МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Агентство недвижимости»

Исполнитель

студент (ка) 2 курса группы 6 Максимчикова Юлия Сергеевна

(Ф.И.О.)

Руководитель работы ассистент Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В Пацей\_

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное программирование и стандарты проектирования"

|  |  |
| --- | --- |
| Специальность: 1-40 01 01 Программное обеспечение информационных технологий  Студент:\_\_\_\_Максимчикова Ю.С. | Группа:\_\_6\_\_ |
| **Тема: Программное средство «Агентство недвижимости»** | |

**1. Срок сдачи студентом законченной работы**: "май 2019 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС должно:

* Управлять базой данных (сделок, сотрудников, недвижимости, клиентов, типов сделок и типов недвижимости)
* Выполнять ввод, редактирование и удаление информации о недвижимости, клиентах агентства и сделках, типах недвижимости и сделок
* Генерировать договор на оказание агентских услуг
* Выполнять поиск продающейся недвижимости по заданным критериям (адрес, площадь, цена)

**2.2.** Программное средство должно быть выполнено с использованием ООП, взаимодействовать с базой данных, реализовано под ОС Windows. Отображение, бизнес логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Язык разработки С#. Управление программой должно быть интуитивно понятным и удобным. Использовать архитектурные и шаблоны проектирования.

**3. Примерное содержание расчетно-пояснительной записки**

(перечень вопросов подлежащих разработке)

* Введение
* Постановка задачи
* Обзор литературы и аналогов (алгоритмы решения, обзор литературы, прототипы)
* Проектирование программного средства (концепция, модель, структура, выполняемые функции и взаимосвязь всех компонентов, диаграммы UML).
* Разработка модели данных программного средства.
* Руководство пользователя
* Тестирование программного средства
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате MS Word. Оформление записки должно быть согласно правилам.
  + Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio, VS или других редакторах.
  + Листинги программы представляются частично в приложении.
  + К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и инсталляцию проекта.

**Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2019 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2019 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2019 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2019 |  |
| 5 | Кодирование программного средства | 23.04.2019 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2019 |  |
| 7 | Оформление пояснительной записки | 7.05.2019 |  |
| 2 | Сдача проекта | 20.05.2019 |  |

**5. Дата выдачи задания \_\_\_\_\_\_\_\_\_\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н.А.Северинчик*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

Содержание

[ВВЕДЕНИЕ 5](#_Toc9574717)

[1 Аналитический обзор прототипов и литературных источников 6](#_Toc9574718)

[2 Анализ требований к программному средству и разработка функциональных требований 12](#_Toc9574719)

[3 Проектирование программного средства 14](#_Toc9574720)

[3.1 Проектирование базы данных 14](#_Toc9574721)

[3.3 Проектирование архитектуры приложения 15](#_Toc9574722)

[4 Реализация программного средства 18](#_Toc9574723)

[4.1 Реализация сущностей 18](#_Toc9574724)

[4.2 Реализация уровня доступа к данным 18](#_Toc9574725)

[4.3 Реализация архитектуры MVVM 19](#_Toc9574726)

[4.4 Реализация представления 20](#_Toc9574727)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 22](#_Toc9574728)

[6 Руководство по установке и использованию 25](#_Toc9574729)

[ЗАКЛЮЧЕНИЕ 30](#_Toc9574730)

[СПИСОК ЛИТЕРАТУРЫ 31](#_Toc9574731)

[ПРИЛОЖЕНИЕ А 32](#_Toc9574732)

[ПРИЛОЖЕНИЕ Б 33](#_Toc9574733)

[ПРИЛОЖЕНИЕ В 34](#_Toc9574734)

[ПРИЛОЖЕНИЕ Г 35](#_Toc9574735)

[ПРИЛОЖЕНИЕ Д 36](#_Toc9574736)

[ПРИЛОЖЕНИЕ Е 37](#_Toc9574737)

[ПРИЛОЖЕНИЕ Ж 38](#_Toc9574738)

[ПРИЛОЖЕНИЕ И 39](#_Toc9574739)

[ПРИЛОЖЕНИЕ К 40](#_Toc9574740)

[ПРИЛОЖЕНИЕ Л 41](#_Toc9574741)

ВВЕДЕНИЕ

Данный курсовой проект посвящён разработке программного средства «Агентство недвижимости». В настоящее время в любом таком агентстве возникает проблема организации управления данными. Достичь наиболее эффективной работы можно с помощью автоматизации. Она позволит хранить, структурировать и систематизировать большие объемы данных.

Средство «Агентство недвижимости» предназначено для комплексной автоматизации работы риэлторов и агентств недвижимости. Данная программа создана для решения всего спектра задач, связанного с оформлением сделок, ведением клиентской базы, учетом недвижимости, документооборотом агентства и многим другим. Исходя из этого, повышается производительность труда, скорость оформления сделок и общая прибыль фирмы. Программа легко и быстро настраивается под конкретные требования заказчика. Она интуитивно проста для пользователей, так как не требует квалифицируемых ИТ-ресурсов.

Немаловажным моментом любого программного средства является язык программирования, на котором оно разрабатывается. Языком разработки данного курсового проекта является C#. На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. C# является объектно-ориентированным языком. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. Для разработки графической составляющей данного продукта используется технология Windows Presentation Foundation (WPF), которая позволяет обеспечить приложению насыщенный дизайн и интерактивность.

1 Аналитический обзор прототипов и литературных источников

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

На сегодняшний день можно встретить большое количество программных решений, разработанных для автоматизации работы риэлторских компаний. Их можно разделить на web-приложения и desktop-приложения.

Среди web-решений популярными являются:

* «Bpm’online real estate»;
* «Intrum CRM»;
* «Real Estate CRM».

«Bpm’online real estate» ⎯ это CRM-система для управления всем комплексом задач агентств недвижимости. CRM‐система (Customer Relationship Management или Управление отношениями с клиентами) — это прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами), в частности, для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путем сохранения информации о клиентах и истории взаимоотношений с ними, установления и улучшения бизнес-процессов и последующего анализа результатов.

Интерфейс «bpm’online real estate» представлен на рисунке 1.1.

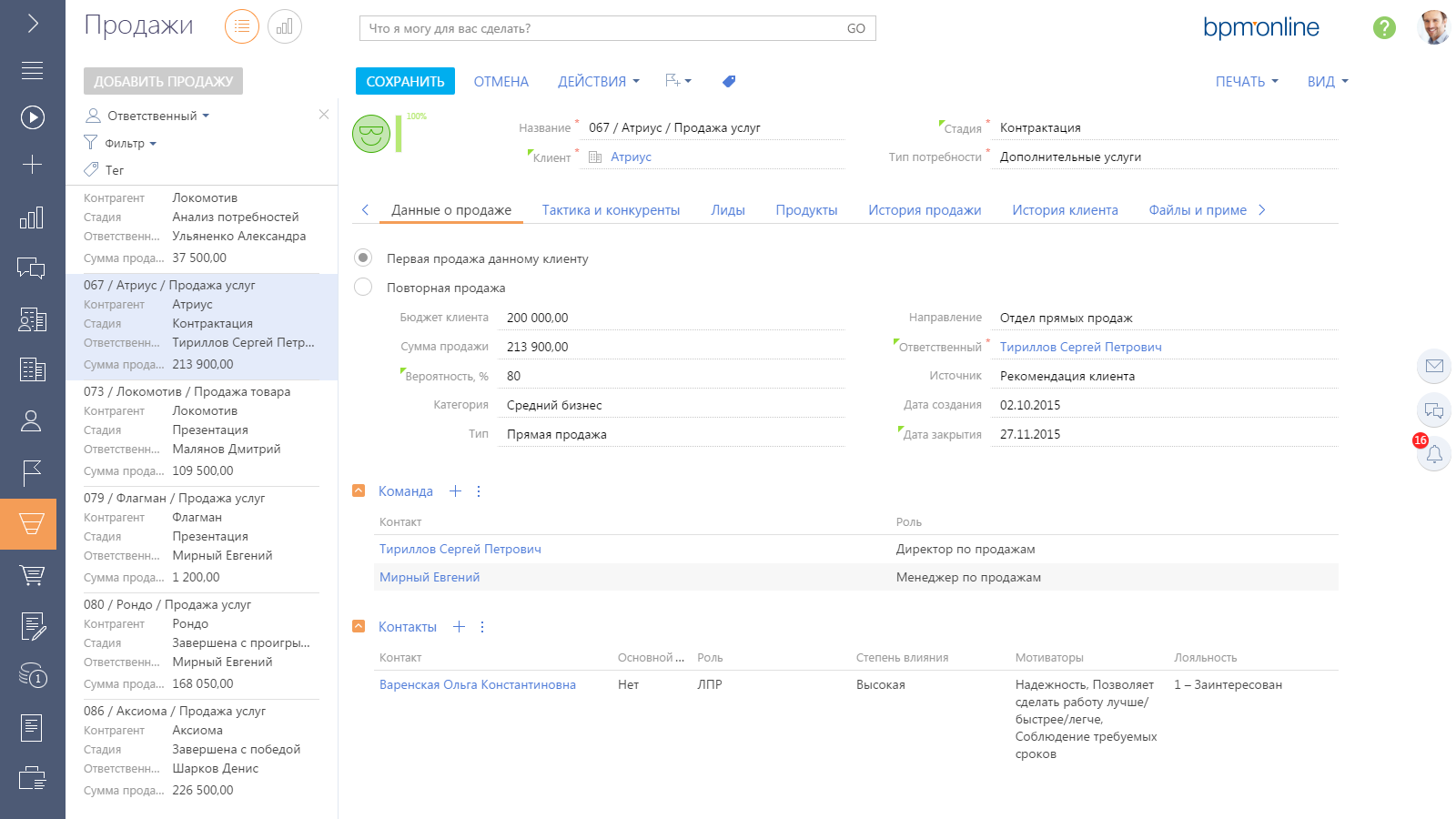


Рисунок 1.1 − Интерфейс «bpm’online real estate»

Проанализировав систему «bpm’online real estate», можно выделить её основные плюсы и минусы.

