Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры ЭИ БГУИР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.П. Котковец |
|  | \_\_\_.\_\_\_\_.2021 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ ОЦЕНКИ ОБЪЕКТОВ НЕДВИЖИМОСТИ»**

БГУИР КП 1-40 01 02-08 015 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 972301  КРУТЬКО Алексей Игоревич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2021  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2021

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc90231921)

[1 Описание предметной области системы автоматизации оценки объектов недвижимости 7](#_Toc90231922)

[2 Постановка задачи и обзор методов её решения 9](#_Toc90231923)

[3 Функциональное моделирование на основе стандарта IDEF0 12](#_Toc90231924)

[4 Информационная модель системы и ее описание 17](#_Toc90231925)

[5 Модели представления системы на основе UML 19](#_Toc90231926)

[5.1 Диаграмма вариантов использования 19](#_Toc90231927)

[5.2 Диаграмма последовательности 20](#_Toc90231928)

[5.3 Диаграмма состояний 21](#_Toc90231929)

[5.4 Диаграмма классов 22](#_Toc90231930)

[5.5 Диаграмма развертывания 24](#_Toc90231931)

[5.6 Диаграмма компонентов 25](#_Toc90231932)

[6 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 27](#_Toc90231933)

[6.1 Алгоритм добавления нового оценщика 27](#_Toc90231934)

[6.2 Схема алгоритма клиент-серверного взаимодействия 27](#_Toc90231935)

[7 Руководство пользователя 28](#_Toc90231936)

[7.1 Вход под пользователем 31](#_Toc90231937)

[7.2 Вход под оценщиком 35](#_Toc90231938)

[7.3 Вход под администратором 41](#_Toc90231939)

[8 Результаты тестирования разработанной системы 46](#_Toc90231940)

[Заключение 48](#_Toc90231941)

[Список использованных источников 49](#_Toc90231942)

[Приложение А (обязательное) Схемы алгоритмов 50](#_Toc90231943)

[Приложение Б (обязательное) Листинг кода 52](#_Toc90231944)

[Приложение В (обязательное) Листинг кода Базы данных 67](#_Toc90231945)

# Введение

В последнее время рынок недвижимости в нашей стране развивается все активнее. У белорусов стало гораздо больше возможностей для того, чтобы по своему усмотрению распоряжаться собственной недвижимостью. Все чаще для этого приходится проводить оценку недвижимости.

Сейчас на рынке недвижимости представлены жилые квартиры и комнаты, офисные здания и помещения, здания производственного и торгового назначения, коттеджи, дачи и сельские дома с земельными участками.

В производственно-экономической практике экономических субъектов появляется множество случаев, когда возникает необходимость оценить рыночную стоимость имущества предприятий. Без оценки стоимости не обходится ни одна операция по купле-продаже имущества, кредитованию под залог, страхованию, разрешению имущественных споров, налогообложению и т. д.

Оценка стоимости недвижимости актуальна всегда до тех пор, пока существует понятие «собственность». С каждым годом всё больше людей и организаций имеют дело с операциями, связанными с недвижимостью, вследствие этого растёт потребность в продукте, который помог бы автоматизировать данный процесс.

Сегодня мы стоим на пороге создания качественно нового общества – информационного. Жизнь и практическая деятельность в нем неразрывно связана с грамотной организацией информационных процессов, освоением и использованием современных информационных технологий.

Современные информационные технологии непосредственно связаны с использованием компьютера. Информационные технологии – это современные виды информационного обслуживания, организованные на базе компьютерной технике и коммуникационных средств, предназначенных для автоматизированного сбора, обработки информации по заранее отработанным алгоритмам, хранения больших объёмов информации на разных носителях, передачи и использования информации для получения определенных, заведомо ожидаемых, результатов.

На сегодняшний день автоматизация деятельности человека проводится в различных сферах, начиная от системы производства, заканчивая сферой услуг.

Объектом исследования являются сервисы оценки объектов недвижимости, а предметом – методы и инструментальные средства разработки программного обеспечения оценки недвижимости.

Целью данного курсового проекта является оптимизация процесса оценки объектов недвижимости, повышение эффективности оценочной деятельности путём сокращение временных издержек.

Для достижения поставленной цели необходимо решить следующие задачи:

* исследовать процесс оценки объектов недвижимости;
* изучить какие подходы к определению стоимости существуют;
* произвести анализ бизнес-процессов сервиса оценки недвижимости;
* разработать программный продукт с удобным интерфейсом;
* провести тестирование программного продукта на соответствие всем поставленным задачам и требованиям;

# Описание предметной области системы автоматизации оценки объектов недвижимости

Недвижимость — вид имущества, признаваемого в законодательном порядке недвижимым.

К недвижимости по происхождению относятся земельные участки, участки недр и все, что прочно связано с землёй, то есть объекты, перемещение которых без несоразмерного ущерба их назначению невозможно, в том числе здания, сооружения, объекты незавершённого строительства.

Различные блага (предметы, вещи, имущество) создаются, продаются и приобретаются для удовлетворения определенных потребностей государства, юридических и физических лиц, а также для получения выгод от владения ими и использования. От возможности того или иного объекта собственности удовлетворить имеющиеся потребности и от преимуществ, которые принесет обладание этим объектом в дальнейшем, зависит его ценность, а следовательно, стоимость.

Все операции и сделки с недвижимостью требуют знания стоимости объекта собственности. В рыночных условиях стоимость недвижимости зависит от факторов, тенденций и изменений в экономике и жизни общества в целом. Необходимость в оценке недвижимости возникает в следующих случаях:

* операции купли-продажи или сдаче в аренду;
* акционирование предприятий и перераспределении имущественных долей;
* привлечение новых пайщиков и дополнительной эмиссии акций;
* кадастровая оценка для целей налогообложения объектов недвижимости: зданий и земельных участков;
* страхование объектов недвижимости;
* кредитование под залог объектов недвижимости;
* внесение объектов недвижимости в качестве вклада в уставный капитал предприятий и организаций;
* разработка инвестиционных проектов и привлечении инвесторов;
* ликвидация объектов недвижимости;
* исполнение прав наследования, судебного приговора, разрешение имущественных споров;
* другие операции, связанные с реализацией имущественных прав на объекты недвижимости.

Основная задача при определении стоимости – прогнозирование количества, качества и продолжительности будущих выгод от владения объектом оценки и пересчет этих выгод в настоящую стоимость. Собственность анализируется исходя из принципов оценки.

Принципы оценки делятся на четыре группы:

* принципы, основанные на представлениях пользователя (принципы полезности, замещения и ожидания);
* принципы, вытекающие из процесса эксплуатации недвижимости (принципы вклада, остаточной продуктивности, предельной производительности, сбалансированности, экономического размера и экономического разделения);
* принципы, обусловленные действием рыночной среды (принципы альтернативности, изменчивости, зависимости, предложения и спроса, конкуренции и соответствия);
* принцип наилучшего и наиболее эффективного использования – главный принцип оценки. Этот принцип соединяет в себе все другие принципы и является фундаментом любой оценки недвижимости.

Оценка недвижимости включает в себя определение одного или нескольких видов стоимости объекта:

* рыночной стоимости - наиболее вероятной стоимости объекта недвижимости при его продаже на открытом рынке;
* восстановительной стоимости - суммы затрат в рыночных ценах на создание объекта, идентичного оцениваемому объекту недвижимости;
* инвестиционной стоимости - стоимости, определяемой исходя из доходности объекта недвижимости при заданных инвестиционных целях.

# Постановка задачи и обзор методов её решения

Для разработки программного продукта сперва необходимо поставить цели и сформулировать задачи, так как это позволит избежать возможных проблем во время разработки, связанных с некорректно сформулированными задачами и/или целями. Исходя из поставленных целей, которые должны быть достигнуты путем решения поставленной задачи, и будет разработан функционал системы.

Программное средство, предназначенное для оценки объектов недвижимости должно иметь простой и понятный каждому интерфейс, а также обладать достаточным функционалом.

В данном курсовом проекте для решения были поставлены следующие задачи:

* разработать клиент-серверное приложение на языке Java;
* реализовать простой и удобный пользовательский интерфейс;
* обеспечить возможность авторизации администратора, а также авторизации и регистрации пользователей;
* предусмотреть возможности добавления, удаления и редактирования данных;
* предусмотреть обработку исключительных ситуаций;
* предоставить пользователю графическую отчётность в виде круговых диаграмм;
* обеспечить сохранность пользовательских данных, используя алгоритм шифрования данных SHA-256;
* обеспечить формирование отчета;
* разработать оконное приложение с использованием JavaFX.

Java разработала компания Sun Microsystems в начале 90-х годов XX века. Ведущую роль в создании языка сыграл канадский инженер Джеймс Гослинг (James Gosling). На ранних этапах разработки язык назывался Oak. Затем его переименовали в честь сорта кофе Java. Связь языка с напитком отражается в логотипе.

Джеймс Гослинг и его единомышленники хотели создать язык с си-подобным синтаксисом. В то же время он должен быть более простым по сравнению с C/C++. Создатели планировали использовать Java для программирования бытовой электроники. Однако практически сразу после выпуска версии 1.0 в 1995 язык стали использовать разработчики серверного и клиентского ПО.

Java — язык программирования общего назначения. Относится к объектно-ориентированным языкам программирования, к языкам с сильной типизацией.

Создатели реализовали принцип WORA: write once, run anywhere или «пиши один раз, запускай везде». Это значит, что написанное на Java приложение можно запустить на любой платформе, если на ней установлена среда исполнения Java (JRE, Java Runtime Environment)[11].

Клиент-сервер — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчиками услуг, называемыми клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением[10].

Клиент-серверное приложение делает возможным, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети. Это позволяет упростить обслуживание вычислительной системы. В частности, замена, ремонт, модернизация или перемещение сервера, не затрагивают клиентов. Также на сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа [1].

Одно из самых существенных преимуществ приложений с архитектурой клиент-сервер состоит в приближенности данных к вычислительным процессам. Большая часть расчетов выполняется на стороне сервера, благодаря чему быстродействие значительно увеличивается.

В качестве СУБД в курсовом проекте используется MySQL.

MySQL Workbench — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL[3].

MySQL является решением для малых и средних приложений. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы [2].

Помимо универсальности и распространенности СУБД MySQL обладает целым комплексом важных преимуществ перед другими системами. В частности, следует отметить такие качества как:

* Простота в использовании. MySQL достаточно легко инсталлируется, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных;
* Обширный функционал. Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта;
* Безопасность. Система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию;
* Масштабируемость. Являясь весьма универсальной СУБД, MySQL в равной степени легко может быть использована для работы и с малыми, и с большими объемами данных;
* Скорость. Высокая производительность системы обеспечивается за счет упрощения некоторых используемых в ней стандартов[13].

1. **Функциональное моделирование на основе стандарта IDEF0**

IDEF0 – нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции. Нотация IDEF0 является одной из самых популярных нотаций моделирования бизнес-процессов[9].

Методология IDEF0 предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция - система разбивается на подсистемы, и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

Каждая сторона блока имеет особое, вполне определенное назначение. Левая сторона блока предназначена для входов, верхняя - для управления, правая - для выходов, нижняя - для механизмов. Такое обозначение отражает определенные системные принципы: входы преобразуются в выходы, управление ограничивает или предписывает условия выполнения преобразований, механизмы показывают, что и как выполняет функция[6].

На контекстной диаграмме верхнего уровня (рисунок 3.1) представлена главная функциональная модель, а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

Входными данными на этой диаграмме являются – договор, информация о пользователях и недвижимости. После соответствующей обработки на выходе мы имеем прибыль, оценённую недвижимость, отказ от соглашения.

Управление - информация, управляющая действиями работы. Обычно управляющие стрелки несут информацию, которая указывает, что должна выполнять работа. На данной диаграмме управлением является закон РБ о работе с клиентами, нормативы оценок, методика анализа.

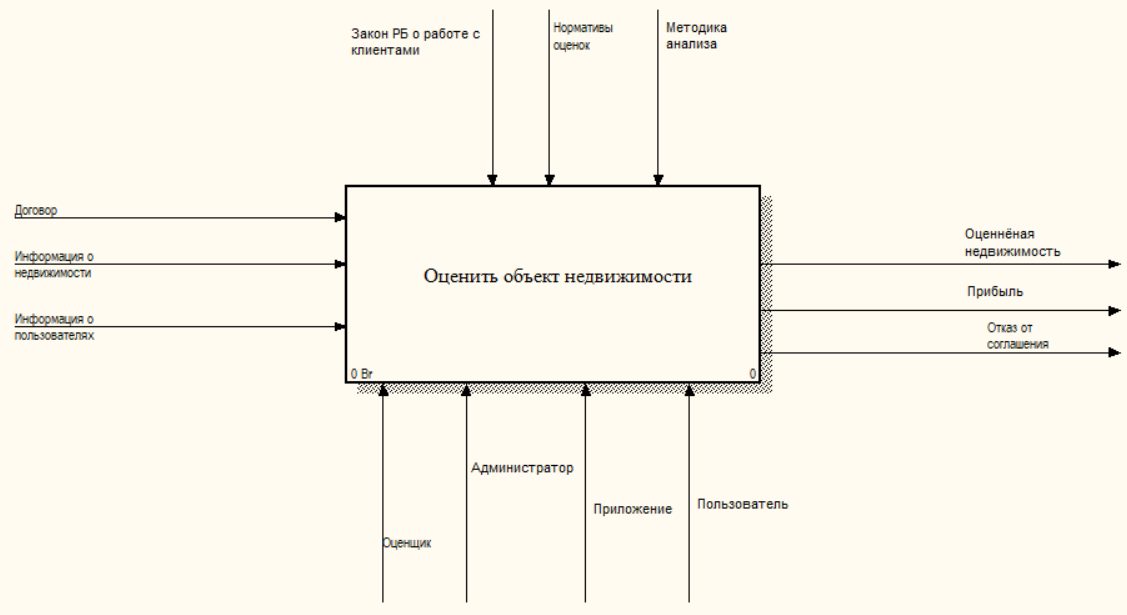


Рисунок 3.1 – Контекстная диаграмма верхнего уровня

Механизмы системы: приложение, оценщик, администратор, пользователь. Это ресурсы, выполняющие работу.

На рисунке 3.2 отображена декомпозиция контекстной диаграммы, состоящая из четырёх блоков: «Проверка данных пользователя», «Заполнить данные об объекте», «Оформление договора», «Оплатить оценку объекта».

Подробное описание процесса «Проверка данных пользователя» представлено на рисунке 3.3 при помощи декомпозиции на 4 подпроцесса: «Проверка личных данных», «Проверка достоверности данных», «Заполнение формы данных», «Добавление в базу данных».

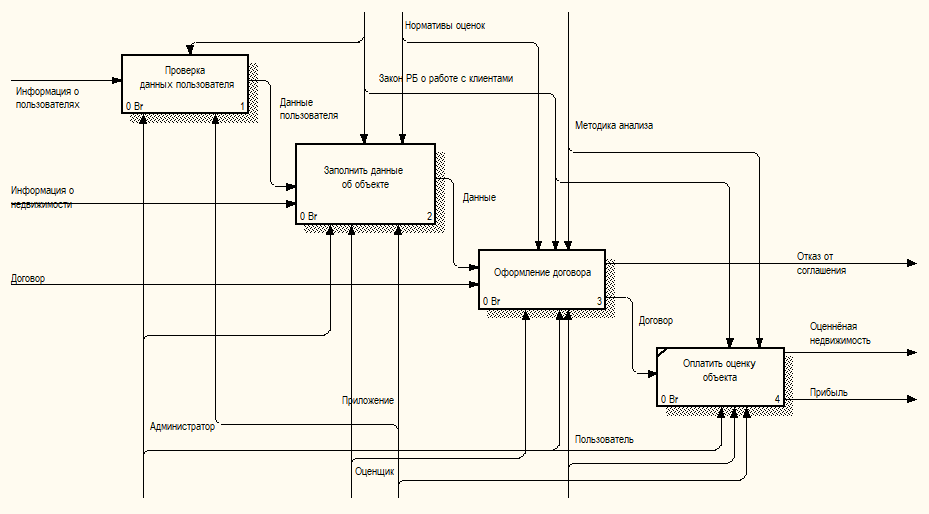


Рисунок 3.2 – Декомпозиция контекстной диаграммы верхнего уровня

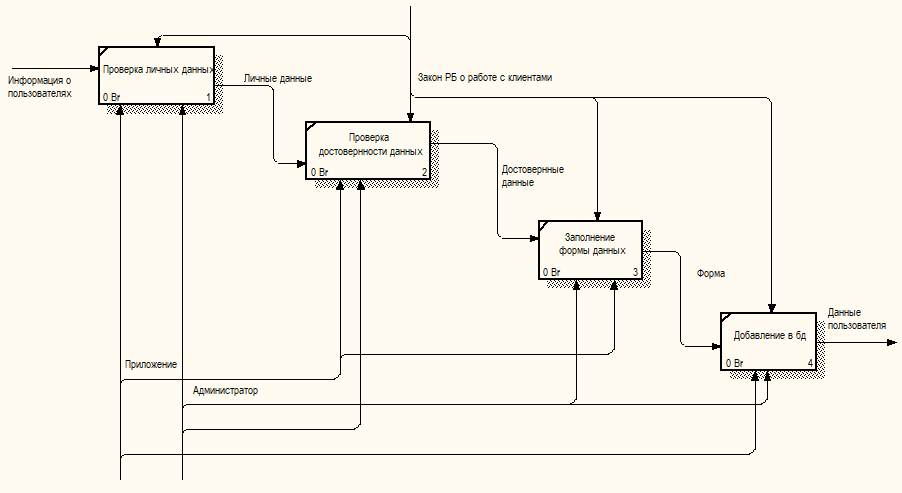


Рисунок 3.3 – Декомпозиция блока «Проверка данных пользователя»

На рисунке 3.4 представлена декомпозиция блока «Заполнить данные об объекте». Она подразделяется на четыре компонента: «Просмотреть список доступных типов», «Выбрать подходящий тип», «Ввести параметры объекта», «Подать договор на рассмотрение».

