Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПО ПРОГРАММЕ БАКАЛАВРИАТА

09.03.04 Программная инженерия

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

«Разработка программно-информационных систем»

Разработка программно-информационный системы агентства недвижимости

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР		А. И. Газинский
	(подпись, дата)	(инициалы, фамилия)
Группа ПО-11б		
Руководитель ВКР		Е. А. Петрик
	(подпись, дата)	(инициалы, фамилия)
Нормоконтроль		А. А. Чаплыгин
	(подпись, дата)	(инициалы, фамилия)
ВКР допущена к защите:		
Заведующий кафедрой		А. В. Малышев
	(подпись, дата)	(инициалы, фамилия)

Курск 2025 г.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

УТВЕРЖДАЮ:

Заведующий кафедрой

_____

(подпись, инициалы, фамилия)

«____» _____ 20____ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ ПО ПРОГРАММЕ БАКАЛАВРИАТА

Студента Газинского А. И., шифр 21-06-0012, группа ПО-11б

1. Тема «Разработка программно-информационный системы агентства недвижимости » утверждена приказом ректора ЮЗГУ от «17» апреля 2025 г. № 1828-с.

2. Срок предоставления работы к защите «11» июня 2025 г.

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

1) провести анализ предметной области;
2) спроектировать базу данных;
3) разработать интерфейс;
4) реализовать приложение.

3.2. Входные данные и требуемые результаты для программы:

1) Входными данными для программной системы являются: информация о сотрудниках агентства недвижимости, информация о клиентах клиентах, информация об объектах недвижимости и их владельцах, предоставляемая ими в процессе регистрации в системе.

2) Выходными данными для программной системы являются: база данных, где содержится вся информация о сотрудниках агентства недвижи-

мости, клиентах и объектах недвижимости; договора купли- продажи объектов недвижимости; договора аренды объектов недвижимости. уведомление о создании договоров; сообщения об ошибках.

4. Содержание работы (по разделам):

4.1. Введение.

4.1. Анализ предметной области.

4.2. Техническое задание.

4.3. Технический проект.

4.4. Рабочий проект.

4.5. Заключение.

4.6. Список использованных источников.

5. Перечень графического материала:

Лист 1. Сведения о ВКРБ.

Лист 2. Цель и задачи разработки.

Лист 3. диаграмма прецедентов.

Лист 4. Концептуальная модель предметной области.

Лист 5. Интерфейс приложения.

Лист 6. Заключение.

Руководитель ВКР		Е. А. Петрик
	(подпись, дата)	(инициалы, фамилия)
Задание принял к исполнению		А. И. Газинский
	(подпись, дата)	(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 85 страницам. Работа содержит 23 иллюстрации, 6 таблиц, 19 библиографических источников и 6 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: SQLite, база данных, концептуальное проектирование базы данных, логическое проектирование базы данных, физическое проектирование базы данных, ER-модель данных, реляционная модель данных.

Результатом выполнения данной работы является приложение для работы пользователей с базой данных агентства недвижимости. При проектировании базы данных использован ER-подход к проектированию реляционных баз данных. Интерфейс приложения содержит: главную форму для вывода таблицы, форму для просмотра данных и редактирования записей базы данных, форму для добавления объектов недвижимости, владельцев, покупателей и заключения договоров.

Приложение предназначено для использования на компьютерах под управлением ОС Windows.

ABSTRACT

The volume of work is 85 pages. The work contains 23 illustrations, 6 tables, 19 bibliographic sources and 6 sheets of graphic material. The number of applications is 2. The graphic material is presented in annex A. The layout of the site, including the connection of components, is presented in annex B.

Keywords list: SQLite, events, Qt, database, conceptual database design, logical database design, physical database design, ER data model, relational data model . The result of this work is an application for users to work with the database of a real estate agency. The ER approach to relational database design was used in database design. The application interface contains: the main form for displaying tables, a form for viewing data and editing database records, a form for adding real estate, owners, buyers and concluding contracts.

The application is intended for use on computers running Windows.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных.

ИС – информационная система.

РП – рабочий проект.

СУБД – система управления базами данных.

ТЗ – техническое задание.

ТП – технический проект.

АН- агентство недвижимости.

ВВЕДЕНИЕ

Агентство недвижимости — это компания, которая предлагает своим клиентам услуги, связанные с продажей, покупкой и арендой различной недвижимости

В современном мире информация является одним из самых ценных ресурсов, и ее эффективное управление имеет ключевое значение для успешного функционирования различных бизнес-секторов. Агентства недвижимости, оперирующие на динамичном рынке, сталкиваются с необходимостью хранения, обработки и анализа больших объемов данных о недвижимости, клиентах и сделках. Создание базы данных для агентства недвижимости становится неотъемлемым шагом, позволяющим оптимизировать рабочие процессы, повысить уровень обслуживания клиентов и улучшить внутреннюю организацию.

Цель данной работы заключается в разработке программно-информационной системы управления агентством недвижимости, с помощью которой можно быстро и эффективно осуществлять необходимые операции. В ходе разработки данной программно-информационной системы необходимо создать базу данных, поэтому в работе также будут рассмотрены ключевые аспекты проектирования баз данных, такие как определение сущностей и их взаимосвязей, выбор подходящей модели данных и применение современных технологий для реализации проекта.

Актуальность темы разработки базы данных для агентства недвижимости обуславливается несколькими ключевыми факторами:

1. Рост рынка недвижимости: В последнее время наблюдается стабильный рост интереса к покупке, продаже и аренде жилья. Это создает потребность в эффективных системах управления информацией, которая будет доступна в режиме реального времени.

2. Увеличение объема данных: В условиях цифровизации объем данных, связанных с недвижимостью, постоянно растет. Базы данных позволя-

ют систематизировать, хранить и обрабатывать эти данные, что делает работу агентства более эффективной.

3. Улучшение клиентского обслуживания: Современные клиенты ожидают быстрого и качественного обслуживания. Наличие удобной и функциональной базы данных позволяет агентствам быстро реагировать на запросы клиентов, предоставляя актуальную информацию о доступных объектах недвижимости и их характеристиках.

4. Конкуренция на рынке: Агентства недвижимости сталкиваются с растущей конкуренцией. Эффективное использование баз данных может стать важным конкурентным преимуществом, позволяя оптимизировать внутренние бизнес-процессы и улучшить маркетинговые стратегии.

5. Автоматизация процессов: База данных позволяет автоматизировать множество рутинных задач, таких как учет объектов, управление заявками, расчеты и аналитика. Это активно снижает затраты времени и ресурсов, повышая общую производительность работы.

6. Анализ рынка: Система управления базами данных предоставляет возможность анализа тенденций и динамики рынка недвижимости, что является важным аспектом стратегического планирования для агентств.

7. Интеграция с другими системами: Возможность интеграции базы данных с другими информационными системами (например, CRM-системами, веб-сайтами и порталами объявления) позволяет создавать единую экосистему, что упрощает работу сотрудников и улучшает взаимодействие с клиентами.

С учетом вышеуказанного, разработка базы данных для агентства недвижимости является не только актуальной, но и необходимой для успешного функционирования бизнеса в условиях быстро меняющегося рынка.

В основные функции данной программно-информационной системы входит: возможность просмотра, добавление, редактирование, удаление информации о недвижимости, владельце недвижимости, покупателе, а также информация о сотрудниках агентства недвижимости. Приложение и база дан-

ных позволят эффективно управлять всеми аспектами агентства, сокращая время обработки и повышая общую производительность.

Основными задачами при проектировании и разработке приложения и БД являются:

- исследование предметной области;
- проектирование базы данных;
- создание базы данных;
- заполнение базы данных информацией;
- разработка интерфейса;
- реализация приложения.

Таким образом, данная работа направлена на создание функциональной и эффективной программно-информационной системы, способствующей успешному развитию агентства недвижимости и улучшению качества предоставляемых услуг.

Организация и объем работы представлены следующим образом: отчет включает введение, четыре раздела основной части, заключение, список литературы и два приложения. Общий объем выпускной квалификационной работы составляет 86 страниц.

Во введении обозначена цель исследования, сформулированы задачи, определена структура и дано краткое описание содержания каждого раздела.

Первый раздел, посвященный описанию технической характеристики предметной области, содержит цели и задачи агентства недвижимости, а также анелиз бизнес-процессов.

Второй раздел, соответствующий стадии технического задания, содержит перечень требований к разрабатываемому приложению. Третий раздел, отражающий стадию технического проектирования, демонстрирует проектные решения для приложения и отображает структуру базы данных.

Четвертый раздел включает в себя результаты тестирования разработанного приложения.

В заключении суммированы ключевые результаты, достигнутые в процессе разработки.

Приложение А содержит графические материалы, а Приложение Б – фрагменты исходного кода.

1 Анализ предметной области

1.1 Описание предметной области

Агентство недвижимости (АН) — это организация, предоставляющая посреднические услуги при совершении сделок с недвижимостью. Деятельность АН охватывает широкий спектр операций, включая:

– покупка и продажа недвижимости: поиск объектов, соответствующих требованиям клиентов, организация просмотров, переговоры о цене, оформление договоров купли-продажи;

– аренда (долгосрочная и краткосрочная): подбор объектов для арендаторов, поиск арендаторов для собственников, составление договоров аренды;

– управление недвижимостью (Property Management): обслуживание объектов недвижимости, контроль платежей, ремонт и техническое обслуживание;

– консультационные услуги: оценка стоимости недвижимости, юридическое сопровождение сделок, помощь в получении ипотеки и т.д.

Эффективная работа АН предполагает обработку и анализ большого объёма информации, относящейся к различным категориям:

– объекты недвижимости: квартиры, дома, земельные участки, коммерческая недвижимость (офисы, магазины, склады) и т.д. Информация об объектах включает адрес, характеристики (площадь, количество комнат, материалы стен, год постройки и т.д.), фотографии, цену, описание, юридические документы;

– клиенты: потенциальные покупатели, продавцы, арендаторы и арендодатели. Информация о клиентах включает контактные данные, предпочтения, требования к недвижимости, историю сделок;

– сотрудники: риелторы, менеджеры, юристы, оценщики и другие специалисты. Информация о сотрудниках включает контактные данные, специализацию, историю работы, комиссионные;

– сделки: договоры купли-продажи, аренды, оказания услуг. Информация о сделках включает сведения об объекте, клиентах, сотрудниках, дате заключения, цене, условиях оплаты, комиссии АН;

– рекламные кампании и источники: информация о размещенных объявлениях, каналах привлечения клиентов, результатах рекламных кампаний.

1.2 Цели и задачи агентства недвижимости

Основными целями АН являются:

– максимизация прибыли: за счет успешных сделок и оказания качественных услуг;

– удовлетворение потребностей клиентов: обеспечение оптимального подбора недвижимости и сопровождение сделок;

– повышение эффективности работы сотрудников: оптимизация рабочих процессов, сокращение временных затрат и повышение производительности;

– расширение клиентской базы: привлечение новых клиентов и удержание существующих;

– укрепление репутации: Предоставление надежных и профессиональных услуг.

Для достижения этих целей АН выполняет следующие задачи:

– поиск и привлечение клиентов: Использование различных каналов рекламы и маркетинга;

– поиск и оценка объектов недвижимости: анализ рынка, поиск подходящих объектов, оценка их стоимости;

– организация просмотров и переговоров: встречи с клиентами, организация показов объектов, ведение переговоров о цене и условиях сделки;

– юридическое сопровождение сделок: проверка юридической чистоты объектов, подготовка и оформление договоров, регистрация сделок;

– финансовый контроль: контроль платежей, ведение бухгалтерского учета, расчет комиссионных;

– анализ рынка: сбор и анализ данных о рынке недвижимости, прогнозирование тенденций.

1.3 Особенности предметной области, влияющие на проектирование БД

При проектировании базы данных для АН необходимо учитывать следующие особенности:

– большой объем данных: база данных должна быть способна обрабатывать большие объемы данных о недвижимости, клиентах и сделках;

– необходимость поиска и фильтрации данных: пользователям нужна возможность быстрого поиска объектов по различным критериям (цена, площадь, местоположение, количество комнат и т. д.), а также фильтрации данных по различным параметрам;

– необходимость формирования отчетов: база данных должна обеспечивать возможность формирования различных отчетов, необходимых для анализа деятельности АН (отчеты о сделках, комиссионных, продажах и т. д.);

– безопасность данных: необходимо обеспечить защиту конфиденциальной информации о клиентах и сотрудниках;

– масштабируемость: база данных должна быть масштабируемой, чтобы учитывать рост агентства и увеличение объема обрабатываемой информации;

– интеграция с другими системами (возможно): интеграция с сайтом АН, CRM-системами, системами учета и отчетности.

1.4 Анализ бизнес-процессов

На основе анализа неформального описания предметной области были сформулированы бизнес-правила:

– у каждого объекта недвижимости должен быть владелец;

– у каждого владельца должен быть телефон для связи;

– каждая сделка имеет определенный тип;

– при регистрации каждой сделки данные о продавце и покупателе обязательны;

– каждый объект недвижимости, сотрудник, владелец, покупатель, сделка должны иметь уникальный код (ID);

– один владелец может иметь несколько квартир в собственности;

– необходимо корректно вводить данные во всех полях.

Ограничения целостности для таблицы ОБЪЕКТ НЕДВИЖИМОСТИ

– код объекта недвижимости является уникальным для каждого объекта недвижимости, разрешены только цифры;

– количество комнат, цена- данные строки могут содержать только значения в виде цифр;

– недопустимы пустые значения во всех полях, кроме срока аренды.

Ограничения целостности для таблицы ПОКУПАТЕЛЬ/АРЕНДАТОР

– код покупателя/арендатора является уникальным для каждого покупателя/арендатора, разрешены только цифры;

– паспортные данные, контактные данные- данные строки могут содержать только значения в виде цифр;

– фамилия, имя, отчество – строка символов, длиной до 50 символов. Может содержать только буквы русского алфавита;

– недопустимы пустые значения во всех полях, кроме отчества клиента.

Ограничения целостности для таблицы ПРОДАВЕЦ/АРЕНДОДАТЕЛЬ

Правила для контроля уникальности в ключевом поле и требования к типам данных и ограничения на допустимые значения данных во всех полях разрабатываются по аналогии с приведенными для таблицы ПОКУПАТЕЛЬ-/АРЕНДОДАТЕЛЬ.

Ограничения целостности для таблицы ДОГОВОР АРЕНДЫ

– код договора аренды является уникальным для каждого договора, разрешены только цифры;

– дата заключения договора- календарная дата;

– при заключении договора обязательно должны быть заполнены данные о арендодателе, арендаторе, сотруднике агентства, объекте недвижимости;

– недопустимы пустые значения во всех полях.

Ограничения целостности для таблицы ДОГОВОР ПРОДАЖИ

Правила для контроля уникальности в ключевом поле и требования к типам данных и ограничения на допустимые значения данных во всех полях разрабатываются по аналогии с приведенными для таблицы ДОГОВОР АРЕНДЫ.

2 Техническое задание

2.1 Основание для разработки

Полное наименование системы: «База данных агентства недвижимости». Основанием для разработки программы является приказ ректора ЮЗГУ от «17» апреля 2025 г. №1828-с «О направлении (допуске) на практику».