Из плюсов особенно выделяются:

* широкий функционал;
* простой и удобный интерфейс с гибкой настройкой;
* гибкие настройки разных этапов работы и отчетов;
* наличие BPM системы.

Основные минусы:

* сложность настройки и доработки системы;
* сложная и запутанная документация;
* сложная API с большим числом ограничений;
* сложная интеграция;
* качество работы поддержки на неудовлетворительном уровне.

Наибольшей популярностью пользуются следующие desktop-программы:

* «1С:Риэлтор»;
* «Феникс»;
* «Идеальный Вариант: Недвижимость».

Продукт «1С:Риэлтор» предназначен для компаний, деятельность которых связана с оказанием услуг в сфере купли-продажи объектов недвижимости либо с продажей собственных объектов недвижимости. Решение учитывает специфику как первичного, так и вторичного рынков недвижимости.

Интерфейс «1С:Риэлтор» представлен на рисунке 1.4.

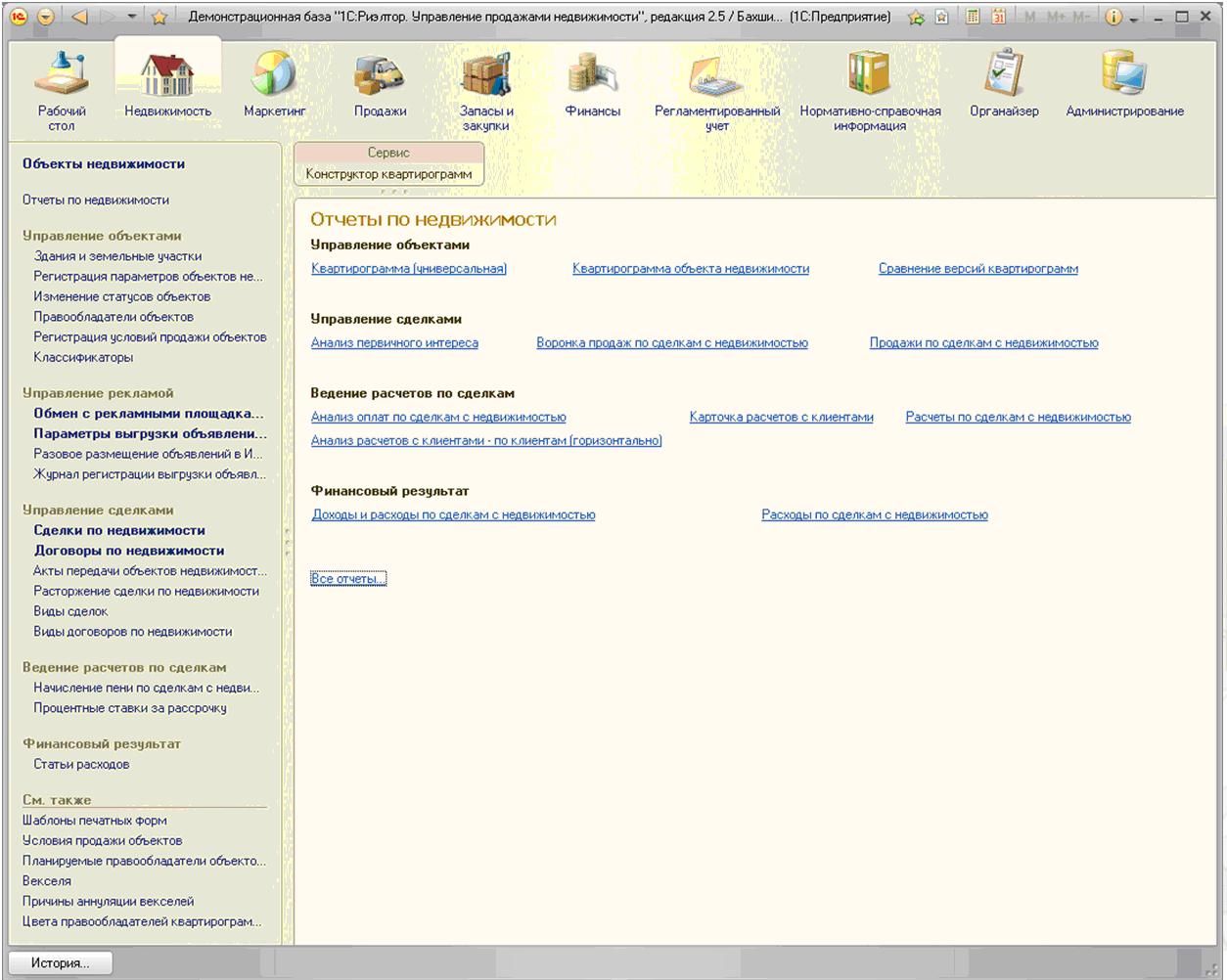


Рисунок 1.4 − Интерфейс «1С:Риэлтор»

Возможность работы в режиме управляемого приложения обеспечивает удаленное подключение к информационной базе через Интернет, что существенно повышает скорость регистрации информации и оперативность формирования отчетных данных.

В целях управления деятельностью, связанной с продажей объектов недвижимости, в конфигурации разработан набор подсистем, обеспечивающих её функциональные возможности.

В целом решение позволяет повысить эффективность подготовки и проведения сделок с недвижимостью в строительных и девелоперских компаниях, в риэлторских компаниях и агентствах недвижимости.

2 Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь данного программного решения должен иметь возможность:

* регистрировать себя в системе;
* входить в приложение, после ввода данных, необходимых для аутентификации;
* редактировать свои личные данные;
* просматривать данные о клиентах, недвижимости, сделках, типах недвижимости и типах сделок;
* вносить новую информацию о клиентах, недвижимости, сделках, типах недвижимости и типах сделок;
* удалять информацию о клиентах, недвижимости, сделках, типах недвижимости и типах сделок;
* подбирать недвижимость по её адресу, цене и площади;
* генерировать договоры по сделкам;
* выводить информацию в формат Excel;
* просматривать информацию с использованием сортировки;
* прикреплять изображения к информации о недвижимости;

Данные требования обобщены в виде диаграммы вариантов использования разрабатываемого программного средства, которая приведена в приложении А. Она отражает функциональность программного средства с точки зрения получения значимого результата для пользователя.

К последнему уровню требований относятся функциональные требования. Функциональные требования определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Данные требования могут быть описаны в виде утверждений, способов взаимодействия и методов реализаций. После проведения анализа были выявлены следующие функциональные требования:

* архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command, Repository, Unit of Work;
* вся информация должна храниться в базе данных;
* интерфейс приложения должен быть полностью локализован. Во время работы с приложением пользователь должен иметь возможность сменить язык на один из представленных языков: английский или русский;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* просмотр данных должен быть реализован в виде таблиц с возможностью сортировки по столбцам;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* приложение должно предоставлять возможность добавлять объекты с помощью отдельной панели;
* приложение должно предоставлять возможность поиска недвижимости по следующим критериям: адрес, цена и площадь.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

3 Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

**3.1 Проектирование базы данных**

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности − это любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 9 сущностей:

* сотрудник;
* фото сотрудника;
* клиент;
* недвижимость;
* тип недвижимости;
* владелец недвижимости;
* фото недвижимости;
* сделка;
* тип сделки.

Также в ER-модели были определены необходимые связи. Например, между сущностями сотрудник и фото сотрудника была установлена связь один-к-одному. Для каждой сущности были выделены атрибуты. Например, для клиента в качестве атрибутов были выделены такие характеристики, как идентификатор, фамилия, имя, отчество, паспорт, адрес, телефон и день рождения.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД. Даталогическая модель базы данных представлена в приложении Б.

**3.2 Проектирование доступа к базе данных**

Для инкапсулирования логики работы с источниками данных в программе используется паттерн Репозиторий. Класс, реализующий данный паттерн, не содержит бизнес-логику, не управляет бизнес-процессами, он только содержит операции над данными. Как правило, репозиторий реализует CRUD-интерфейс, то есть представляет операции по извлечению, добавлению, редактированию и удалению данных.

Для организации доступа к одному подключению для всех репозиториев приложения используется другой паттерн − Unit Of Work.

## **3.3 Проектирование архитектуры приложения**

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбранных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.1 – Структура шаблона MVVM

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Представление выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновлённое значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Model или Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения. В данном решении в роли модели будет выступать Репозиторий, который организовывает доступ к хранилищу данных − базе данных.

**3.4 Проектирование уровня представления**

Пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, поэтому разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей.

Общие требования к интерфейсу программы:

* любое окно может быть закрыто;
* главное окно должно быть масштабируемым;
* главное окно может быть свёрнуто;
* главное и стартовое окно должны предоставлять возможность смены языка программы;
* все ошибки валидации должны быть отображены и видны пользователю.

После запуска программы должно быть открыто окно, с помощью которого пользователь может осуществить регистрацию или вход в систему. Стартовое окно должно быть интуитивно понятным и простым. После регистрации или аутентификации пользователя в случае успеха должно быть открыто главное окно, при этом стартовое окно должно закрыться. В случае неуспеха должно быть показано окно с информацией, описывающей ошибку, которая привела к этой ситуации.

Главное окно должно быть реализовано в виде контейнера для страниц. Оно должно содержать удобное меню для навигации по страницам и панель управления. Для каждого пункта меню будет вызываться соответствующая страница, при этом такие элементы, как меню и панель управления будут оставаться неизменными.