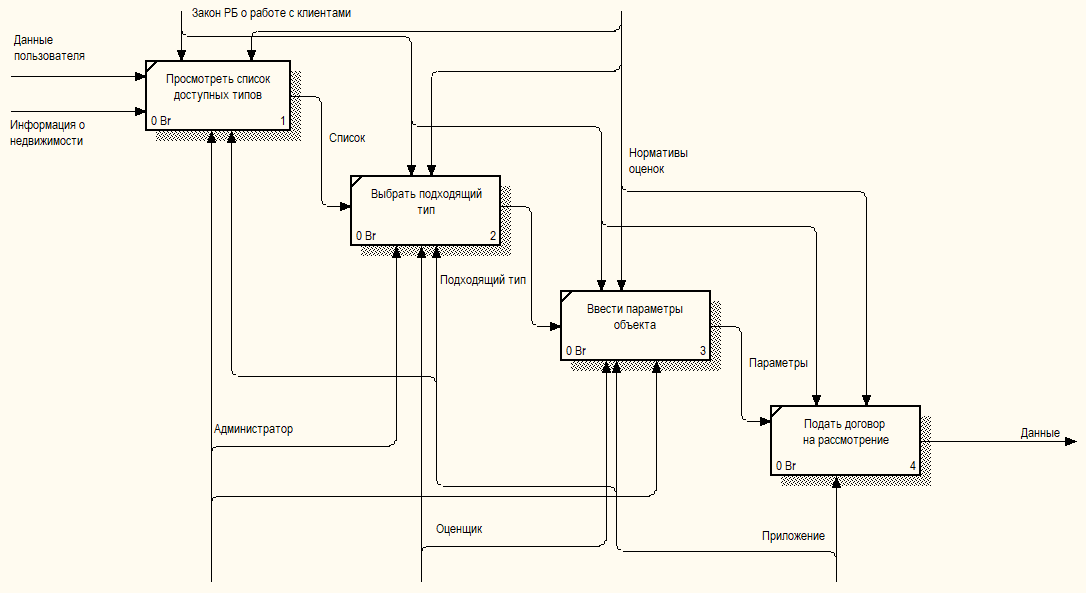


Рисунок 3.4 – Декомпозиция блока «Заполнить данные об объекте»

Декомпозиция блока «Оформление договора» отображена на рисунке 3.5. Она состоит из пяти компонентов: «Подготовка проекта договора с заказчиком», «Юридическое оформление договора», «Согласование договора со сторонами», «Согласование договора у директора», «Заключение договора».

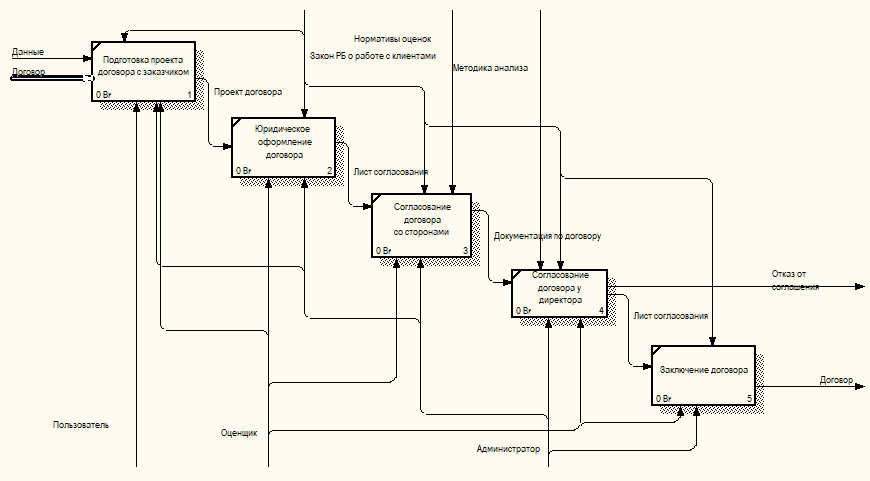


Рисунок 3.5 – Декомпозиция блока «Оформление договора»

1. **Информационная модель системы и ее описание**

Для отображения структуры хранимых данных в данной курсовой работе была использована информационная методология IDEF1.X.

Методология моделирования IDEF1X, являясь расширением стандарта IDEF1, предназначена для описания данных (информации). В ее основе лежит язык семантического моделирования, основанного на концепции "сущность — связь", позволяющей определять данные и связи между ними. Методология используется для создания информационной модели предметной области с помощью идентификации ее сущностей и связей между ними[7].

Основными элементами модели IDEF1X являются сущности, атрибуты и отношения.

В процессе информационного моделирования были выделены следующие сущности:

* Пользователи;
* Оценщики;
* Объекты для оценки;
* Типы объектов;
* Соглашение на оценку.

На рисунке 4.1. отображена модель IDEF1. X.

Сущность «users» используется для авторизации пользователей, оценщиков и администратора.

Сущность «appraisers» хранит данные о зарплатах оценщиков.

Сущность «objectstoappraise» хранит данные об объектах для оценки.

Сущность «objectstypes» хранит данные о типах объектов.

Сущность «appraisalagreement» хранит данные о соглашении на оценку.

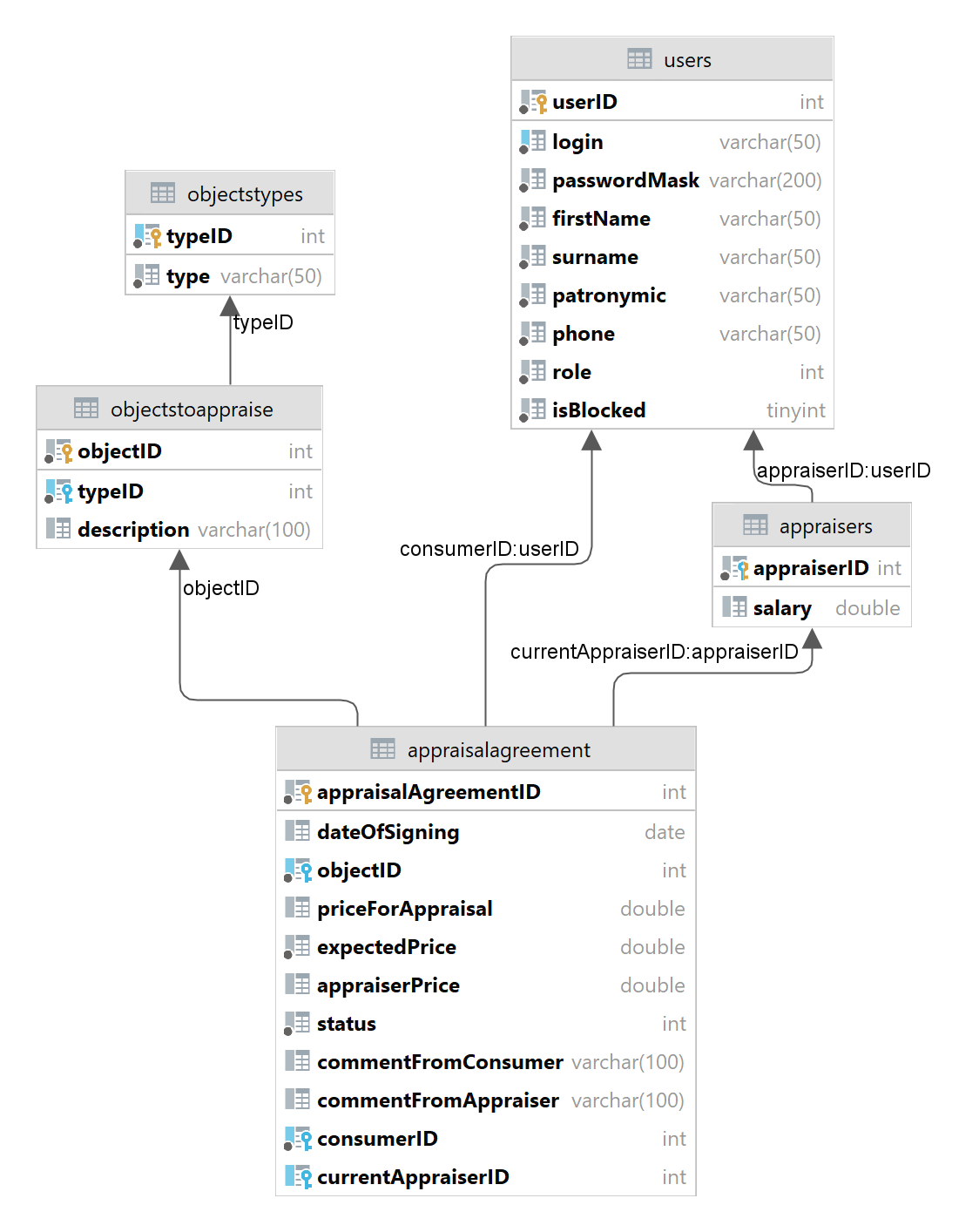


Рисунок 4.1 – Модель IDEF1. X.

1. **Модели представления системы на основе UML**
   1. **Диаграмма вариантов использования**

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (действующим лицом, актантом) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

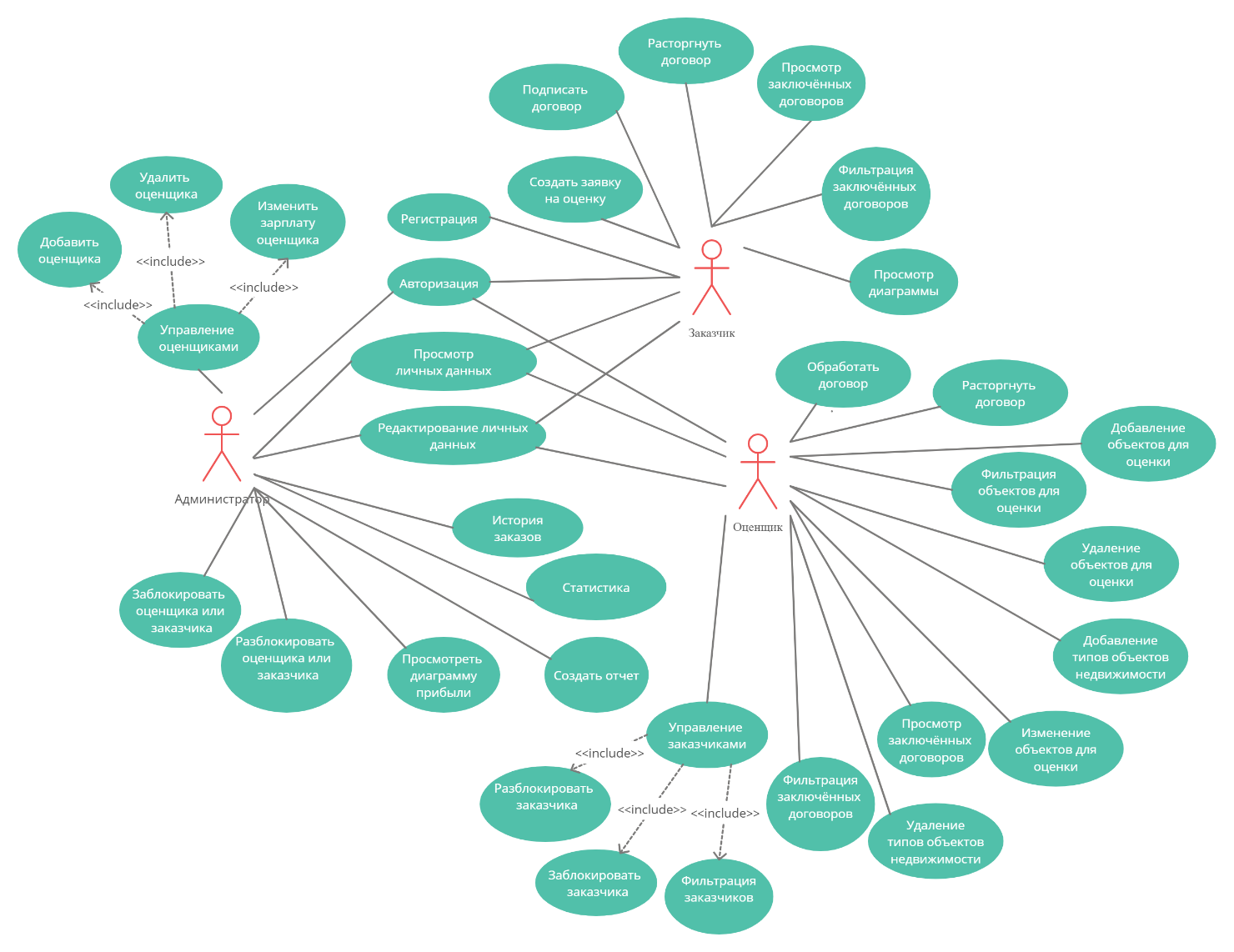


Рисунок 5.1 – Диаграмма вариантов использования программного средства

* 1. **Диаграмма последовательности**

Диаграмма последовательности (Sequence Diagram) – используется для уточнения диаграмм прецедентов – описывает поведенческие аспекты системы. Диаграмма последовательности отражает взаимодействие объектов в динамике, во времени. При этом информация принимает вид сообщений, а взаимодействие объектов подразумевает обмен этими сообщениями в рамках сценария. Данная диаграмма представлена на рисунке 5.2[4].

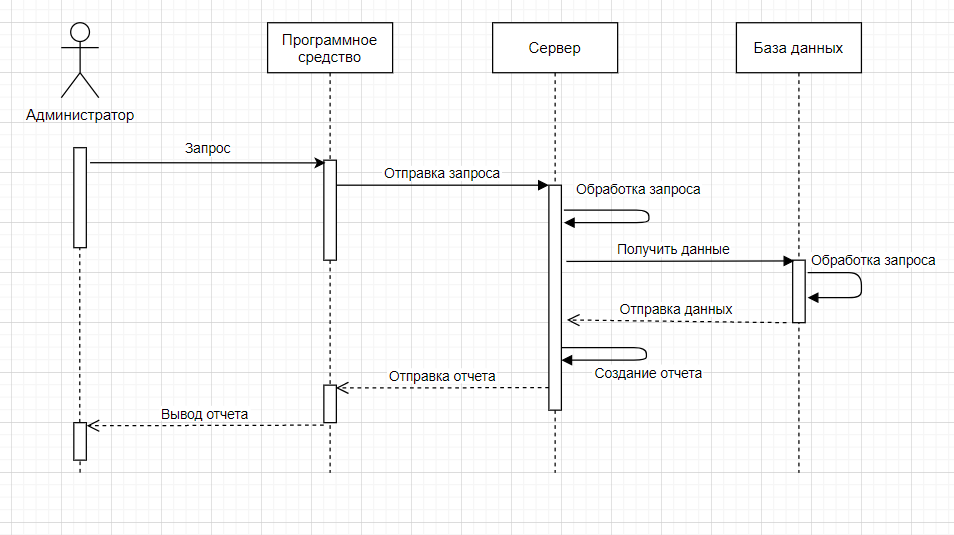


Рисунок 5.2 – Диаграмма последовательности

Процесс начинается с запроса пользователя, после чего запрос отправляется серверу. Сервер обрабатывает запрос и отправляет запрос на получение необходимой информации из базы данных. База данных обрабатывает запрос и отправляет необходимую информацию серверу. Сервер в свою очередь создает ответ и отправляет его клиенту.

* 1. **Диаграмма состояний**

Данная диаграмма необходима для того, чтобы видеть, как объект переходит из одного состояния в другое. Она служит для моделирования динамических аспектов системы, полезна при моделировании жизненного цикла объекта[5].

На рисунке 5.3 представлена диаграмма взаимодействий бизнес-процесса «Подача заявки на заключение договора» .

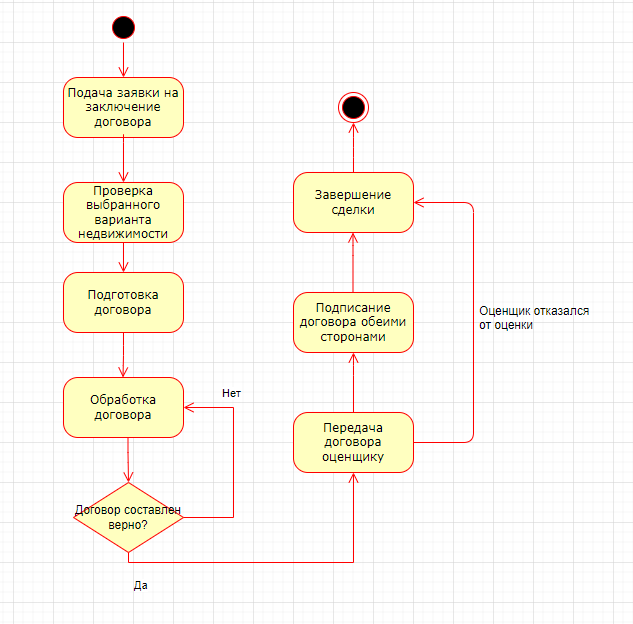


Рисунок 5.3 – Диаграмма состояний

* 1. **Диаграмма классов**

Центральное место в объектно-ориентированном программировании занимает разработка логической модели системы в виде диаграммы классов. Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений[8].

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений. Диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи.

Диаграмма классов клиента представлена на рисунке 5.4. Классы клиентской части отображают действия и возможности, которыми обладают администратор, оценщик и заказчик.

