МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ
Государственное профессиональное образовательное учреждение
«Воркутинский политехнический техникум»

КУРСОВОЙ ПРОЕКТ

По дисциплине МДК.05.02 Разработка кода информационных систем

Учет риелторский операций

Выполнил студент гр. ИСП-20 /_____/ Мадонов М. А. /
(подпись) (Ф.И.О.)

ОЦЕНКА:_____

Дата:_____

ПРОВЕРИЛ

Научный руководитель_____/ Егоров Данил Павлович/
(подпись) (Ф.И.О.)

Воркута

2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тема данной курсовой работы является на сегодняшний день очень актуальной, т.к. внедрение информационных технологий в повседневную деятельность агентств недвижимости упрощает, ускоряет работу риелторов, а также помогает руководству данных агентств отслеживать все проходящие сделки в рамках агентства и соответственно не упускать выгоду для своего предприятия.

На сегодняшний день на первый план выходит информатизация и компьютеризация деятельности человека, начиная от переноса бумажного документооборота в электронный формат, заканчивая глобальным общением и обменом опыта среди подразделений компании посредством глобальной сети. Использование автоматизированных систем для агентств недвижимости позволяет отслеживать все проходящие сделки в агентстве, вести полную базу клиентов, в мгновение публиковать объявления недвижимости на разных площадках и на сайте агентства. Все более или менее крупные агентства на данный момент используют в своей работе разного рода информационные системы. Это уменьшает время расчётов, повышает качество обслуживания и т.д.

Объект - информационная система “Учет риелторских операций”.

Предмет – автоматизация учета риелторских операций.

Цель - разработать программное обеспечение автоматизированной информационной системы «Учет риелторских операций».

Задачи:

- выбрать инструментарий;
- спроектировать базу данных;
- разработать информационную систему.

ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

1.1 Платформа .NET

Платформа .NET Framework – это специальная технология, предназначенная для выполнения различных веб-служб и создания приложений на ОС Windows.

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows. При разработке платформы .NET Framework учитывались следующие цели.

Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.

Предоставление среды выполнения кода, в которой:

- сведена к минимуму вероятность конфликтов в процессе развертывания программного обеспечения и управления его версиями;
- гарантируется безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем;
- исключаются проблемы с производительностью сред выполнения скриптов или интерпретируемого кода;
- обеспечиваются единые принципы разработки для разных типов приложений, таких как приложения Windows и веб-приложения;
- обеспечивается взаимодействие на основе промышленных стандартов, которое гарантирует интеграцию кода платформы .NET Framework с любым другим кодом.

Прошло уже целых 20 лет, но .Net до сих пор пользуется популярностью несмотря на то, что есть платформы нового поколения, например .Net Core. Сегодня .Net позволяет использовать одни и те же пространства имен, библиотеки и API для разных языков:

- C#;
- Visual Basic;
- Visual C++;
- F#.

Когда программист создает программу на одном из этих языков, то в первую очередь ему необходимо подключить пространство имен System. Это позволяет организовать код программы в логические блоки, объединить и отделить от остального кода некоторую функциональность. Если бы не было .Net, то приходилось бы создавать отдельный System для каждого языка программирования, а это бы уже нарушало один из главных принципов

программирования: «Не повторяйся».

1.2 Язык программирования C#

C# (читается как «Си шарп») — это язык программирования от компании Microsoft. Изначально его создавали для проектов под Windows, но теперь это по-настоящему универсальный язык: на нём пишут игры, десктопные приложения, веб-сервисы, нейросети и даже графику для метавселенных.

Один из ведущих разработчиков языка — легендарный Андерс Хейлсберг, который до C# успел сделать Turbo Pascal и Delphi, а после — TypeScript (майкрософтовский JS на стероидах).

Если коротко, этот язык:

- Кроссплатформенный — запускается почти на любом железе.
- Объектно-ориентированный — состоит из классов и объектов, которые умеют передавать свойства друг другу.
- Постоянно развивается — для тех, кто любит учиться.
- Дружит с экосистемой Windows — для этого и был написан.

Есть две основных версии, зачем Microsoft стала делать свой язык программирования.

Официальная: чтобы упростить разработку приложений под Windows. А то языки в то время были сложные и плохо адаптировались под разные Windows-компьютеры.

Правдоподобная: чтобы заменить Java, на который у Microsoft не было лицензии. В итоге получился почти такой же язык, но с интеграцией под Windows. Как и Java, он основан на языке C и легко запускается на любом устройстве. А название «Си» с решёткой как бы говорит: «У нас тут не копия Java, а новый язык в линейке „Си“».

C# — пятый по популярности язык программирования в мире. Его используют банки, диджитал-агентства, провайдеры связи и крупные IT-компании. Вот что пишут на C# российские и зарубежные корпорации:

- Microsoft — приложения для Windows и Xbox.
- Tesla — корпоративные веб-сервисы и программы.
- Stack Overflow — серверную логику сайтов.
- Сбербанк — 3D-графику и программы виртуальной реальности.
- Ozon — складские и логистические системы.
- «Яндекс» — приложения для автоматизации продаж.

Преимущества C#

Независимость от железа. Программисту не надо адаптировать программу под разные платформы и системы — за него это делает виртуальная машина, вшитая в .NET Framework. В итоге один и тот же код можно запускать на любых устройствах — смартфонах, компьютерах, серверах, банкоматах и даже умных часах.

Отличная совместимость с Windows. Не зря же язык разработали именно в Microsoft. Так же как Swift идеально подходит для программирования под экосистему Apple, C# прекрасно вписывается в экосистему Windows.

Управление памятью. Чтобы программа работала стабильно, её надо иногда чистить от ненужных объектов, ссылок, кэша и прочего мусора. В C# это происходит автоматически — разработчику не надо следить за расходом памяти, бороться с её утечками или удалять мёртвые куски кода.

Строгая типизация. Когда вы объявляете переменную в C#, надо сначала указать, что в ней лежит — строка, число или массив. Так разрабатывать чуть дольше, зато ваш код работает предсказуемо — числа взаимодействуют с числами, строки со строками и так далее. В языках со слабой типизацией свободы и драйва больше, но есть шанс пропустить ошибку, которая всплывёт в готовой программе.

Большое сообщество. На C# пишут более миллиона программистов по всему миру. В соцсетях полно чатов и сообществ «шарпистов», где можно задать вопрос, обсудить сложную тему или найти готовое решение. В теории можно даже найти ментора, который поделится знаниями и поможет быстрее освоить язык.

Синтаксический сахар. В C# есть много способов сократить код, не нарушая логику программы. Программисты называют такие приёмы «синтаксическим сахаром» — они помогают сделать код проще, понятнее и в целом симпатичнее. Сравните, например, как выглядит сложение чисел с «сахаром» и без.

1.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

Windows Presentation Foundation (или сокращенно — WPF) — это платформа для построения пользовательского интерфейса, не зависящая от разрешения и использующая векторный механизм визуализации, способный использовать все преимущества современного графического оборудования. Так, если при разработке графического приложения с использованием WinForms преимущественно используются GDI/GDI+, то WPF использует DirectX.

Преимущества WPF

Широкая интеграция

До появления WPF разработчикам в Windows, которые хотели использовать одновременно 3D-графику, видео, речевые технологии, работу с документами и так далее могло понадобиться изучать несколько независимых технологий, которые могли быть, к тому же, плохо совместимы между собой. В WPF все это входит в состав согласованной модели программирования, поддерживающей композицию и визуализацию разнородных элементов. Одни и те же эффекты применимы к различным видам мультимедийной информации, а один раз освоенная техника может использоваться и для других целей.

Независимость от разрешения экрана

Благодаря использованию векторной графики, WPF обеспечивает и дает возможность уменьшать или увеличивать элементы на экране независимо от его разрешения.

Аппаратное ускорение

Поскольку в основе WPF лежит использование технологии DirectX, то содержимое в WPF-приложении, будь то кнопка, изображение или текст, преобразуется объекты Direct3D, а затем отрисовываются, используя графический процессор видеокарты . Таким образом, WPF-приложения задействуют все возможности аппаратного ускорения графики, что позволяет добиться более качественного изображения и одновременно повысить производительность (поскольку часть работы перекладывается с центральных процессоров на графические).

При этом, WPF не требует обязательного наличия высокопроизводительной видеокарты. В ней имеется и собственный программный конвейер визуализации. Это позволяет использовать возможности, которые пока еще не поддерживаются аппаратно (например, осуществлять высокоточное отображение любого содержимого на экране). Программная реализация используется и как запасной вариант в случае отсутствия аппаратных ресурсов (например, если в системе стоит устаревшая графическая карта, или карта современная, но GPU не хватает ресурсов, скажем, видеопамяти).