Каждая страница должна соответствовать следующим требованиям:

* страница должна содержать таблицу, отображающую данные об одной конкретной сущности;
* на странице должна быть панель для добавления нового объекта сущности;
* на странице должна быть реализована возможность удаления и редактирования данных.

4 Реализация программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

**4.1 Реализация сущностей**

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Code-First. При данном подходе модель EDMX не используется. Создание базы данных происходит из созданной вручную модели объектов C#. Созданные модели объектов совпадают с сущностями, которые были сформированы раннее в разделе 3.1.

Также стоит отметить, что именно на этом этапе происходит валидация данных. Она реализована с помощью интерфейса IDataErrorInfo, который имплементируют сущностные классы. Данный интерфейс требует реализации свойства Error и индексатора. Свойство используется для указания общей ошибки, а индексатор указывает на конкретное свойство, в котором произошла ошибка валидации.

Диаграмма классов UML для сущностных классов представлена в приложении В.

Подробный код класса, соответствующего сущности сделки, представлен в приложении Г.

**4.2 Реализация уровня доступа к данным**

В программном решении доступ к данным должен быть организован с помощью паттернов Repository и Unit of Work.

Паттерн Repository позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. В соответствии с паттерном Repository был создан обобщенный интерфейс IRepository<TEntity> и обобщённый класс, реализующий этот интерфейс, GenericRepository<TEntity>. Код интерфейса IRepository <TEntity> представлен в приложении Д.

Паттерн Unit of Work представлен классом UnitOfWork, в котором находятся реализации методов сохранения и создания обобщённого репозитория. Также этот класс реализует интерфейс IDisposable. Интерфейс IDisposable объявляет один единственный метод Dispose, в котором при реализации интерфейса в классе должно происходить освобождение неуправляемых ресурсов. Код класса UnitOfWork представлен в приложении Е.

Полная структура элементов, а именно двух классов и одного интерфейса, данного уровня представлена на рисунке 4.1.

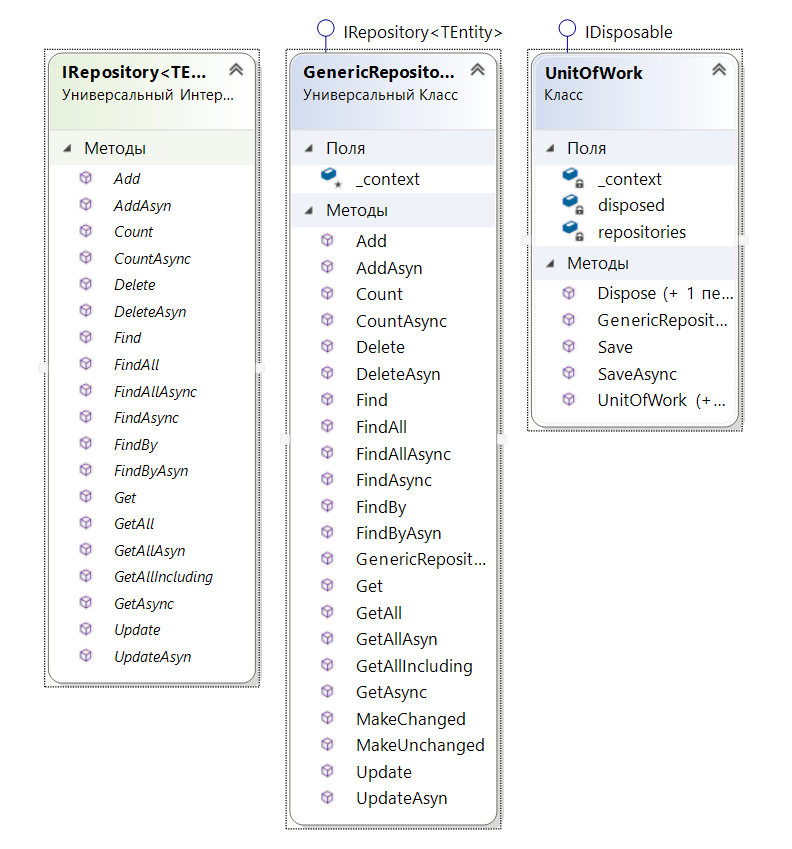


Рисунок 4.1 – Repository и Unit of Work

Стоит отметить, что каждый метод для работы с базой данных представлен в двух вариантах: синхронном и асинхронном.

**4.3 Реализация архитектуры MVVM**

Для ускорения создания и разработки приложений с архитектурой MVVM существует ряд различных библиотек. В качестве такой библиотеки в данном решении был выбран инструментарий MVVM Light Toolkit.

Для реализации паттерна MVVM файлы программы были распределены по соответствующим пространствам имён и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 4.2.

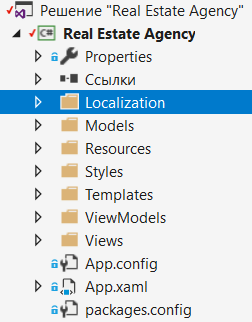


Рисунок 4.2 – Логические модули проекта

В папке Models содержатся сущностные классы, которые используются для создания БД. Их описание представлено в разделе 4.1. Также в этой папке находится реализация паттернов Repository и Unit of Work. Их подробная реализация приводится в разделе 4.2.

В папке также находится статический класс Hasher, который является вспомогательным классом. Он содержит методы, которые нужны для хэширования пароля.

Хеширование паролей является одним из самых основных соображений безопасности, которые необходимо делать, при разработке приложения, принимающего пароли от пользователей. Применяя хеширующий алгоритм к пользовательским паролям перед сохранением их в базе данных, разгадывание оригинального пароля для атакующего базу данных становится невозможным, в то же время сохраняется возможность сравнения полученного хеша с оригинальным паролем. В данном программном решении был использован хэширующий алгоритм MD5.

В папке ViewModels находятся все классы ViewModel для каждого представления. Программа спроектирована таким образом, что в ней предполагается одна и та же работа над разными сущностями, поэтому для удобства был выделен абстрактный класс BasePageVM, в который вынесен общий функционал для работы с некоторыми страницами. Также в эту папку вынесены классы для работы с навигацией по страницам.

В папке Views хранятся все используемые представления.

**4.4 Реализация представления**

Для разработки графической части приложения была выбрана технология WPF.

Windows Presentation Foundation (WPF) — это библиотека для создания пользовательских интерфейсов для интеллектуальных клиентских приложений. В основе WPF лежит мощная инфраструктура, основанная на DirectX — API-интерфейсе графики с аппаратным ускорением, который обычно используется в современных компьютерных играх. Это означает, что применение развитых графических эффектов не приведёт к снижению производительности.

Одной из важных особенностей WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Разработка с использованием XAML позволяет отделить графический интерфейс от логики приложения, а также создавать насыщенный интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#.

В конечном итоге в приложении «Агентство недвижимости» было реализовано четыре окна:

* стартовое окно (регистрация и вход в систему);
* главное окно (содержит различные страницы);
* окно для вывода ошибок;
* окно для просмотра изображения.

А также восемь страниц:

* страница для редактирования профиля;
* стартовая страница;
* страница для работы с клиентами;
* страница для работы с недвижимостью;
* страница для работы с типами недвижимости;
* страница для работы со сделками;
* страница для работы с типами сделок;

В результате выполнения данного этапа было создано функционирующее программное средство.

5 Тестирование, проверка работоспособности и анализ полученных результатов

Прежде всего были проведены тесты авторизации: проверка на пустые и неверные данные. Валидация выполнена таким образом, что кнопка для входа или регистрации будет недоступна до тех пор, пока введённые данные не будут корректными. При таком способе валидации шанс ввести невалидные данные минимален. Проверки работоспособности валидации представлены на рисунках 5.1− 5.4.