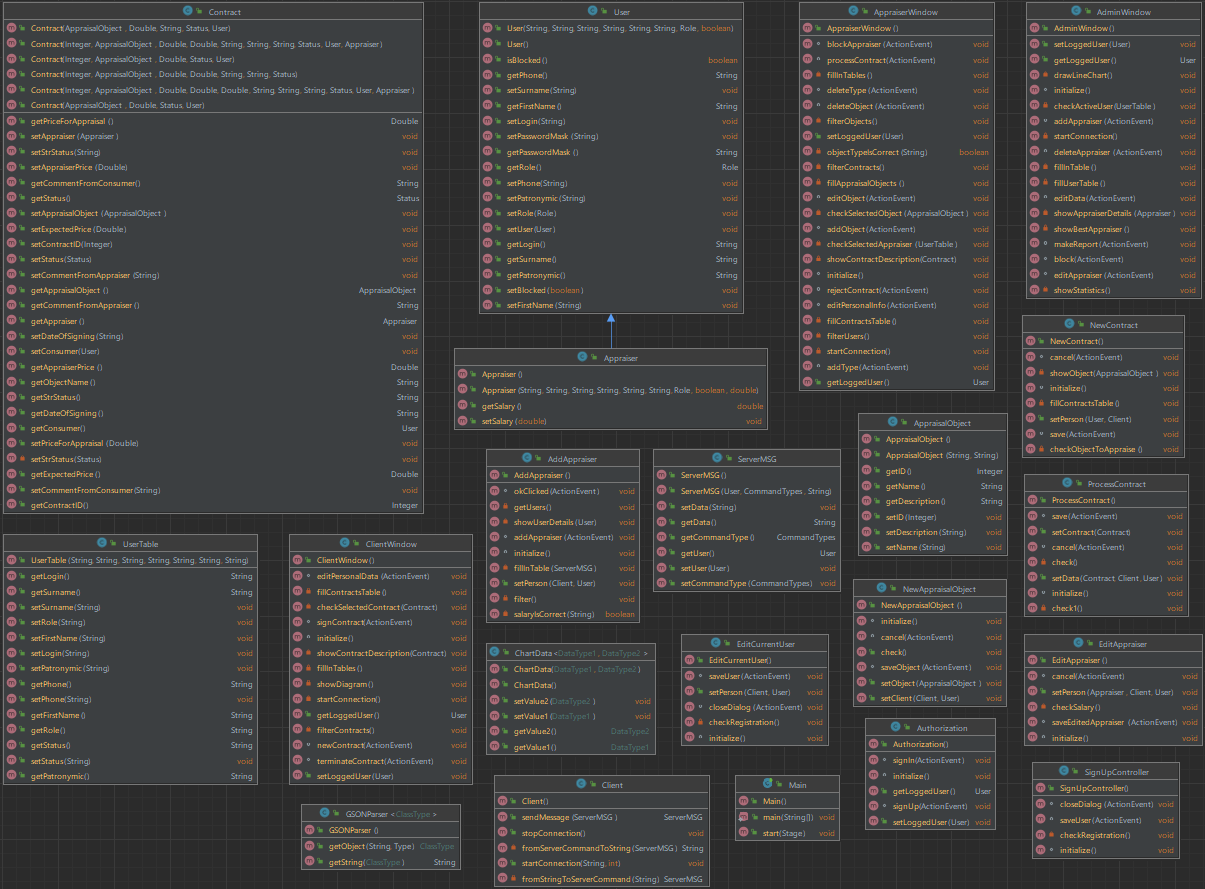


Рисунок 5.4 – Диаграмма классов клиента

Диаграмма классов серверной части представлена на рисунке 5.5. Здесь расположены классы, которые обеспечивают запуск серверного приложения, и обеспечивают подключение сервера к СУБД MySQL.

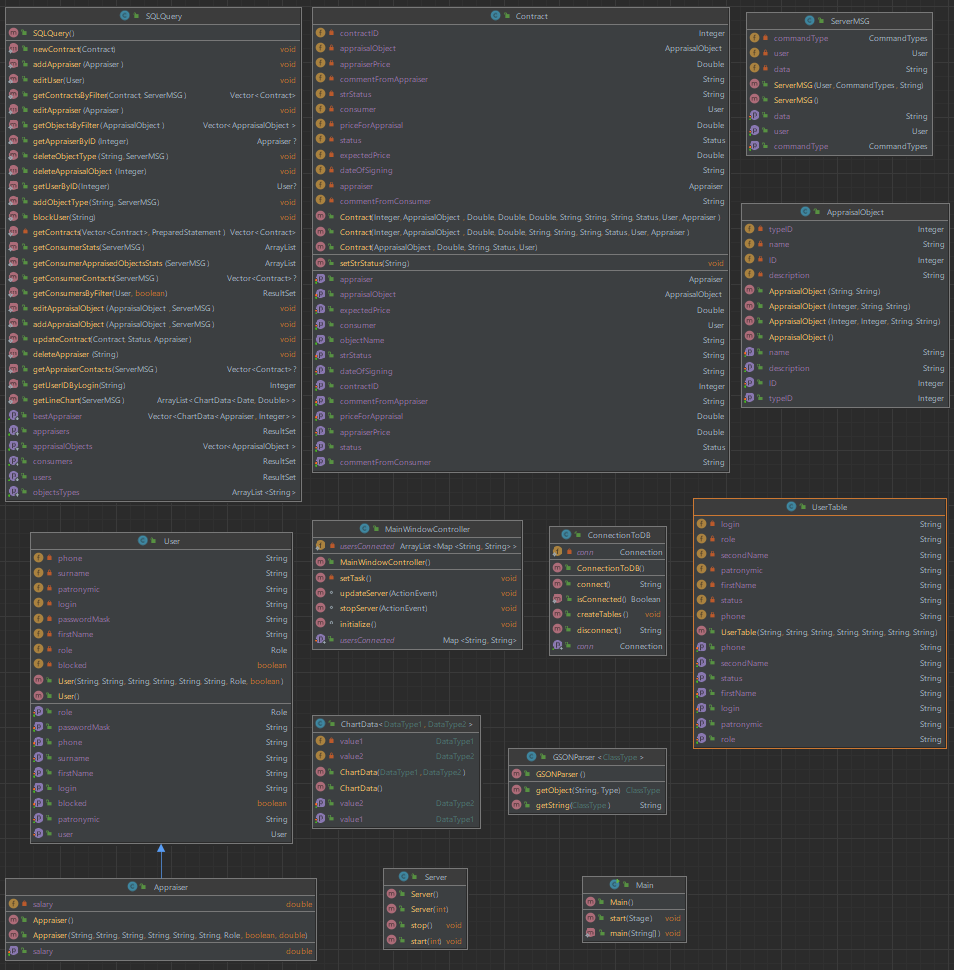


Рисунок 5.5 – Диаграмма классов сервера

* 1. **Диаграмма развертывания**

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Используя его, вы можете понять, как система будет физически развернута на аппаратном обеспечении.

Диаграмма развёртывания показана на рисунке 5.6.

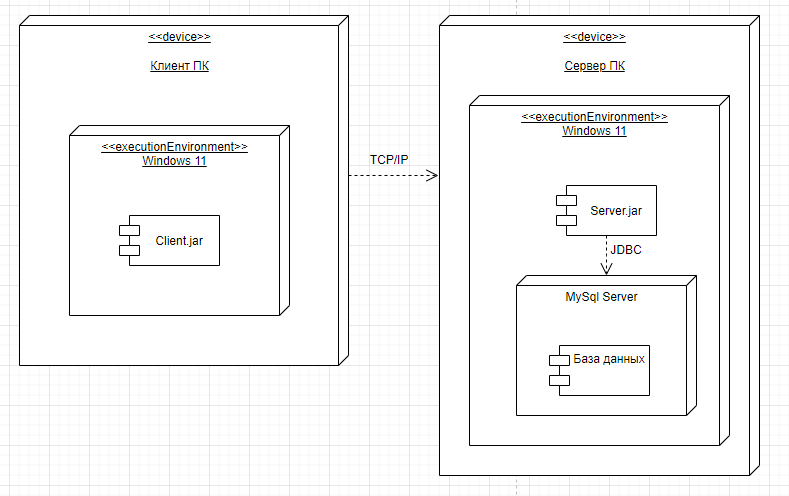


Рисунок 5.6 – Диаграмма развёртывания

* 1. **Диаграмма компонентов**

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код.

Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

На рисунке 5.7 представлена диаграмма компонентов системы.

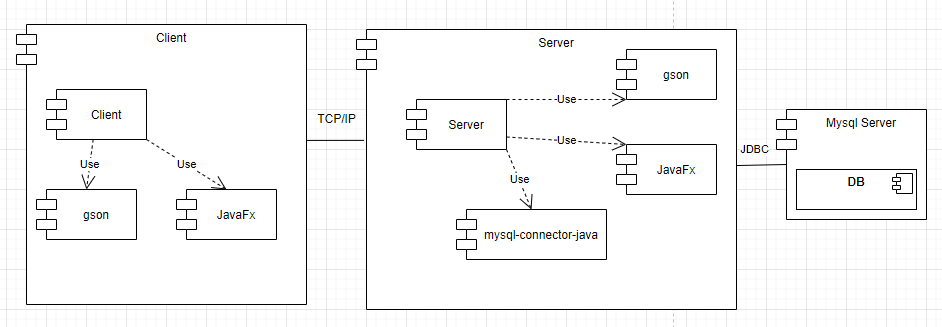


Рисунок 5.7 – Диаграмма компонентов

1. **Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы**
   1. **Алгоритм добавления нового оценщика**

Схема, показанная на рисунке А.1 приложения А, представляет собой алгоритм добавления нового оценщика.

Для добавления нового оценщика администратор выбирает соответствующий пункт в меню, после чего клиентская часть приложения устанавливает соединение с сервером и отправляет ему соответствующий запрос на добавление. Далее осуществляется подключение к базе данных и запрос на добавление данных в соответствующую таблицу. Сервер определяет статус пользователя, если данный пользователь заблокирован, то выведется ошибка, если же он активный, то нам предложит ввести зарплату оценщику и после этого добавит данные в базу. Клиенту передается ответ о конечном результате операции добавления.

* 1. **Схема алгоритма клиент-серверного взаимодействия**

Алгоритм клиент-серверного взаимодействия необходим для полного понимания работы клиента и сервера. Данный алгоритм начинается с отправки запроса на подключение, и если подключение проходит успешно, то клиент может отправлять запросы серверу до тех пор, пока сервер доступен, после отправки запроса необходимо получить ответ и далее обработать его. Далее отправка и получение запроса, а также получение ответа продолжается пока в этом есть необходимость[12].

Схема алгоритма клиент-серверного взаимодействия представлена на рисунке А.2 приложения А.

1. **Руководство пользователя**

Запуск приложения начинается с запуска серверной части «Окно сервера», (рисунок 7.1), в данном окне мы можем увидеть статистику подключений, состояние сервера, подключён ли сервер к базе данных, а так же номер порта используемом при соединении.

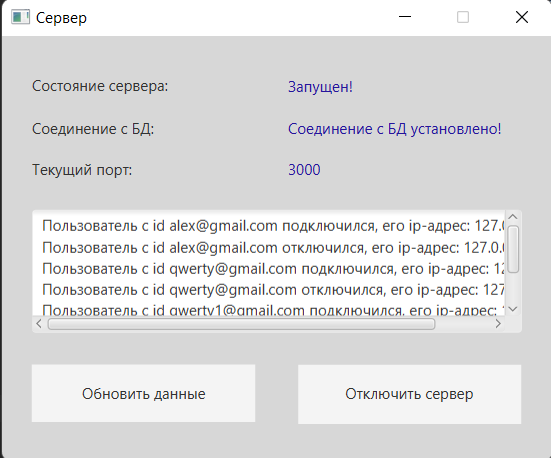


Рисунок 7.1 – Окно сервера

После запуска клиентской части открывается главное меню (рисунок 7.2), где можно авторизоваться или зарегистрироваться, если ещё нет аккаунта. Окно регистрации представлено на рисунке 7.3.

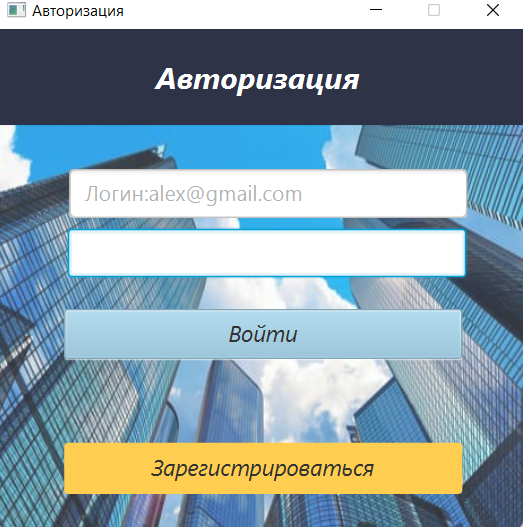


Рисунок 7.2 – Начальное окно

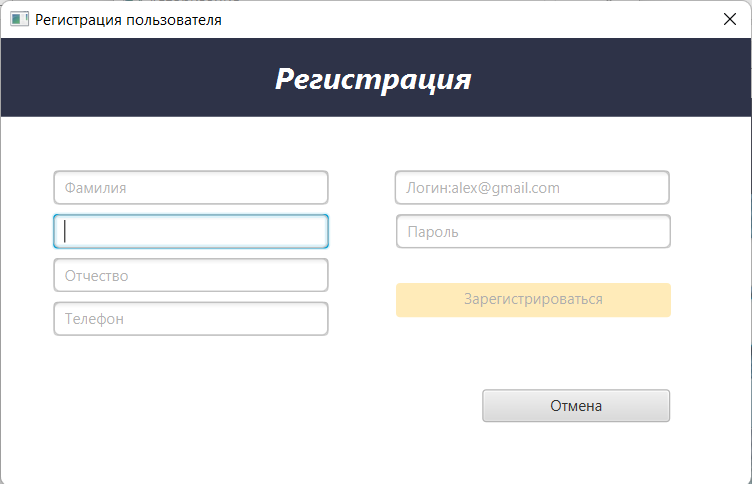


Рисунок 7.3 – Окно регистрации

Если мы заполним хоть одно из полей неверно или оставим поля без заполненных данных, то выскочит соответствующая ошибка (рисунок 7.4) и система не продолжит работу, пока пользователь не введет правильные данные. После того как все данные введены верно, то кнопка зарегистрироваться становится активной (рисунок 7.5).

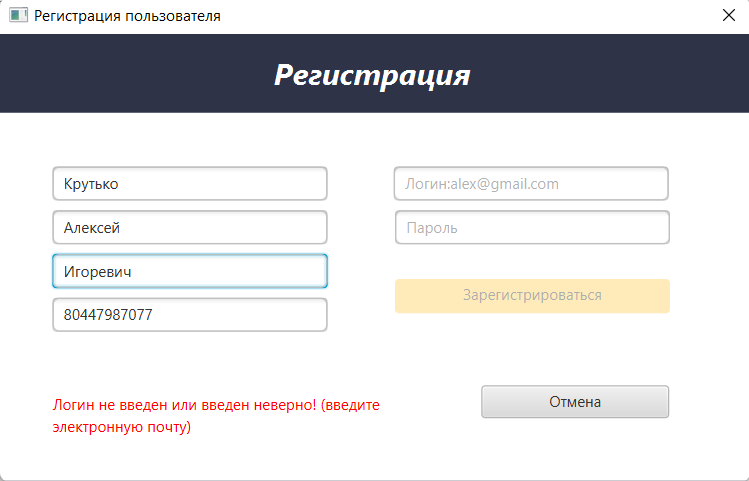


Рисунок 7.4 – Ошибки при регистрации

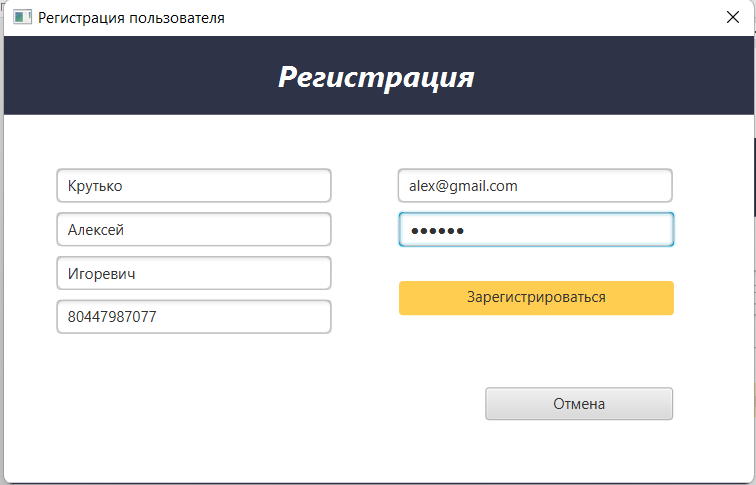


Рисунок 7.5 – Данные введены верно

## **Вход под пользователем**

При входе под пользователем(заказчиком оценки) нам открывается окно, в котором можем подавать договор на рассмотрение оценщику(кнопка новый договор) (рисунок 7.7), если оценщик одобрит договор, внесёт необходимые данные, то пользователь будет решать подписать или расторгнуть договор (рисунок 7.6).

В следующем окне пользователь может просмотреть и отфильтровать уже заключённые с ним договора на оценку объектов недвижимости (рисунок 7.8).

Далее пользователю доступна диаграмма соотношения ожидаемой цены и цены (рисунок 7.9), предложенной оценщиком из которой заказчик может сделать вывод получит ли он больше или меньше денег за продажу чем ожидал.