2.2 Цель и назначение разработки

Программно-информационная система предназначена для создания договоров аренды и договоров купли-продажи в агентстве недвижимости. С системой должны работать следующие группы пользователей:

– сотрудник агентства недвижимости;

– руководство агентства недвижимости.

Сотрудник должен иметь возможность добавлять, удалять и редак- тировать клиентов и объекты недвижимости. Руководство агентства недвижимости должно иметь те же возможности, что и сотрудник, а также добавлять и удалять данные сотрудников.

Данная разработка направлена на оптимизацию деятельности агентства недвижимости.

В рамках этой разработки предусмотрены следующие задачи:

– создание структуры базы данных;

– разработка дизайна пользовательского интерфейса;

– разработка методов отображения данных из базы данных;

– разработка инструментов для администрирования базы данных, чтобы поддерживать актуальность информации.

2.3 Требования к программной системе

2.3.1 Требования к данным

Входными данными для системы являются:

– информация о сотруднике, предоставляемая им в процессе регистрации в системе;

– информация о покупателе, предоставляемая им в процессе оформления сделки;

– информация о владельце объекта недвижимости, предоставляемая им в процессе регистрации в системе;

– информация об объекте недвижимости.

Выходными данными для системы являются:

– список сотрудников;

– список клиентов;

– список объектов недвижимости;

– созданный договор аренды ;

– созданный договор купли-продажи;

– сообщения об ошибках.

2.3.2 Функциональные требования

Приложение имеет две группы пользователей с разными правами: руководство и сотрудники.

Сотрудникам должны быть доступны следующие функции:

– добавление покупателей;

– добавление владельцев недвижимости;

– добавление объектов недвижимости;

– просмотр информации о сотрудниках;

– просмотр информации о покупателях;

– просмотр информации о владельцах объектов недвижимости;

– просмотр информации о объектах недвижимости;

– удаление различной информации;

– создание договоров аренды;

– создание договоров купли-продажи;

– удаление договоров.

На рисунке 2.1 изображены прецеденты для сотрудника агентства.

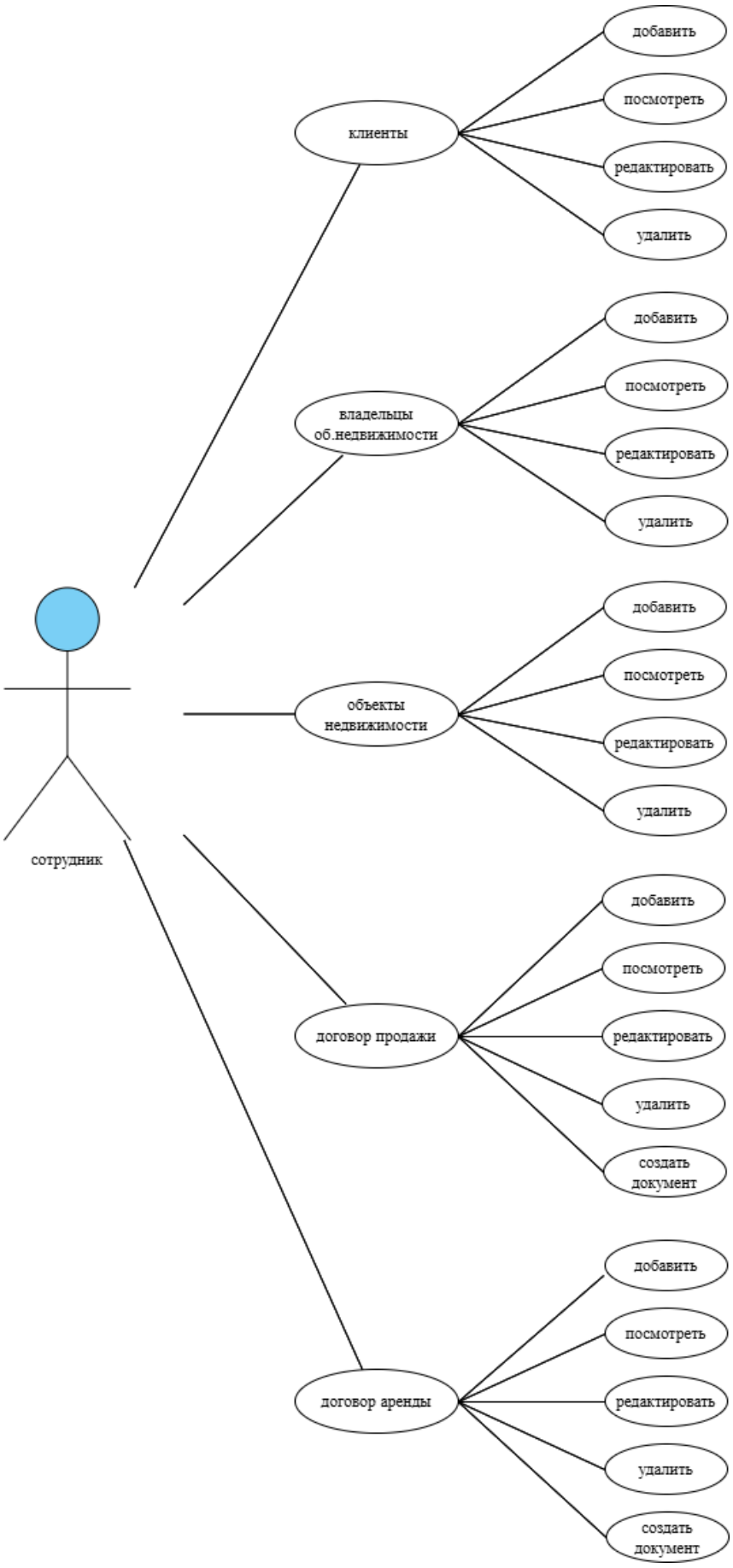

Рисунок 2.1 – Прецеденты для сотрудника

Руководству должны быть доступны следующие функции:

– добавление сотрудников;
– удаление сотрудников;
– добавление покупателей;
– добавление владельцев недвижимости;
– добавление объектов недвижимости;
– просмотр информации о сотрудниках;
– просмотр информации о покупателях;
– просмотр информации о владельцах объектов недвижимости;
– просмотр информации о объектах недвижимости;
– удаление различной информации;
– создание договоров аренды;
– создание договоров купли-продажи;
– удаление договоров.

2.3.2.1 Сценарий прецедента сотрудника «добавление информации о покупателе/арендаторе»

Основной успешный сценарий для прецедента «добавление информации о покупателе/арендаторе».

1. Открыть окно «добавить покупателя/арендатора».
2. Заполнить данные покупателя/арендатора.
3. Нажать кнопку «добавить»

2.3.2.2 Сценарий прецедента сотрудника «просмотр информации о покупателе/арендаторе»

Основной успешный сценарий для прецедента «просмотр информации о покупателе/арендаторе».

1. Открыть окно «посмотреть покупателей/арендаторов».

2.3.2.3 Сценарий прецедента сотрудника «добавление информации о продавце/арендодателе»

Основной успешный сценарий для прецедента «добавление информации о продавце/арендодателе».

1. Открыть окно «добавить продавца/арендодателя».
2. Заполнить данные продавца/арендодателя.
3. Нажать кнопку «добавить».

2.3.2.4 Сценарий прецедента сотрудника «просмотр информации о продавце/арендодателе»

Основной успешный сценарий для прецедента «просмотр информации о продавце/арендодателе».

1. Открыть окно «посмотреть продавца/арендодателя».

2.3.2.5 Сценарий прецедента сотрудника «добавление объекта недвижимости»

Основной успешный сценарий для прецедента «добавление объекта недвижимости».

1. Открыть окно «добавить объект недвижимости».
2. Заполнить информацию об объекте недвижимости.
3. Нажать кнопку «добавить»

2.3.2.6 Сценарий прецедента сотрудника «просмотр объектов недвижимости»

Основной успешный сценарий для прецедента «просмотр объектов недвижимости».

1. Открыть окно «просмотреть объекты недвижимости».

2.3.2.7 Сценарий прецедента сотрудника «добавить договор купли/продажи»

Основной успешный сценарий для прецедента «добавить договор купли/продажи».

1. Открыть окно «добавить договор купли/продажи».
2. Заполнить все данные.
3. Нажать кнопку «добавить».

2.3.2.8 Сценарий прецедента сотрудника «посмотреть договора купли/продажи»

Основной успешный сценарий для прецедента «посмотреть договор купли/продажи».

1. Открыть окно «посмотреть договор купли/продажи».
2. Выбрать договор.
3. Нажать кнопку «создать договор».
4. Сохранить договор на устройстве.

2.3.2.9 Сценарий прецедента сотрудника «добавить договор аренды»

Основной успешный сценарий для прецедента «добавить договор аренды».

1. Открыть окно «добавить договор аренды».
2. Заполнить все данные.
3. Нажать кнопку «добавить».

2.3.2.10 Сценарий прецедента сотрудника «посмотреть договор аренды»

Основной успешный сценарий для прецедента «просмотреть договора аренды».

1. Открыть окно «посмотреть договор аренды».

2. Выбрать договор.
3. Нажать кнопку «создать договор».
4. Сохранить договор на устройстве.

2.3.2.11 Сценарий прецедента руководителя «Добавление сотрудника»

Основной успешный сценарий для прецедента «Добавление сотрудника».

1. Открыть окно «добавить сотрудника».
2. Заполнить данные сотрудника.
3. Нажать кнопку «добавить».

2.3.2.12 Сценарий прецедента руководителя «Просмотр сотрудников»

Основной успешный сценарий для прецедента «Просмотр сотрудников».

1. Открыть окно «просмотреть сотрудников агентства».

2.3.3 Требования пользователя к интерфейсу приложения

Приложение должно иметь следующие окна:

– Главное окно;
– Окно с добавлением покупателя/арендатора;
– Окно с просмотром покупателей/арендаторов;
– Окно с добавлением продавцов/арендодателей;
– Окно с просмотром продавцов/арендодателей;
– Окно с добавлением объектов недвижимости;
– Окно с просмотром объектов недвижимости;
– Окно с добавлением сотрудников;
– Окно с просмотром сотрудников;
– Окно с добавлением договоров продажи;
– Окно с просмотром договоров продажи;

– Окно с добавлением договоров аренды;
– Окно с просмотром договоров аренды.

На рисунке 2.2 представлен интерфейс приложения.

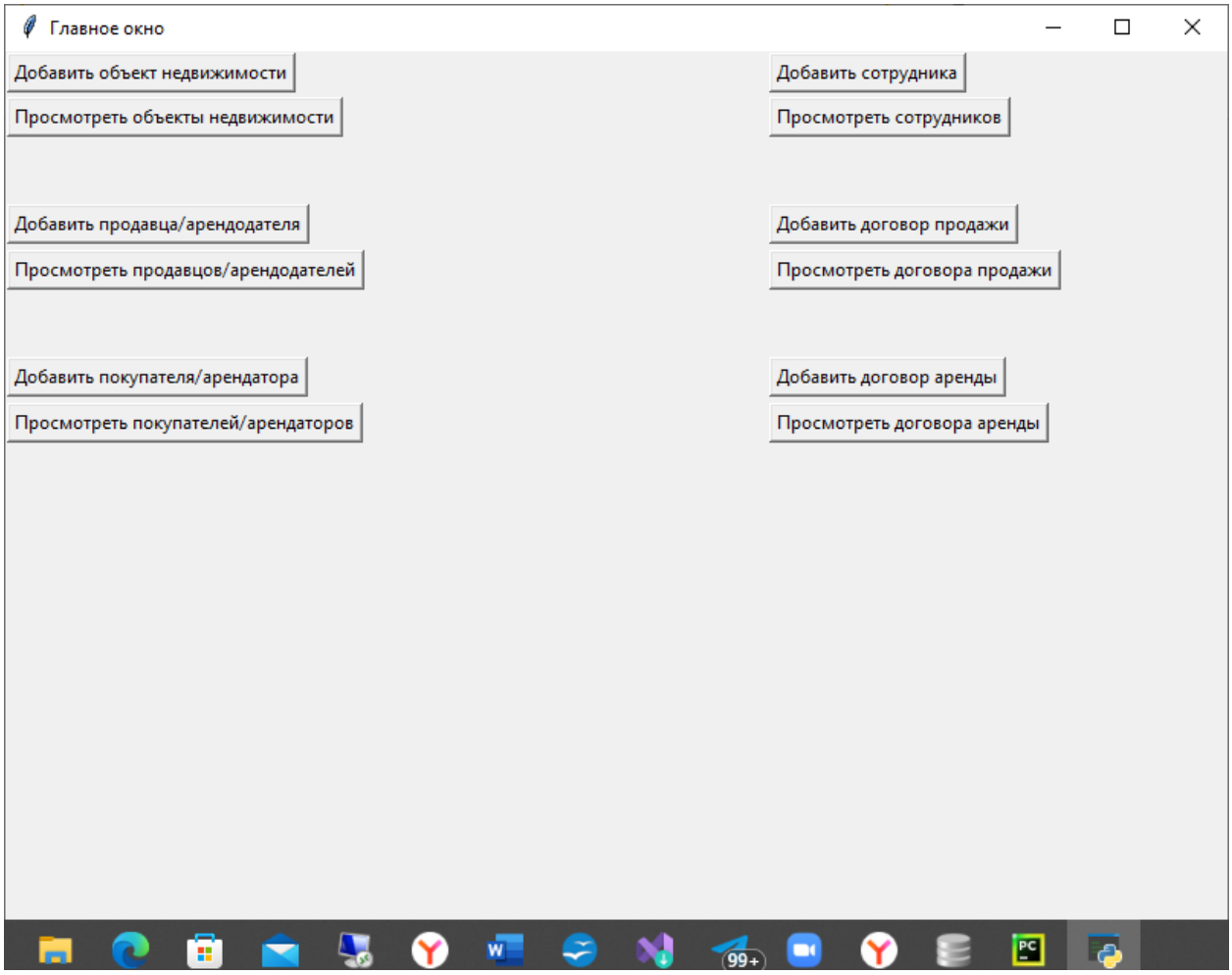

Рисунок 2.2 – Интерфейс приложения

2.3.4 Нефункциональные требования

2.3.4.1 Требования к безопасности

Необходимо устранить уязвимости, возможные для приложений:

– роли пользователей: Установить разные роли (администратор, агент, клиент) с соответствующими правами доступа;

– защита от атак с перебором паролей: Внедрить блокировку учетной записи после нескольких неудачных попыток входа;

– шифрование данных: Использовать алгоритмы шифрования для хранения конфиденциальных данных, таких как пароли и личная информация;

– защита передачи данных: Все данные, передаваемые по сети, должны быть защищены с помощью HTTPS;

– защита от SQL-инъекций: Использовать технологии ORM (Object-Relational Mapping) и параметризованные запросы для защиты от SQL-инъекций;

– ведение журналов: Реализовать механизмы ведения журналов действий пользователей и администраторов для последующего анализа;

– управление доступом к данным: Ограничить доступ пользователей к конфиденциальной информации (например, информации о других пользователях, сделках);

– регулярные обновления: Обеспечить своевременное обновление всех библиотек и зависимостей для защиты от известных уязвимостей;

– безопасность сервера: Хранить серверы в защищенных помещениях с ограниченным доступом;

– резервное копирование: Регулярно делать резервные копии базы данных и важных данных;

– обучение сотрудников: Проводить тренинги по вопросам безопасности для пользователей и сотрудников агентства;

– соблюдение законодательства: Соблюдать местное и международное законодательство по защите персональных данных (такие как GDPR).