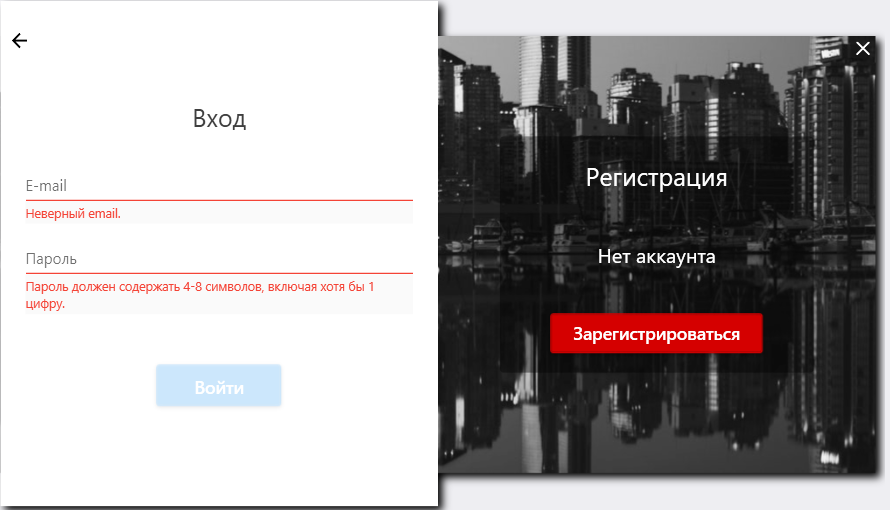


Рисунок 5.1 – Валидация формы входа

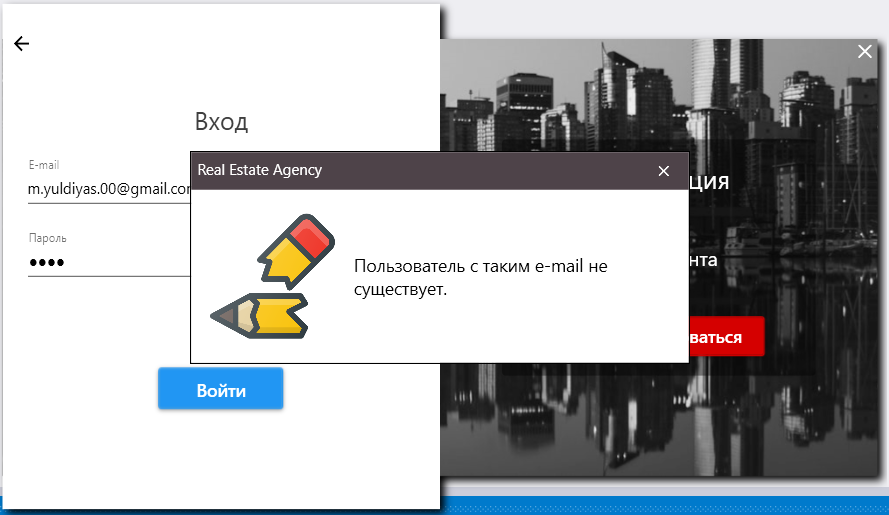


Рисунок 5.2 – Попытка входа без регистрации

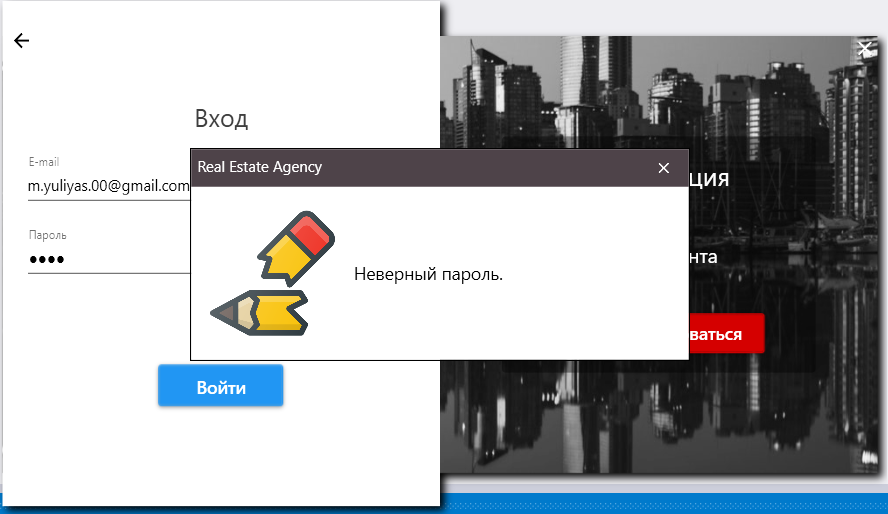


Рисунок 5.3 – Вход с неверным паролем

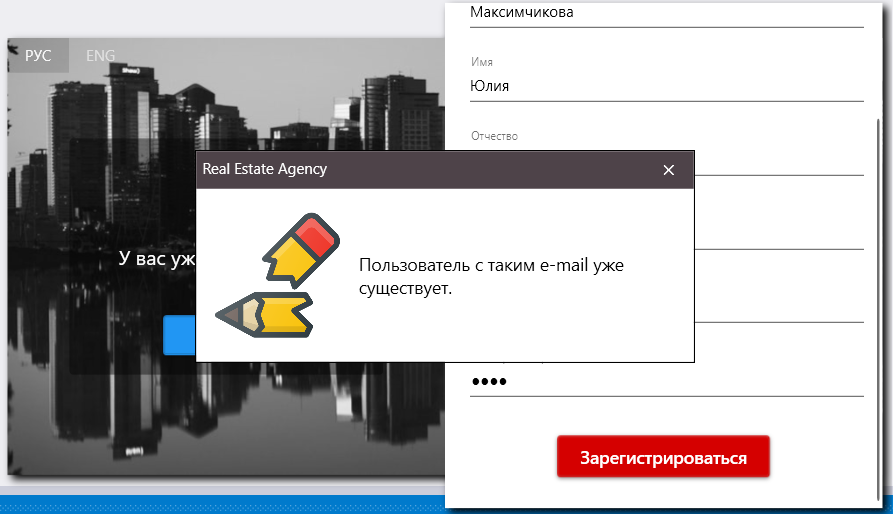


Рисунок 5.4 – Регистрация уже существующего пользователя

Также были проведены тесты на работу валидации внутри приложения. Наличие валидации является обязательным в связи с тем, что в приложении постоянно ведётся работа с базой данных.

Валидация организованна таким же образом, как и на форме авторизации, т.е. кнопки добавить и обновить будут недоступны до тех пор, пока не будут введены валидные данные. Все ошибки, возникшие при валидации, доступны и отображены удобным образом, что позволяет пользователю легко понять какие данные не валидны и быстро исправить их.

Пример результата валидации данных внутри приложения приведен на рисунке 5.5.

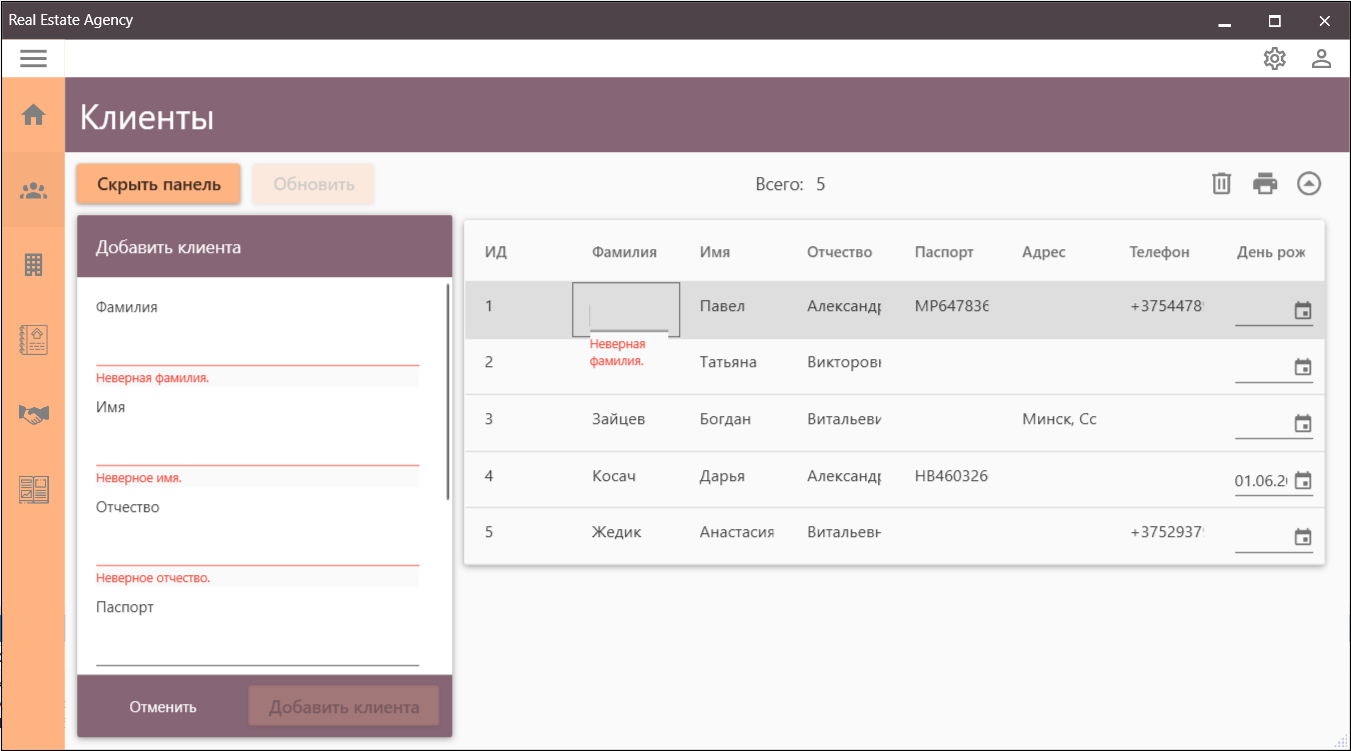


Рисунок 5.5 – Демонстрация валидации внутри приложения

На данном этапе были выполнены тесты на проверку работоспособности приложения, а именно на проверку валидации. Были проведены анализы результатов, которые показали, что валидация в приложении работает в соответствии с тем, как она задумывалась.

6 Руководство по установке и использованию

С помощью утилиты Microsoft Visual Studio Installer Projects был создан инсталлятор программы. Процесс установки программы представлен на рисунке 6.1.

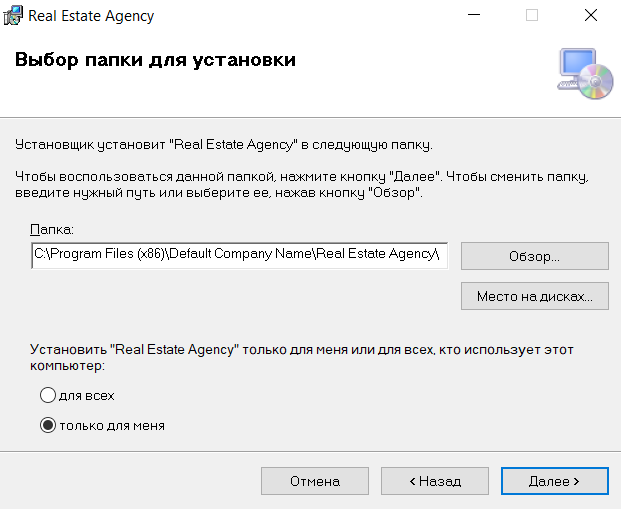


Рисунок 6.1 – Установка программы

Запустим программу из установленной папки. При запуске программы первым откроется окно авторизации. Оно приведено на рисунке 6.2.