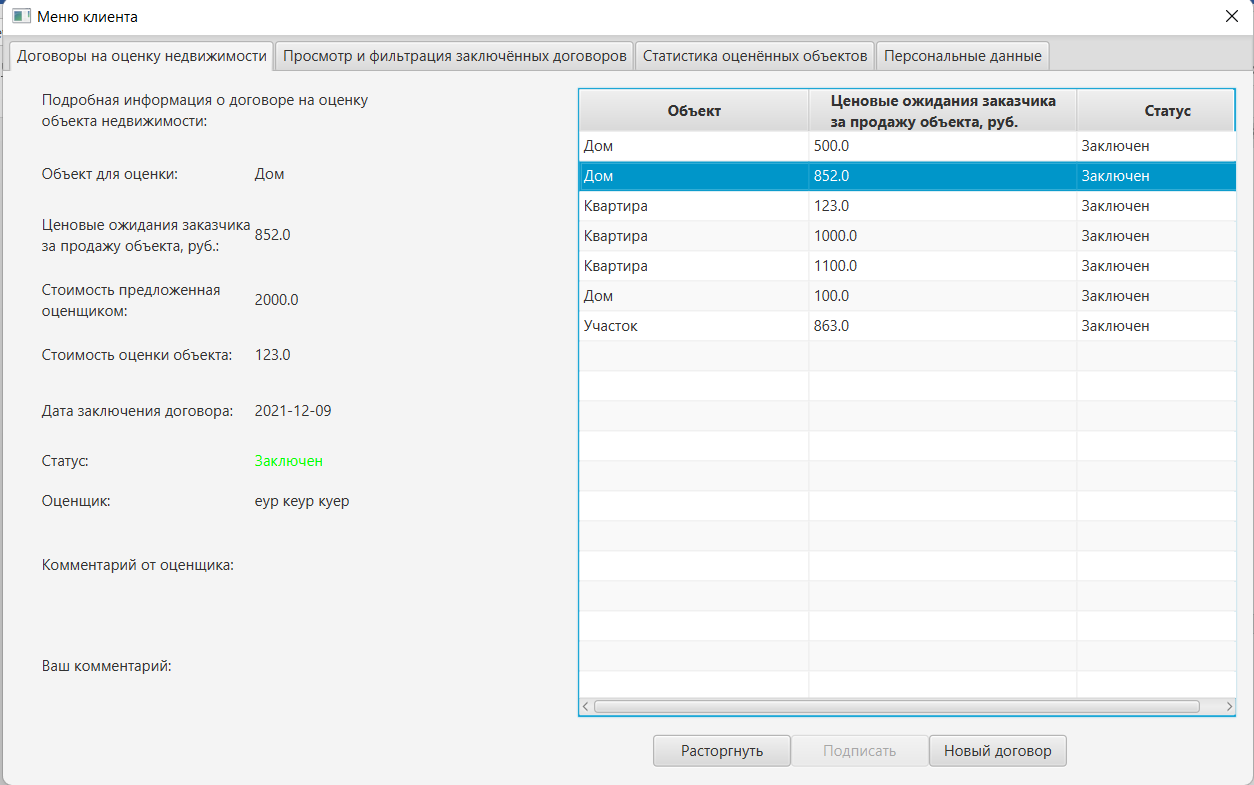


Рисунок 7.6 – Окно пользователя

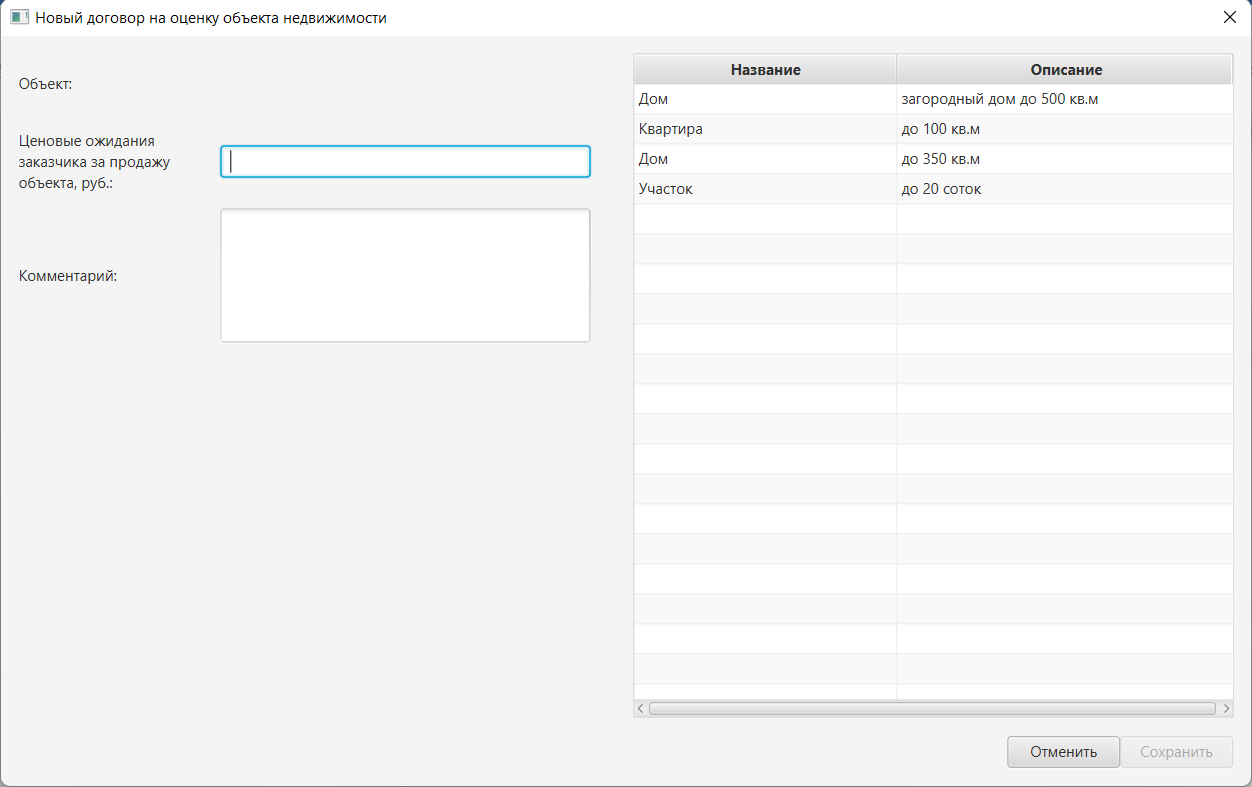


Рисунок 7.7 – Добавление нового договора

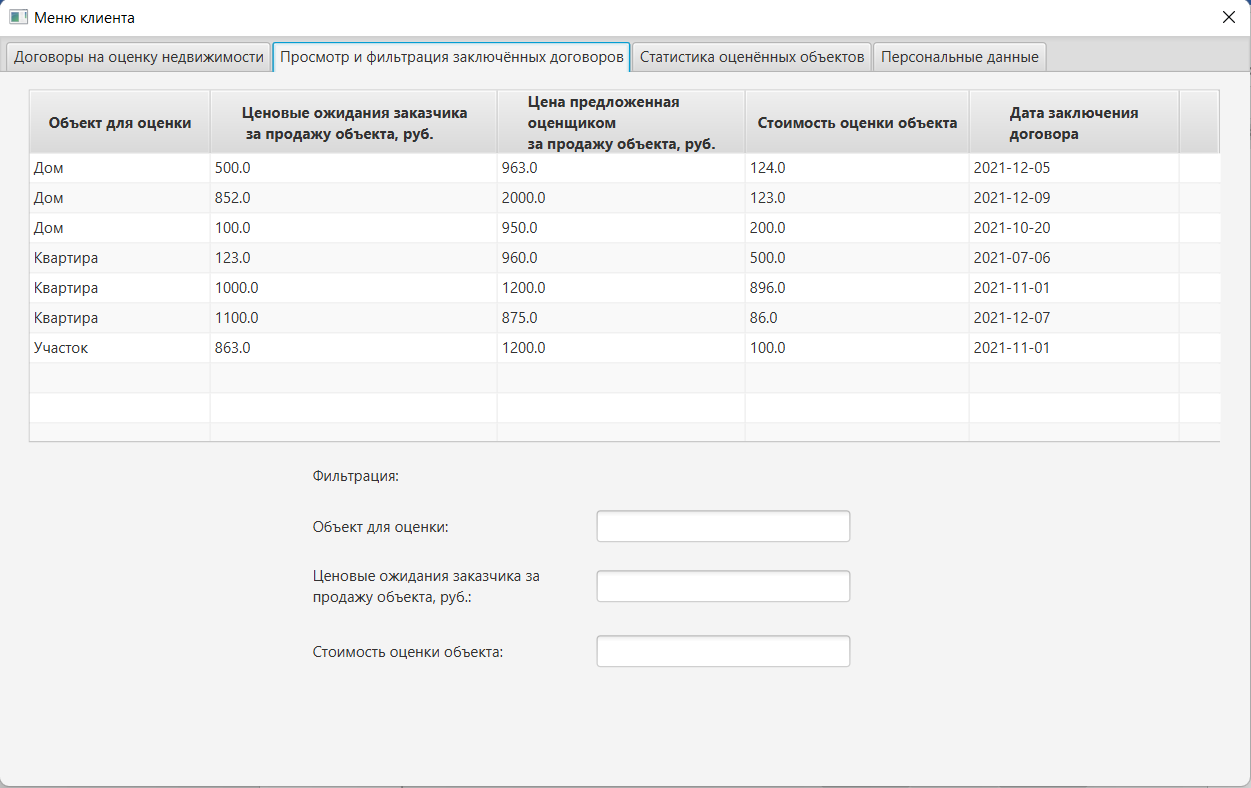


Рисунок 7.8 – Просмотр и фильтрация заключённых договоров

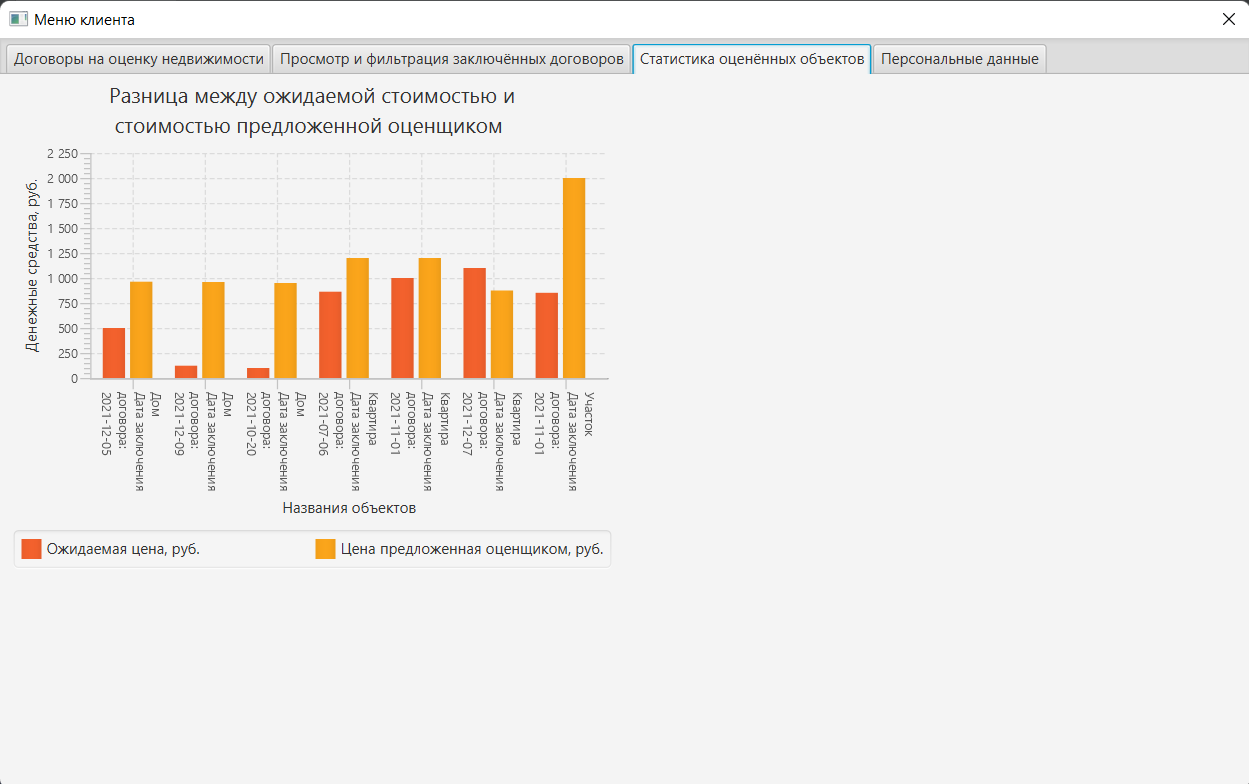


Рисунок 7.9 – Просмотр диаграммы

Также пользователю доступен просмотр личных данных (рисунок 7.10), а также при необходимости их редактирование (рисунок 7.11).

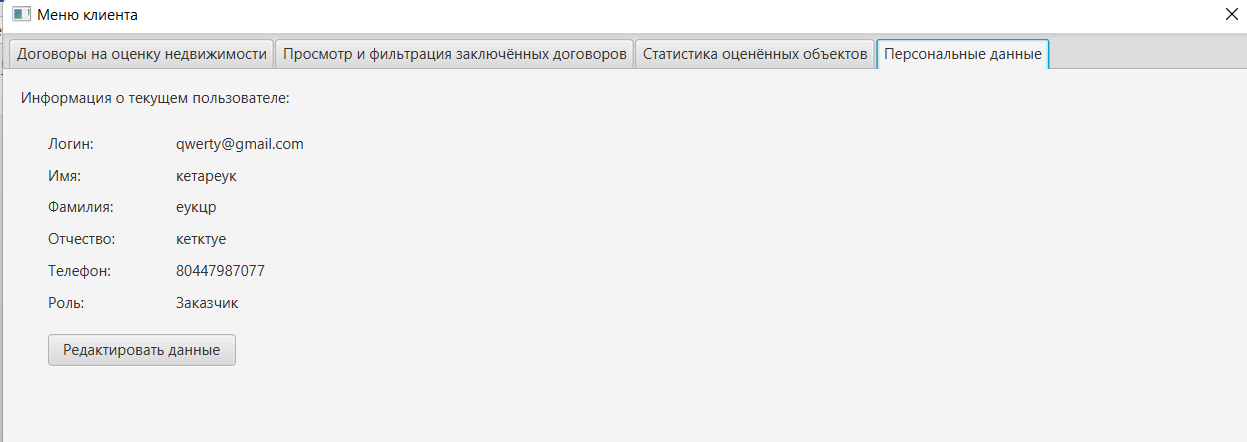


Рисунок 7.10 – Просмотр личных данных

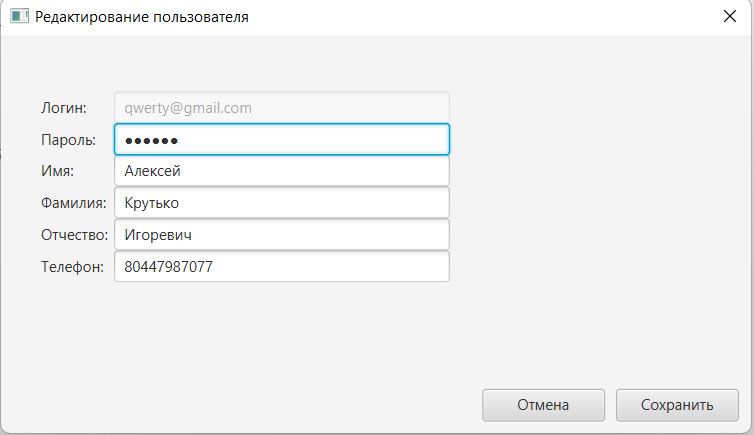


Рисунок 7.11 – Редактирование личных данных

## **Вход под оценщиком**

На первом окне (рисунок 7.12) мы можем видеть, что оценщик может отклонить или обработать договор (рисунок 7.13), а также просмотреть подробную информацию по каждому договору.