2.3.4.2 Требования к программному обеспечению

Для создания программного решения потребуется подготовить ряд ключевых элементов:

– фреймворк Laravel, обеспечивающий структуру и упрощающий разработку;

– среду исполнения Python, необходимую для запуска кода;

– систему управления базами данных.

Laravel демонстрирует широкую совместимость, работая на актуальных операционных системах, таких как Windows, macOS и Linux.

2.3.4.3 Требования к аппаратному обеспечению

Для работы приложения требуется дисковое пространство не менее 1 Гб. Рекомендуется использовать процессор с 2 или более ядрами и частотой 2 ГГц или выше.

2.4 Требования к оформлению документации

Стадии разработки программного обеспечения и требования к программной документации для вычислительной техники, комплексов и систем любого назначения и области применения регламентируются ГОСТ 19.102–77. В состав программной документации входят:

- анализ предметной области;
- техническое задание;
- технический проект;
- рабочий проект.

3 Технический проект

3.1 Выбор технологии проектирования

3.1.1 Паттери MVC

MVC (Model-View-Controller) — это широко распространённый архитектурный шаблон, используемый при разработке графических пользовательских интерфейсов (GUI) и веб-приложений. Он разделяет приложение на три взаимосвязанных компонента: модель, представление и контроллер. Такое разделение позволяет упорядочить код, упростить разработку, тестирование и поддержку, а также повысить эффективность повторного использования компонентов. Рассмотрим MVC как архитектурный шаблон для разработки, выделив его сильные и слабые стороны, а также области применения.

3.2 Выбор средств разработки

3.2.1 Python

Python – это высокоуровневый, интерпретируемый, объектно-ориентированный язык программирования, который выделяется своей читаемостью и простотой. Его универсальность и богатая экосистема делают его отличным выбором для широкого спектра задач разработки. Рассмотрим Python с точки зрения выбора средств разработки, выделив его сильные и слабые стороны, а также области применения.

Преимущества Python для разработки:

– читаемость и простота синтаксиса: Python разработан с акцентом на читаемость кода. Его синтаксис интуитивно понятен, что упрощает обучение и понимание кода, особенно для начинающих разработчиков. Меньше времени тратится на разбор сложных конструкций, позволяя сосредоточиться на логике программы. Это снижает порог вхождения и ускоряет процесс разработки;

– большая и активная экосистема библиотек и фреймворков: Python обладает огромным количеством библиотек и фреймворков для решения практически любой задачи. Это позволяет разработчикам использовать готовые решения вместо написания кода с нуля, что значительно экономит время и ресурсы. Примеры:

– NumPy, SciPy, Pandas: для научных вычислений и анализа данных;

– Django, Flask: для веб-разработки;

– TensorFlow, PyTorch: для машинного обучения и искусственного интеллекта;

– Requests: для работы с HTTP-запросами;

– Beautiful Soup, Scrapy: для парсинга веб-страниц;

– кроссплатформенность: Python работает на различных операционных системах (Windows, macOS, Linux и др.), что обеспечивает гибкость при разработке и развертывании приложений. Код, написанный на Python, может быть легко перенесен на другую платформу без значительных изменений;

– поддержка различных парадигм программирования: Python поддерживает объектно-ориентированное, императивное и функциональное программирование, что позволяет разработчикам выбирать наиболее подходящий стиль для решения конкретной задачи;

– интерпретируемость: Python – интерпретируемый язык, что означает, что код выполняется построчно без предварительной компиляции. Это упрощает отладку и позволяет быстро вносить изменения в код;

– большое сообщество и широкая поддержка: Python имеет огромное и активное сообщество разработчиков, которые предоставляют помощь, создают библиотеки и фреймворки, а также активно участвуют в развитии языка. Это обеспечивает широкую поддержку и доступность ресурсов для решения возникающих проблем;

– быстрая разработка (Rapid Prototyping): благодаря простоте синтаксиса и доступности библиотек Python идеально подходит для быстрой разработки прототипов и MVP (минимально жизнеспособного продукта).

Недостатки Python для разработки:

– производительность: Python, как интерпретируемый язык, часто уступает в производительности компилируемым языкам, таким как C++ или Java. Это может быть критично для задач, требующих высокой вычислительной мощности или низкой задержки. Однако для многих задач разница в производительности не является решающим фактором, и преимущества Python в скорости разработки перевешивают этот недостаток. Кроме того, можно использовать библиотеки, написанные на C/C++, для оптимизации критических участков кода;

– глобальная блокировка интерпретатора (GIL): GIL ограничивает возможность параллельного выполнения кода Python в многопоточных приложениях, что может снижать производительность на многоядерных процессорах. Однако для задач, связанных с вводом-выводом (I/O), GIL не является существенным ограничением, и для достижения параллелизма можно использовать многопроцессорность или асинхронное программирование;

– веб-разработка: создание веб-приложений и API с использованием фреймворков, таких как Django и Flask;

– анализ данных и машинное обучение: обработка и анализ больших объемов данных, построение моделей машинного обучения с использованием библиотек, таких как NumPy, SciPy, Pandas, Scikit-learn, TensorFlow и PyTorch;

– автоматизация и скрипты: написание скриптов для автоматизации рутинных задач, администрирования систем и сетей;

– тестирование: автоматизация тестирования программного обеспечения с использованием фреймворков, таких как pytest и unittest;

– разработка игр: создание игр с использованием библиотек, таких как Pygame;

– научные вычисления: моделирование физических процессов, обработка изображений и сигналов с использованием библиотек, таких как NumPy, SciPy и Matplotlib;

– DevOps: Автоматизация процессов развертывания, мониторинга и управления инфраструктурой.

Python в сравнении с другими языками:

– Python против Java: Python обычно проще в изучении и использовании, чем Java, но Java может быть более производительной для некоторых задач. Java также имеет более развитую систему статической типизации, которая помогает обнаруживать ошибки на этапе компиляции;

– Python против C++: C++ обеспечивает более высокую производительность, чем Python, но разработка на C++ требует больше времени и усилий. Python часто используется для создания прототипов и быстрого решения задач, а C++ — для задач, требующих максимальной производительности;

– Python против JavaScript: JavaScript является основным языком для разработки веб-интерфейсов, а Python чаще используется для серверной части веб-приложений. Однако с появлением Node.js JavaScript также можно использовать для разработки серверной части.

Python — это мощный и универсальный язык программирования, который отлично подходит для широкого спектра задач разработки. Его читаемость, простота и богатая экосистема позволяют учитывать специфику проекта и требования к производительности, но в большинстве случаев Python будет отличным выбором.

3.2.2 pgAdmin4

pgAdmin 4 — это современный и многофункциональный инструмент администрирования и разработки для СУБД PostgreSQL. Это один из самых популярных и широко используемых графических пользовательских интерфейсов (GUI) для работы с PostgreSQL, предоставляющий удобный и интуитивно понятный интерфейс для выполнения различных задач, от администрирования и мониторинга до разработки и отладки. Рассмотрим pgAdmin 4 с точки зрения выбора средств разработки, подчеркнув его сильные и слабые стороны, а также целевую аудиторию.

Преимущества pgAdmin 4 для разработки:

– кроссплатформенность: pgAdmin 4 можно запускать как веб-приложение в браузере, что обеспечивает кроссплатформенность и позволя-

ет использовать его в различных операционных системах (Windows, macOS, Linux и др.) без необходимости установки дополнительных компонентов;

– удобный и интуитивно понятный интерфейс: интерфейс pgAdmin 4 разработан с учетом удобства пользователя. Он предоставляет визуальные инструменты для управления серверами, базами данных, таблицами, представлениями, функциями и другими объектами PostgreSQL. Навигация по интерфейсу интуитивно понятна, что облегчает выполнение различных задач;

– мощный редактор SQL: pgAdmin 4 оснащен мощным редактором SQL с подсветкой синтаксиса, автодополнением кода и возможностью выполнения нескольких запросов одновременно. Редактор также предоставляет инструменты для отладки SQL-кода и анализа планов запросов;

– визуальные инструменты для управления базами данных: pgAdmin 4 предоставляет визуальные инструменты для создания, изменения и удаления баз данных, таблиц, представлений, функций и других объектов PostgreSQL. Это упрощает процесс проектирования и разработки баз данных, особенно для начинающих разработчиков;

– поддержка расширений PostgreSQL: pgAdmin 4 поддерживает расширения PostgreSQL, что позволяет разработчикам использовать дополнительные функциональные возможности СУБД;

– мониторинг производительности: pgAdmin 4 предоставляет инструменты для мониторинга производительности PostgreSQL, что позволяет выявлять узкие места и оптимизировать работу базы данных;

– управление пользователями и ролями: pgAdmin 4 упрощает управление пользователями и ролями PostgreSQL, что позволяет контролировать доступ к данным и обеспечивать безопасность базы данных;

– резервное копирование и восстановление данных: pgAdmin 4 предоставляет инструменты для резервного копирования и восстановления данных PostgreSQL, что обеспечивает защиту данных от потери;

– бесплатность и открытый исходный код: pgAdmin 4 — это бесплатный инструмент с открытым исходным кодом, что позволяет использовать его без ограничений и модифицировать в соответствии с потребностями.
pgAdmin 4 — отличный выбор для разработки и администрирования баз данных PostgreSQL. Он предоставляет удобный и интуитивно понятный интерфейс с широким набором инструментов для выполнения различных задач. pgAdmin 4 особенно полезен для разработчиков, администраторов баз данных и аналитиков данных, работающих с PostgreSQL. Хотя существуют и другие альтернативные инструменты, pgAdmin 4 остается одним из самых популярных и востребованных клиентов для PostgreSQL благодаря своей функциональности, кроссплатформенности и бесплатному доступу. При выборе инструмента следует учитывать требования к ресурсам, уровень знаний и специфические потребности проекта.

3.2.3 Фреймворк Laravel

Laravel – это бесплатный, с открытым исходным кодом PHP-фреймворк, предназначенный для разработки современных веб-приложений, следующих архитектурному шаблону MVC (Model-View-Controller). Известный своей элегантностью, выразительностью и богатым набором встроенных инструментов, Laravel значительно упрощает и ускоряет процесс разработки, делая его популярным выбором среди PHP-разработчиков. Рассмотрим Laravel с точки зрения выбора средств разработки, выделив его сильные и слабые стороны, а также целевую аудиторию.

Преимущества Laravel для разработки:

– элегантный синтаксис и выразительность: Laravel известен своим чистым и выразительным синтаксисом, который делает код легко читаемым и поддерживаемым. Это снижает когнитивную нагрузку на разработчиков и позволяет им быстрее понимать и изменять код;

– MVC-архитектура: Laravel использует архитектурный шаблон MVC, который разделяет приложение на три основных компонента: модель (данные), представление (интерфейс пользователя) и контроллер (логика прило-

жения). Это облегчает организацию кода, повторное использование компонентов и тестирование;

– встроенная система маршрутизации: Laravel предоставляет мощную и гибкую систему маршрутизации, которая позволяет легко определять правила обработки HTTP-запросов и связывать их с соответствующими контроллерами;

– шаблонизатор Blade: Blade – это простой, но мощный шаблонизатор в Laravel, который позволяет создавать динамические веб-страницы с использованием PHP-кода и специальных директив. Blade обеспечивает наследование шаблонов, компоненты и другие полезные функции;

– миграции баз данных: Laravel предоставляет систему миграций баз данных, которая позволяет легко создавать и изменять структуру базы данных, а также отслеживать изменения в ее истории. Миграции упрощают развертывание приложения на различных средах;

– Artisan Console: Artisan – это консольная утилита Laravel, которая предоставляет множество полезных команд для автоматизации рутинных задач, таких как создание контроллеров, моделей, миграций, генерация кода и многое другое.

– тестирование: Laravel разработан с учетом тестирования и предоставляет встроенные инструменты для написания модульных и интеграционных тестов;

– безопасность: Laravel предоставляет встроенные инструменты для защиты от распространенных веб-угроз, таких как CSRF (Cross-Site Request Forgery), XSS (Cross-Site Scripting) и SQL-инъекции;

– авторизация и аутентификация: Laravel упрощает реализацию систем авторизации и аутентификации пользователей, предоставляя готовые компоненты для регистрации, входа в систему, сброса пароля и т.д.;

– очереди: Laravel предоставляет систему очередей для обработки задач в фоновом режиме, что позволяет разгрузить основной поток приложения и улучшить его отзывчивость;

– активное сообщество и документация: Laravel имеет большое и активное сообщество разработчиков, которые создают пакеты, делятся знаниями и оказывают помощь. Официальная документация Laravel хорошо написана и содержит множество примеров;

– пакеты (Composer): расширяемость за счет использования Composer и обширной экосистемы пакетов.

Недостатки Laravel для разработки:

– изучение: хотя Laravel имеет элегантный синтаксис, изучение фреймворка может потребовать времени, особенно для начинающих PHP-разработчиков. Необходимо освоить концепции MVC, ORM, шаблонизатора и другие компоненты Laravel;

– производительность: Laravel, как и другие PHP-фреймворки, может уступать в производительности компилируемым языкам, таким как Java. Однако для большинства веб-приложений разница в производительности не является критической, и можно принять меры для оптимизации приложений на Laravel;

– размер: Laravel — довольно большой фреймворк, что может привести к увеличению размера приложения и времени загрузки.

Laravel — отличный выбор для PHP-разработчиков, которые хотят создавать современные, элегантные и масштабируемые веб-приложения. Его выразительный синтаксис, богатый набор встроенных инструментов и активное сообщество делают его одним из самых популярных PHP-фреймворков в мире. При выборе Laravel стоит учитывать требования к производительности, сложность проекта и опыт команды разработчиков. Для проектов, требующих высокой производительности или глубокой интеграции с существующей инфраструктурой, могут подойти другие фреймворки. Но в целом Laravel обеспечивает отличный баланс между функциональностью, простотой использования и производительностью.

3.3 Архитектура программной системы

Система состоит из следующих основных компонентов:

3.3.1 Клиент-Приложение

Описание: Этот компонент объединяет уровень представления (пользовательский интерфейс) и уровень приложений (бизнес-логику). Он отвечает за взаимодействие с пользователем, обработку запросов и передачу данных в базу данных.

ТЕХНОЛОГИИ:

Python: Основной язык программирования для реализации приложения.

psycopg2: Библиотека Python для подключения и взаимодействия с сервером PostgreSQL.

Tkinter: для создания простого графического интерфейса (GUI) или текстового интерфейса.

ИЕТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ:

Отображение данных о недвижимости, клиентах и сделках в виде таблицы или списка.

Предоставление форм для добавления, редактирования и удаления данных.

Реализация функциональности поиска и фильтрации данных по различным критериям.

Навигация по данным.

БИЗНЕС ЛОГИКА:

Получение данных от пользователя.

