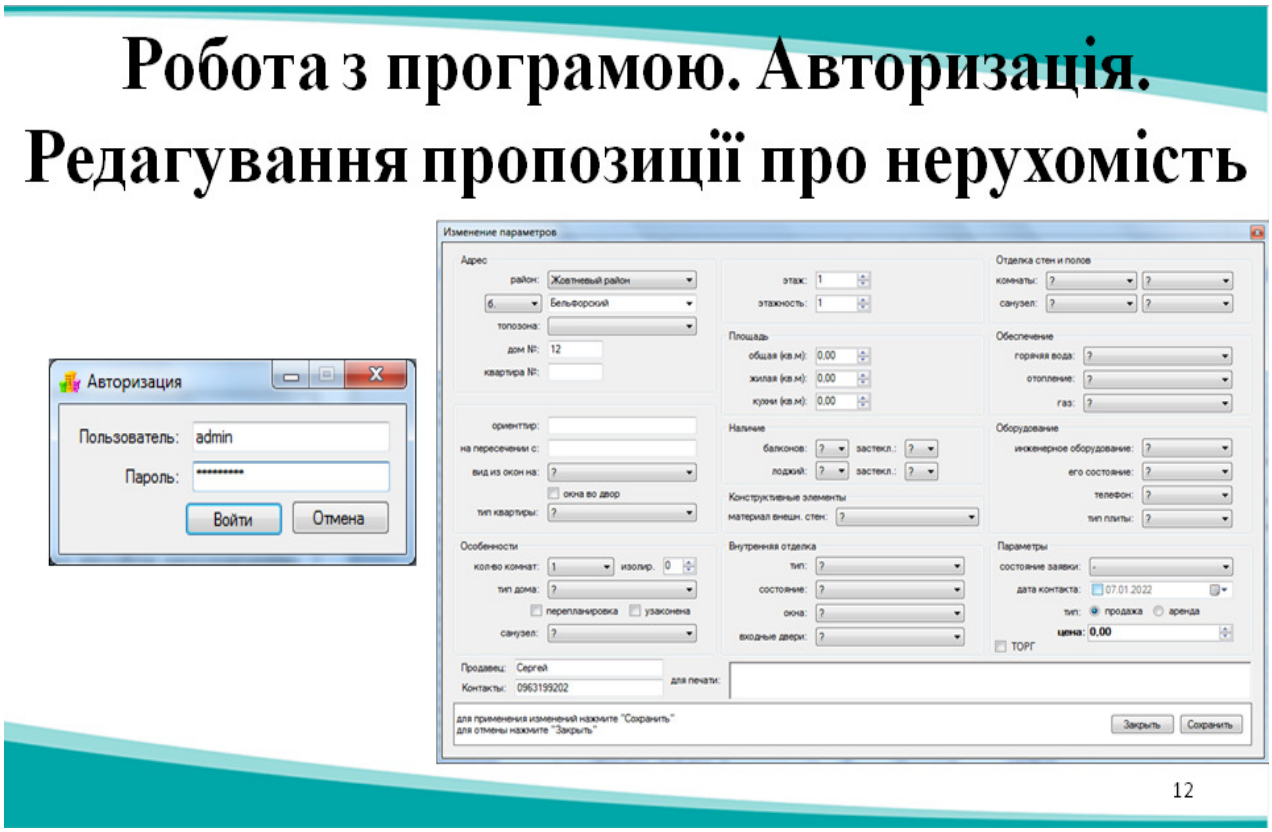
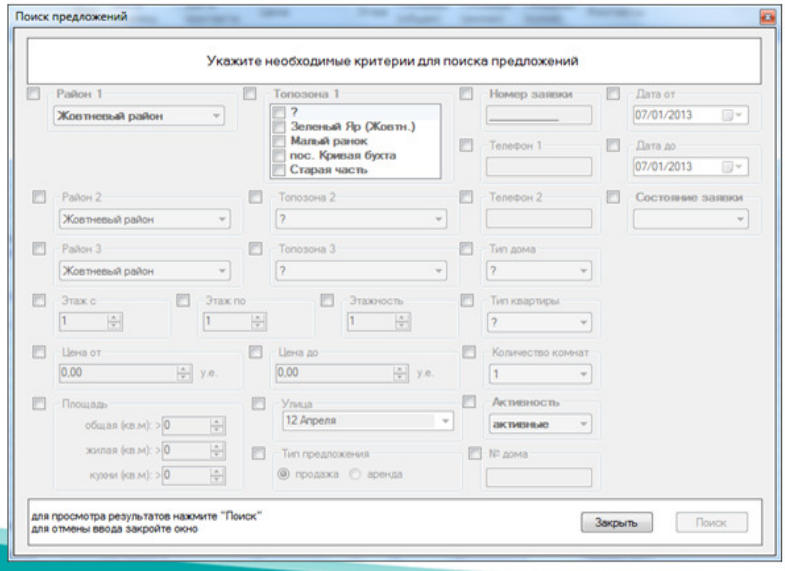


Рисунок Д.12 – Слайд 12

Робота з програмою. Пошук пропозиції про нерухомість



13

Рисунок Д.13 – Слайд 13

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для агенцій нерухомості.

Проаналізовано поняття про ріелторську діяльність та ріелторські агентства. Досліджено та проаналізовано програмні засоби для ріелторських агентств.

Розроблено програмне забезпечення ріелторського агентства. Для створення програмного забезпечення ріелторського агентства було обрано мову програмування C#.

При розробці програмного забезпечення ріелторського агентства було прийнято рішення використовувати середовище Microsoft Visual Studio, що включає інтегроване середовище розробки програмного забезпечення і багато інших інструментів тадозволяє створювати програмне забезпечення різного призначення на різних мовах програмування, зокрема, і на мові C#.

14

Рисунок Д.14 – Слайд 14

Висновки

Для створення програми ріелторського агентства обрано систему керування базами даних Microsoft Access.

Результати тестування програмного забезпечення ріелторського агентства показали, що програма працює коректно, вся інформація програмного забезпечення ріелторського агентства правильно виводиться в інтерфейсних формах та при реєстрації користувача, дані про об'єкти нерухомості коректно заносяться до бази даних.

Усі завдання випускної дипломної кваліфікаційної роботи бакалавра повністю виконано.

15

Рисунок Д.15 – Слайд 15

Дякую за увагу!

16

Рисунок Д.16 – Слайд 16