Декларативное программирование

Хотя декларативное программирование не является уникальной особенностью WPF, в WPF применение декларативного программирования вышло на новый уровень благодаря языку XAML (extensible Application Markup Language — расширяемый язык разметки приложений). Сочетание WPF и XAML можно сравнить с использованием HTML для описания пользовательского интерфейса, но с гораздо более широкими выразительными возможностями. И эта выразительность выходит далеко за рамки описания интерфейса.

В WPF язык XAML, в основном, применяется в качестве описания интерфейса приложения. Использование XAML, с одной стороны, может стать небольшим препятствием для изучения и работы с WPF, особенно, если до этого вы долгое время использовали WinForms. Однако, с другой стороны, благодаря такому подходу, мы можем полностью отделить интерфейс приложения от логики — дизайнеры «рисуют» красивый привлекательный интерфейс приложения, не задумываясь о бизнес-логике, программисты — пишут бизнес-логику.

Богатые возможности композиции и настройки

В WPF элементы управления могут сочетаться самыми немыслимыми и причудливыми способами. Например, можно создать комбинированный список, содержащий анимированные кнопки, или добавить в кнопку рисунок и так далее. Раньше, для таких «чудес» нам бы пришлось писать довольно много кода, в WPF — это делается достаточно просто и, в некотором роде, даже интуитивно понятно. Кроме того, стоит отметить, что в WPF можно без особого труда изменить тему оформления (скин, обложку) приложения.

Недостатки и ограничения WPF

1. Несмотря на то, что WPF поддерживает трехмерную визуализацию, если планируется разработка приложений с большим количеством трехмерных объектов, например игр, то будет лучшем использование других средств — специальных фреймворков, например, Monogame или Unity.
2. По сравнению с приложениями на WinForms объем программ на WPF и потребление ими памяти в процессе работы в среднем может быть несколько выше. Но, при этом, см. выше — WPF дает ряд преимуществ, реализовать которые в WinForms может стоить большого количество человеко-часов.
3. На данный момент создавать приложения на WPF можно только под ОС Windows.

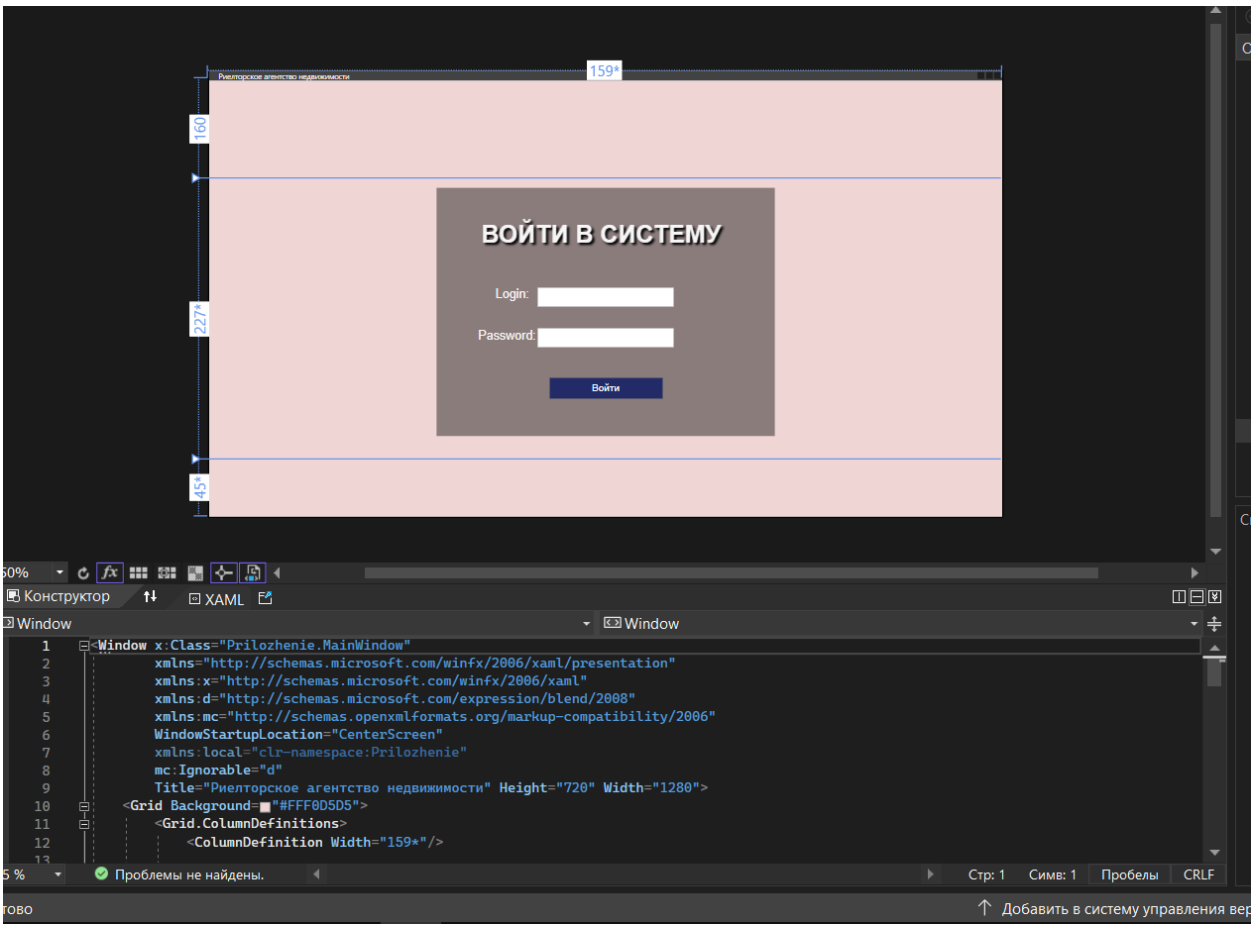

Рис. 1.3 Окно WPF

1.4 СУБД SQL server

СУБД SQL server используются для создания, размещения, хранения и управления реляционными (табличными) базами данных на специальных серверах или в облаке. Они работают через настольные приложения и web-сайты. К основным преимуществам их функционирования относятся:

- высокоскоростной доступ к данным, обеспечиваемый надежной клиент-серверной архитектурой СУБД;
- простота работы и администрирования, обусловленные понятной структурой языка программирования SQL;
- безопасность хранения информации в БД - благодаря возможности шифрования данных и резервного копирования.

Специфика работы сервера базы данных SQL server заключается в транзакционной обработке данных. Это означает, что по каждому запросу от СУБД обрабатывается и сохраняется небольшое количество информации.

Применение SQL server позволяет автоматизировать решение различных бизнес-

задач, поддерживать проведение аналитики данных в режиме онлайн, отслеживать направление ресурсов СУБД, управлять транзакциями (операциями по обработке данных).

ВИДЫ SQL-СЕРВЕРОВ

Для реализации функций СУБД на сегодняшний день чаще всего используются следующие SQL-серверы:

MS SQL server - многопользовательский программный продукт, разработанный компанией Microsoft, обладающий высокой производительностью и отказоустойчивостью, тесно интегрированный с ОС Windows. Этот сервер поддерживает удаленные подключения, работает с многими популярными типами данных, дает возможность создавать триггеры и хранимые данные, имеет практичные и удобные утилиты для настройки;

Oracle Database server - СУБД, предназначенная для создания, консолидации и управления базами данных в облачной среде. Используя этот сервер, можно как автоматизировать обычные бизнес-операции, так и выполнять динамический многомерный анализ данных (OLAP), проводить операции с документами xml-формата и управлять разделенной и локальной информацией;

IBM DB2 - семейство СУБД для работы с реляционными базами данных, признанное самым производительным, имеющим высокие технические показатели и возможности масштабирования. SQL-серверы этой группы характеризуются мультиплатформенностью, способностью к мгновенному созданию резервных копий и восстановлению БД, реорганизации таблиц в онлайн-режиме, разбиению баз данных, определению пользователями новых типов данных;

MySQL - СУБД, разработанная и поддерживаемая компанией Oracle. В основном она используется локальными или удаленными клиентами, позволяя им работать с таблицами разных типов, поддерживающих полнотекстовый поиск или выполняющих транзакции на уровне отдельных записей;

PostgreSQL - СУБД с открытым исходным кодом, работающая с объектно-реляционными (поддерживающими пользовательские объекты) базами данных. Также PostgreSQL предназначена для создания, хранения и извлечения сложных структур данных. Она поддерживает самые различные типы данных (среди них - числовые, текстовые, булевы, денежные, бинарные данные, сетевые адреса, xml и другие).

1.5 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа, настройки, администрирования и разработки всех компонентов SQL Server, Базы данных SQL Azure, управляемого экземпляра SQL Azure, SQL Server на виртуальной машине Azure и Azure Synapse Analytics.