Проверка введенных данных (например, проверка формата даты, типа данных и т. д.).

Формирование SQL-запросов для взаимодействия с базой данных.

Обработка результатов запросов, полученных от базы данных.

Вывод результатов на экран.

3.3.2 Уровень данных

Описание: Отвечает за хранение и управление данными системы.

Технологии:

PostgreSQL: выбрана в качестве СУБД. PostgreSQL — мощная объектно-реляционная СУБД с открытым исходным кодом, отличающаяся высокой надежностью, производительностью и поддержкой стандартов SQL.

ФУНКЦИОНАЛЬНОСТЬ:

Хранение данных о недвижимости, клиентах и сделках в структурированном виде.

Обеспечение целостности данных с использованием ограничений, ключей и транзакций.

Предоставление доступа к данным через язык SQL.

Индексирование данных для ускорения поиска.

3.3.3 Технологии и инструменты разработки

Язык программирования: Python СУБД: PostgreSQL

Python-библиотека для PostgreSQL: psycopg2

GUI-библиотека (опционально): Tkinter

Инструменты разработки: IDE (PyCharm), система контроля версий (Git, GitHub)

3.3.4 Функциональность

Операции CRUD: реализация функций для создания, чтения, обновления и удаления данных в таблицах realty, client и owner.

Поиск и фильтрация: реализация возможности поиска объектов недвижимости по различным критериям (например, по типу, адресу, цене).

Вывод данных: отображение данных в удобном формате (таблица, список).

Добавление сделок: реализация возможности добавления информации о сделках в таблицу.

3.3.5 Перспективы развития

Реализация более продвинутых функций поиска и фильтрации данных.

Интеграция с внешними сервисами (например, картографическими сервисами).

3.4 Структура базы данных

На основе анализа неформального описания предметной области были определены наборы объектов и связей.

ОПРЕДЕЛЕНИЯ ОБЪЕКТОВ

Потенциальные объекты и атрибуты:

Каждый объект недвижимости характеризуется следующими параметрами:

- код объекта недвижимости;
- тип сделки;
- регион;
- город;
- улица;
- номер дома;
- номер квартиры (если есть);
- площадь, кв.м;
- кол-во комнат;
- срок сдачи (если сдается);
- цена.

Информация о владельце объекта недвижимости включает следующее:

- ФИО;
- паспортные данные;
- контактные данные;
- ID недвижимости.

Информация о покупателе/арендаторе включает следующее:

- ФИО;
- паспортные данные;
- контактные данные.

Информация о сотруднике агентства включает следующее:

– ФИО;
– паспортные данные;
– контактные данные.

Каждая сделка по покупке/аренде характеризуется следующими параметрами:

– номер сделки сделки;
– код объекта недвижимости;
– данные владельца;
– данные покупателя/арендатора;
– данные сотрудника агентства;
– тип сделки;
– срок аренды (если есть);
– цена;
– дата сделки.

Сущности и отношения между ними отображены на ER-диаграмме.

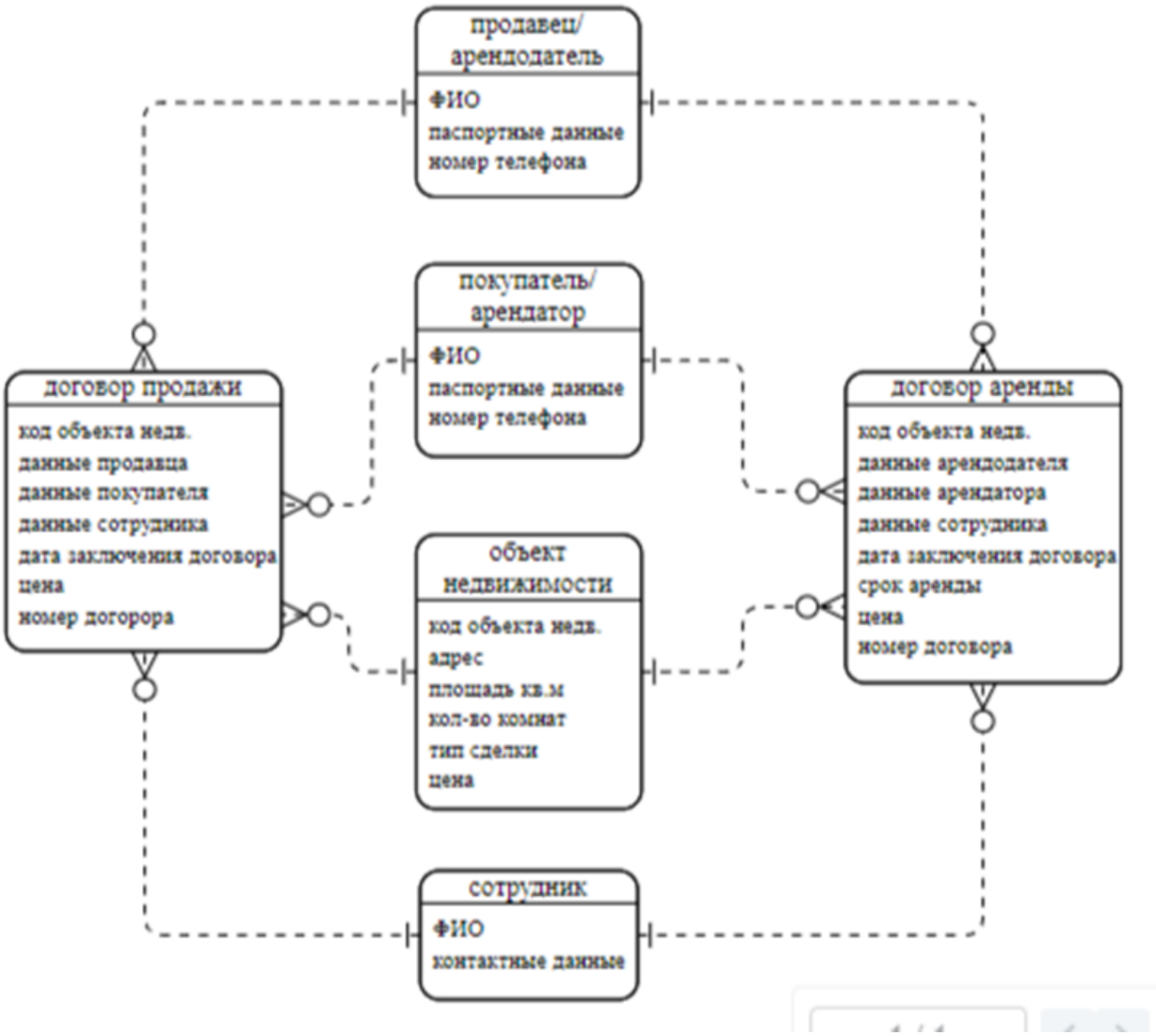

Рисунок 3.1 – ER-диаграмма

3.5 Логическое проектирование базы данных

3.5.1 Нормализация ER-модели данных

Первая нормальная форма: каждый атрибут данной модели имеет только одно значение.

Вторая форма: каждый атрибут зависит только от полного UID объекта.

Третья форма: каждый атрибут зависит только от UID своего объекта.

Объект «Продавец/арендодатель» соответствует 1NF, так как каждый атрибут имеет только одно значение.

Объект «Продавец/арендодатель» соответствует 2NF, так как каждый атрибут зависит от полного UID данного объекта.

Объект «Продавец/арендодатель» соответствует 3NF, так как каждый атрибут зависит только от UID данного объекта.

Объект «Покупатель/арендатор» соответствует 1NF, так как каждый атрибут имеет только одно значение.

Объект «Покупатель/арендатор» соответствует 2NF, так как каждый атрибут зависит от полного UID данного объекта.

Объект «Покупатель/арендатор» соответствует 3NF, так как каждый атрибут зависит только от UID данного объекта.

Объект «Объект недвижимости» соответствует 1NF, так как каждый атрибут имеет только одно значение.

Объект «Объект недвижимости» соответствует 2NF, так как каждый атрибут зависит от полного UID данного объекта.

Объект «Объект недвижимости» соответствует 3NF, так как каждый атрибут зависит только от UID данного объекта.

Объект «Сотрудник» соответствует 1NF, так как каждый атрибут имеет только одно значение.

Объект «Сотрудник» соответствует 2NF, так как каждый атрибут зависит от полного UID данного объекта.

Объект «Сотрудник» соответствует 3NF, так как каждый атрибут зависит только от UID данного объекта.

Объект «Договор продажи» соответствует 1NF, так как каждый атрибут имеет только одно значение.

Объект «Договор продажи» соответствует 2NF, так как каждый атрибут зависит от полного UID данного объекта.

Объект «Договор продажи» соответствует 3NF, так как каждый атрибут зависит только от UID данного объекта.

Объект «Договор аренды» соответствует 1NF, так как каждый атрибут имеет только одно значение.

На основе ER-модели данных в онлайн-сервисе Lucidcharts построена реляционная модель данных, показанная на рисунке 3.2.

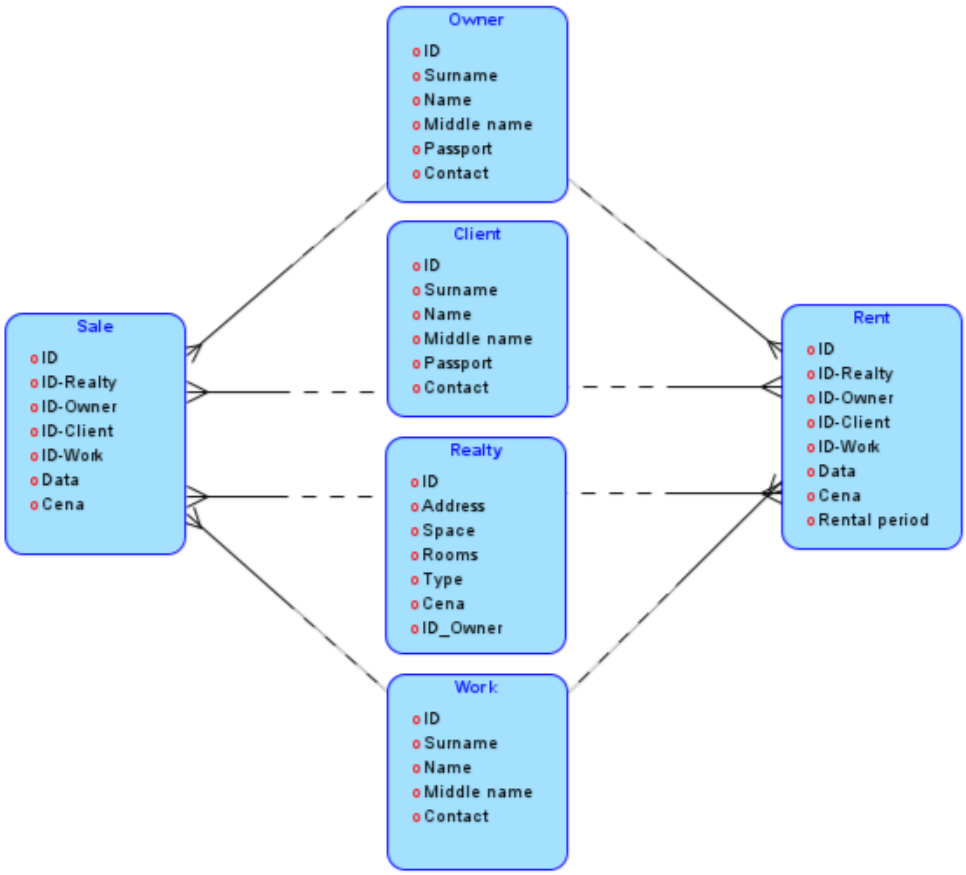

Рисунок 3.2 – Реляционная модель данных

Названия таблиц и названия столбцов каждой таблицы приведены в таблицах 3.1 -3.6.

Таблица 3.1 – Владелец объекта недвижимости

Key Type	Optionality	Column name	Data type
1	2	3	4
pk	*	Surname	TEXT
	*	Name	TEXT
	*	Middle name	TEXT
	*	Passport	INTEGER
	*	Contact	INTEGER

Таблица 3.2 – Клиент

Key Type	Optionality	Column name	Data type
1	2	3	4
pk	*	Surname	TEXT
	*	Name	TEXT
	*	Middle name	TEXT
	*	Passport	INTEGER
	*	Contact	INTEGER

Таблица 3.3 – Объект недвижимости

Key Type	Optionality	Column name	Data type
1	2	3	4
pk	*	Address	TEXT
	*	Space	TEXT
	*	Rooms	INTEGER
	*	Type	TEXT
	*	Cena	INTEGER
	*	ID owner	INTEGER

Таблица 3.4 – Сотрудник агентства

Key Type	Optionality	Column name	Data type
1	2	3	4
pk	*	Surname	TEXT
	*	Name	TEXT
	*	Middle name	TEXT
	*	Contact	INTEGER

Таблица 3.5 – Договор продажи

Key Type	Optionality	Column name	Data type
1	2	3	4
pk	*	Realty	TEXT
	*	Owner	TEXT
	*	Client	TEXT
	*	Work	TEXT
	*	Data	DATE
	*	Cena	INTEGER

Таблица 3.6 – Договор аренды

Key Type	Optionality	Column name	Data type
1	2	3	4
pk	*	Realty	TEXT
	*	Owner	TEXT
	*	Client	TEXT
	*	Work	TEXT
	*	Data	DATE
	*	Cena	INTEGER
	*	Rental period	TEXT