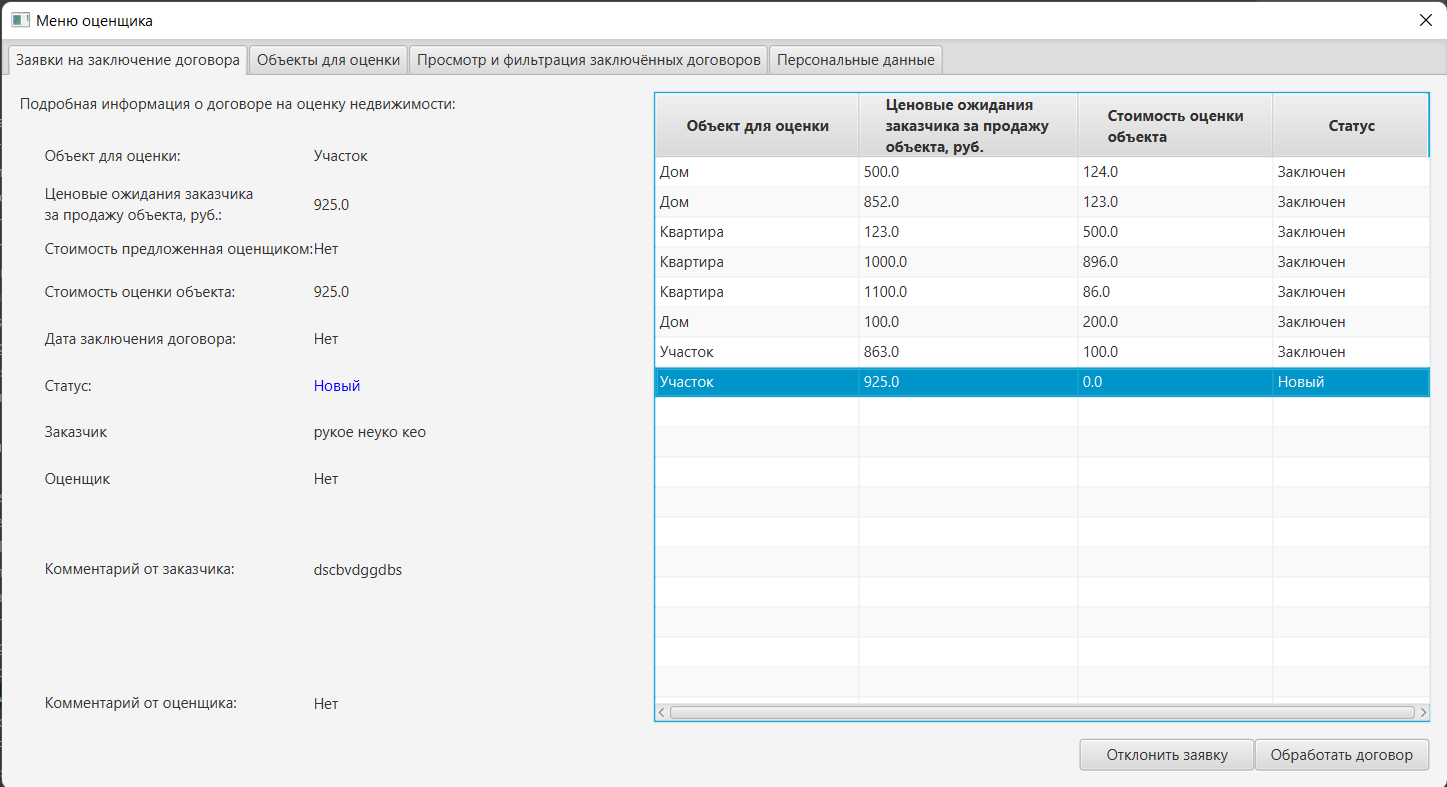


Рисунок 7.12 – Работа с договорами

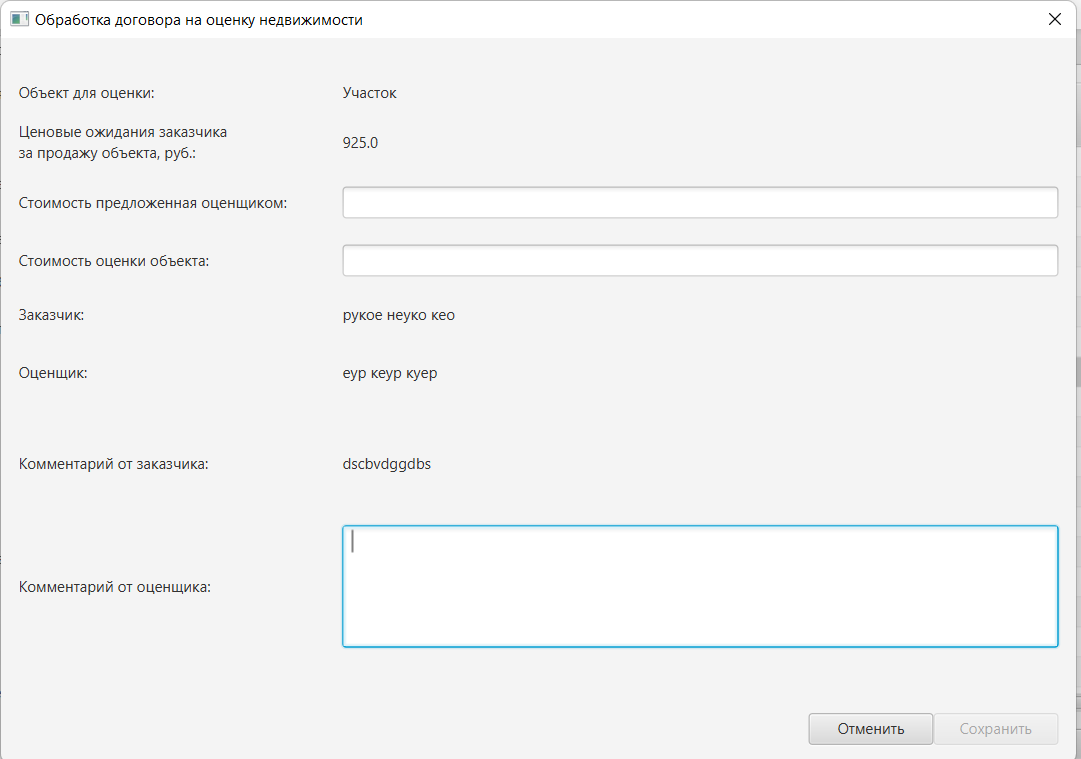


Рисунок 7.13 – Обработка договора

Во втором окне (рисунок 7.14) мы можем видеть, что оценщик может добавлять (рисунок 7.15) и удалять типы объектов для оценки (рисунок 7.16), а также работать с доступными объектами для оценки, то есть добавлять(рисунок 7.17), удалять, изменять(рисунок 7.18).

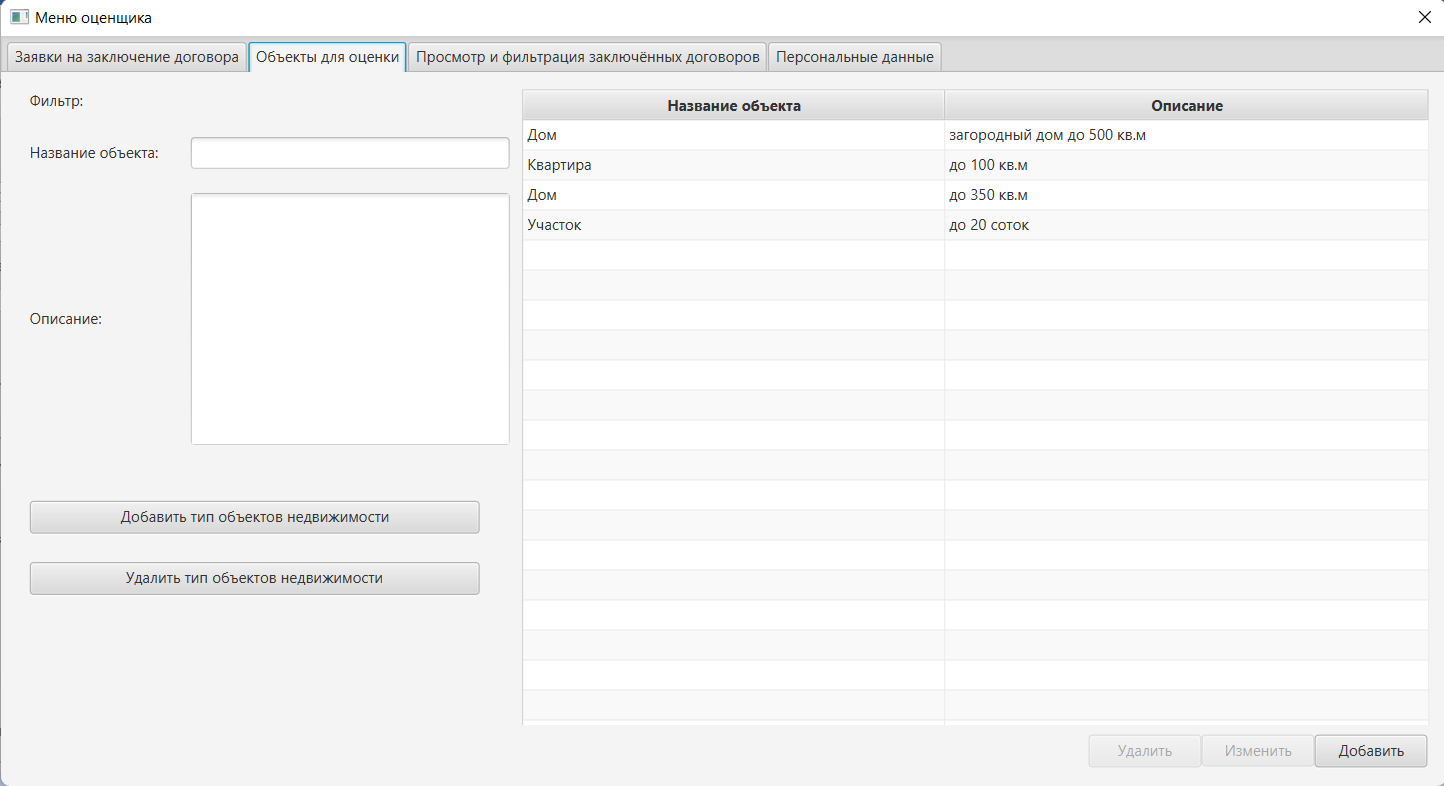


Рисунок 7.14 – Второе окно оценщика

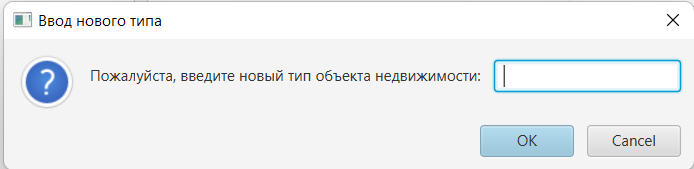


Рисунок 7.15 – Добавление типа

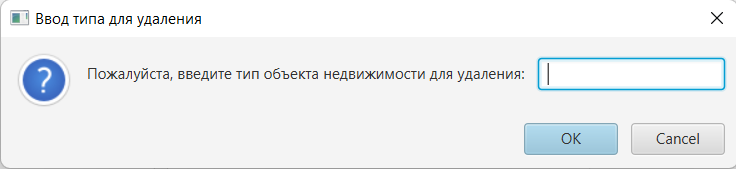


Рисунок 7.16 – Удаление типа

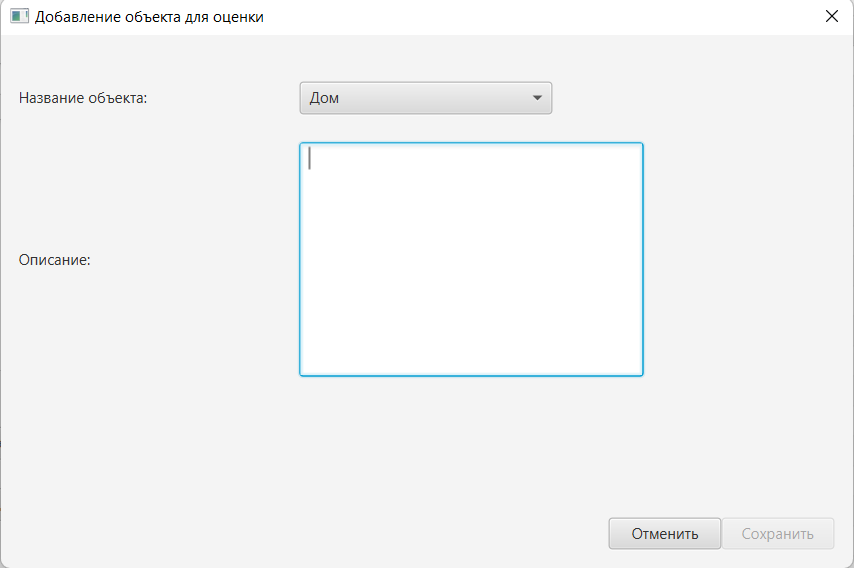


Рисунок 7.17 – Добавление объекта для оценки

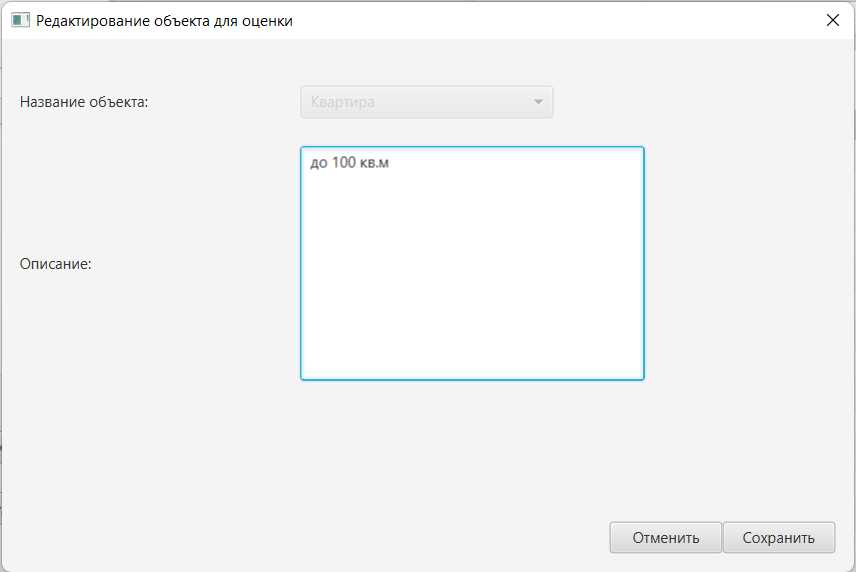


Рисунок 7.18 – Редактирование объекта для оценки

Далее мы можем просмотреть и отфильтровать заключенные этим оценщиком договора (рисунок 7.19). В следующем окне (рисунок 7.20) оценщик может блокировать/разблокировать пользователей(заказчиков), редактировать личные данные (рисунок 7.21), а также фильтровать заказчиков.