Среда SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе обширную группу графических инструментов с рядом многофункциональных редакторов скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех профессиональных уровней.

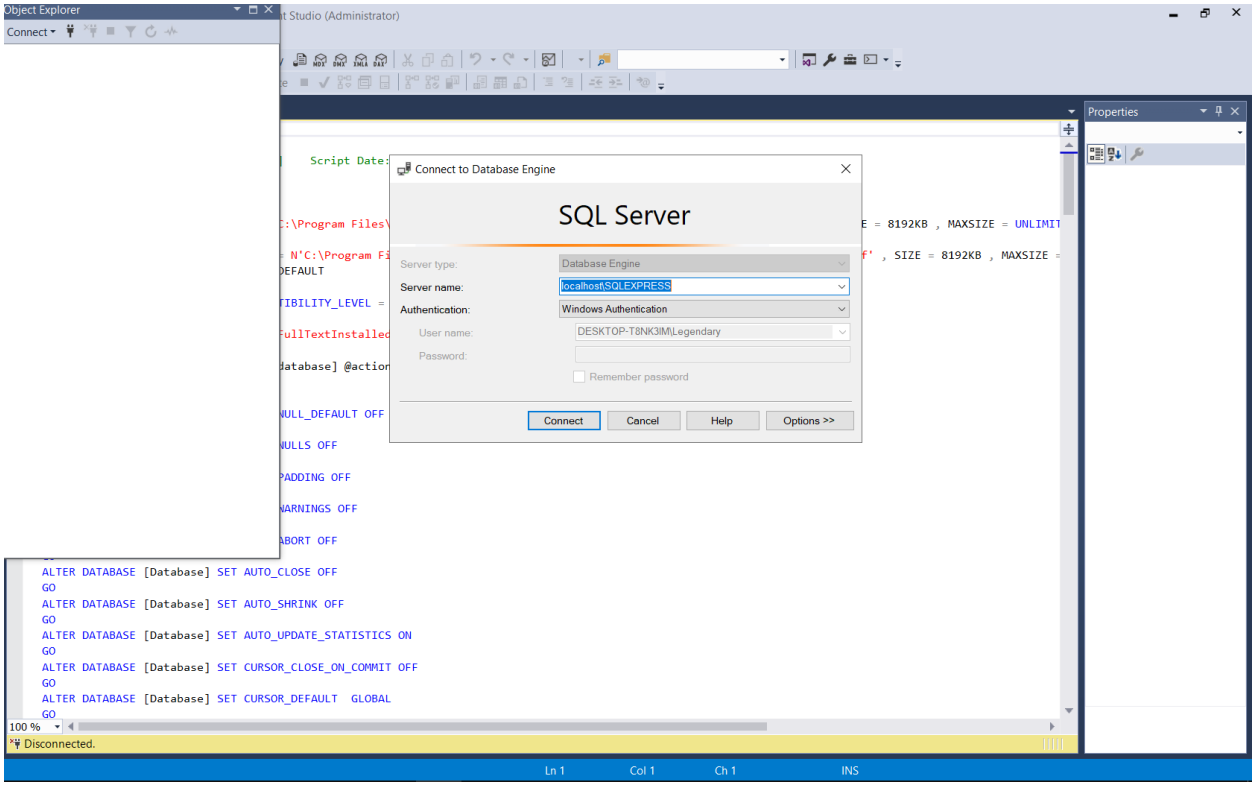

Рис. 1.5 Внешний вид Microsoft SQL Server Management Studio

1.6 Entity Framework

Entity Framework — это решение для работы с базами данных, которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Также код с использованием EF пишется гораздо быстрее.

Например, работая с базами данных напрямую, разработчик должен беспокоиться

о подключении, подготовке SQL и параметров, отправке запросов и транзакций. На Entity Framework всё это делается автоматически — программист же работает непосредственно с сущностями и только говорит EF, что нужно сохранить изменения.

Entity Framework позволяет значительно сократить код работы с базами данных. При этом он предоставляет большие возможности.

Например, можно использовать:

- foreign keys;
- связи one-to-one, one-to-many и many-to-many;
- параметризацию запросов;
- хранимые процедуры.

Однако стоит учитывать, что EF выступает прослойкой между приложением и базой данных, поэтому может ухудшаться производительность. Для небольших проектов это допустимо, но если программа должна работать под большой нагрузкой, то лучше использовать чистый ADO.NET.

ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

2.1 Разработка диаграммы ERD

Данная диаграмма — (ER-модель данных) обеспечивает стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Диаграммы «сущность— связь» в отличие от функциональных диаграмм определяют спецификации структур данных программного обеспечения. Первый вариант модели «сущность — связь» был предложен П. Ченом. В дальнейшем многими авторами были разработаны свои варианты подобных моделей. Все варианты диаграмм «сущность— связь» исходят из одной идеи — графическое изображение нагляднее текстового описания. Все такие диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов) и взаимосвязей между сущностями. Наиболее распространенной является нотация Баркера, ее и будем придерживаться.

Базовыми понятиями ER-модели данных (ER — Entity— Relationship) являются сущность, атрибут и связь.

Сущность — это класс однотипных реальных или абстрактных объектов (людей, событий, состояний, предметов и т.п.), информация о которых имеет существенное значение для рассматриваемой предметной области. Структурой данных называют совокупность правил и ограничений, которые отражают связи, существующие между отдельными частями (элементами) данных. Каждая сущность должна иметь:

- уникальное имя;
- один или несколько атрибутов, которые либо принадлежат сущности, либо наследуются через связь;
- один или несколько атрибутов, которые однозначно идентифицируют каждый экземпляр сущности.

Связь — это отношение одной сущности к другой или к самой себе. Каждая связь может иметь одну из двух модальностей связей.

Пример разработки диаграммы «сущность — связь» ИС “Учет риелторских операций” (рис. 2.1). Основными сущностями для решения указанной задачи являются: Пользователи, Роли Договоры, Помещения, Заявки. Каждая сущность будет впоследствии отвечать при создании таблицы в базе данных.

У каждой сущности существует первичный ключ и атрибуты. Между сущностями установлены связи. В данной диаграмме установлены связи 1 ко многим. Так как может

быть один или несколько пользователей, договоров, помещений, запросов и ролей.

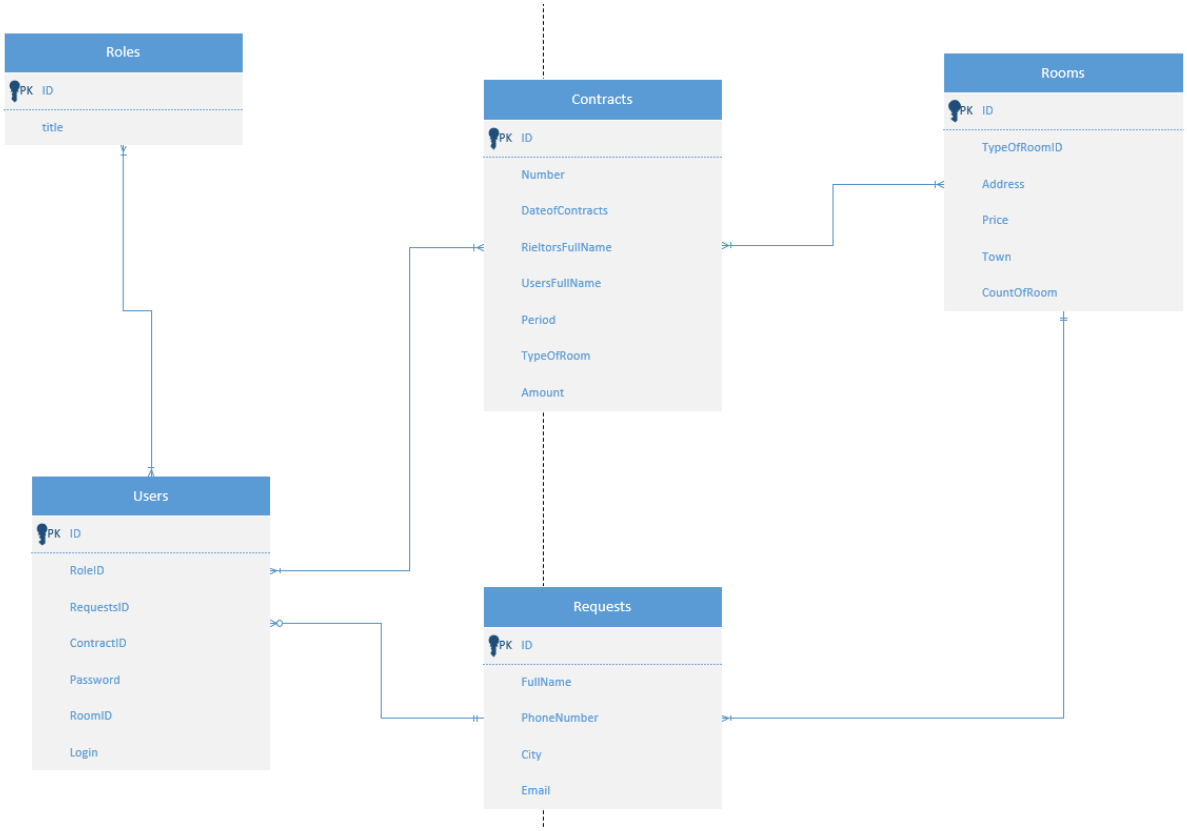

Рис. 2.1 Диаграмма сущность-связь (ERD)

2.2 Разработка базы данных

Для того, чтобы создать базу данных нужно построить диаграмму ERD со всеми сущностями и связями. У каждого атрибута есть свой тип данных. В данном случае используется лишь 2 типа данных – integer (числовой), для обозначения ID идентификаторов и nvchar, для остальных атрибутов.