4 Рабочий проект

4.1 Описание классов и функций

```
def open_add_window(): Открывает новое окно (Toplevel) для добавления информации о недвижимости в базу данных.

def add_realty(): Добавляет информацию об объекте недвижимости в базу данных. Эта функция определена внутри

open_add_window() и имеет доступ к виджетам этого окна.

def open_salee_window(): открывает новое окно (Toplevel) для просмотра данных о продажах из таблицы sale базы данных.

def create_agreement(): создает договор купли-продажи на основе выбранной записи в списке продаж.

def open_view_window(): эта функция отвечает за создание и отображение окна просмотра объектов недвижимости. Она извлекает данные об объектах из базы данных и отображает их в окне с возможностью прокрутки.

def open_work_window(): эта функция создает окно для добавления информации о новом сотруднике.

def add_employee(): эта функция отвечает за получение данных о новом сотруднике из полей ввода, проверку (хотя бы минимальную) и добавление этой информации в таблицу employee в базе данных.


def open_worker_window(): эта функция отвечает за создание и отображение окна для просмотра списка сотрудников, хранящегося в базе данных.

def open_buy_window(): эта функция отвечает за создание окна, предназначенного для добавления информации о новом покупателе (или арендаторе). Она создает элементы интерфейса, необходимые для ввода данных о покупателе.

def add_client(): эта функция предназначена для добавления информации о новом клиенте (покупателе/арендаторе) в базу данных. Она собирает данные из полей ввода, выполняет минимальную проверку и добавляет запись в таблицу client.

def open_buyer_window(): эта функция открывает окно для просмотра списка клиентов (покупателей/арендаторов), извлекая данные из базы данных и отображая их.

def open_sell_window(): эта функция отвечает за создание окна, в котором пользователь может ввести данные о новом владельце (продавце/арендодателе) недвижимости.

def open_seller_window(): эта функция создает окно для просмотра списка владельцев (продавцов/арендодателей) недвижимости, извлекая данные из базы данных и отображая их в виде списка.
```

```
29
30 def open_sale_window(): эта функция отвечает за создание окна для ввода
       данных о новой продаже (заключении договора купли-продажи) и добавления
       этой информации в базу данных.
31
32 def add_sale(): Определяет функцию, которая будет вызываться при нажатии
       кнопки «Добавить». Эта функция собирает данные из полей ввода, проверяет
       их и добавляет новую запись в таблицу sale в базе данных.
33
34 def generate_sale_agreement(row): эта функция генерирует текст договора купли
       -продажи на основе данных, полученных из базы данных. Она принимает строку
        данных (row) в качестве аргумента и формирует текстовое представление
       договора, готовое к отображению или печати.
35
36 def open_salee_window(): эта функция отвечает за создание окна для просмотра
       информации о существующих договорах купли-продажи. Она извлекает данные о
       продажах из базы данных, отображает их в виде списка и предоставляет
       возможность сгенерировать текст договора для выбранной продажи.
37
38 def save_agreement(): эта функция позволяет сохранить сгенерированный текст
       договора (хранящийся в переменной agreement) в текстовый файл, выбранный
       пользователем с помощью диалогового окна сохранения файла.
39
40 def open_rent_window(): эта функция отвечает за создание окна,
       предназначенного для ввода информации о договоре аренды.
41
42 def add_renta(): эта функция предназначена для добавления данных о новом
       договоре аренды в базу данных. Она собирает информацию из полей ввода в
       графическом интерфейсе, а затем вставляет эту информацию в таблицу renta.
       Также добавляет сообщение об успешном выполнении и закрывает окно.
43
44 def generate_rent_agreement(row): эта функция генерирует текст договора
       аренды на основе данных, полученных из строки базы данных. Функция
       формирует текст договора, который можно отобразить пользователю или
       сохранить в файл.
45
46 def open_rentt_window(): эта функция создает окно для просмотра договоров
       аренды. Она извлекает данные из базы данных и отображает их в виджете
       Listbox. Также предоставляет возможность сгенерировать договор аренды на
       основе выбранной записи.
47
48 save_agreement(): эта функция позволяет сохранить сгенерированный текст
       договора аренды (который хранится в переменной agreement) в текстовом
       файле. Пользователь может выбрать место и имя файла с помощью стандартного
        диалогового окна сохранения файла.
```

4.2 Тестирование разработанной программной системы

Для работы пользователей с созданной базой данных было разработано десктопное приложение для ОС Windows. Для разработки приложения использовался язык программирования Python с фреймворком Qt, в состав кото-

рого входят библиотеки для разработки графического интерфейса и средства взаимодействия с базами данных.

Структура приложения показана на рисунках 4.13 – 4.30.

На рисунке 4.1 представлено главное окно программы.

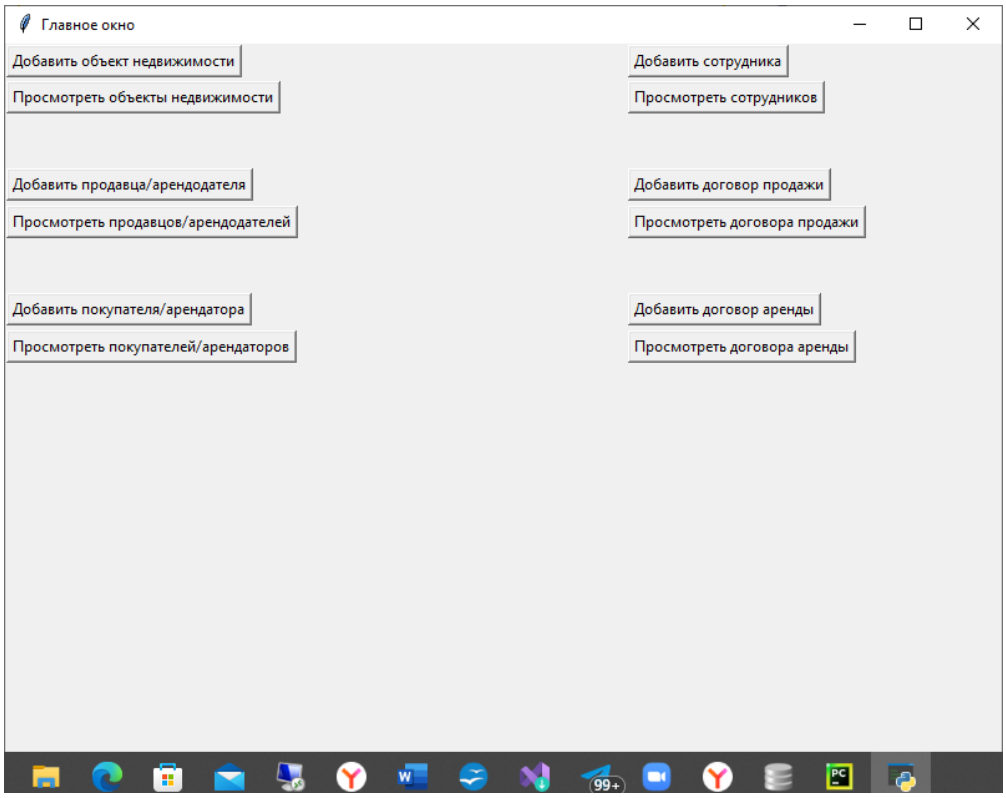

Рисунок 4.1 – Главное окно программы

На рисунке 4.2 представлено окно для добавления объектов недвижимости.

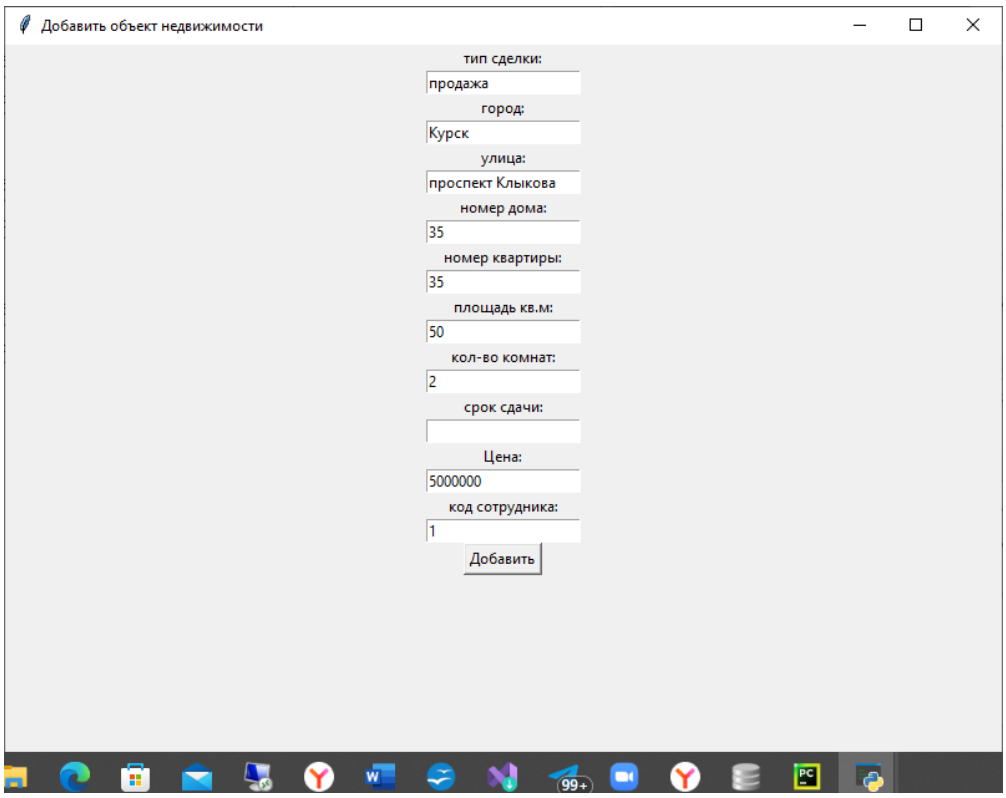

Рисунок 4.2 – Окно для добавления объектов недвижимости

На рисунке 4.3 представлен результат добавления объекта недвижимости.

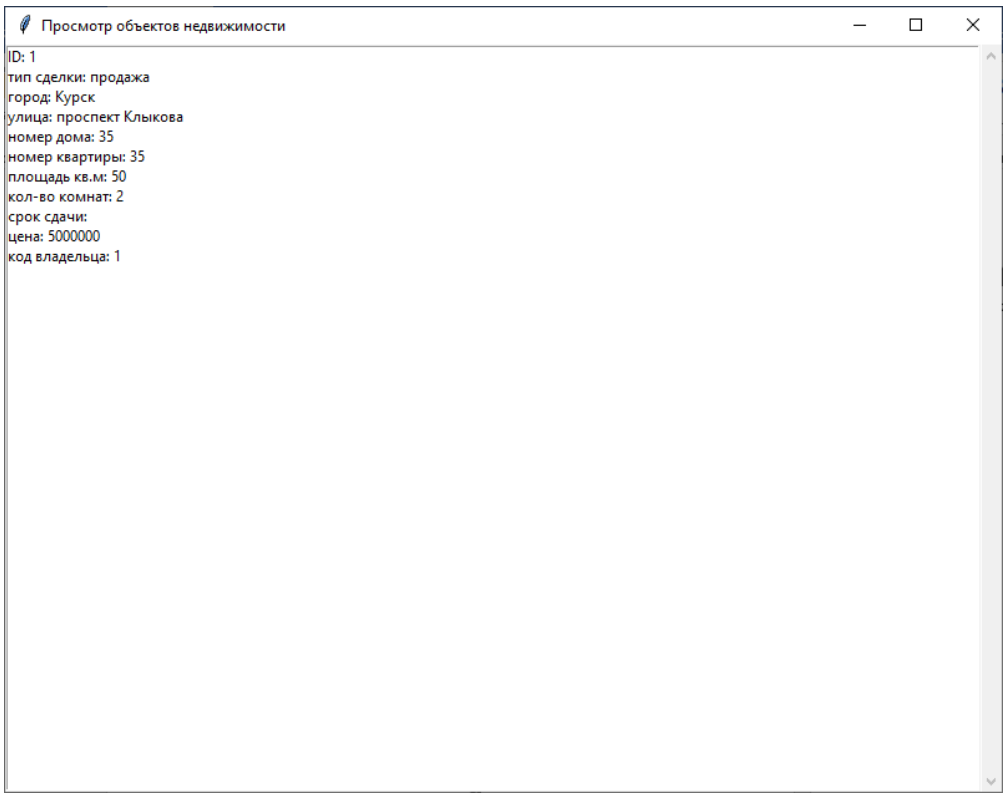

Рисунок 4.3 – Результат добавления объекта недвижимости

На рисунке 4.4 представлена попытка ввести некорректные данные в поле номер дома.

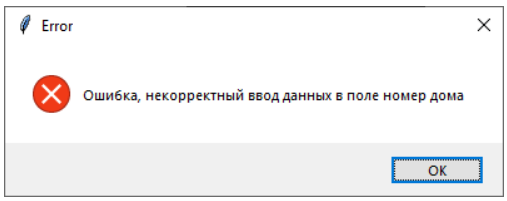

Рисунок 4.4 – Попытка ввести некорректные данные в поле номер дома

На рисунке 4.5 представлена попытка добавить данные, не заполнив все поля.

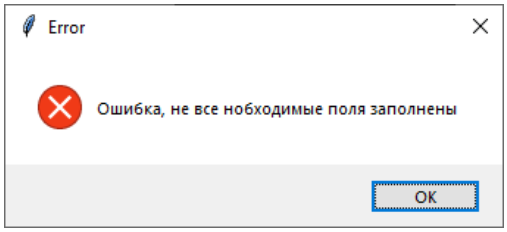

Рисунок 4.5 – Попытка добавить данные, не заполнив все поля

На рисунке 4.6 представлено окно для добавления данных продавца/арендадателя.

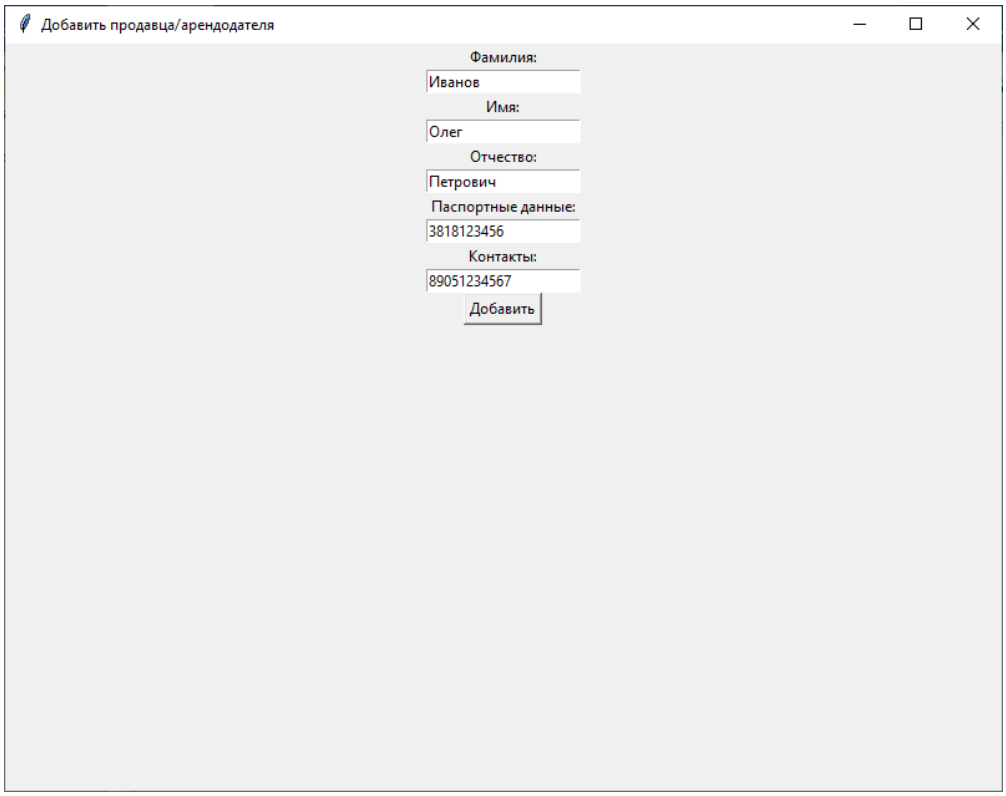

Рисунок 4.6 – Окно для добавления данных продавца/арендадателя

На рисунке 4.7 представлен результат добавления данных продавца/арендадателя.