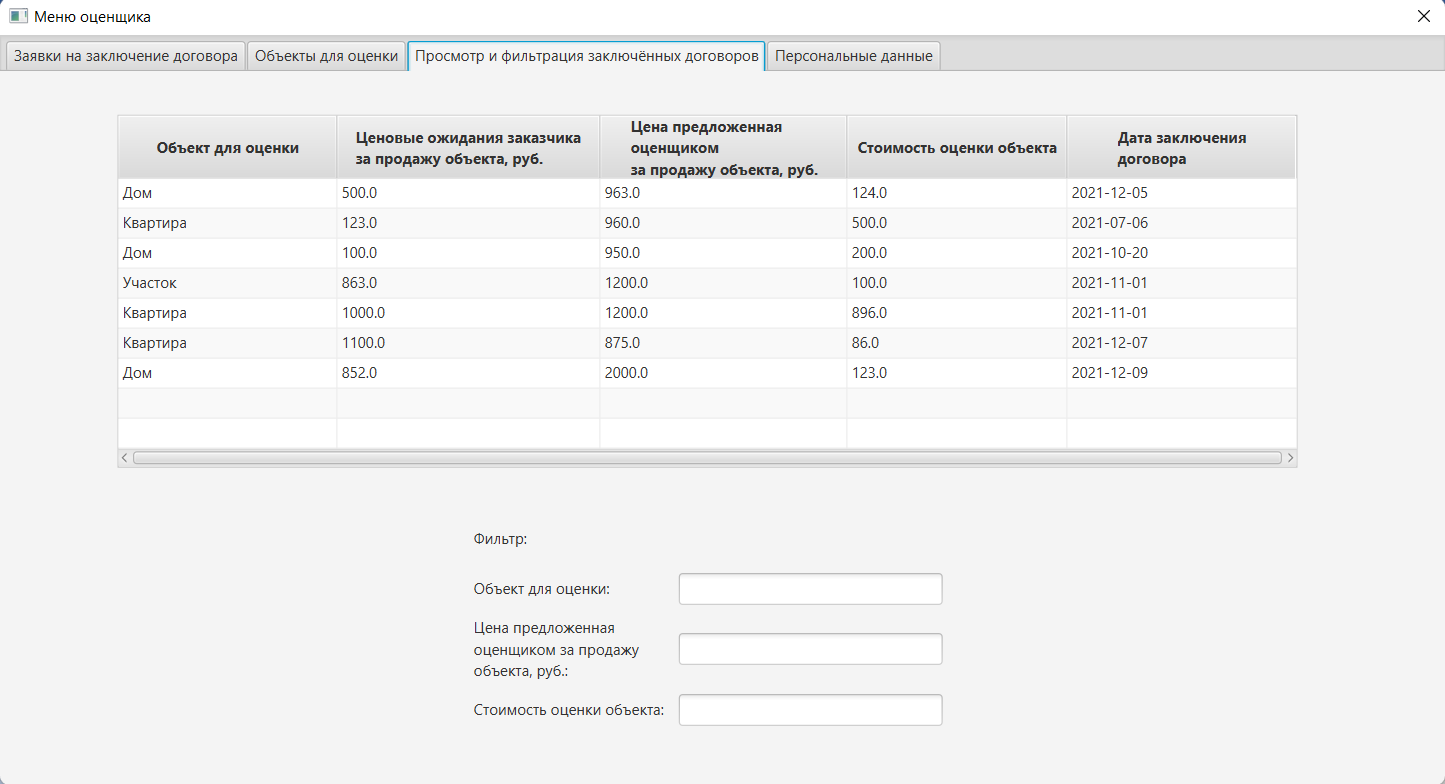


Рисунок 7.19 – Просмотр и фильтрация заключённых договоров

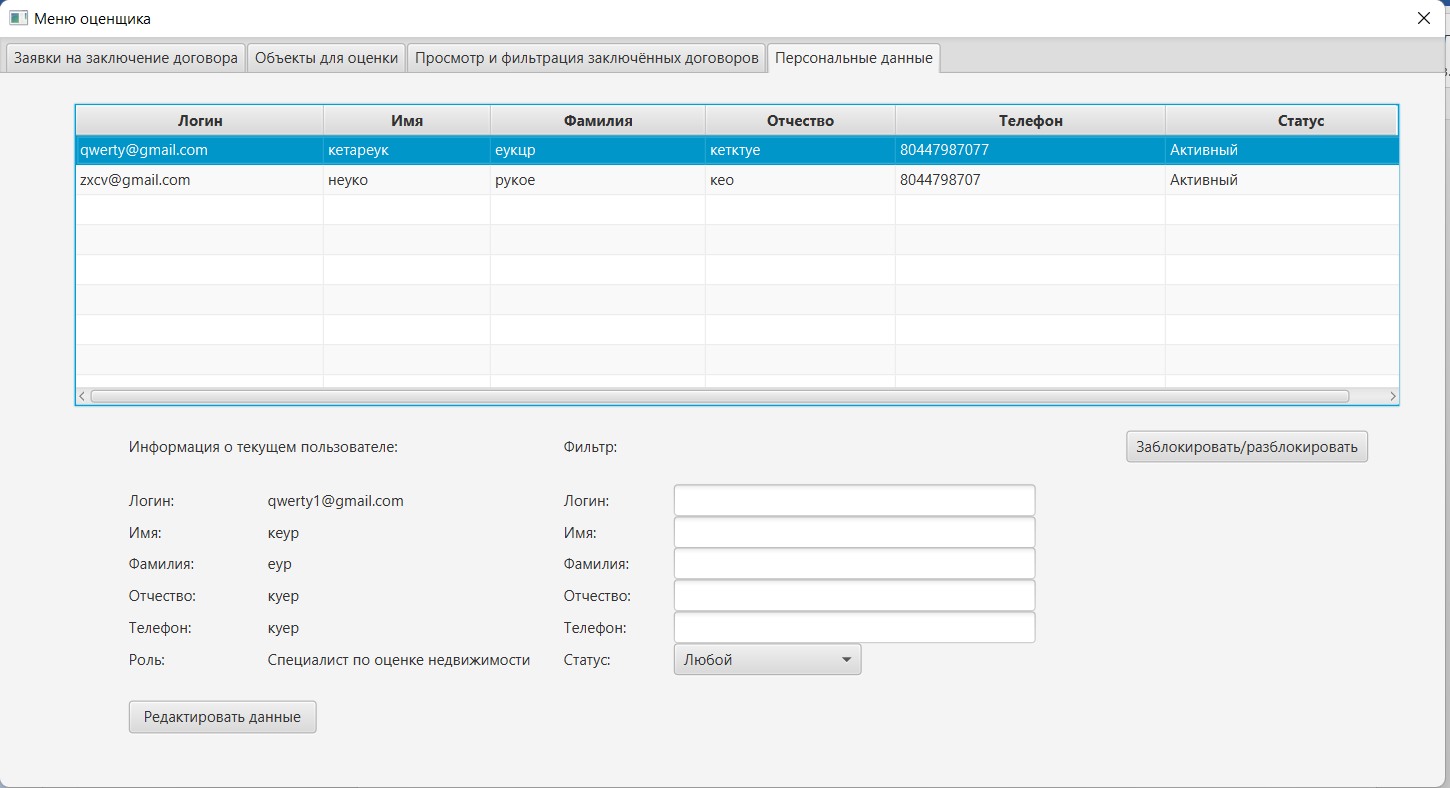


Рисунок 7.19 – Четвёртое окно оценщика

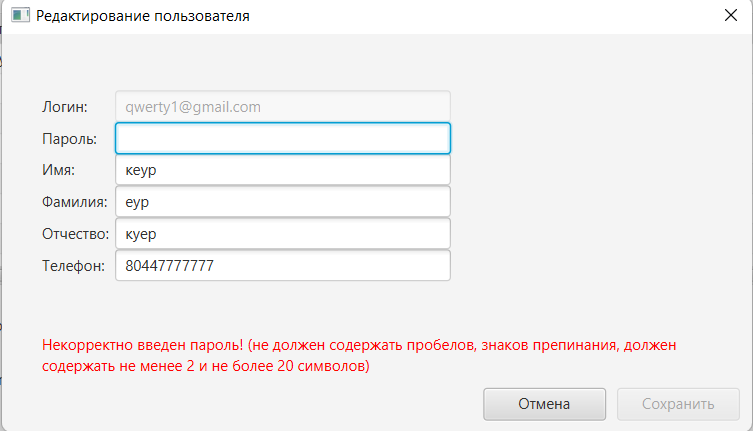


Рисунок 7.20 – Редактирование данных оценщика

## **Вход под администратором**

Первое окно администратора представлено на рисунке 7.21, в нём администратор может управлять оценщиками, то есть добавлять (рисунок 7.22), удалять, изменять (рисунок 7.24). После того как будет выбран пользователь, которому нужно выдать права оценщика появится окно, в котором надо будет ввести зарплату (рисунок 7.23).

Во втором окне администратор может просмотреть статистику прибыли за последнее полугодие (рисунок 7.25), а также вывести отчет в pdf-файл(рисунок 7.26).

В третьем окне (рисунок 7.27) можно редактировать личные данные (рисунок 7.28), блокировать/разблокировать заказчиков и оценщиков.