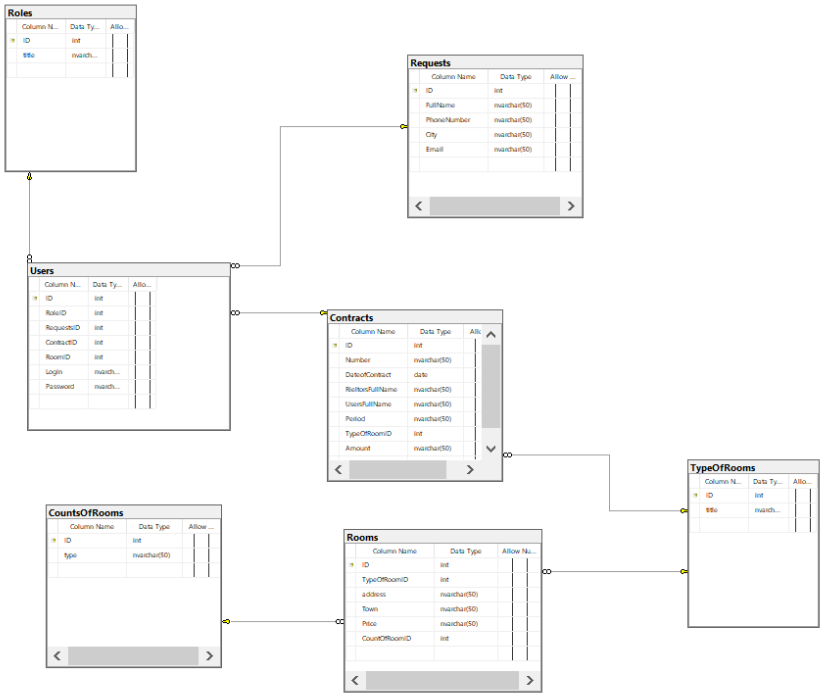

Рис. 2.2 Конечный вид диаграммы ERD

После создания диаграммы, во вкладке Tables (Рис.2.3) появляются добавленные атрибуты. Именно эти атрибуты будут хранить в себе данные, занесенные в таблицы информационной системы.

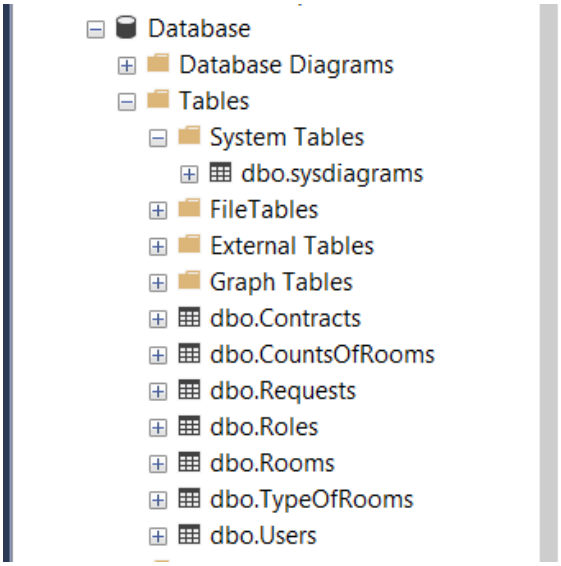

Рис. 2.3 Атрибуты, добавленные в базу данных

ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Разработка интерфейса информационной системы

В данном окне пользователь вводит существующий логин. После происходит проверка логина, после которого открывается возможность ввести пароль. Если данные верны, то появляется окно проверки на капчу. Она обновляется каждые 10 секунд. Также предусмотрена возможность вручную обновить капчу, нажав на специальную кнопку. После правильного ввода кода проверки, пользователь входит в аккаунт.

Существует разграничение для пользователей. Всего существует 2 типа ролей: Пользователь и риелтор, который выступает в роли администратора с большими возможностями.

Алгоритм авторизации:

- сотрудник вводит логин и пароль;
- при вводе пароля сотрудником и нажатии кнопки Войти на служебный телефон отправляется СМС с единоразовым кодом доступа;
- сотрудник вводит код и далее получает доступ к необходимому функционалу.

Основная разметка <Grid> для логина, пароля, капчи и <Button> для перехода на следующие окно.