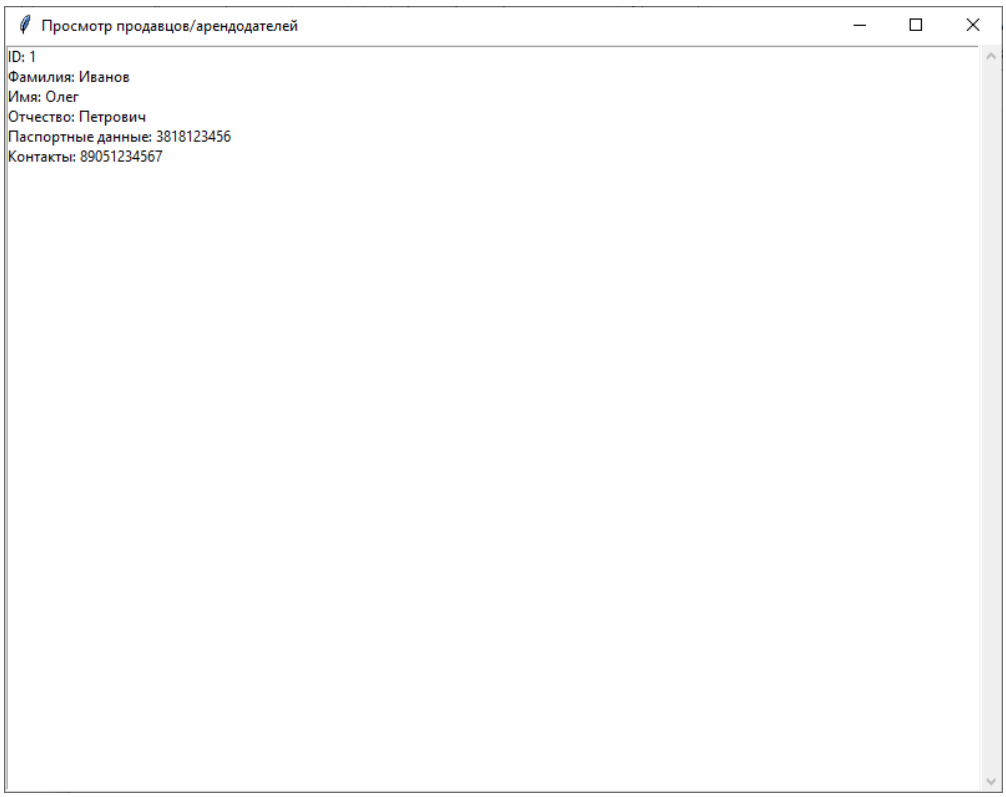

Рисунок 4.7 – Результат добавления данных продавца/арендадателя

На рисунке 4.8 представлено окно для добавления данных покупателя/арендатора.

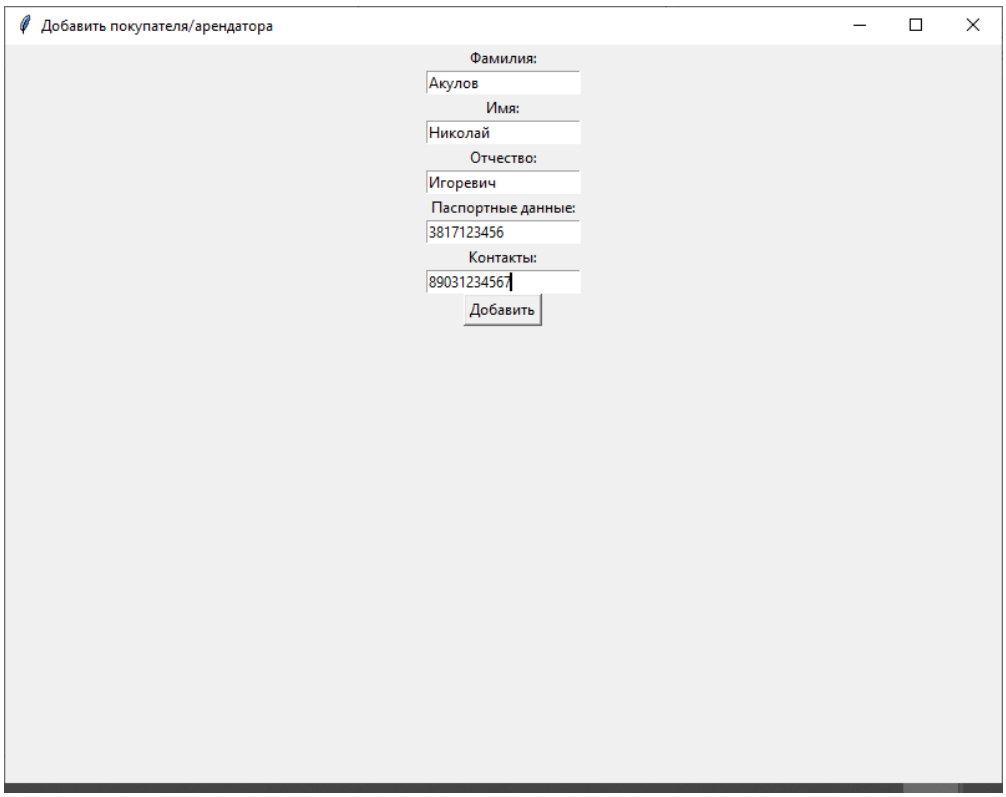

Рисунок 4.8 – Окно для добавления данных покупателя/арендатора

На рисунке **??** представлен результат добавления данных покупателя/арендатора.

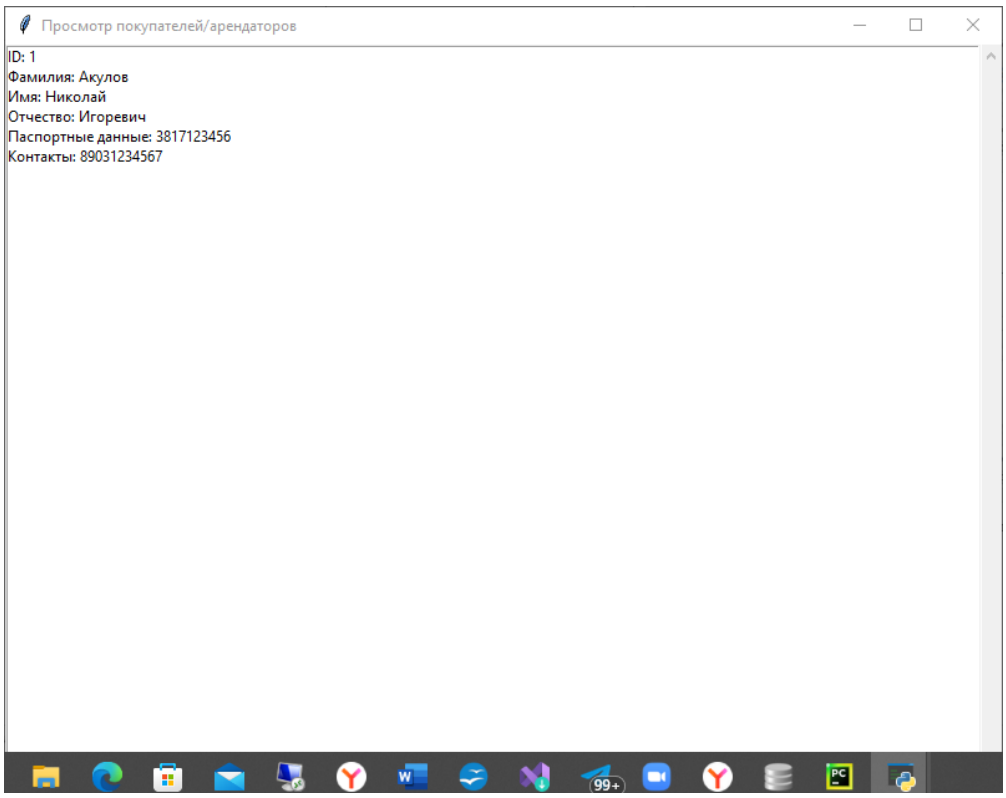

Рисунок 4.9 – Результат добавления данных покупателя/арендатора

На рисунке 4.10 представлено окно для добавления данных сотрудника.

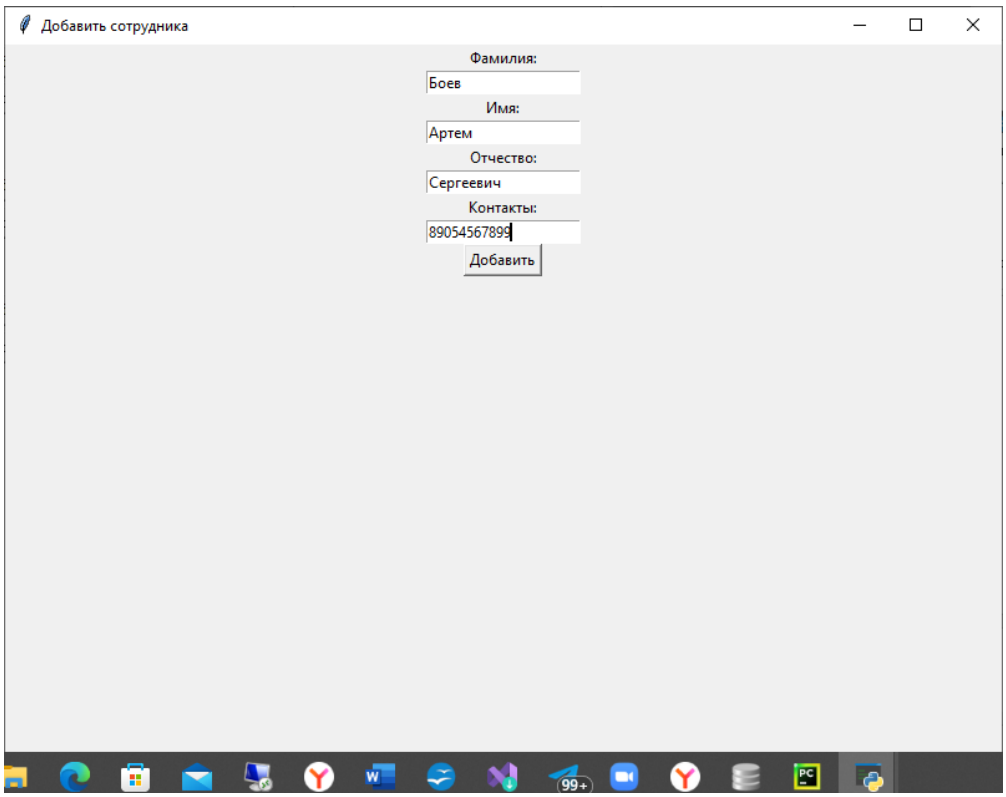

Рисунок 4.10 – Окно для добавления данных сотрудника

На рисунке 4.11 представлен результат добавления данных сотрудника.

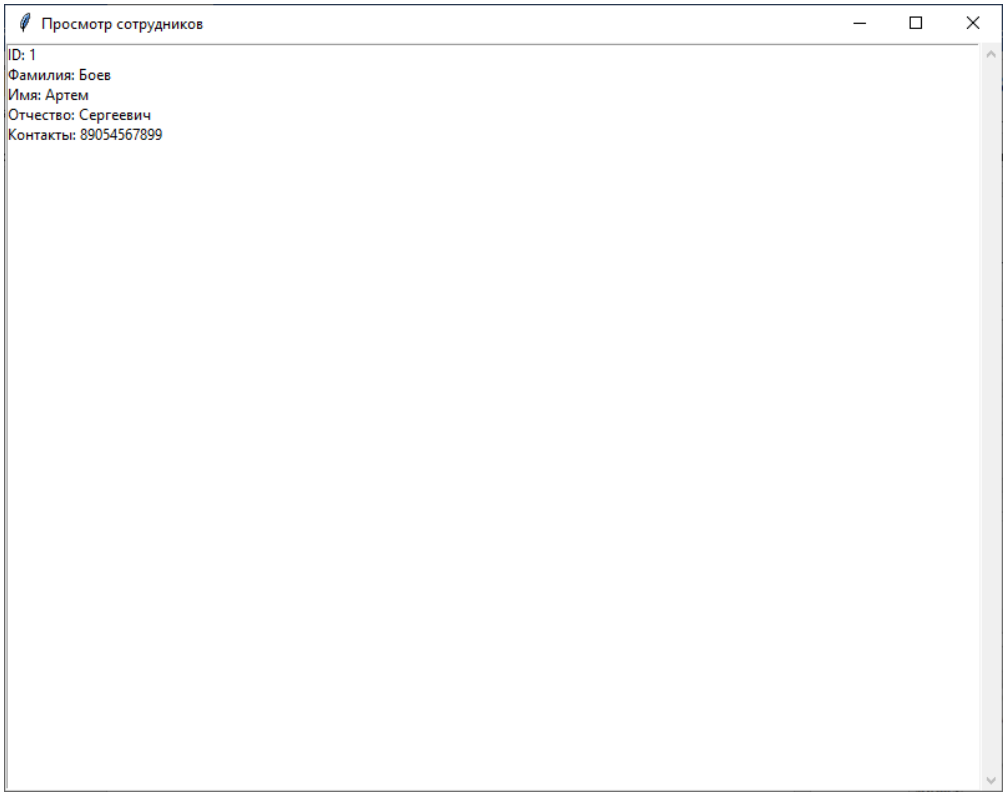

Рисунок 4.11 – Результат добавления данных сотрудника

На рисунке 4.12 представлено окно для добавления данных договора продажи.

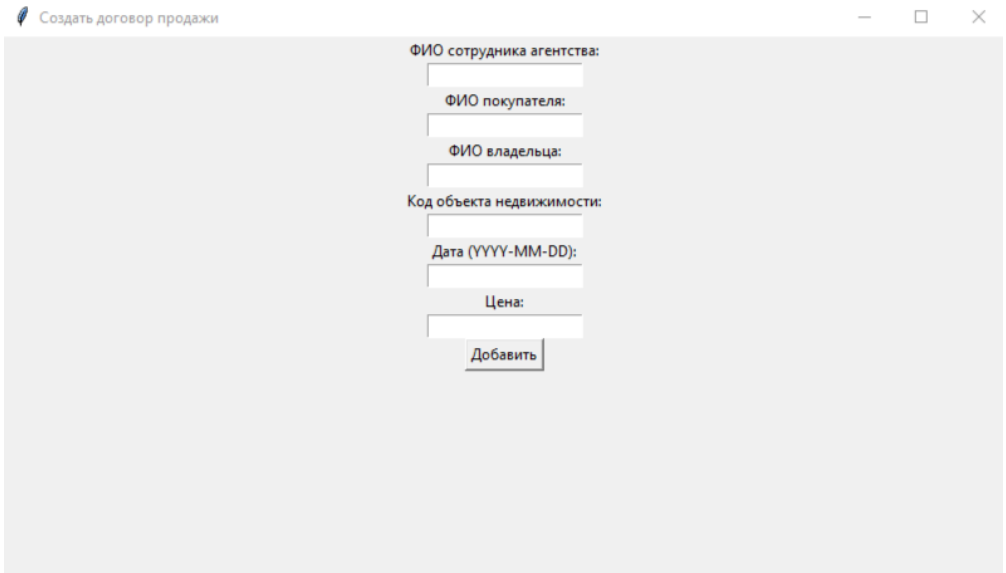

Рисунок 4.12 – Окно для добавления данных договора продажи

На рисунке 4.13 представлены данные для создания договора продажи.