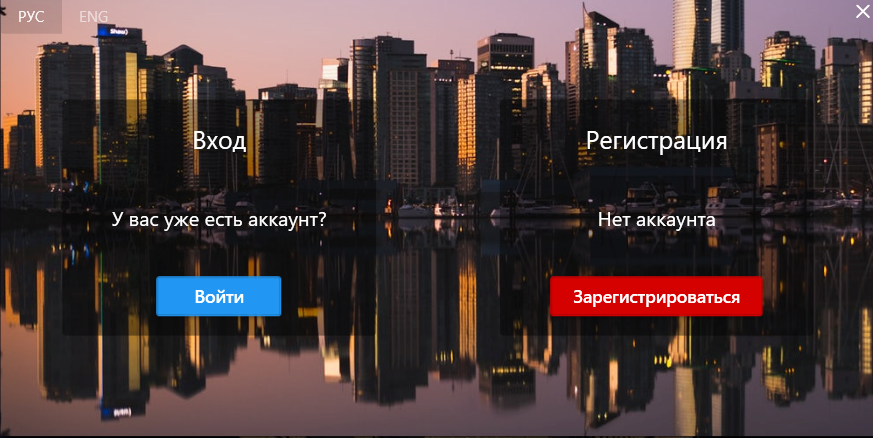


Рисунок 6.2 – Стартовое окно

Далее, если вход или регистрация успешна, открывается главное окно со стартовой страницей. На стартовой странице содержится краткое описание возможностей программы. На панели управления главного окна находится кнопка для перехода на страницу редактирования профиля сотрудника, а также кнопка настроек. В настройках можно сменить текущий язык приложения. В данный момент доступно два языка: русский и английский. По умолчанию установлен русский язык. Слева на главном окне расположено меню, с помощью которого мы можем перемещаться по страницам.

Главное окно и страница редактирования профиля сотрудника представлены на рисунках 6.3 и 6.4 соответственно.

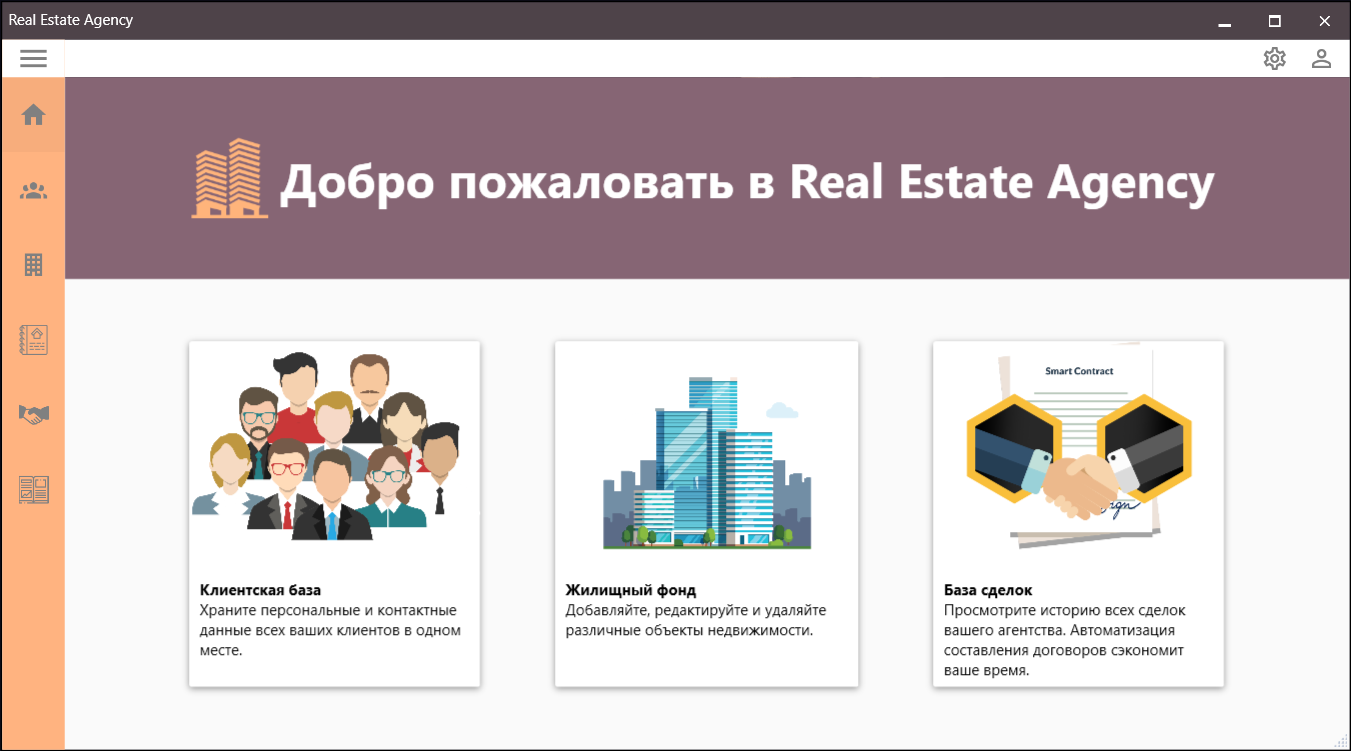


Рисунок 6.3 – Главное окно и стартовая страница

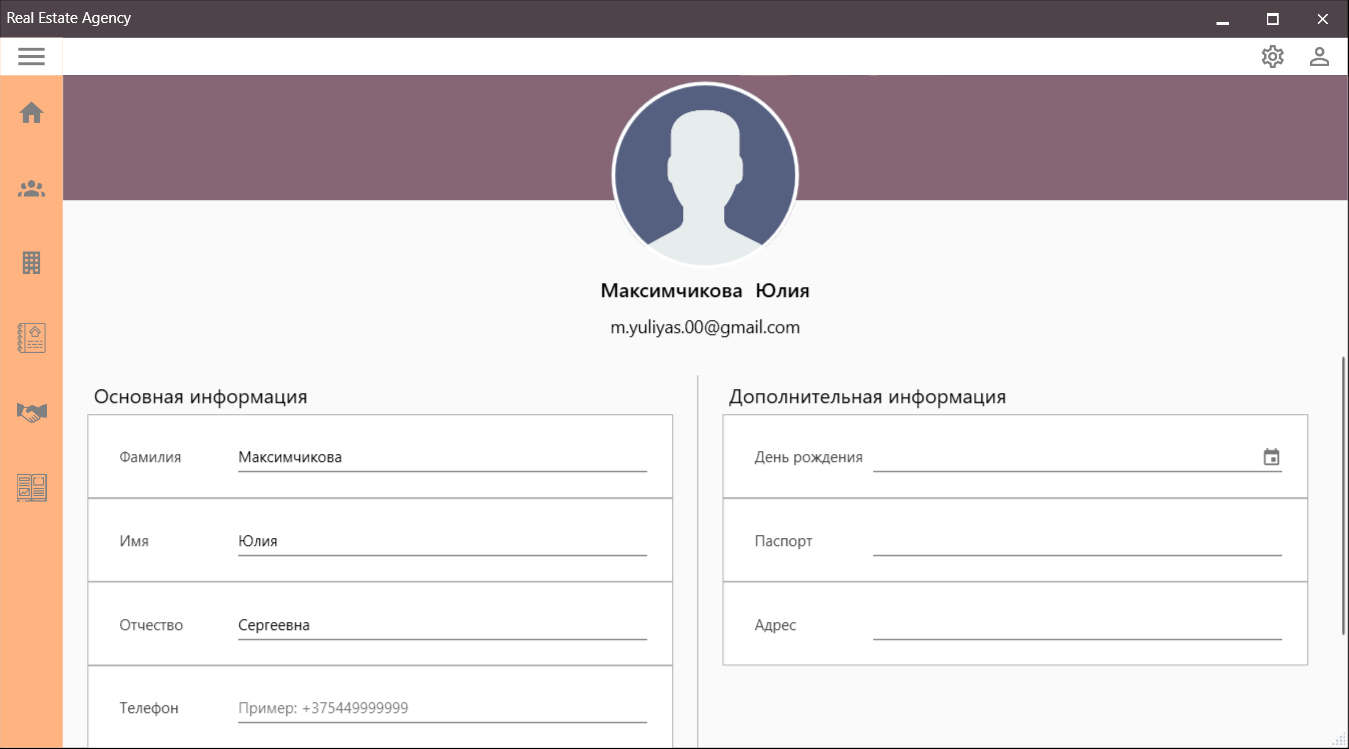


Рисунок 6.4 – Страница редактирования профиля сотрудника

На страницах Клиенты, Недвижимость, Сделки, Типы недвижимости, Типы сделок можно добавлять новые объекты в базу, удалять и редактировать уже существующие. Все данные представлены в виде таблиц. На каждой странице имеется кнопка для вывода информации в формат Excel.

Вышеперечисленные страницы показаны на рисунках 6.5-6.9.