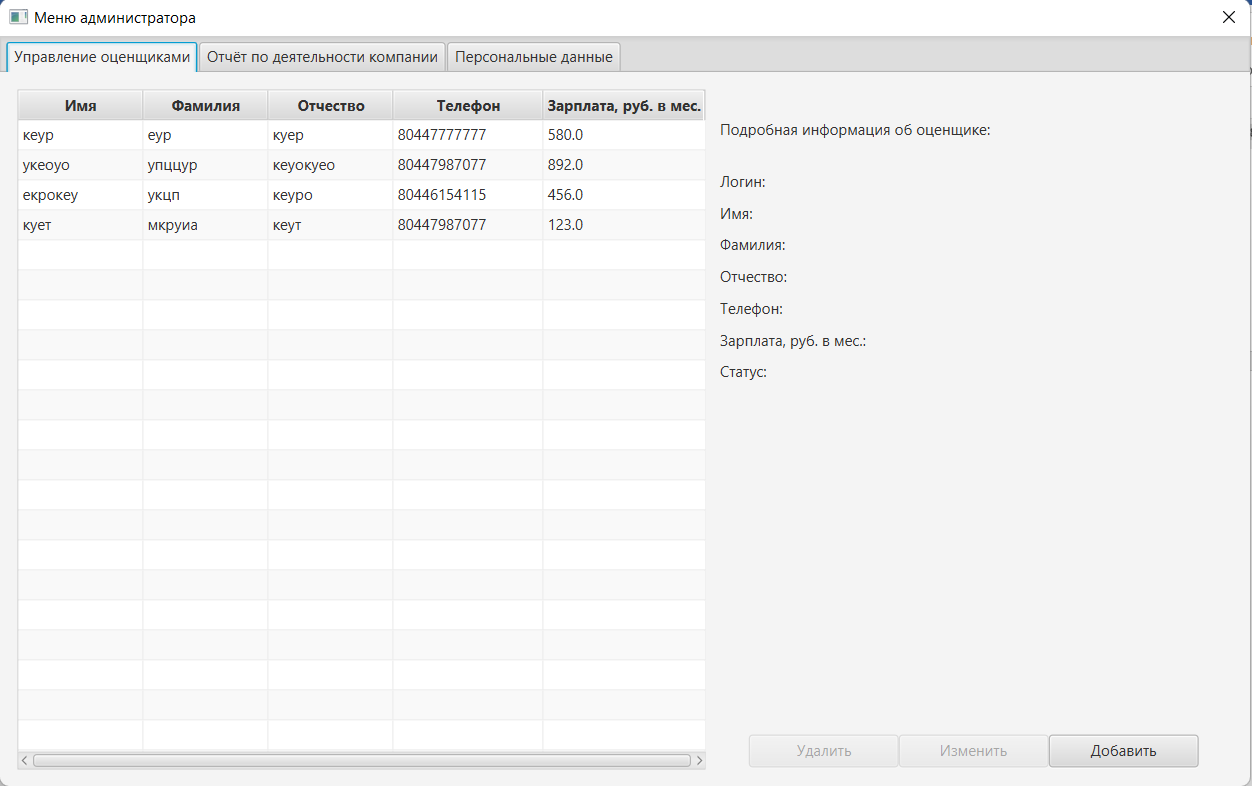


Рисунок 7.21 – Управление оценщиками

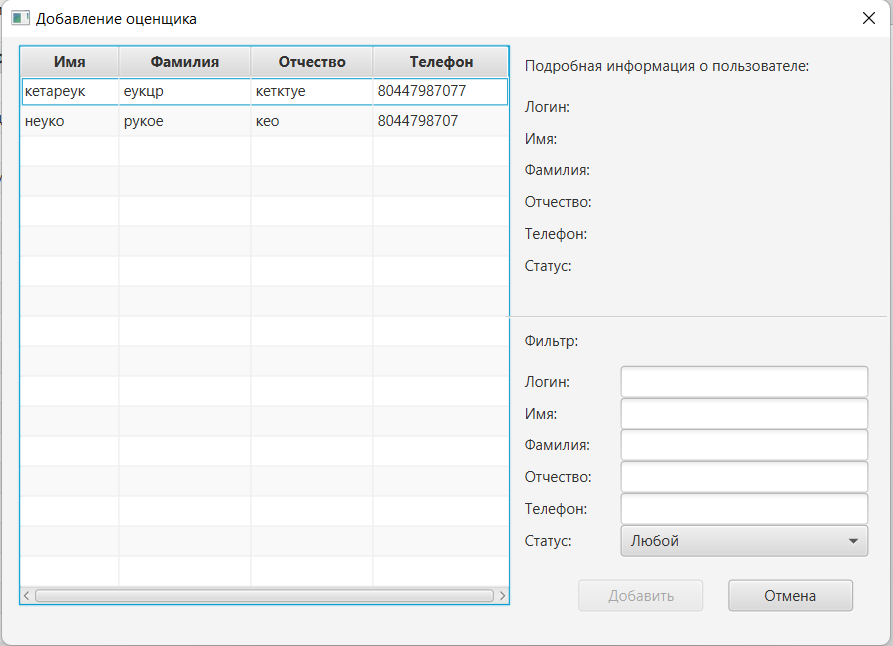


Рисунок 7.22 – Добавление оценщика

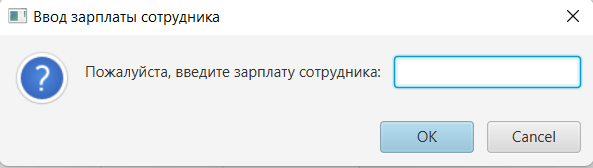


Рисунок 7.23 – Ввод зарплаты оценщика

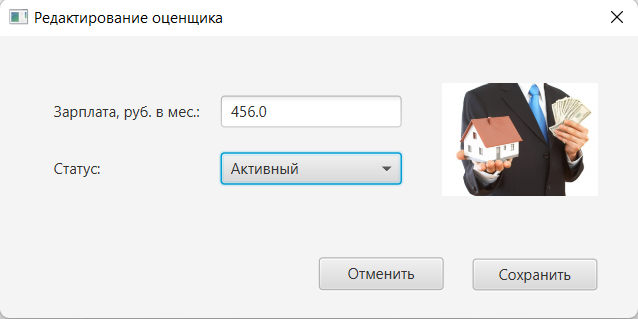


Рисунок 7.24 – Редактирование данных оценщика

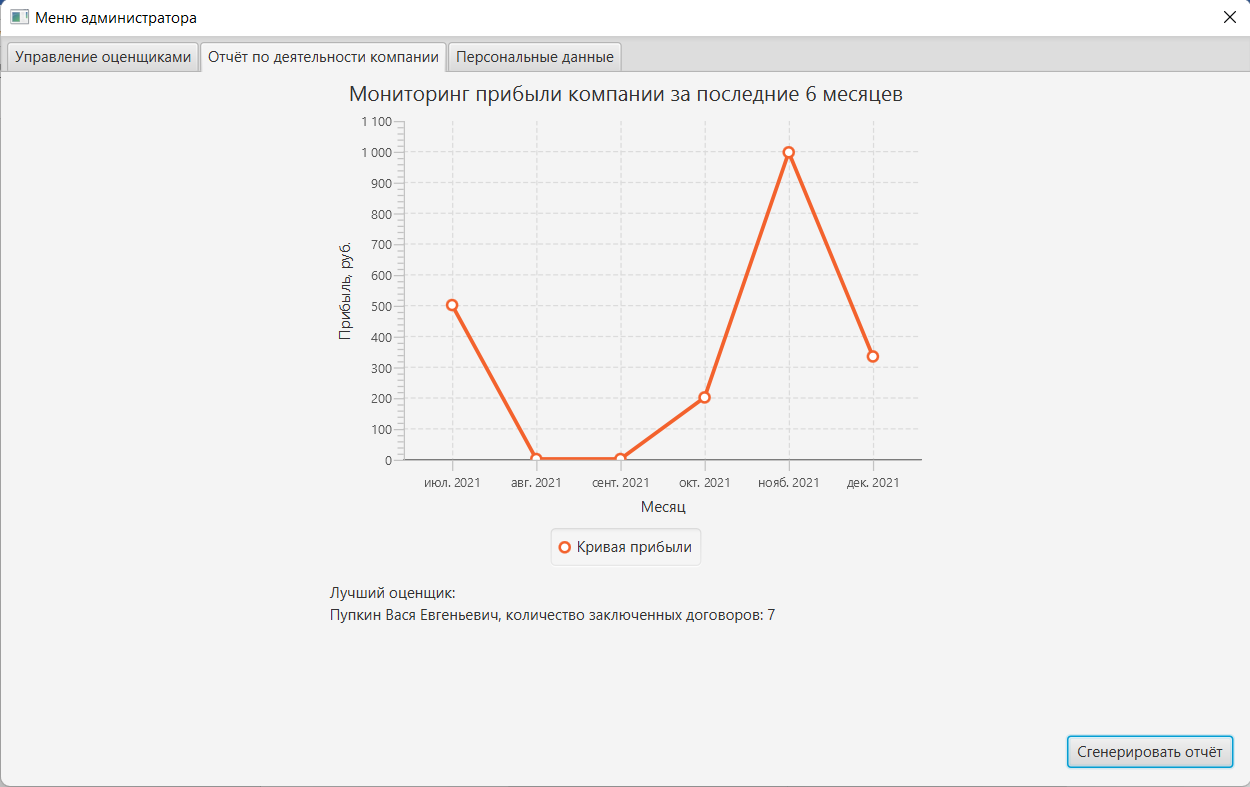


Рисунок 7.25 – Диаграмма прибыли компании за полугодие

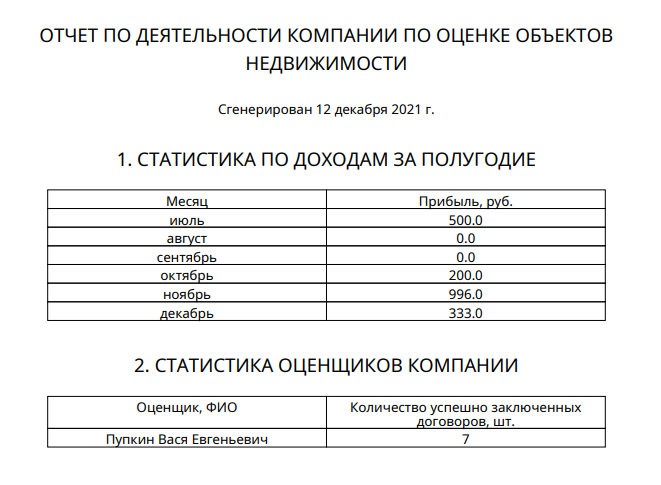


Рисунок 7.26 – Отчет в pdf

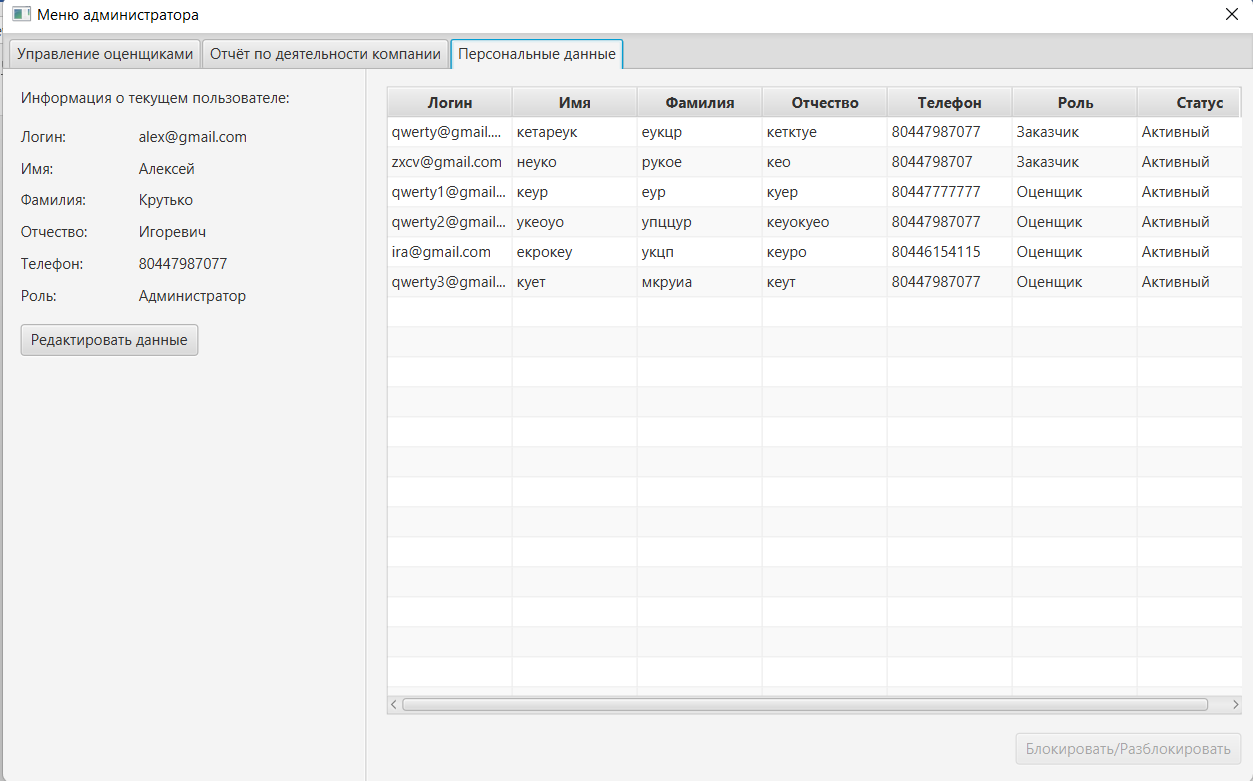


Рисунок 7.27 – Третье окно администратора

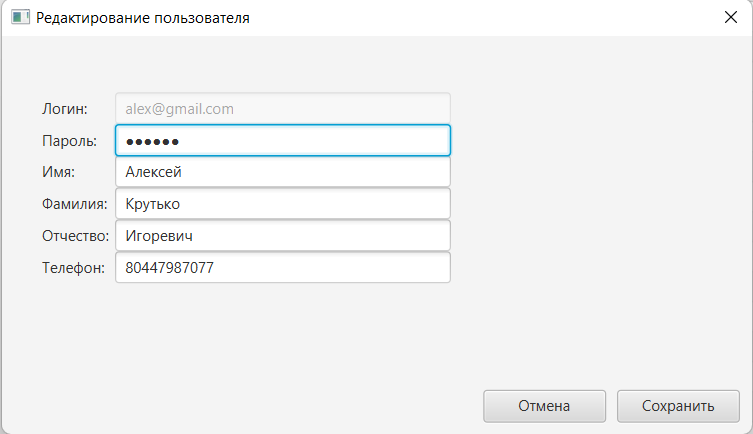


Рисунок 7.28 – Редактирование личных данных

1. **Результаты тестирования разработанной системы**

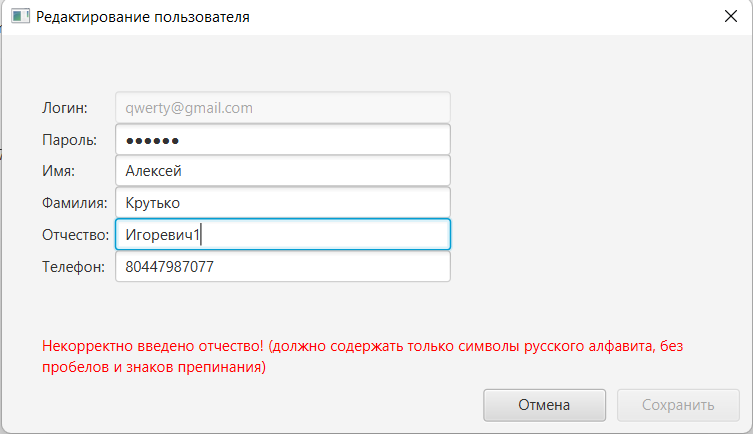


Рисунок 8.1 – Ошибка при редактировании личных данных

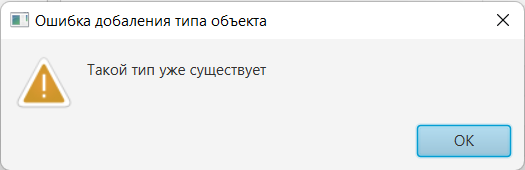


Рисунок 8.2 – Ошибка при добавлении типа объекта

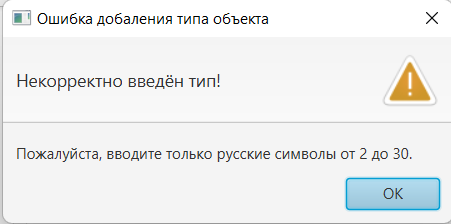


Рисунок 8.3 – Ошибка при добавлении типа объекта

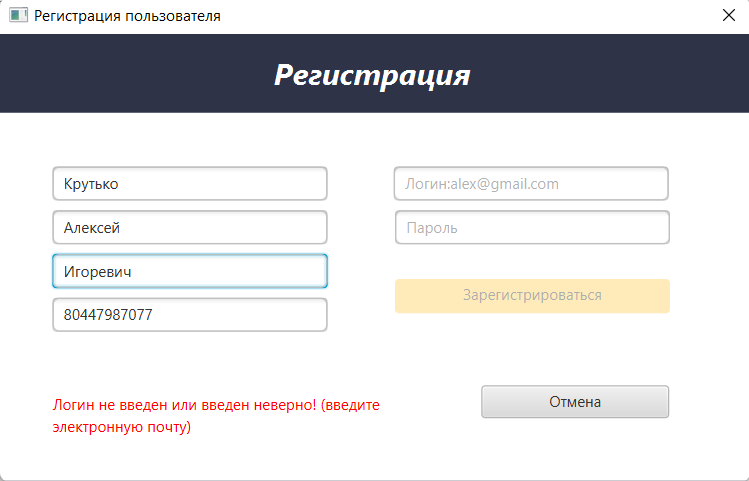


Рисунок 8.4 – Ошибка при регистрации

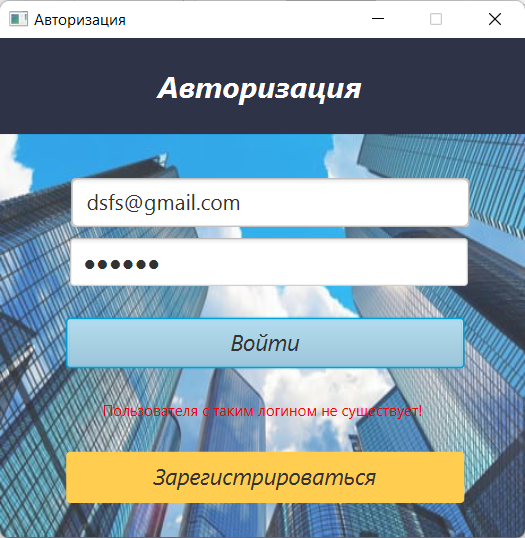


Рисунок 8.5 – Ошибка при авторизации

# Заключение

В процессе разработки данного курсового проекта было проведено исследование предметной области, разработаны основные алгоритмы и функции, были достигнуты поставленные цели и задачи.

Помимо этого, был разработан программный продукт для оценки объектов недвижимости, основанный на клиент-серверной архитектуре, представляющий собой приложение с удобным и понятным пользовательским интерфейсом, что позволяет быстро начать работу.

Для того, чтобы пользователь мог быстро освоиться в программном продукте, было разработано краткое руководство пользователя, которое позволяет в кратчайшие сроки разобраться с функционалом и особенностями разработанного программного продукта. Помимо этого, было создано руководство по развертыванию, которое позволяет быстро и без ошибок разобраться с установкой программного продукта.

В результате разработанная программа оснащена простым и удобным для пользователей интерфейсом, несложной системой навигации, необходимыми функциями для выполнения основной задачи курсового проекта, она является пригодной для использования юридическими лицами, которые нуждаются в оценке объектов недвижимости, например риэлторским агентством.

# Список использованных источников

1. IT-black. Технология клиент-сервер [Электронный ресурс]. – Электронные данные. – Режим доступа: https://it-black.ru/tekhnologiya-kliyent-server/.
2. MySQL [Электронный ресурс]. – Электронные данные. – Режим доступа:  <https://www.tadviser.ru/index.php/Продукт:MySQL>.
3. StudFiles [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://studfile.net/preview/1444532/>.
4. lucidchart [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.lucidchart.com/pages/ru/erd-диаграмма>.
5. Web-creator.ru [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://web-creator.ru/articles/about_client_side>.
6. Васильев А.Н. Java. – СПб «Питер», 2011. – 400 с.
7. Живицкая, Е. Н. Системный анализ и проектирование информационных систем: учебно-методическое пособие / Е. Н. Живицкая, Т. М. Унучек, Н. П. Мытник. – Мн.: БГУИР, 2016
8. UML. Классика CS. 2-у изд./Пер. с англ.; Под общей редакцией проф. С.Орлова - СПб.: Питер, 2006. - 736 с.: ил.
9. Шилдт, Г. Java 8: руководство для начинающих / Г. Шилдт; пер. с англ. – 6-е изд. – М.: Вильямс, 2015.
10. Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org>.
11. Руководство по языку программирования Java [Электронный ресурс]. Режим доступа: <https://metanit.com/java/tutorial/>.
12. Клиент-сервер шаг-за-шагом [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/330676/>.
13. Базы данных [Электронный ресурс]. Режим доступа: <https://ru.hostings.info/schools/bazy-dannyh.html>.
14. Черемных С.В., Семенов И.О. Моделирование и анализ систем. IDEF – технологии: практикум. М.: Финансы и статистика, 2006. – 188 с.
15. Учебные материалы [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://works.doklad.ru/view/SNMY_zdnmOo.html>.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Схемы алгоритмов**

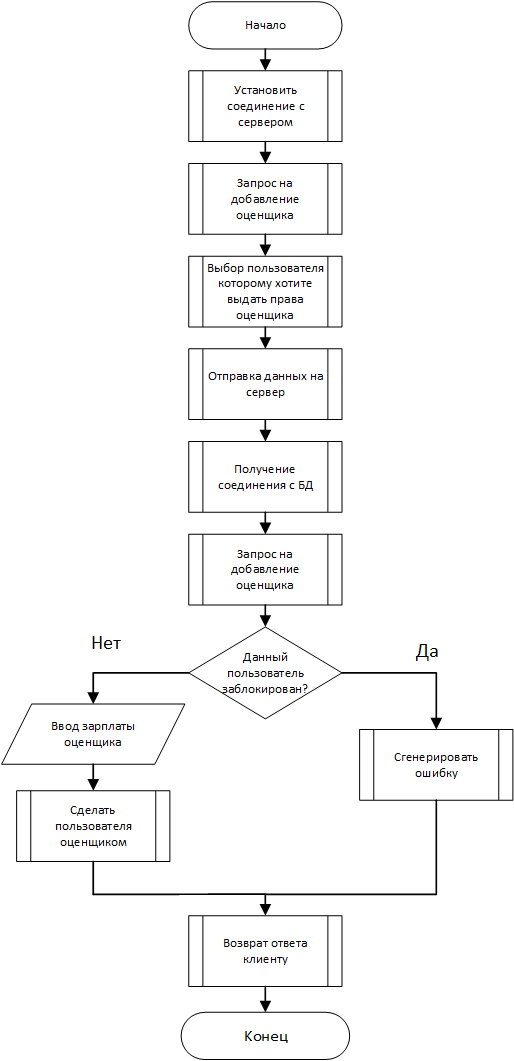


Рисунок А.1 – Блок-схема добавления оценщика

Продолжение приложения А

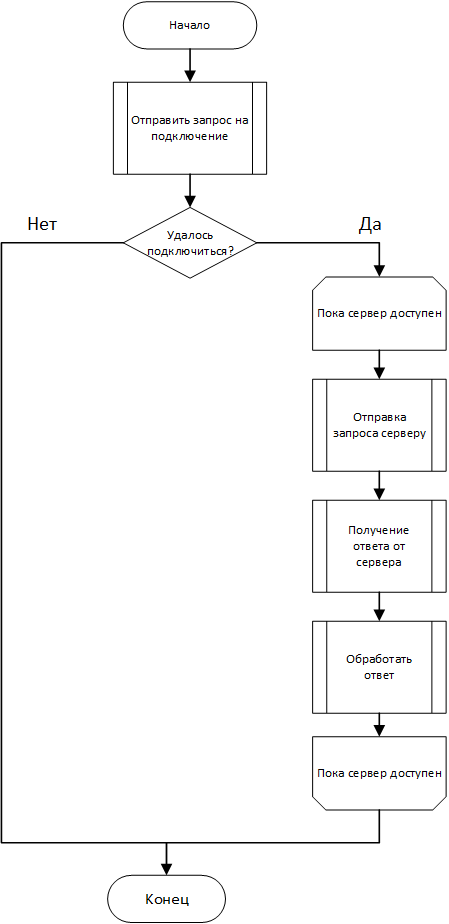


Рисунок А.2 – Схема алгоритма клиент-серверного взаимодействия

# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Листинг кода**

import appraisal.AppraisalObject;  
import appraisal.ChartData;  
import appraisal.Contract;  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import com.google.gson.reflect.TypeToken;  
import controllers.MainWindowController;  
import gsonParser.GSONParser;  
import appraisal.Contract.Status;  
import org.mindrot.jbcrypt.BCrypt;  
import personalInfo.Appraiser;  
import personalInfo.User;  
import personalInfo.User.Role;  
import serverFiles.ServerMSG.CommandTypes;  
import workingWithDB.SQLQuery;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintWriter;  
import java.lang.reflect.Type;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.sql.\*;  
import java.sql.Date;  
import java.util.\*;

public class Server {  
 private ServerSocket serverSocket;  
  
 public Server() {  
 }  
  
 public Server(int port) {  
 start(port);  
 }  
  
 public void start(int port) {  
 try {

serverSocket = new ServerSocket(port);  
 while (true) {  
 try {  
 new EchoClientHandler(serverSocket.accept()).start();  
 }  
 catch (Exception e){

Продолжение приложения Б

}  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void stop() {  
 try {  
 serverSocket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private static class EchoClientHandler extends Thread {  
 private Socket clientSocket;  
 private PrintWriter out;  
 private BufferedReader in;  
  
 public EchoClientHandler(Socket socket) {  
 this.clientSocket = socket;  
 }  
  
 public void run() {  
 ServerMSG answer = null;  
 try {  
 out = new PrintWriter(clientSocket.getOutputStream(), true);  
 in = new BufferedReader(  
 new InputStreamReader(clientSocket.getInputStream()));  
 String inputLine;  
  
 while ((inputLine = in.readLine()) != null) {  
  
 ServerMSG serverMSG = fromStringToServerCommand(inputLine);  
 answer = new ServerMSG();  
  
 if (serverMSG.getCommandType() == CommandTypes.SIGNIN) {  
 answer.getUser().setLogin(serverMSG.getUser().getLogin());  
 answer.getUser().setPasswordMask(serverMSG.getUser().getPasswordMask());  
 correctSignIn(answer);  
 if(answer.getData() == "success"){  
 Map usersConnected = new HashMap<>();  
 usersConnected.put("connected", answer.getUser().getLogin());

Продолжение приложения Б

MainWindowController.setUsersConnected(usersConnected);

}  
 out.println(fromServerCommandToString(answer));  
 } else if (serverMSG.getCommandType() ==

CommandTypes.SIGNUP) {  
 newUser(serverMSG);  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.GETAPPRAISERS) {  
 ResultSet rs = SQLQuery.getAppraisers();  
 Vector<Appraiser> appraisers = new Vector<>();  
 while (rs.next()) {  
 appraisers.add(new Appraiser(rs.getString("login"), rs.getString("passwordMask"),  
 rs.getString("firstName"), rs.getString("surname"),  
 rs.getString("patronymic"), rs.getString("phone"),  
 Role.valueOf(rs.getInt("role")),  
 rs.getBoolean("isBlocked"), rs.getDouble("salary")));  
 }  
 GSONParser<Vector<Appraiser>> gsonParser = new GSONParser<>();  
 serverMSG.setData(gsonParser.getString(appraisers));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.DELETEAPPRAISER) {  
 SQLQuery.deleteAppraiser(serverMSG.getData());  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.BLOCKUSER) {  
 SQLQuery.blockUser(serverMSG.getData());  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.EDITAPPRAISER) {  
 GSONParser<Appraiser> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<Appraiser>() {  
 }.getType();  
 Appraiser appraiser = gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.editAppraiser(appraiser);  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.NEWAPPRAISER) {  
 GSONParser<Appraiser> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<Appraiser>() {

Продолжение приложения Б

}.getType();  
 Appraiser appraiser = gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.addAppraiser(appraiser);  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.GETCONSUMERS) {  
 ResultSet rs = SQLQuery.getConsumers();  
 Vector<User> users = new Vector<>();  
  
 while (rs.next()) {  
 users.add(new User(rs.getString("login"), rs.getString("passwordMask"),  
 rs.getString("firstName"), rs.getString("surname"),  
 rs.getString("patronymic"), rs.getString("phone"),  
 Role.valueOf(rs.getInt("role")),  
 rs.getBoolean("isBlocked")));  
 }  
 GSONParser<Vector<User>> gsonParser = new GSONParser<>();  
 serverMSG.setData(gsonParser.getString(users));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.FILTERUSERS) {  
 GSONParser<HashMap<Boolean, User>> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<HashMap<Boolean, User>>() {  
 }.getType();  
  
 HashMap<Boolean, User> filterUser = gsonParser.getObject(serverMSG.getData(), type);  
 Set<Boolean> set = filterUser.keySet();  
 ResultSet rs = null;  
 for (Boolean bool : set) {  
 rs = SQLQuery.getConsumersByFilter(filterUser.get(bool), bool);  
 }  
 Vector<User> users = new Vector<>();  
 while (rs.next()) {  
 users.add(new User(rs.getString("login"), rs.getString("passwordMask"),  
 rs.getString("firstName"), rs.getString("surname"),  
 rs.getString("patronymic"),rs.getString("phone"),  
 Role.valueOf(rs.getInt("role")),

Продолжение приложения Б

rs.getBoolean("isBlocked")));  
 }  
 GSONParser<Vector<User>> vectorGSONParser = new GSONParser<>();  
 serverMSG.setData(vectorGSONParser.getString(users));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.GETUSERS) {  
 ResultSet rs = SQLQuery.getUsers();  
 Vector<User> users = new Vector<>();  
 while (rs.next()) {  
 users.add(new User(rs.getString("login"),

rs.getString("passwordMask"),  
 rs.getString("firstName"), rs.getString("surname"),  
 rs.getString("patronymic"),rs.getString("phone"),  
 Role.valueOf(rs.getInt("role")),  
 rs.getBoolean("isBlocked")));  
 }  
 GSONParser<Vector<User>> gsonParser = new GSONParser<>();  
 serverMSG.setData(gsonParser.getString(users));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.EDITUSER) {  
 GSONParser<User> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<User>() {  
 }.getType();  
 User user = gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.editUser(user);  
 out.println(fromServerCommandToString(serverMSG));  
 }  
 else if (serverMSG.getCommandType() == CommandTypes.ADDOBJECT) {  
 GSONParser<AppraisalObject> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<AppraisalObject>() {  
 }.getType();  
 AppraisalObject appraisalObject = gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.addAppraisalObject(appraisalObject, serverMSG);  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.GETOBJECTS) {  
 GSONParser<Vector<AppraisalObject>> parser = new GSONParser<>();

Продолжение приложения Б

Vector<AppraisalObject> appraisalObjects = SQLQuery.getAppraisalObjects();  
 serverMSG.setData(parser.getString(SQLQuery.getAppraisalObjects()));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.DELETEOBJECT) {  
 SQLQuery.deleteAppraisalObject(Integer.parseInt(serverMSG.getData()));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.FILTEROBJECTS) {  
 GSONParser<AppraisalObject> parser = new GSONParser<>();  
 GSONParser<Vector<AppraisalObject>> answerParser = new GSONParser<>();  
 Type type = new TypeToken<AppraisalObject>() {

}.getType();  
 Vector<AppraisalObject> appraisalObjects =  
 SQLQuery.getObjectsByFilter(parser.getObject(serverMSG.getData(), type));  
 serverMSG.setData(answerParser.getString(appraisalObjects));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.EDITOBJECT) {  
 GSONParser<AppraisalObject> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<AppraisalObject>() {  
 }.getType();  
 AppraisalObject appraisalObject = gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.editAppraisalObject(appraisalObject, serverMSG);  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.NEWOBJECTTYPE) {  
 GSONParser<String> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<String>() {  
 }.getType();  
 String objectType = gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.addObjectType(objectType, serverMSG);  
 out.println(fromServerCommandToString(serverMSG));  
 }else if (serverMSG.getCommandType() == CommandTypes.DELETEOBJECTTYPE) {  
 GSONParser<String> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<String>() {  
 }.getType();  
 String objectType =