Рисунок 4.13 – Данные для создания договора продажи

На рисунке 4.14 представлен созданный договор продажи.

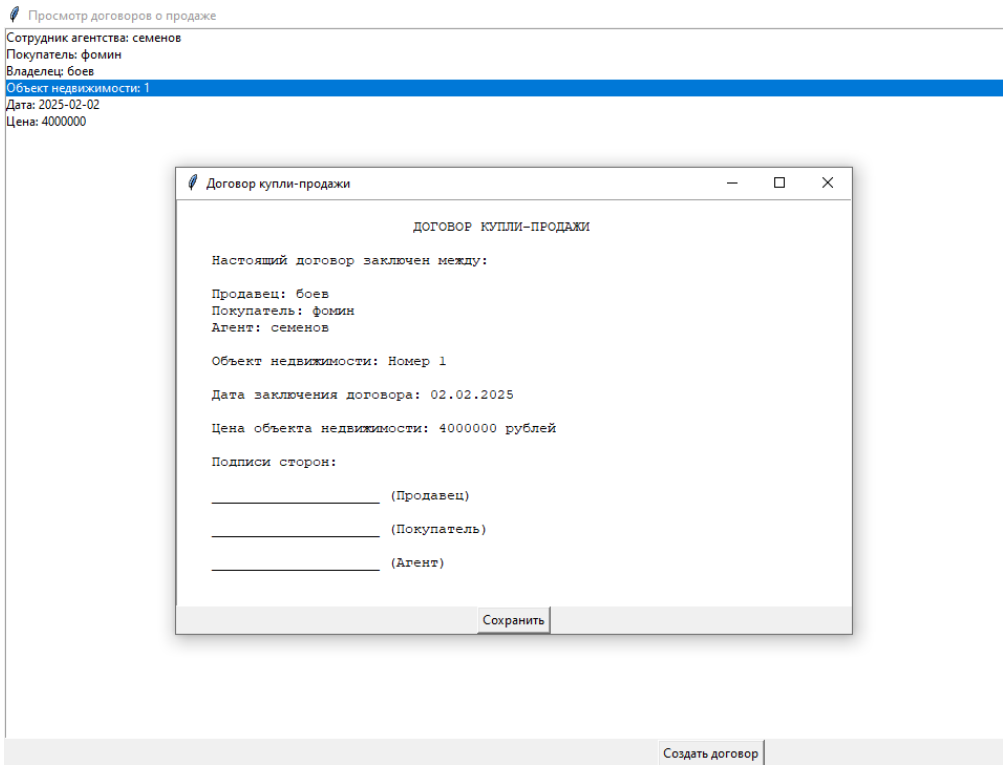

Рисунок 4.14 – Созданный договор продажи

На рисунке 4.15 представлено окно для добавления данных договора аренды.

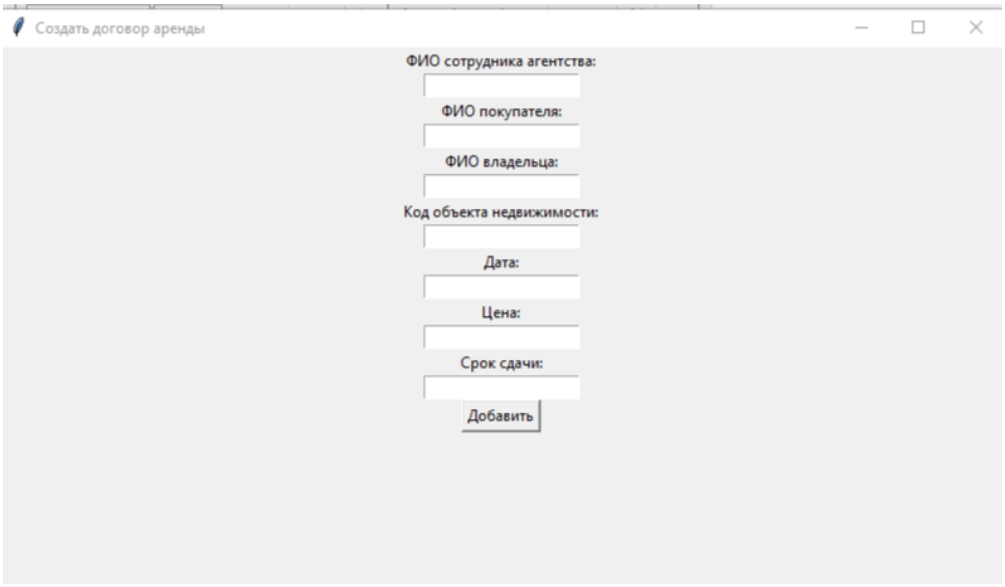

Рисунок 4.15 – Окно для добавления данных договора аренды

На рисунке 4.16 представлены данные для создания договора аренды.

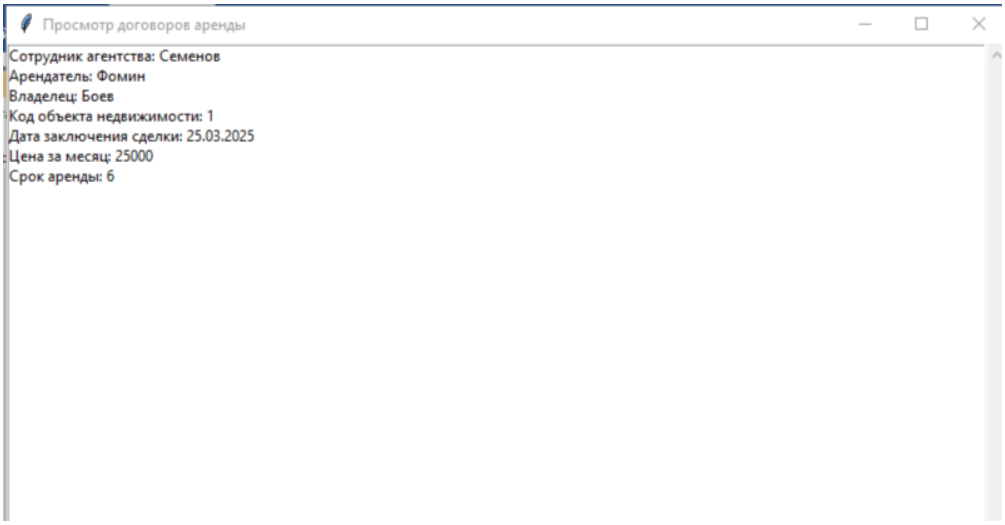

Рисунок 4.16 – Данные для создания договора аренды

На рисунке 4.17 представлен созданный договор аренды.

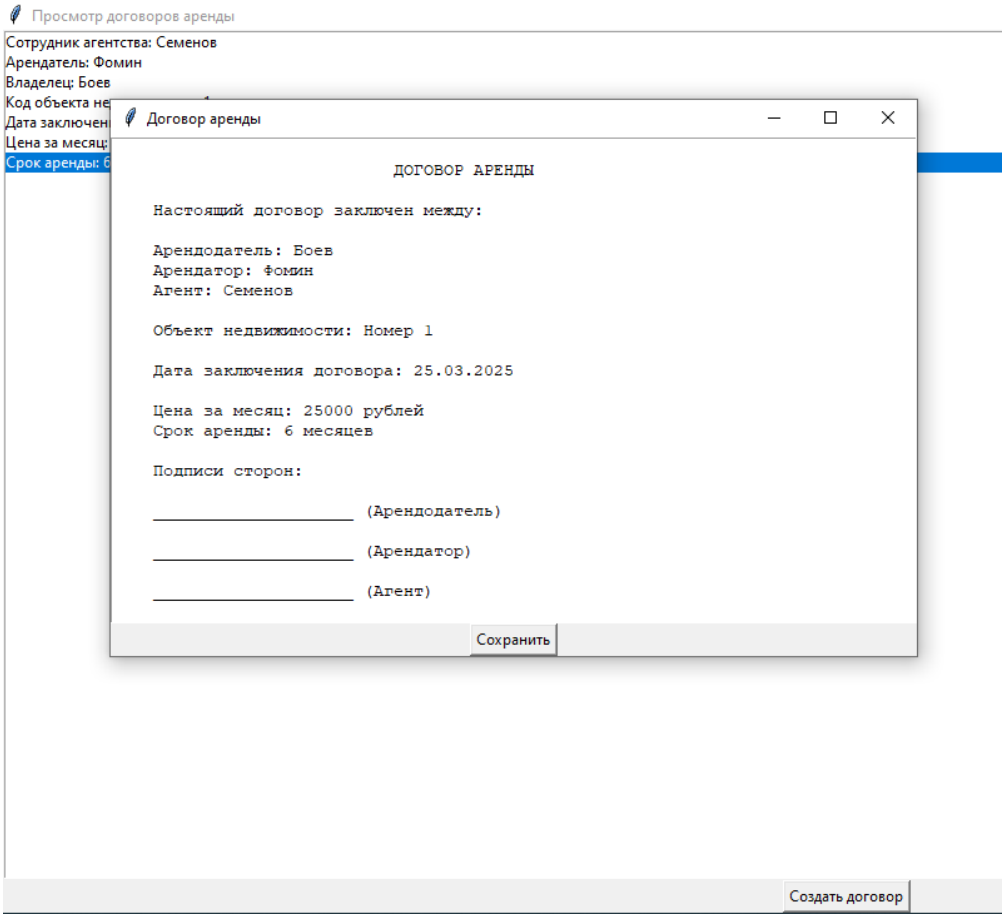

Рисунок 4.17 – Созданный договор аренды

На рисунке 4.18 представлено уведомление об успешной загрузке данных в базу данных.

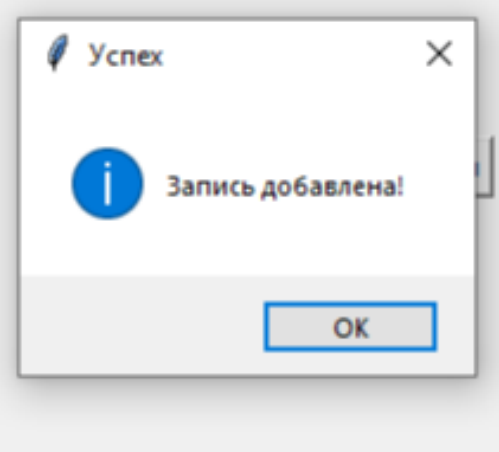

Рисунок 4.18 – Уведомление об успешной загрузке данных в базу данных

На рисунке 4.19 представлена попытка создать пустой договор.

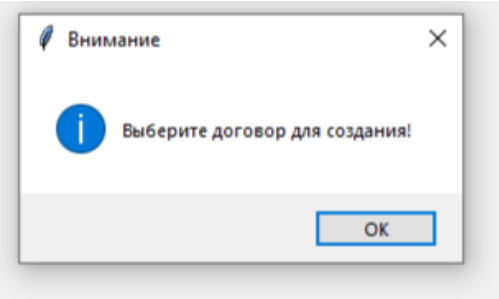

Рисунок 4.19 – Попытка создать пустой договор

ЗАКЛЮЧЕНИЕ

Целью данной работы являлось проектирование базы данных агентства недвижимости и разработка приложения для доступа к этой базе данных.

При проектировании базы данных и разработке приложения были решены следующие задачи:

- исследование предметной области;
- проектирование базы данных;
- создание базы данных;
- заполнение базы данных информацией;
- разработка интерфейса;
- реализация приложения.

Результатом выполнения данной работы является десктопное приложение для работы пользователей с базой данных агентства недвижимости. При проектировании базы данных использован ER-подход к проектированию реляционных баз данных. Интерфейс приложения содержит: главную форму для вывода таблицы, форму для просмотра данных и редактирования записей базы данных, форму для добавления объектов недвижимости, их владельцах, покупателях и тд.

Все требования были полностью реализованы в данном программном продукте.

Все задачи, поставленные в начале проектирования проекта, были также решены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кузнецов С.Д. Базы данных: учебник для студ. учреждений высш. проф. образования / С.Д. Кузнецов. — 2-е изд., испр. — М.: Издательский центр «Академия», 2010. — 496 с.

2. Дейт, К. Дж. Введение в системы баз данных = An Introduction to Database Systems / К. Дж. Дейт. — 8-е изд. — М.: Вильямс, 2006. — 1328 с. (Общая теория баз данных)

3. Коннолли, Т., Бегг, К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е изд. / Т. Коннолли, К. Бегг. — М.: Вильямс, 2003. — 816 с. (Общая теория баз данных)

4. Рамкришнан Р., Гехре Д. Управление базами данных = Database Management Systems / Р. Рамкришнан, Д. Гехре. — 3-е изд. — М.: Вильямс, 2003. — 1056 с. (Общая теория баз данных)

5. Васильев А.Н. Python на примерах. Практический курс программирования. — СПб.: Наука и техника, 2017. — 432 с. (Основы языка Python)

6. Бизли Д.М. Справочник по Python (4-е издание). — Addison-Wesley Professional, 2009. — 704 страницы. (Справочник по Python)

7. Расперри П.Л., Расин Б. Сборник рецептов по анализу данных на Python — O’Reilly Media, 2015. — 514 страниц. (Рецепты по анализу данных на Python)

8. Петров А.А. Анализ предметной области «Агентство недвижимости» и разработка концептуальной модели базы данных // Вестник [Название университета/организации]. — 2020. — № 3. — С. 45-52. (Анализ предметной области)

9. Сидорова Е.В. Проектирование реляционной базы данных для управления недвижимостью // Информационные технологии. — 2019. — № 12. — С. 67-74. (Проектирование реляционных БД)

10. Иванов И.И., Козлов П.С. Методы оптимизации запросов в базах данных агентств недвижимости // Системы управления базами данных. — 2021. — № 2. — С. 102-110. (Оптимизация запросов)

11. Смирнов П.А. Применение ER-диаграмм для моделирования баз данных агентств недвижимости // Современные научные исследования и инновации. – 2018. – № 5. (ER-диаграммы)

12. Васильев С.В. Особенности построения баз данных для информационных систем агентств недвижимости // Наука и образование: сохраняя прошлое, создаём будущее. – 2022. – С. 88-90. (Особенности БД для агентств)

13. Борисова О.И. Проблемы внедрения и использования современных баз данных в агентствах недвижимости // Экономика и управление в XXI веке. – 2019. – С. 123-126. (Проблемы внедрения)

14. Официальная документация Python. — URL: https://docs.python.org/3/ (дата обращения: 15.05.2024). (Официальная документация)

15. Официальная документация SQLite. — URL: https://www.sqlite.org/docs.html (дата обращения: 15.05.2024). (Если вы используете SQLite)

16. psycopg.org — адаптер PostgreSQL для Python. — URL: https://www.psycopg.org/ (дата обращения: 15.05.2024). (Если вы используете PostgreSQL)

17. Учебник по SQL [Электронный ресурс] // W3Schools. — URL: https://www.w3schools.com/sql/ (дата обращения: 15.05.2024). (Основы SQL)

18. Настоящие уроки Python — URL: https://realpython.com/ (дата обращения: 15.05.2024) (Уроки Python)

19. pandas.pydata.org — библиотека Python для анализа данных. — URL: https://pandas.pydata.org/ (дата обращения: 15.05.2024) (Если вы используете Pandas)

ПРИЛОЖЕНИЕ А

Представление графического материала

Графический материал, выполненный на отдельных листах, изображен на рисунках А.1–А.6.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПО ПРОГРАММЕ БАКАЛАВРИАТА

09.03.04 Программная инженерия

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

«Разработка программно-информационных систем»

Разработка программно-информационной системы управления агентством недвижимости

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР	_____ (подпись, дата)	А. И. Газинский (инициалы, фамилия)
Группа ПО-116		
Руководитель ВКР	_____ (подпись, дата)	Е. А. Петрик (инициалы, фамилия)
Нормоконтроль	_____ (подпись, дата)	А. А. Чаплыгин (инициалы, фамилия)

Рисунок А.1 – Сведения о ВКРБ

Цель и задачи разработки

Цель данной работы: разработка программно-информационной системы управления агентством недвижимости

Основными задачами при проектировании и разработке базы данных и приложения являются:

- исследование предметной области;
- проектирование базы данных;
- создание базы данных;
- заполнение базы данных информацией;
- разработка интерфейса;
- реализация приложения.