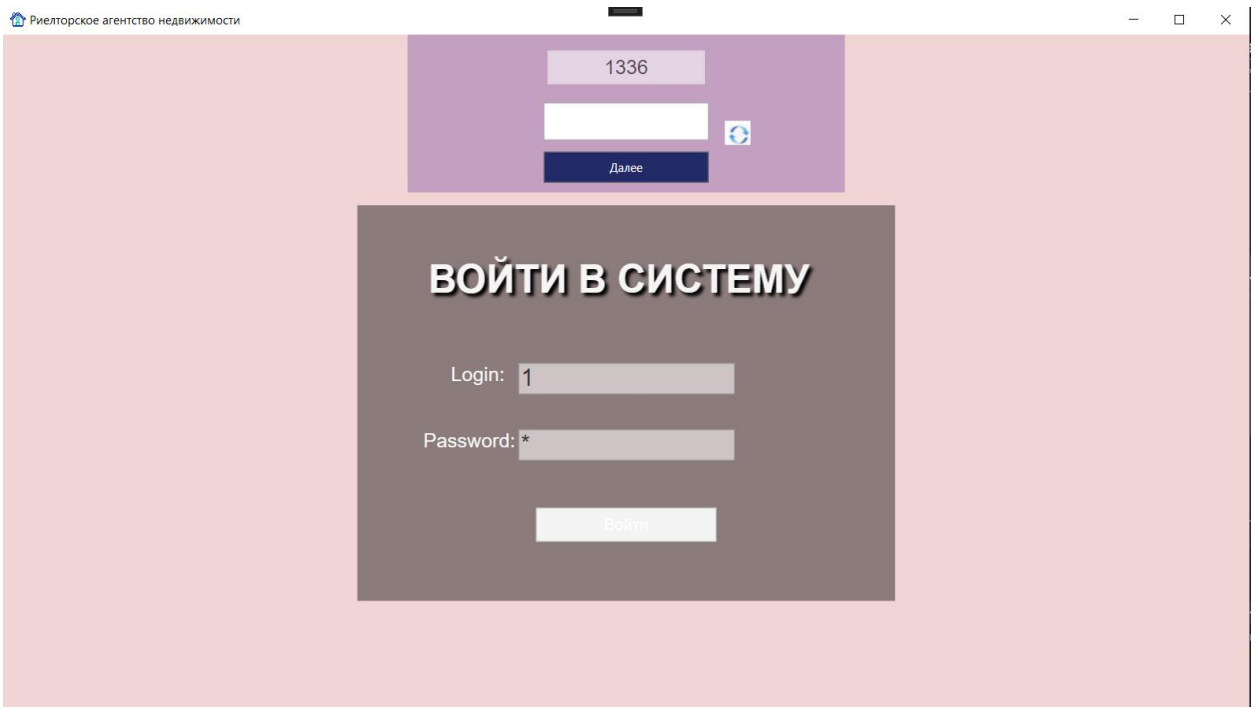

Рис. 3.1 Окно авторизации с проверкой капчи

```
        </Grid>
    <Image Margin="0,13,0,0" Source="/Images/Rectangle 6.png" Stretch="Fill" Grid.Row="1" HorizontalAlignment="Center" VerticalAlignment="Top" Width="546" Height="401"/>
    <TextBlock Text="Login:" HorizontalAlignment="Center" Margin="0,173,0,0" VerticalAlignment="Top" Foreground="#FFF7F6F6" FontFamily="Arial" FontSize="20" Width="356" Height="32" RenderTransformOrigin="0.128,0.514" Grid.Row="1"/>
    <TextBlock Text="Password:" HorizontalAlignment="Center" Margin="0,240,0,0" VerticalAlignment="Top" Foreground="#FFF7F6F6" FontFamily="Arial" FontSize="20" Width="412" Height="32" Grid.Row="1"/>
    <TextBox x:Name="login" HorizontalAlignment="Center" Margin="0,173,0,0" Grid.Row="1" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="220" Height="32" FontFamily="Arial" FontSize="24" />
    <PasswordBox x:Name="password" HorizontalAlignment="Center" Grid.Row="1" PasswordChar="*" VerticalAlignment="Top" Width="220" Height="32" Margin="0,240,0,0" FontFamily="Arial" FontSize="24"/>
    <Button x:Name="knopka" Content="Войти" HorizontalAlignment="Center" Grid.Row="1" VerticalAlignment="Top" Click="BtnSignIn_Click" Width="184" FontSize="16" FontFamily="Arial" Height="35" Background="#FF222B68" Foreground="White" Margin="0,319,0,0"/>
            <Button Visibility="Hidden" x:Name="knopka_2" > //Если проверки на логин и пароль не пройдут, то капча не появится
Content="Войти" HorizontalAlignment="Center" Grid.Row="1" VerticalAlignment="Top" Click="BtnSignIn_Click_2" Width="184" FontSize="16" FontFamily="Arial" Height="35" Background="#FF222B68" Foreground="White" Margin="0,319,0,0" Grid.Column="1"/>
    <Label Content="ВОЙТИ В СИСТЕМУ" HorizontalAlignment="Center" Margin="0,60,0,0" VerticalAlignment="Top" Foreground="#FFF7F6F6" FontFamily="Arial" FontSize="40" Width="412" Height="59" FontWeight="Bold" Grid.Row="1" Grid.Column="1">
        </Label>
          </Grid>
```

После авторизации пользователь попадает на главную страницу приложения. Здесь ему доступны 4 активные кнопки: Заявки, Помещения, Договоры, а также есть возможность сменить аккаунт.

В разметке XAML прописаны кнопки навигации по приложению. Доступен переход на страницы заявок, помещений и договоров.

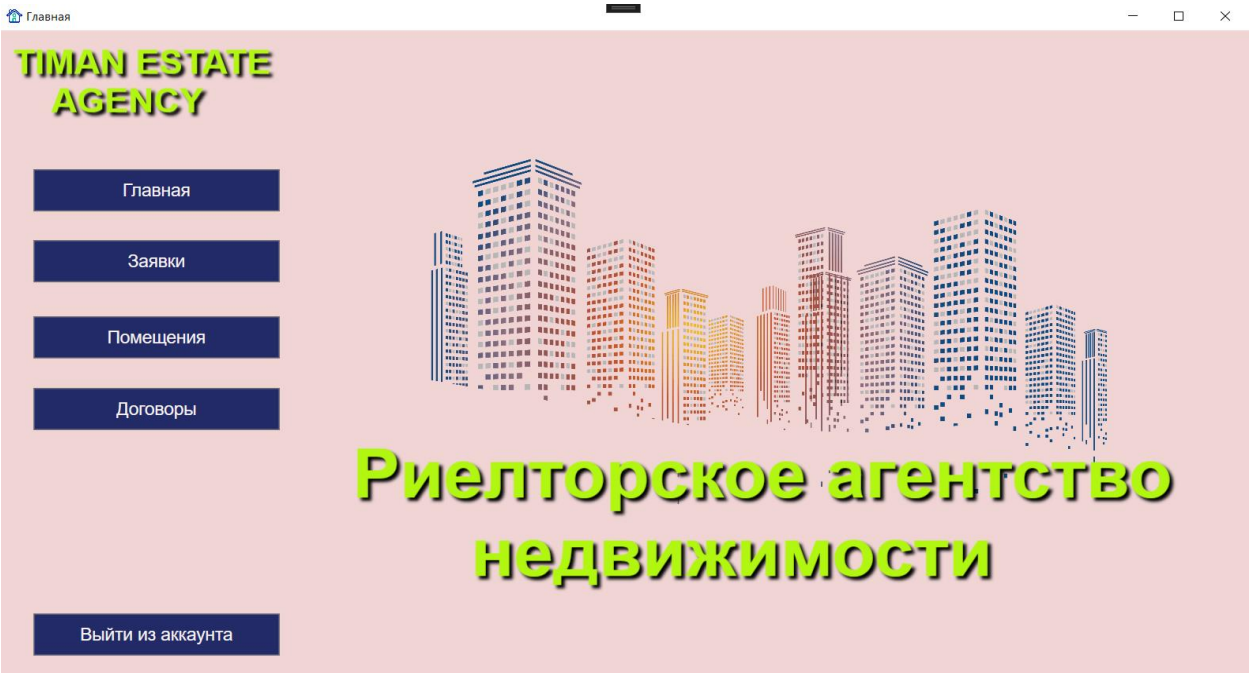

Рис. 3.2 Главная страница приложения

<Button Content="Выйти из аккаунта" HorizontalAlignment="Left" Margin="34,0,0,19" VerticalAlignment="Bottom" Width="250" Height="45" Click="Exit_Click" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="2"/>

<Button Content="Главная" HorizontalAlignment="Left" Margin="34,30,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1"/>

<Button Content="Заявки" HorizontalAlignment="Left" Margin="34,105,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1" Click="Button_Click_1"/>

<Button Content="Помещения" HorizontalAlignment="Left" Margin="34,186,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1" Click="Button_Click_2"/>

<Button x:Name="invisible" Content="Договоры" HorizontalAlignment="Left" Margin="34,262,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1" Click="Button_Click"/>

<Label Content="TIMAN ESTATE
 AGENCY" HorizontalAlignment="Center" VerticalAlignment="Center" Foreground="#FFB2F712" FontFamily="Arial" FontSize="36" FontWeight="Bold" Height="102" Width="276" RenderTransformOrigin="0.564,0.521">

Во вкладке “Заявки” пользователь может оставить заявку риелтору указав свои личные данные, а именно: ФИО, номер телефона, город проживания и Email адрес.

Существует возможность добавления, редактирования и удаления записей.

На данной странице XAML прописаны кнопки добавления, изменения, удаления и формирования отчётов. Для установки связи с базой данных использовано {Binding}, прописанные в /DataGrid>.

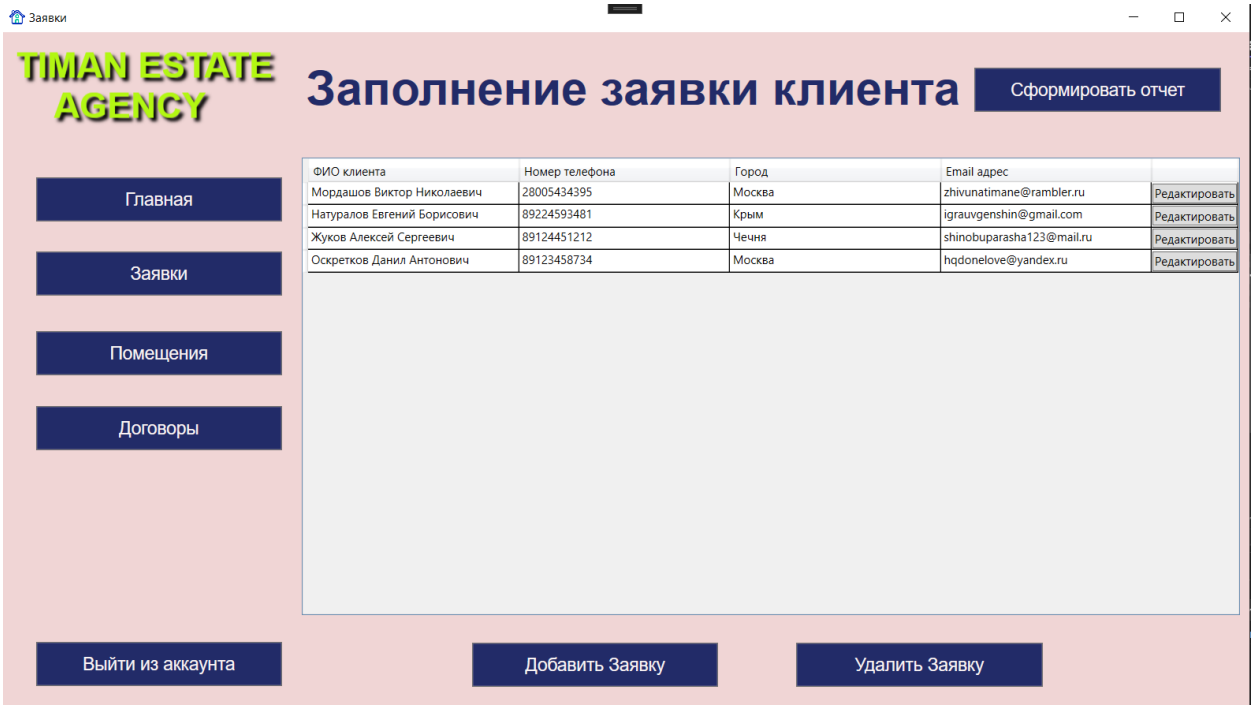

Рис. 3.3 Страница заполнения заявки клиента

```
<DataGrid
AutoGenerateColumns="False"
x:Name="Grid3" IsReadOnly="True" Margin="10,10,10,10" Grid.Row="1" Grid.Column="1">
<DataGrid.Columns>
  <DataGridTextColumn Header="ФИО клиента" Width="*" Binding="{Binding FullName}"/>
  <DataGridTextColumn Header="Номер телефона" Width="*" Binding="{Binding PhoneNumber}"/>
  <DataGridTextColumn Header="Город" Width="*" Binding="{Binding City}"/>
  <DataGridTextColumn Header="Email адрес" Width="*" Binding="{Binding Email}"/>
  <DataGridTemplateColumn Width="auto">
    <DataGridTemplateColumn.CellTemplate>
      <DataTemplate>
        <Button Content="Редактировать" Name="BtnEdit" Click="Edit_Click"></Button>
      </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
  </DataGridTemplateColumn>
</DataGrid.Columns>
</DataGrid>
```