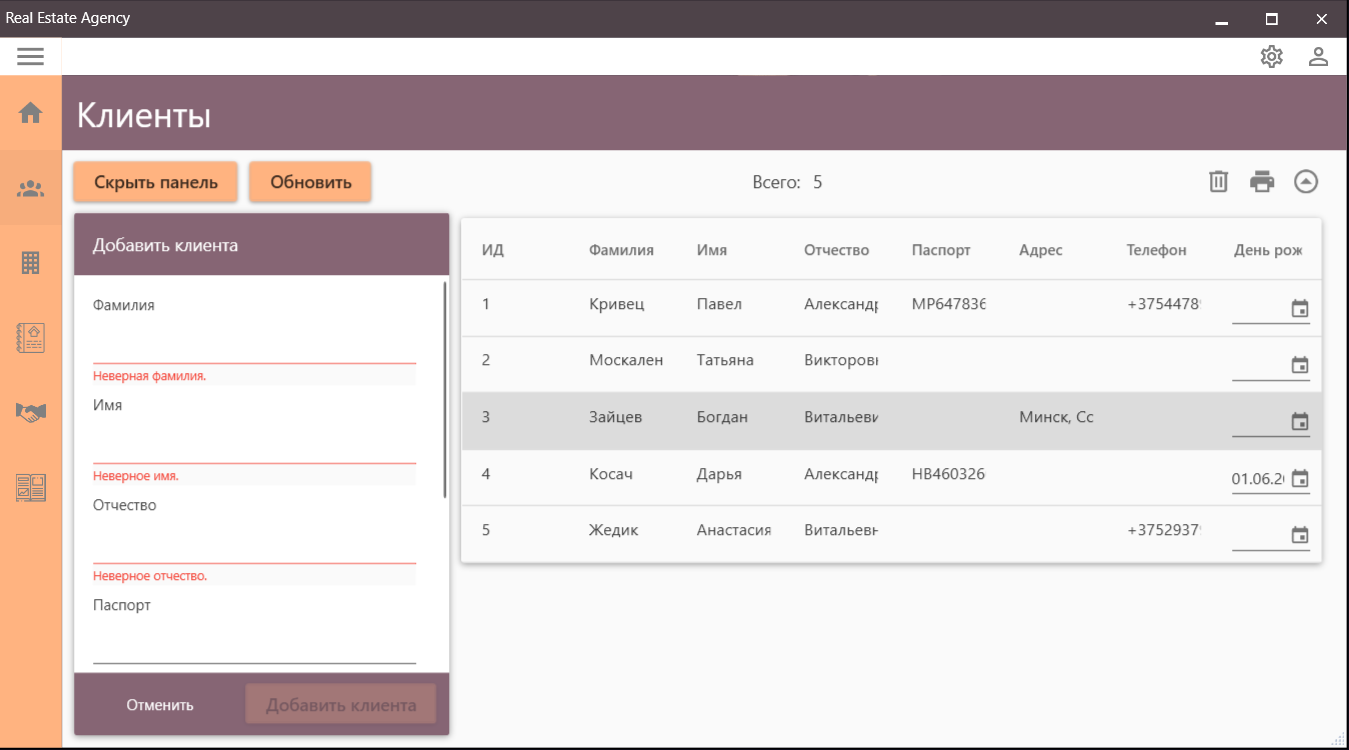


Рисунок 6.5 – Страница Клиенты

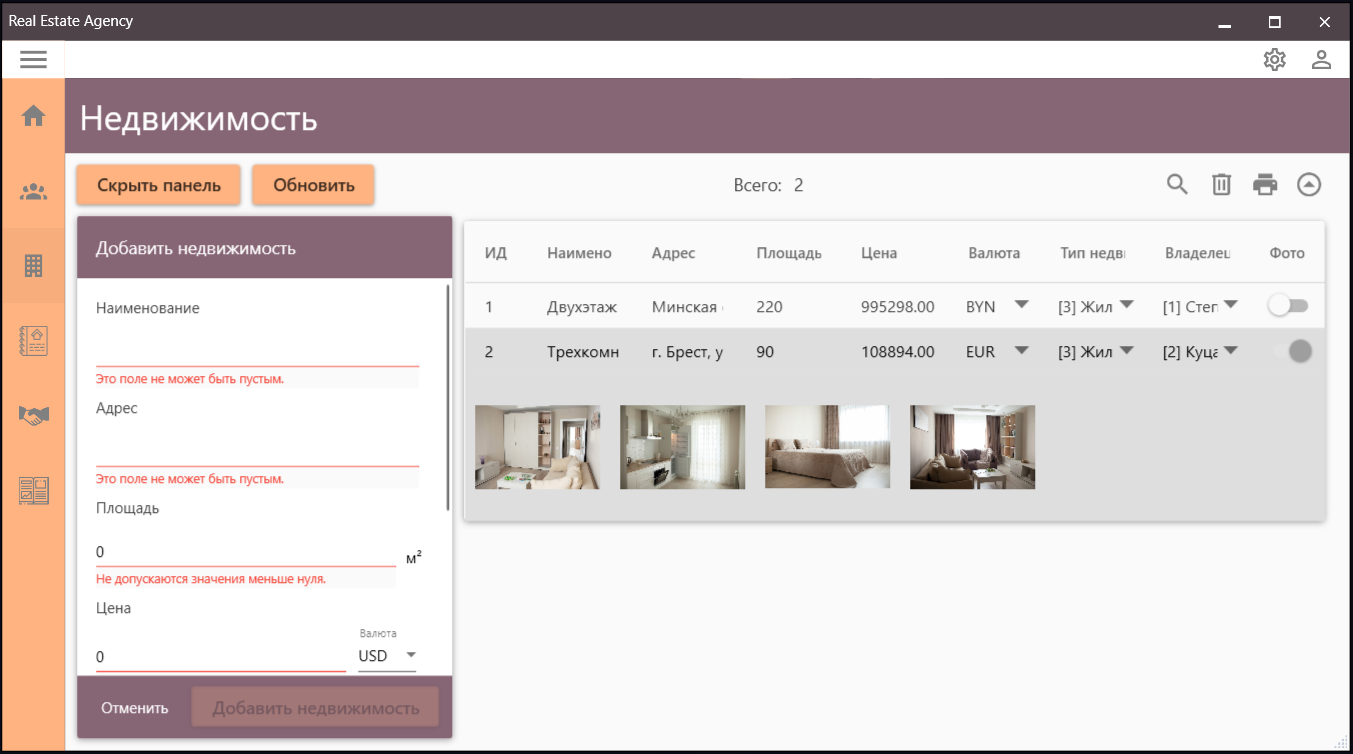


Рисунок 6.6 – Страница Недвижимость

На странице Недвижимость при добавлении объекта можно прикреплять фото. Таблица спроектирована таким образом, чтобы прикреплённые фото можно было просмотреть. При двойном клике на фото открывается окно с этим фото. Также на данной странице можно выполнить поиск по одному или нескольким перечисленным критериям: адрес, площадь, цена и валюта.