Продолжение приложения Б

gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.deleteObjectType(objectType, serverMSG);  
 out.println(fromServerCommandToString(serverMSG));  
 }  
 else if (serverMSG.getCommandType() == CommandTypes.GETOBJECTSTYPES) {  
 GSONParser<ArrayList<String>> parser = new GSONParser<>();  
 serverMSG.setData(parser.getString(SQLQuery.getObjectsTypes()));  
 out.println(fromServerCommandToString(serverMSG));  
 }  
 else if (serverMSG.getCommandType() == CommandTypes.GETUSERCONTRACTS ||  
 serverMSG.getCommandType() == CommandTypes.GETALLCONTRACTS) {  
 GSONParser<Vector<Contract>> gsonParser = new

GSONParser<>();  
 Vector<Contract> contracts;  
 if (serverMSG.getCommandType().equals(CommandTypes.GETUSERCONTRACTS)) {  
 contracts = SQLQuery.getConsumerContacts(serverMSG);  
 } else {  
 contracts = SQLQuery.getAppraiserContacts(serverMSG);  
 }  
 serverMSG.setData(gsonParser.getString(contracts));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.NEWCONTRACT) {  
 GSONParser<Contract> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<Contract>() {  
 }.getType();  
 Contract contract = gsonParser.getObject(serverMSG.getData(), type);  
 SQLQuery.newContract(contract);  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.CONTRACTSIGNED ||  
 serverMSG.getCommandType() == CommandTypes.CONTRACTTERMINATED ||  
 serverMSG.getCommandType() == CommandTypes.CONTRACTWAITFORAPPRAISER ||  
 serverMSG.getCommandType() == CommandTypes.CONTRACTWAITFORCONSUMER) {  
  
 GSONParser<Contract> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<Contract>() {

Продолжение приложения Б

}.getType();  
 Status status = null;  
 if (serverMSG.getCommandType() == CommandTypes.CONTRACTSIGNED) {  
 status = Status.SIGNED;  
 } else if (serverMSG.getCommandType() == CommandTypes.CONTRACTTERMINATED) {  
 status = Status.TERMINATED;  
 } else if (serverMSG.getCommandType() == CommandTypes.CONTRACTWAITFORCONSUMER) {  
 status = Status.WAITFORCONSUMER;  
 } else if (serverMSG.getCommandType() == CommandTypes.CONTRACTWAITFORAPPRAISER) {  
 status = Status.WAITFORAPPRAISER;  
 }  
 Contract contract = gsonParser.getObject(serverMSG.getData(), type);  
 Appraiser appraiser = SQLQuery.getAppraiserByID(SQLQuery.getUserIDByLogin(serverMSG.getUser().getLogin

()));  
 SQLQuery.updateContract(contract, status, appraiser);  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.CONSUMERDIAGRAM) {  
 if(serverMSG.getData() == null){  
 GSONParser<ArrayList<Map<String, String>>> gsonParser = new GSONParser<>();  
 serverMSG.setData(gsonParser.getString(SQLQuery.getConsumerAppraisedObjectsStats(serverMSG)));  
 out.println(fromServerCommandToString(serverMSG));  
 }else{  
 GSONParser<ArrayList<ChartData<Double, Double>>> gsonParser = new GSONParser<>();  
 serverMSG.setData(gsonParser.getString(SQLQuery.getConsumerStats(serverMSG)));  
 out.println(fromServerCommandToString(serverMSG));  
 }  
 } else if (serverMSG.getCommandType() == CommandTypes.FILTERCONTRACTS) {  
 GSONParser<Contract> parser = new GSONParser<>();  
 GSONParser<Vector<Contract>> answerParser = new GSONParser<>();  
 Type type = new TypeToken<Contract>() {  
 }.getType();  
 Vector<Contract> contractVector =  
 SQLQuery

Продолжение приложения Б

.getContractsByFilter(parser.getObject(serverMSG.getData(), type), serverMSG);  
 serverMSG.setData(answerParser.getString(contractVector));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.LINECHART) {  
 GSONParser<ArrayList<ChartData<Date, Double>>> gsonParser = new GSONParser<>();  
 serverMSG.setData(gsonParser.getString(SQLQuery.getLineChart(serverMSG)));  
 out.println(fromServerCommandToString(serverMSG));  
 } else if (serverMSG.getCommandType() == CommandTypes.BESTAPPRAISER) {  
 GSONParser<Vector<ChartData<Appraiser, Integer>>>

gsonParser = new GSONParser<>();  
 serverMSG.setData(gsonParser.getString(SQLQuery.getBestAppraiser()));

out.println(fromServerCommandToString(serverMSG));  
 }  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 try {  
 in.close();  
 out.close();  
 clientSocket.close();  
 Map usersConnected = new HashMap<>();  
 usersConnected.put("disconnected", answer.getUser().getLogin());  
 MainWindowController.setUsersConnected(usersConnected);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void correctSignIn(ServerMSG answer) throws SQLException {  
 Connection connection =  
 DriverManager.getConnection("jdbc:mysql://localhost:3306/cw5sem", "root", "123456");  
 String sql = "SELECT login, passwordMask, firstName, surname, patronymic, phone, role, isBlocked FROM users\n" +  
 "WHERE login = '" + answer.getUser().getLogin() + "';";  
 Statement stmt = connection.createStatement();  
 ResultSet rs = stmt.executeQuery(sql);  
 while (rs.next()) {  
 User user = new User(rs.getString("login"), rs.getString("passwordMask"),

Продолжение приложения Б

rs.getString("firstName"), rs.getString("surname"),  
 rs.getString("patronymic"), rs.getString("phone"), Role.valueOf(rs.getInt("role")),  
 rs.getBoolean("isBlocked"));  
  
 String checkingPassword = answer.getUser().getPasswordMask();  
  
 answer.setUser(user);  
 answer.setCommandType(CommandTypes.SIGNIN);  
 if (user.isBlocked()) {  
 answer.setData("blocked");  
 } else if (BCrypt.checkpw(checkingPassword, answer.getUser().getPasswordMask())) {  
 answer.setData("success");  
 } else {  
 answer.setData("password");  
 }  
 return;  
 }  
 answer.setUser(null);  
 answer.setCommandType(CommandTypes.SIGNIN);  
 answer.setData("login");  
 return;  
 }

private ServerMSG fromStringToServerCommand(String json) {  
 GsonBuilder builder = new GsonBuilder();  
 Gson gson = builder.create();  
 ServerMSG serverMSG = gson.fromJson(json, ServerMSG.class);  
 System.out.println(json);  
 return serverMSG;  
 }  
  
 private String fromServerCommandToString(ServerMSG serverMSG) {  
 Gson gson = new Gson();  
 String json = gson.toJson(serverMSG);  
 return json;  
 }  
  
 private void newUser(ServerMSG answer) {  
 String sql = "INSERT INTO users (login, passwordMask, firstName, surname, patronymic, phone, role)" +  
 " VALUES(?,?,?,?,?,?,?)";  
 try {  
 Connection connection =  
 DriverManager.getConnection("jdbc:mysql://localhost:3306/cw5sem", "root", "123456");

Продолжение приложения Б

String hashed = BCrypt.hashpw(answer.getUser().getPasswordMask(), BCrypt.gensalt(12));  
 answer.getUser().setPasswordMask(hashed);  
  
 Role roleType = Role.ADMINISTRATOR;  
 int roleTypeID = answer.getUser().getRole().getValue();  
  
 PreparedStatement statement = connection.prepareStatement(sql);  
 statement.setString(1, answer.getUser().getLogin());  
 statement.setString(2, answer.getUser().getPasswordMask());  
 statement.setString(3, answer.getUser().getFirstName());  
 statement.setString(4, answer.getUser().getSurname());  
 statement.setString(5, answer.getUser().getPatronymic());  
 statement.setString(6, answer.getUser().getPhone());  
 statement.setInt(7, roleTypeID);  
 statement.executeUpdate();  
 answer.setData("success");  
 return;  
 } catch (SQLException throwables) {  
 answer.setData("login exists");  
 return;  
 }  
 }

}  
}

private void showDiagram() {  
 try{  
 LocalDate date = LocalDate.now();  
 statisticsPane.getChildren().remove(objectsPricesChart);  
 GSONParser< ArrayList<Map<String, String>>> gsonParser = new GSONParser<>();  
 GSONParser<ArrayList<ChartData<Double, Double>>> gsonParser1 = new GSONParser<>();  
 Type type = new TypeToken<ArrayList<Map<String, String>>>() {  
 }.getType();  
 Type type1 = new TypeToken<ArrayList<ChartData<Double, Double>>>() {  
 }.getType();  
 ArrayList<Map<String, String>> result = new ArrayList<>();  
 ArrayList<ChartData<Double,Double>> result1 = new ArrayList<>();  
 ServerMSG serverMSG = null;  
 ServerMSG answer = null;  
  
 serverMSG = new ServerMSG(loggedUser, CommandTypes.CONSUMERDIAGRAM, null);  
 answer = client.sendMessage(serverMSG);  
 result.addAll(gsonParser.getObject(answer.getData(), type));  
  
 serverMSG = new ServerMSG(loggedUser, CommandTypes.CONSUMERDIAGRAM,

Продолжение приложения Б

"1");  
 answer = client.sendMessage(serverMSG);  
 result1.addAll(gsonParser1.getObject(answer.getData(), type1));  
  
  
  
 final CategoryAxis xAxis = new CategoryAxis();  
 final NumberAxis yAxis = new NumberAxis();  
 objectsPricesChart = new BarChart<String, Number>(xAxis, yAxis);  
 objectsPricesChart.setTitle("Разница между ожидаемой стоимостью и\n стоимостью предложенной оценщиком");  
 xAxis.setLabel("Названия объектов");  
 yAxis.setLabel("Денежные средства, руб.");  
  
 XYChart.Series series1 = new XYChart.Series();  
 series1.setName("Ожидаемая цена, руб.");  
 XYChart.Series series2 = new XYChart.Series();  
 series2.setName("Цена предложенная оценщиком, руб.");  
  
 Iterator<Map<String, String>> iterator = result.iterator();  
  
 ArrayList<String> sr1 = new ArrayList<>();  
 ArrayList<Double> sr2 = new ArrayList<>();  
  
 for (ChartData<Double, Double> chartData : result1) {  
 if(iterator.hasNext()){  
 Map<String, String> map = new HashMap<>();  
 map = iterator.next();  
 String appraisalObjectNameAndDate = String.valueOf(map.keySet());  
 StringBuffer stringBuffer = new StringBuffer(appraisalObjectNameAndDate);  
 stringBuffer.delete(0,1);  
 stringBuffer.delete(stringBuffer.length()-1,stringBuffer.length());  
 appraisalObjectNameAndDate = stringBuffer.toString();

Double expectedPrice = chartData.getValue1();  
 Double appraiserPrice = chartData.getValue2();  
 series1.getData().add(new XYChart.Data(appraisalObjectNameAndDate+ "\nДата заключения\nдоговора:\n" + map.get(appraisalObjectNameAndDate), expectedPrice));  
 series2.getData().add(new XYChart.Data(appraisalObjectNameAndDate + "\nДата заключения\nдоговора:\n" + map.get(appraisalObjectNameAndDate), appraiserPrice));  
 }  
 }  
  
 objectsPricesChart.getData().addAll(series1, series2);

Продолжение приложения Б

statisticsPane.getChildren().addAll(objectsPricesChart);  
 }  
 catch (Exception e) {  
 Alert alert = new Alert(AlertType.ERROR);  
 alert.setTitle("Ошибка соединения с сервером");  
 alert.setHeaderText(null);  
 alert.setContentText("Пожалуйста, проверьте подключение к серверу.");  
 alert.showAndWait();  
 }  
 }

public class Client {  
 private Socket clientSocket;  
 private PrintWriter out;  
 private BufferedReader in;  
  
 public void startConnection(String ip, int port) {  
 try {  
 clientSocket = new Socket(ip, port);  
 out = new PrintWriter(clientSocket.getOutputStream(), true);  
 in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
 } catch (ConnectException e) {  
 e.printStackTrace();  
 } catch (UnknownHostException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }

public ServerMSG sendMessage(ServerMSG msg) {  
 String answerString = new String();  
 ServerMSG answer = new ServerMSG();  
 try {  
 out.println(fromServerCommandToString(msg));  
  
 answerString = in.readLine();  
 answer = fromStringToServerCommand(answerString);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return answer;  
 }  
  
 public void stopConnection() {  
 try {  
 in.close();  
 out.close();

Продолжение приложения Б

clientSocket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private ServerMSG fromStringToServerCommand(String json){  
 GsonBuilder builder = new GsonBuilder();  
 Gson gson = builder.create();  
 ServerMSG serverMSG = gson.fromJson(json, ServerMSG.class);  
 return serverMSG;  
 }  
  
 private String fromServerCommandToString(ServerMSG serverMSG){  
 Gson gson = new Gson();  
 String json = gson.toJson(serverMSG);  
 System.out.println(json);  
 return json;  
 }  
}

private void drawLineChart() {  
 try {

final CategoryAxis xAxis = new CategoryAxis();  
 final NumberAxis yAxis = new NumberAxis();  
 xAxis.setLabel("Месяц");  
 yAxis.setLabel("Прибыль, руб.");  
  
 incomeLineChart = new LineChart<String, Number>(xAxis, yAxis);  
  
 incomeLineChart.setTranslateX(250);  
 incomeLineChart.setTitle("Мониторинг прибыли компании за последние 6 месяцев");  
  
 XYChart.Series series = new XYChart.Series();  
 series.setName("Кривая прибыли");  
  
 GSONParser<Vector<ChartData<Date, Double>>> gsonParser = new GSONParser<>();  
 Type type = new TypeToken<Vector<ChartData<Date, Double>>>() {  
 }.getType();  
 Vector<ChartData<Date, Double>> result;  
 ServerMSG serverMSG = new ServerMSG(loggedUser, CommandTypes.LINECHART, null);  
 ServerMSG answer = client.sendMessage(serverMSG);  
  
 result = gsonParser.getObject(answer.getData(), type);  
  
 for (ChartData<Date, Double> entry : result) {

Продолжение приложения Б

String month = null;  
 try {  
 java.util.Date bufDate = new SimpleDateFormat("yyyy-MM-dd").parse(String.valueOf(entry.getValue1()));  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("MMM yyyy", new Locale("ru"));  
 month = simpleDateFormat.format(bufDate);  
 }catch (Exception e) {  
 //ignore  
 }  
 series.getData().add(new XYChart.Data(month, entry.getValue2()));  
 }  
  
 incomeLineChart.getData().add(series);  
 reportPane.getChildren().add(incomeLineChart);  
 }  
 catch (Exception e){  
 Alert alert = new Alert(AlertType.ERROR);  
 alert.setTitle("Ошибка соединения с сервером");  
 alert.setHeaderText(null);  
 alert.setContentText("Пожалуйста, проверьте подключение к серверу.");  
 alert.showAndWait();  
 }  
}

# ПРИЛОЖЕНИЕ В

**(обязательное)**

**Листинг кода Базы данных**

CREATE DATABASE IF NOT EXISTS `cw5sem`;

use cw5sem;

CREATE TABLE IF NOT EXISTS `users` (

`userID` int NOT NULL AUTO\_INCREMENT,

`login` varchar(50) NOT NULL,

`passwordMask` varchar(200) NOT NULL,

`firstName` varchar(50) NOT NULL,

`surname` varchar(50) NOT NULL,

`patronymic` varchar(50) NOT NULL,

`phone` varchar(50) NOT NULL,

`role` int NOT NULL DEFAULT '0',

`isBlocked` tinyint NOT NULL DEFAULT '0',

PRIMARY KEY (`userID`),

UNIQUE KEY `login\_UNIQUE` (`login`));

CREATE TABLE IF NOT EXISTS `appraisers` (

`appraiserID` int NOT NULL,

`salary` double DEFAULT '0',

PRIMARY KEY (`appraiserID`),

CONSTRAINT `appraiserID` FOREIGN KEY (`appraiserID`) REFERENCES `users` (`userID`) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `objectstypes` (

`typeID` int NOT NULL AUTO\_INCREMENT,

`type` varchar(50) NOT NULL,

PRIMARY KEY (`typeID`));

CREATE TABLE IF NOT EXISTS `objectstoappraise` (

`objectId` int NOT NULL AUTO\_INCREMENT,

`typeId` int NOT NULL,

`description` varchar(100) DEFAULT NULL,

PRIMARY KEY (`objectId`),

UNIQUE KEY `name\_UNIQUE` (`name`),

CONSTRAINT `objectTypeId` FOREIGN KEY (`objectTypeId`) REFERENCES `objectstypes` (`typeID`));

CREATE TABLE IF NOT EXISTS `appraisalagreement` (

`appraisalAgreementID` int NOT NULL AUTO\_INCREMENT,

`dateOfSigning` date DEFAULT NULL,

`objectID` int NOT NULL,

`priceForAppraisal` double DEFAULT NULL,

`expectedPrice` double NOT NULL,

`appraiserPrice` double DEFAULT NULL,

Продолжение приложения В

`status` int NOT NULL DEFAULT '0',

`commentFromConsumer` varchar(100) DEFAULT NULL,

`commentFromAppraiser` varchar(100) DEFAULT NULL,

`consumerID` int NOT NULL,

`currentAppraiserID` int DEFAULT NULL,

PRIMARY KEY (`appraisalAgreementID`),

CONSTRAINT `appraiserIDFK` FOREIGN KEY (`currentAppraiserID`) REFERENCES `appraisers` (`appraiserID`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `object` FOREIGN KEY (`objectID`) REFERENCES `objectstoappraise` (`objectId`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `userIDFK` FOREIGN KEY (`consumerID`) REFERENCES `users` (`userID`) ON DELETE CASCADE ON UPDATE CASCADE);