Во вкладке “Помещения” пользователь может посмотреть доступные помещения для покупки. Данная страница позволяет увидеть тип помещения, адрес, город, его цену, а

также количество комнат.

На данной вкладке у пользователя ограниченные возможности – существует разграничение. Ему не доступно добавление, редактирование и удаление записей, так как у пользователя недостаточно прав.

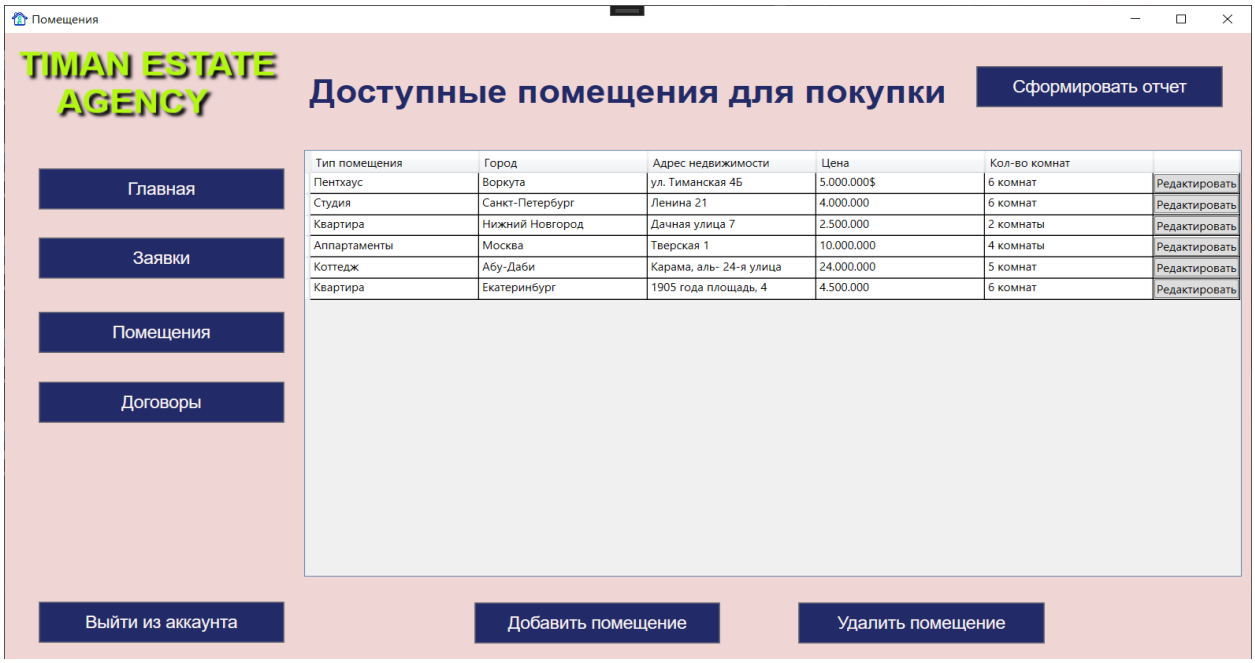

Рис. 3.4 Страница с доступными помещениями для покупки

<Button Content="Выйти из аккаунта" HorizontalAlignment="Left" Margin="34,0,0,19" VerticalAlignment="Bottom" Width="250" Height="45" Click="Exit_Click" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="2"/>

<Button Content="Главная" HorizontalAlignment="Left" Margin="34,30,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1" Click="Button_Click_1"/>

<Button Content="Заявки" HorizontalAlignment="Left" Margin="34,105,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1" Click="Button_Click_2"/>

<Button Content="Помещения" HorizontalAlignment="Left" Margin="34,186,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1"/>

<Button x:Name="invisible2" Content="Договоры" HorizontalAlignment="Left" Margin="34,262,0,0" VerticalAlignment="Top" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="1" Click="Button_Click"/>

<Button x:Name="invisible" Content="Добавить помещение" HorizontalAlignment="Left" Margin="183,0,0,0" VerticalAlignment="Center" Width="250" Height="45" Click="Add_Click"

```
Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="2" Grid.Column="1"/>
    <Button x:Name="invisible1" Content="Удалить помещение" HorizontalAlignment="Left" VerticalAlignment="Center" Width="250" Height="45" Click="Delete_Click" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Row="2" Grid.Column="1" Margin="512,0,0,0"/>
    <Button Content="Сформировать отчет" HorizontalAlignment="Left" Margin="693,0,0,0" VerticalAlignment="Center" Width="250" Height="45" Background="#FF222B68" Foreground="#FFFAF4F4" FontSize="18" FontFamily="Arial" RenderTransformOrigin="1.6,-1.76" Grid.Column="1" Click="Export_Click"/>
    <DataGrid
      AutoGenerateColumns="False"
      x:Name="Grid4" IsReadOnly="True" Margin="10,10,10,10" Grid.Row="1" Grid.Column="1">
      <DataGrid.Columns>
        <DataGridTextColumn Header="Тип помещения" Width="*" Binding="{Binding TypeOfRooms.title}"/>
        <DataGridTextColumn Header="Город" Width="*" Binding="{Binding Town}"/>
        <DataGridTextColumn Header="Адрес недвижимости" Width="*" Binding="{Binding address}"/>
        <DataGridTextColumn Header="Цена" Width="*" Binding="{Binding Price}"/>
        <DataGridTextColumn Header="Кол-во комнат" Width="*" Binding="{Binding CountsOfRooms.type}"/>
        <DataGridTemplateColumn Width="auto" x:Name="invisible3"> // Скрытая кнопка редактирования для пользователей, у которых нет прав.
          <DataGridTemplateColumn.CellTemplate>
            <DataTemplate>
              <Button Content="Редактировать" x:Name="BtnEdit" Click="Edit_Click"></Button>
            </DataTemplate>
          </DataGridTemplateColumn.CellTemplate>
        </DataGridTemplateColumn>
      </DataGrid.Columns>
        </DataGrid>
```

Страница договоры доступна только риелтору (администратору). Он может добавлять, редактировать или удалять договоры для покупки недвижимости. Данная страница содержит в себе: Номер договора, ФИО риелтора, заключившего договор с клиентом, ФИО самого клиента, период действия договора, дата его заключения, тип помещения, а также его стоимость. Существует возможность сформировать отчет и экспортировать его в файл Word.