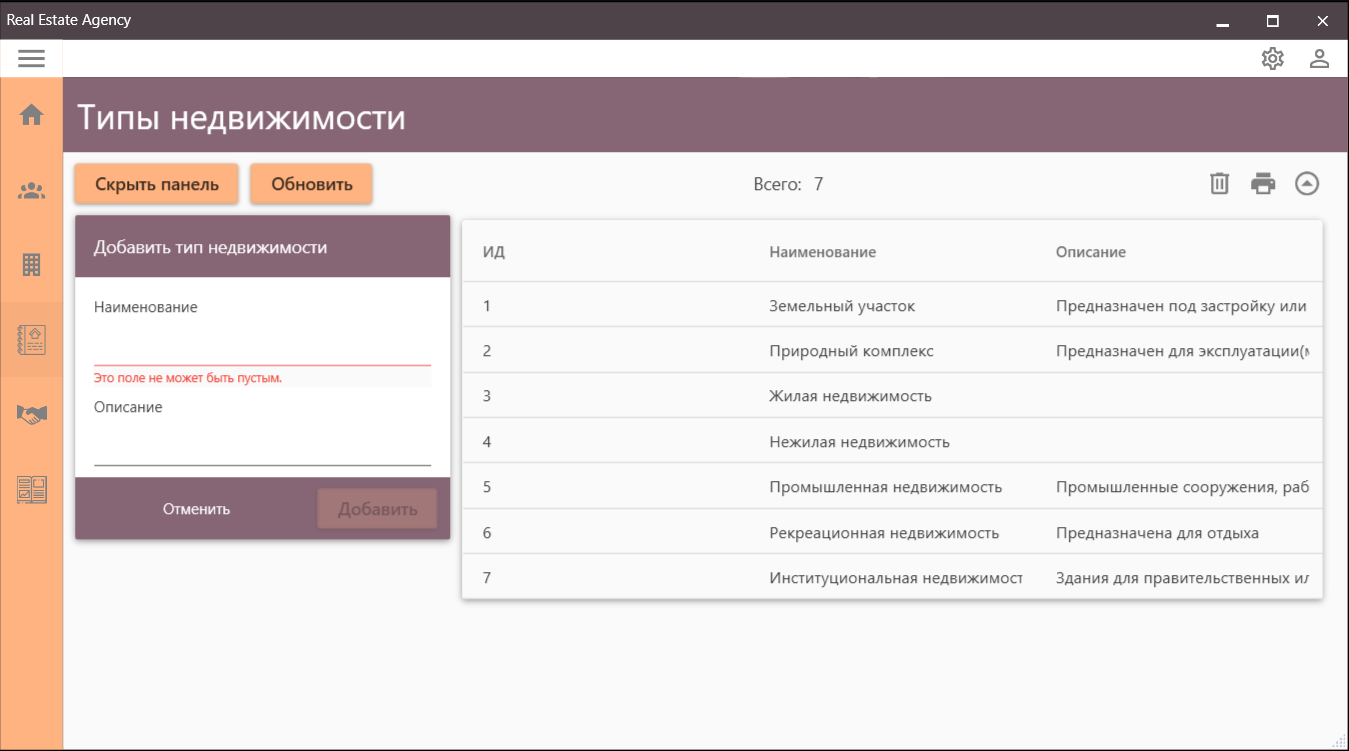


Рисунок 6.7 – Страница Типы недвижимости

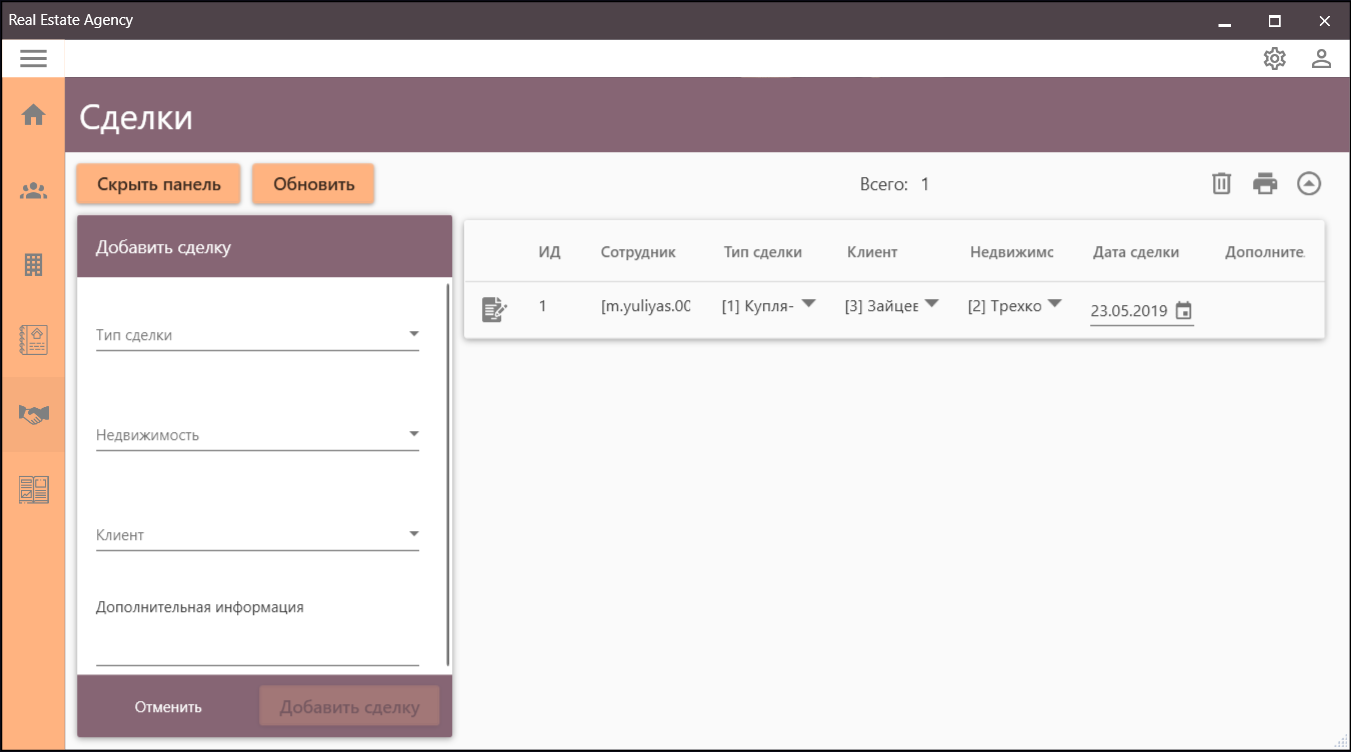
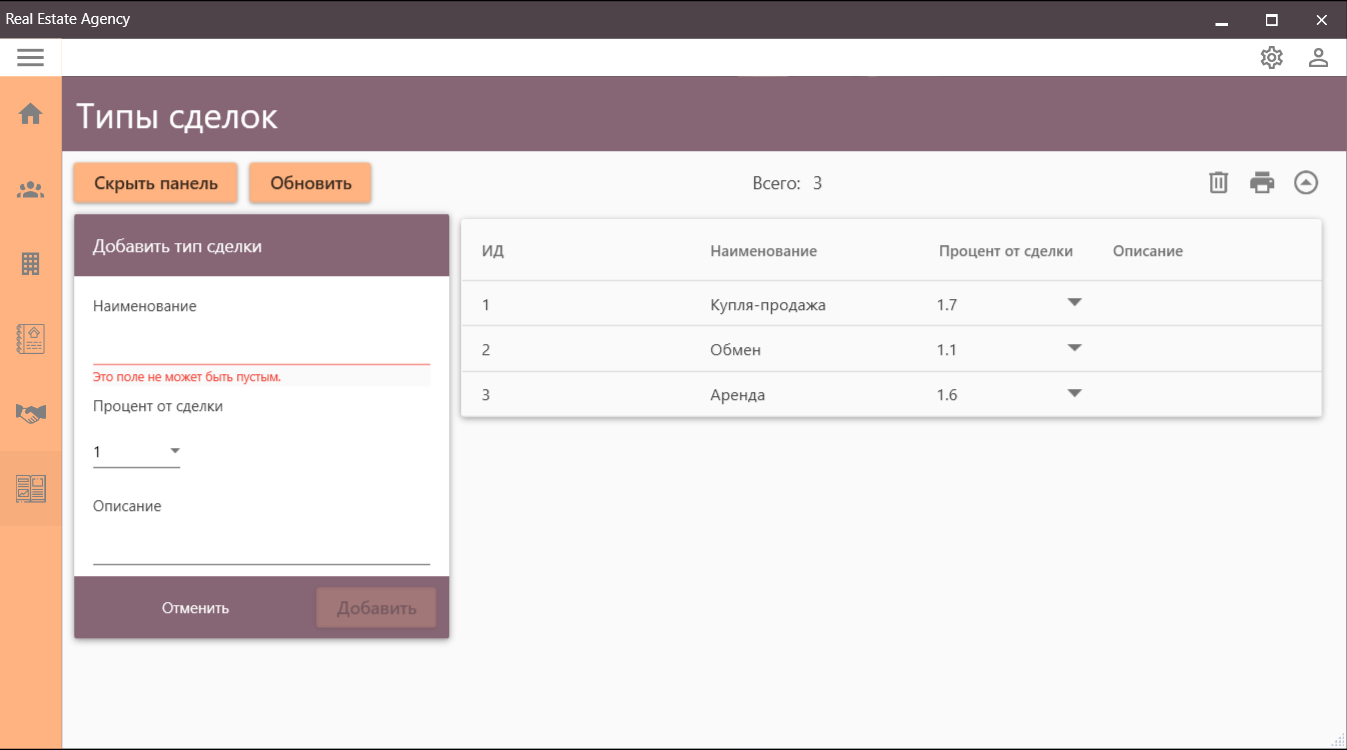


Рисунок 6.8 – Страница Сделки

На странице Сделки внутри каждой строки таблицы есть кнопка Составить договор. При её нажатии происходит автоматическая генерация договора по данной сделке.

Рисунок 6.9 – Страница Типы сделок

На каждой странице есть информация об общем количестве объектов, что позволяет при большом объём данных легко просмотреть количество объектов, находящихся в хранилище.

ЗАКЛЮЧЕНИЕ

В процессе решения поставленной задачи была достигнута поставленная цель по созданию программного средства «Агентство недвижимости». При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения.

В программном средстве были реализованы следующие функции:

* регистрация пользователя в системе;
* вход пользователя в систему;
* пользователь может редактировать свои данные, а также загружать своё фото;
* смена языка приложения;
* добавление, редактирование, удаление клиентов;
* добавление, редактирование, удаление недвижимости;
* добавление, редактирование, удаление сделок;
* добавление, редактирование, удаление типов недвижимости;
* добавление, редактирование, удаление типов сделок;
* прикрепление фотографий к объекту недвижимости при его добавлении;
* автоматическая генерация договора по сделке;
* перенос данных в формат Excel.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает, верно, а требования технического задания выполнены в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.

2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.

3. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: http://msdn.microsoft.com/library/rus/. Дата доступа: 20.04.2019

4. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com. Дата доступа: 20.04.2019

5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru Дата доступа: 13.04.2019

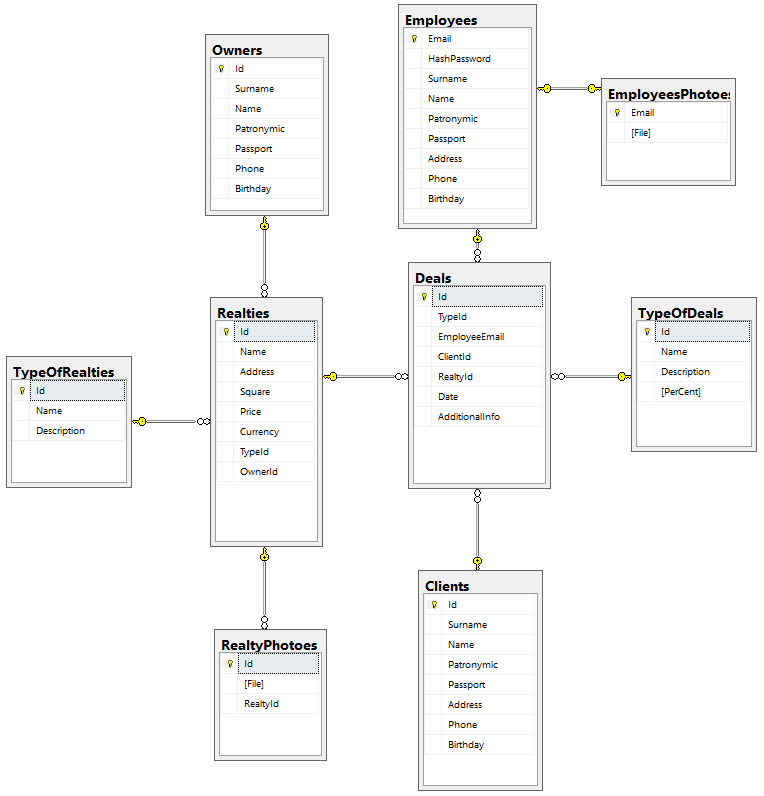
ПРИЛОЖЕНИЕ А

Диаграмма использования для сотрудника агентства



ПРИЛОЖЕНИЕ Б

Даталогическая модель базы данных



ПРИЛОЖЕНИЕ В

UML диаграмма сущностных классов



ПРИЛОЖЕНИЕ Г

Код класса, соответствующего сущности сделка

public class Deal : IDataErrorInfo

{

[NotMapped]

private string \_additionalInfo;

public int Id { get; set; }

public int TypeId { get; set; }

[Required]

public virtual TypeOfDeal Type { get; set; }

[Required]

public string EmployeeEmail { get; set; }

[ForeignKey("EmployeeEmail")]

public virtual Employee Employee { get; set; }

public int ClientId { get; set; }

[Required, ForeignKey("ClientId")]

public virtual Client Client { get; set; }

public int RealtyId { get; set; }

[Required, ForeignKey("RealtyId")]

public virtual Realty Realty { get; set; }

public DateTime Date { get; set; }

public string AdditionalInfo

{

get => \_additionalInfo;

set

{

if (value == "")