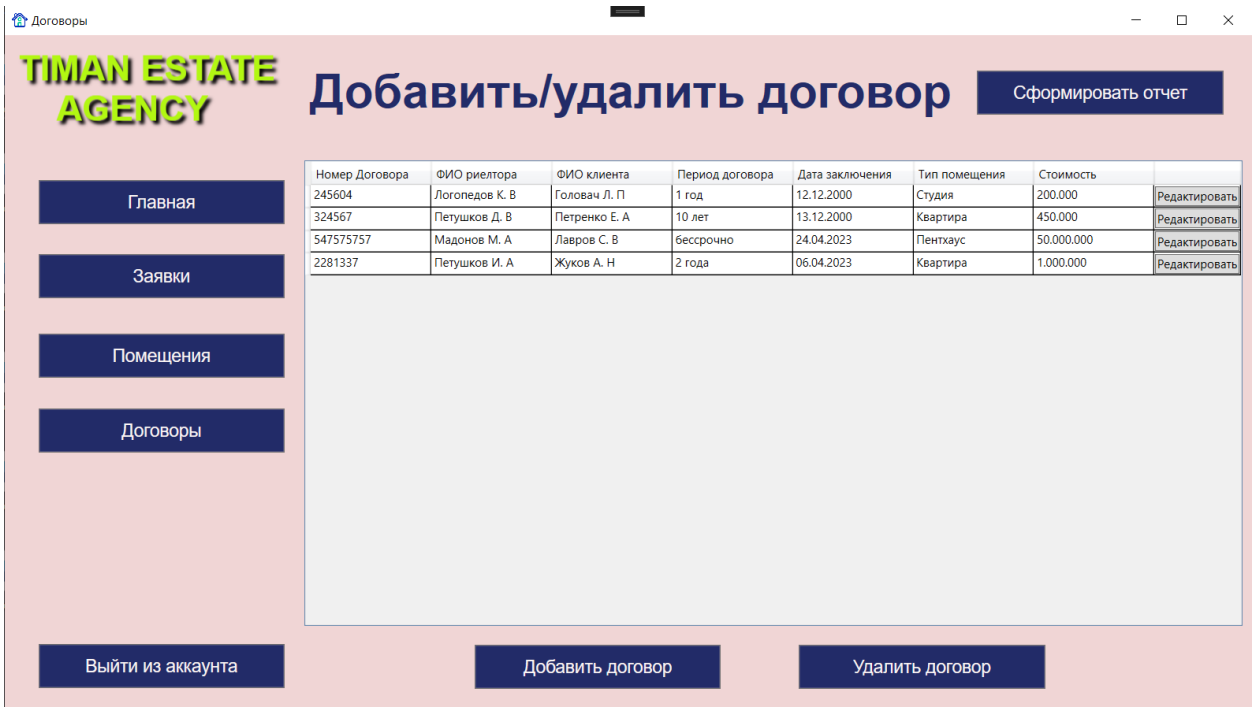

Рис. 3.5 Страница со всеми доступными договорами

```
<DataGrid
  AutoGenerateColumns="False"
  x:Name="Grid2" IsReadOnly="True" Margin="10,10,10,1" Grid.Row="1" Grid.Column="1">
<DataGrid.Columns>
  <DataGridTextColumn Header="Номер Договора" Width="*" Binding="{Binding Number}"/>
  <DataGridTextColumn Header="ФИО риелтора" Width="*" Binding="{Binding RieltorsFullName}"/>
  <DataGridTextColumn Header="ФИО клиента" Width="*" Binding="{Binding UsersFullName}"/>
  <DataGridTextColumn Header="Период договора" Width="*" Binding="{Binding Period}"/>
  <DataGridTextColumn Header="Дата заключения" Width="*" Binding="{Binding DateofContract, StringFormat={}{0:dd\.}{0:MM\.}{0:yyyy}}"/>
  <DataGridTextColumn Header="Тип помещения" Width="*" Binding="{Binding TypeOfRooms.title}"/>
  <DataGridTextColumn Header="Стоимость" Width="*" Binding="{Binding Amount}"/>
  <DataGridTemplateColumn Width="auto">
    <DataGridTemplateColumn.CellTemplate>
      <DataTemplate>
        <Button Content="Редактировать" Name="BtnEdit" Click="Edit_Click"></Button>
      </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
  </DataGridTemplateColumn>
</DataGrid.Columns>
</DataGrid>
```

Администратор может отредактировать уже имеющиеся данные, находящиеся в таблице. Окно добавления и редактирования позволяет добавлять или изменять информацию в таблице.

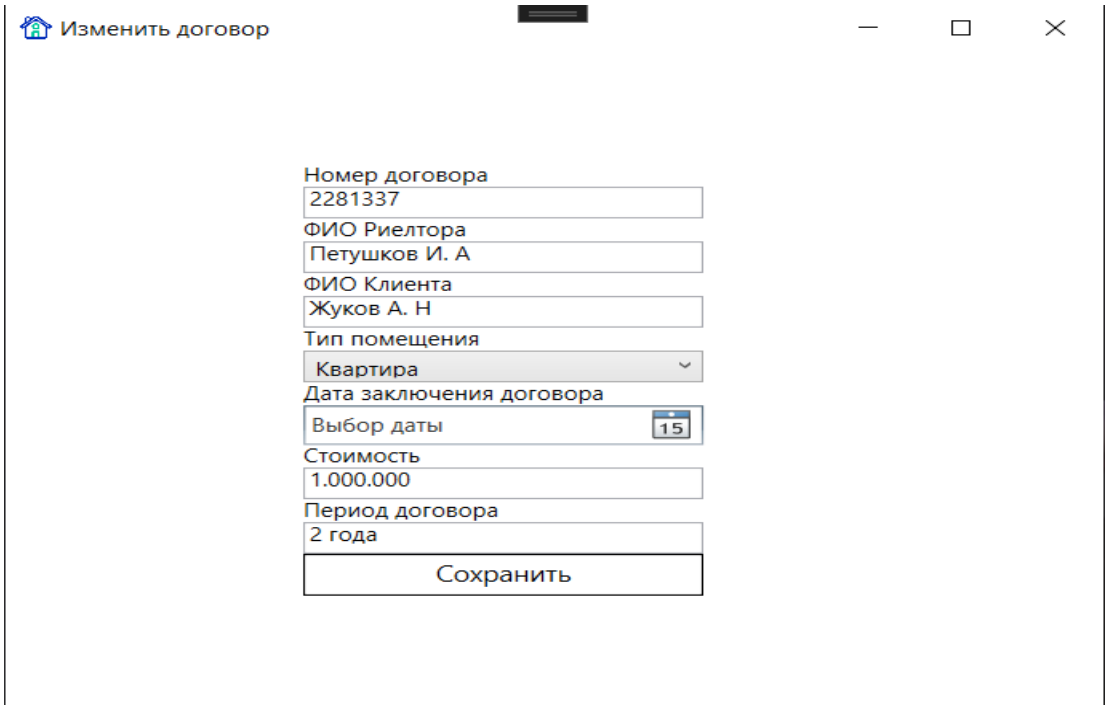

Рис. 3.6 Окно добавления и редактирования

```
<Grid Background="White" Margin="0,0,51,0">
<StackPanel VerticalAlignment="Center"
        HorizontalAlignment="Center"
        MinWidth="200">
<TextBlock FontFamily="TeX Gyre Adventor"><Run Language="ru-ru" Text="Номер договора"/></TextBlock>
<TextBox x:Name="number" FontFamily="TeX Gyre Adventor" Height="20"/>
<TextBlock FontFamily="TeX Gyre Adventor"><Run Language="ru-ru" Text="ФИО Риелтора"/></TextBlock>
<TextBox x:Name="realtor" FontFamily="TeX Gyre Adventor" Height="20"/>
<TextBlock FontFamily="TeX Gyre Adventor"><Run Language="ru-ru" Text="ФИО Клиента"/></TextBlock>
<TextBox x:Name="user" FontFamily="TeX Gyre Adventor" Height="20"/>
<TextBlock FontFamily="TeX Gyre Adventor"><Run Language="ru-ru" Text="Тип помещения"/></TextBlock>
<ComboBox x:Name="title" DisplayMemberPath="title" Height="20"/>
<TextBlock FontFamily="TeX Gyre Adventor"><Run Language="ru-ru" Text="Дата заключения договора"/></TextBlock>
```

```
        <DatePicker x:Name="Date" Text="{Binding Birthdate}" Height="25" />
        <TextBlock FontFamily="TeX Gyre Adventor"><Run Language="ru-ru" Text="Стоимость"/></TextBlock>
        <TextBox x:Name="amount" FontFamily="TeX Gyre Adventor" Height="20"/>
        <TextBlock FontFamily="TeX Gyre Adventor"><Run Language="ru-ru" Text="Период договора"/></TextBlock>
        <TextBox x:Name="period" FontFamily="TeX Gyre Adventor" Height="20"/>
        <Button Content="Сохранить"
            Click="Save_Btn_Click" FontFamily="TeX Gyre Adventor" FontSize="14" Background="White" Height="26" BorderBrush="Black" Foreground="Black"/>
    </StackPanel>
  </Grid>
```

3.2 Программирование информационной системы

Данная часть кода реализует окно реализации. Происходит разграничение прав пользователя и объявление глобальных переменных

```
public static class Globals
{
    public static int UserRole;
    public static Users userinfo { get; set; }
}
```

В данной части происходит проверка логина и пароля, путем подключения к Users. В случае, если пользователя нет в базе, вылезет окно с уведомлением.

```
private void BtnSignIn_Click(object sender, RoutedEventArgs e)  // кнопка входа
{
    var CurrentUser = AppData.db.Users.FirstOrDefault(u => u.Login == login.Text);  // подключение к Users
    if (CurrentUser != null)
    {
        Globals.UserRole = CurrentUser.RoleID;  // глобальный класс для пользователя
        Globals.userinfo = CurrentUser;
        password.IsEnabled = true;  // разблокировка пароля при проверке логина
        knopka.Visibility = Visibility.Hidden;
        knopka_2.Visibility = Visibility.Visible;
    }
    else
    {
        MessageBox.Show("Такого пользователя нет в базе!");  // уведомление
    }
}
```