\_additionalInfo = null;

else

\_additionalInfo = value;

}

}

[NotMapped]

public string Error { get => string.Empty; }

[NotMapped]

public string this[string propertyName]

{

get

{

string result = string.Empty;

propertyName = propertyName ?? string.Empty;

if (propertyName == "Type" && Type == null)

result = App.Current.Resources["\_NotNull"].ToString();

if (propertyName == "Realty" && Realty == null)

result = App.Current.Resources["\_NotNull"].ToString();

if (propertyName == "Client" && Client == null)

result = App.Current.Resources["\_NotNull"].ToString();

return result;

}

}

public double GetCommission() => (double)Realty.Price \* (Type.PerCent / 100.0);

}

ПРИЛОЖЕНИЕ Д

Код интерфейса GenericRepository<TEntity>

interface IRepository<TEntity> where TEntity : class

{

TEntity Add(TEntity t);

Task<TEntity> AddAsyn(TEntity t);

int Count();

Task<int> CountAsync();

void Delete(TEntity entity);

Task<int> DeleteAsyn(TEntity entity);

TEntity Find(Expression<Func<TEntity, bool>> match);

Task<TEntity> FindAsync(Expression<Func<TEntity, bool>> match);

ICollection<TEntity> FindAll(Expression<Func<TEntity, bool>> match);

Task<ICollection<TEntity>> FindAllAsync(Expression<Func<TEntity, bool>> match);

IQueryable<TEntity> FindBy(Expression<Func<TEntity, bool>> predicate);

Task<ICollection<TEntity>> FindByAsyn(Expression<Func<TEntity, bool>> predicate);

TEntity Get(object id);

Task<TEntity> GetAsync(object id);

IQueryable<TEntity> GetAll();

Task<ICollection<TEntity>> GetAllAsyn();

IQueryable<TEntity> GetAllIncluding(params Expression<Func<TEntity, object>>[] includeProperties);

TEntity Update(TEntity t, object key);

Task<TEntity> UpdateAsyn(TEntity t, object key);

}

ПРИЛОЖЕНИЕ Е

Код класса UnitOfWork

public class UnitOfWork : IDisposable

{

private readonly REAContext \_context;

private Dictionary<string, object> repositories;

private bool disposed;

public UnitOfWork()

{

\_context = new REAContext();

}

public UnitOfWork(REAContext context)

{

\_context = context;

}

public GenericRepository<T> GenericRepository<T>() where T : class

{

if (repositories == null)

{

repositories = new Dictionary<string, object>();

}

var type = typeof(T).Name;

if (!repositories.ContainsKey(type))

{

var repositoryType = typeof(GenericRepository<>);

var repositoryInstance = Activator.CreateInstance(repositoryType.MakeGenericType(typeof(T)), \_context);

repositories.Add(type, repositoryInstance);

}

return (GenericRepository<T>)repositories[type];

}

public virtual void Save()

{

\_context.SaveChanges();

}

public async virtual Task<int> SaveAsync()

{

return await \_context.SaveChangesAsync();

}

public virtual void Dispose(bool disposing)

{

if (!this.disposed)

{

if (disposing)

{

\_context.Dispose();

}

this.disposed = true;

}

}

public void Dispose()

{

Dispose(true);

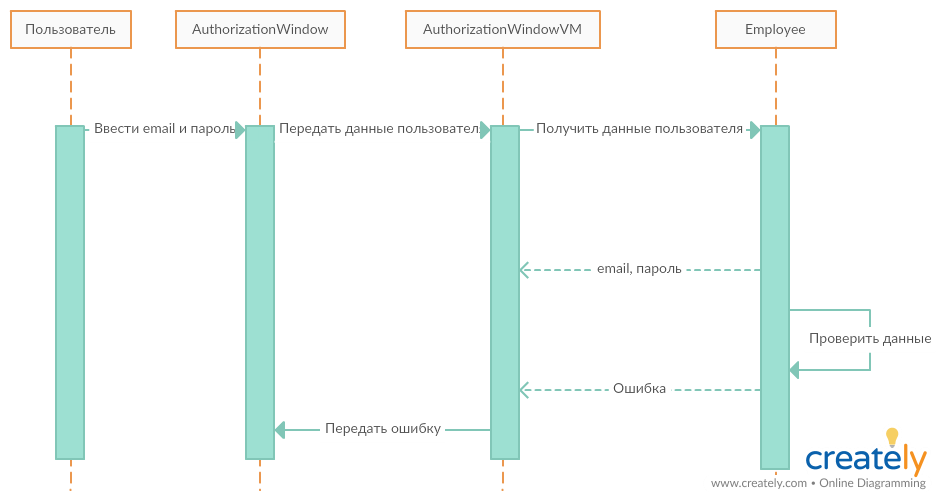
GC.SuppressFinalize(this);

}

}

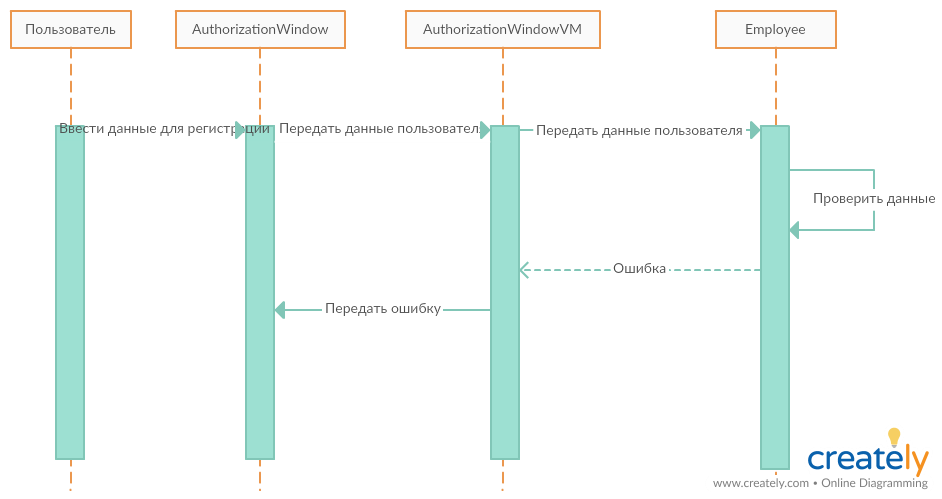
ПРИЛОЖЕНИЕ Ж

Диаграмма последовательности входа в систему



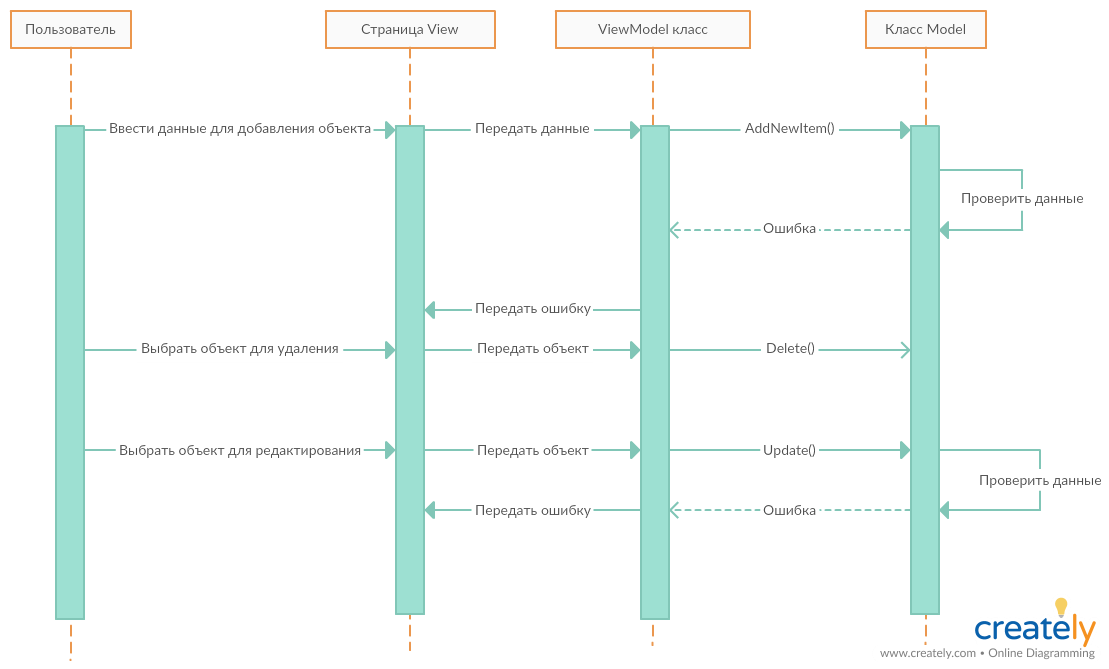
ПРИЛОЖЕНИЕ И

Диаграмма последовательности регистрации



ПРИЛОЖЕНИЕ К

Диаграмма последовательности работы с объектами



ПРИЛОЖЕНИЕ Л

Алгоритм работы с программой