В данной части кода происходит финальная часть проверки пользователя. После ввода правильного логина и пароля открывается окно введения капчи, которая

обновляется в течении 10 секунд.

```
private async void BtnSignIn_Click_2(object sender, RoutedEventArgs e) // кнопка входа после проверки логина и пароля
  {
    var CurrentUser1 = AppData.db.Users.FirstOrDefault(u => u.Login == login.Text && u.Password == password.Password) ;
    if (CurrentUser1 != null)
    {
      Globals.UserRole = CurrentUser1.RoleID;  // ввод капчи
      Globals.userinfo = CurrentUser1;
      if (kapcha.Visibility == Visibility.Hidden)
        kapcha.Visibility = Visibility.Visible;
      login.IsEnabled = false;
      password.IsEnabled = false;
      knopka_2.IsEnabled = false;
      while (true)
      {
        Random x = new Random();  // генератор случайных чисел в капче
        int a = x.Next(1000, 9999);
        TXB1.Text = a.ToString();
        await Task.Delay(10000);
      }
    }
    else
    {
      MessageBox.Show("Неверный пароль!");
    }
  }
```

Данный код реализует переход между страницами.

```
private void Button_Click(object sender, RoutedEventArgs e) // кнопка перехода
  {
    Window1 zxc = new Window1();
    zxc.Show();
    this.Close();
  }
```

Данная часть отвечает за разграничение пользователей. Для пользователей, у которых нет прав, кнопки могут быть скрыты.

```
// Разграничение для пользователя и админа
        if (MainWindow.Globals.UserRole == 1)
            {
               invisible.Visibility = Visibility.Visible;
            }
            else
            {
               invisible.Visibility = Visibility.Collapsed;
             }
```

Кнопка удаления, с уведомлением

```
private void Delete_Click(object sender, RoutedEventArgs e)  // кнопка для удаления
  {
     if (MessageBox.Show("Вы точно хотите удалить следующие элементы?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)
     {
        var CurrentSotrudnikii = Grid2.SelectedItem as Contracts;
        AppData.db.Contracts.Remove(CurrentSotrudnikii);
        AppData.db.SaveChanges();

        Grid2.ItemsSource = AppData.db.Contracts.ToList();
        MessageBox.Show("Удалено");
     }
      }

// функция для обновления данных на странице
private void Window_Loaded(object sender, RoutedEventArgs e)
  {
     Grid2.ItemsSource = AppData.db.Contracts.ToList();
      }
```

Этот код нужен для добавления информации в таблицу.

```
private void Save_Btn_Click(object sender, RoutedEventArgs e)  // кнопка сохранения

  {
     Contracts clients = new Contracts();

     var zxc = title.SelectedItem as TypeOfRooms;
```

```
        if (_currentHotel.ID >= 0)
            DatabaseEntities.GetContext().Contracts.AddOrUpdate(_currentHotel);

        try
// окно с уведомлением о сохранении
        {
            DatabaseEntities.GetContext().SaveChanges();
            MessageBox.Show("Информация сохранена!");
            Window1 qwe = new Window1();
            qwe.Show();
            this.Close();

        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
             }

    // обращение к другой сущности
    title.ItemsSource = DatabaseEntities.GetContext().TypeOfRooms.ToList();
```

Данная часть кода реализует добавление информации в таблицу.

```
//кнопка сохранения
private void Save_Btn_Click(object sender, RoutedEventArgs e)
      {
         Contracts clients = new Contracts();
//заполнение данных
         clients.Number = number.Text;
         clients.RieltorsFullName = realtor.Text;
         clients.UsersFullName = user.Text;
         clients.Period = period.Text;
         clients.Amount = amount.Text;

// функция для даты
         var CurrentDate = Date.SelectedDate.Value;
         clients.DateofContract = CurrentDate;

         var CurrentOffice = title.SelectedItem as TypeOfRooms;
         clients.TypeOfRoomID = CurrentOffice.ID;
```

```
// уведомления об успешном добавлении
        AppData.db.Contracts.Add(clients);
        AppData.db.SaveChanges();
        MessageBox.Show("Добавлено");

// переход на страницу
        Window1 glavnoe = new Window1();
        glavnoe.Show();
        this.Close();

         }
```

В последней части кода происходит формирование отчётов с разметкой и оформлением в Word документе (кнопка export_click).

```
private void Export_Click(object sender, RoutedEventArgs e)  // кнопка формирования отчета в ворде
      {
        var allRequest = DatabaseEntities.GetContext().Requests.ToList();

        var application = new Word.Application();

        Word.Document document = application.Documents.Add();

        Word.Paragraph userParagraph = document.Paragraphs.Add();
        Word.Range userRange = userParagraph.Range;
        userRange.Text = "Отчет по заявкам";

        userRange.InsertParagraphAfter();

        Word.Paragraph tableParagraph = document.Paragraphs.Add();
        Word.Range tableRange = tableParagraph.Range;
        Word.Table paymentsTable = document.Tables.Add(tableRange, allRequest.Count() + 1, 4);   // размер
максимального кол-ва столбцов
        paymentsTable.Borders.InsideLineStyle = paymentsTable.Borders.OutsideLineStyle
           = Word.WdLineStyle.wdLineStyleSingle;
        paymentsTable.Range.Cells.VerticalAlignment =
Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

        Word.Range cellRange;

        cellRange = paymentsTable.Cell(1, 1).Range;  // названия столбцов в экспортируемой таблице
        cellRange.Text = "ФИО клиента";
```

```
            cellRange = paymentsTable.Cell(1, 2).Range;
            cellRange.Text = "Номер телефона";
            cellRange = paymentsTable.Cell(1, 3).Range;
            cellRange.Text = "Город";
            cellRange = paymentsTable.Cell(1, 4).Range;
            cellRange.Text = "Email адрес";

            paymentsTable.Rows[1].Range.Bold = 1;
            paymentsTable.Rows[1].Range.ParagraphFormat.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphCenter;

            for (int i = 0; i < allRequest.Count(); i++)
            {
               var currentCategory = allRequest[i];  // добавление данных в столбцы
               {
                  cellRange = paymentsTable.Cell(i + 2, 1).Range;
                  cellRange.Text = Convert.ToString(currentCategory.FullName);
                  cellRange.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

                  cellRange = paymentsTable.Cell(i + 2, 2).Range;
                  cellRange.Text = Convert.ToString(currentCategory.PhoneNumber);

                  cellRange = paymentsTable.Cell(i + 2, 3).Range;
                  cellRange.Text = Convert.ToString(currentCategory.City);

                  cellRange = paymentsTable.Cell(i + 2, 4).Range;
                  cellRange.Text = Convert.ToString(currentCategory.Email);

               }
            }

            application.Visible = true;

          }
```

ЗАКЛЮЧЕНИЕ

Данная автоматизированная информационная система была разработана для работников риелтоского агентства и агентства недвижимости. В процессе выполнения информационной системы "Учет риелторских операций", была спроектирована диаграмма Entity Relationship.

Разработка диаграммы системы выполнена с помощью Visio 2016. База данных для информационной системы была построена в утилите SQL Server Management Studio. Информационная система построена в интегрированной среде разработки Visual Studio.

При написании программы основное внимание было уделено удобству работы пользователя с программой и построению дружественного интерфейса.

Исходя из общего положения и опираясь на совокупность всех ранее вышеперечисленных и упомянутых фактов можно сделать вывод, что поставленные цель и задачи при проектировании информационной системы были выполнены в полной мере.

Ссылка на репозиторий:
https://github.com/TheCoolerBoychik/KURSOVOYPROJECT.git

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Литература

1. Рудаков А.В., Федорова Г.Н. Технология разработки программных продуктов. – Москва: 2014. – 192 с.
2. Мэтью Мак-Дональд WPF Windows Presentation Foundation в .NET 4.5 с примерами на C#: для профессионалов/ Мэтью Мак-Дональд: Санкт-Петербург, 2016.

Интернет-ресурсы

1. Stack overflow exception. Интерфейсы // C# - Киберфорум – Режим доступа: https://www.cyberforum.ru/csharp
2. Newest 'c#' Questions // Stack Overflow. – Режим доступа: https://stackoverflow.com/questions
3. Документация по C#. Начало работы, руководства, справочные материалы. // Microsoft Learn. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/
4. Особые исключения в .NET и как их готовить // Хабр – Режим доступа: https://habr.com/ru/companies/jugru/articles/426045
5. Что такое Windows Presentation Foundation - WPF .NET // Microsoft Learn – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0