Рисунок А.2 – Цель и задачи разработки

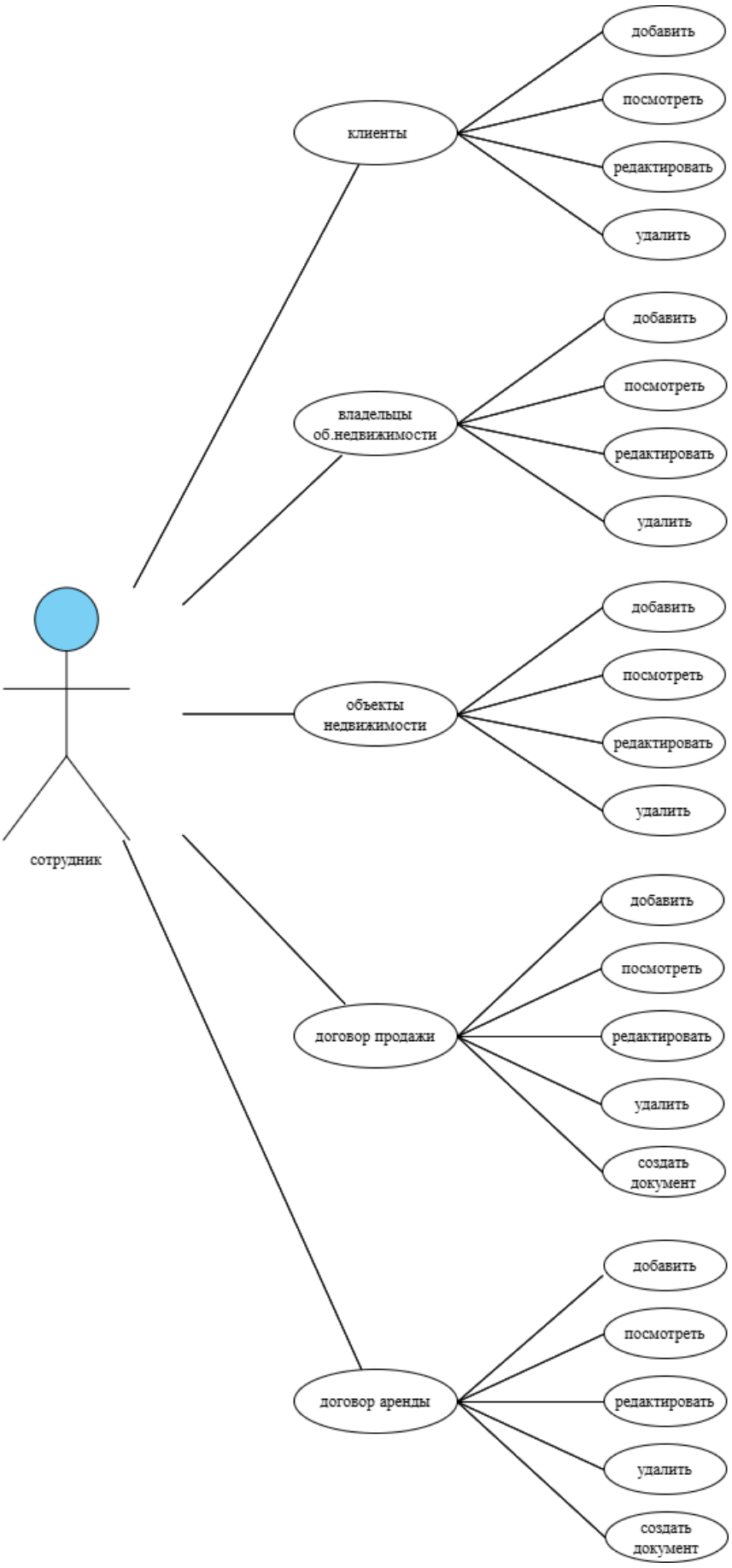

Рисунок А.3 – диаграмма прецедентов

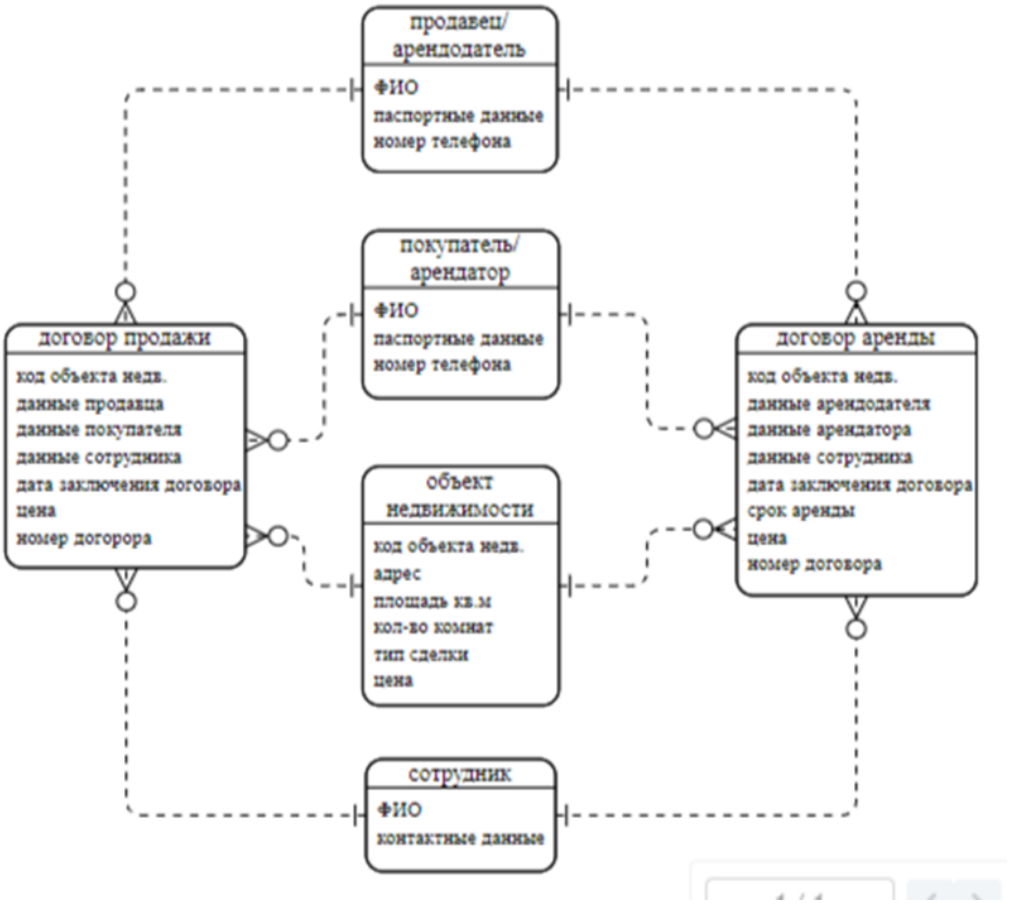

Рисунок А.4 – Концептуальная модель предметной области

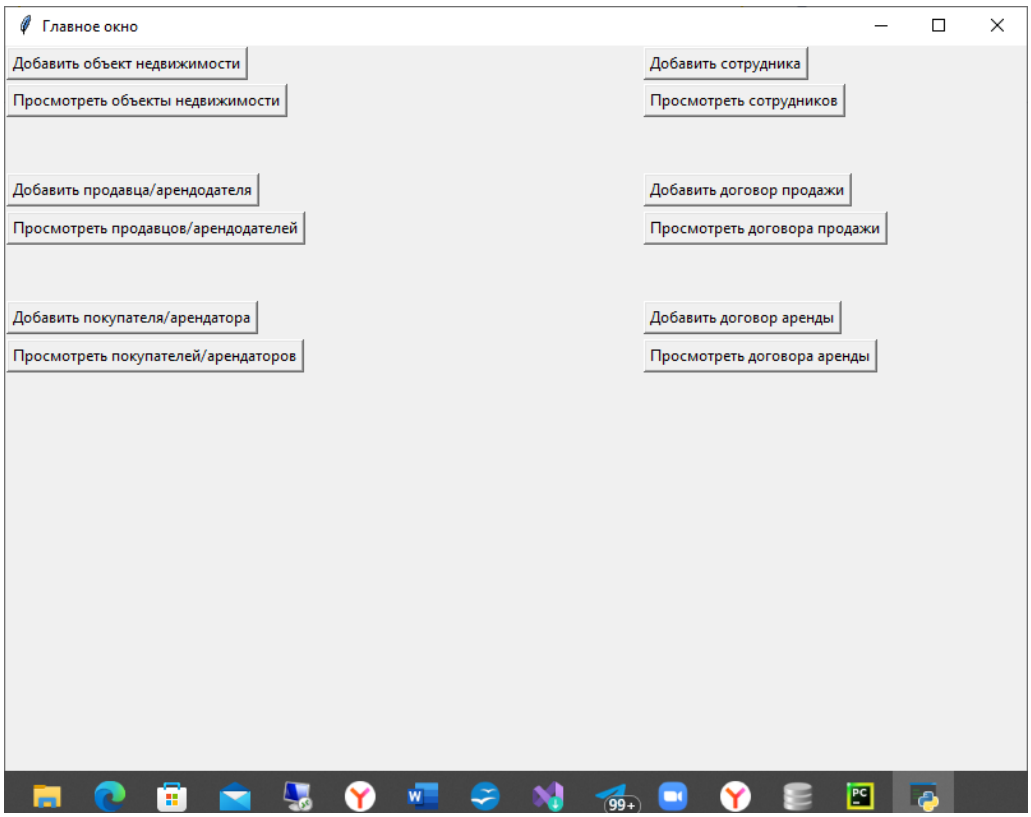

Рисунок А.5 – Интерфейс приложения

ЗАКЛЮЧЕНИЕ

Целью данной работы являлось проектирование базы данных агентства недвижимости и разработка приложения для доступа к этой базе данных.

При проектировании базы данных и разработке приложения были решены следующие задачи:

- исследование предметной области;
- проектирование базы данных;
- создание базы данных;
- заполнение базы данных информацией;
- разработка интерфейса;
- реализация приложения.

Результатом выполнения данной работы является десктопное приложение для работы пользователей с базой данных агентства недвижимости. При проектировании базы данных использован ER-подход к проектированию реляционных баз данных. Интерфейс приложения содержит: главную форму для вывода таблицы, форму для просмотра данных и редактирования записей базы данных, форму для добавления объектов недвижимости, их владельцах, покупателях и тд.

Все требования были полностью реализованы в данном программном продукте.

Все задачи, поставленные в начале проектирования проекта, были также решены.

Рисунок А.6 – Заключение

ПРИЛОЖЕНИЕ Б

Фрагменты исходного кода программы

main.py

```
import tkinter as tk
import sqlite3
from datetime import datetime
from tkinter import Tk, Toplevel, filedialog, Label, Button, Entry, Listbox, messagebox, Scrollbar, END


# Создание базы данных и таблицы
conn = sqlite3.connect('database.db')
c = conn.cursor()
c.execute('''CREATE TABLE IF NOT EXISTS realty (id INTEGER PRIMARY KEY, type CHARACTER VARYING, city CHARACTER VARYING, street CHARACTER VARYING, home INTEGER, flat INTEGER, space INTEGER, rooms INTEGER, time TEXT, price INTEGER, owner INTEGER)''')
c.execute('''CREATE TABLE IF NOT EXISTS employee (id INTEGER PRIMARY KEY, surname CHARACTER VARYING, name CHARACTER VARYING, middlename CHARACTER VARYING, telephone BIGINT)''')
c.execute('''CREATE TABLE IF NOT EXISTS client (id INTEGER PRIMARY KEY, surname CHARACTER VARYING, name CHARACTER VARYING, middlename CHARACTER VARYING, passport BIGINT, telephone BIGINT)''')
c.execute('''CREATE TABLE IF NOT EXISTS owner (id INTEGER PRIMARY KEY, surname CHARACTER VARYING, name CHARACTER VARYING, middlename CHARACTER VARYING, passport BIGINT, telephone BIGINT)''')
c.execute('''CREATE TABLE IF NOT EXISTS sale (id INTEGER PRIMARY KEY, id_employee INTEGER, id_client INTEGER, id_owner INTEGER, id_realty INTEGER, data DATE, cena INTEGER)''')
c.execute('''CREATE TABLE IF NOT EXISTS renta (id INTEGER PRIMARY KEY, id_employee INTEGER, id_client INTEGER, id_owner INTEGER, id_realty INTEGER, data DATE, cena INTEGER, period CHARACTER VARYING)''')
conn.commit()

# Создание главного окна
root = tk.Tk()
root.title("Главное окно")
root.geometry("800x600")

# раздел о недвижимости
# Функция для открытия окна добавления данных
def open_add_window():
    add_window = tk.Toplevel(root)
    add_window.title("Добавить объект недвижимости")
    add_window.geometry("800x600")



    # Добавление элементов интерфейса для ввода данных
    type_label = tk.Label(add_window, text="тип сделки:")
    type_label.pack()
    type_entry = tk.Entry(add_window)
```

```
    type_entry.pack()

    city_label = tk.Label(add_window, text="город:")
    city_label.pack()
    city_entry = tk.Entry(add_window)
    city_entry.pack()

    street_label = tk.Label(add_window, text="улица:")
    street_label.pack()
    street_entry = tk.Entry(add_window)
    street_entry.pack()

    home_label = tk.Label(add_window, text="номер дома:")
    home_label.pack()
    home_entry = tk.Entry(add_window)
    home_entry.pack()

    flat_label = tk.Label(add_window, text="номер квартиры:")
    flat_label.pack()
    flat_entry = tk.Entry(add_window)
    flat_entry.pack()

    space_label = tk.Label(add_window, text="площадь кв.м:")
    space_label.pack()
    space_entry = tk.Entry(add_window)
    space_entry.pack()

    rooms_label = tk.Label(add_window, text="кол-во комнат:")
    rooms_label.pack()
    rooms_entry = tk.Entry(add_window)
    rooms_entry.pack()

    time_label = tk.Label(add_window, text="срок сдачи:")
    time_label.pack()
    time_entry = tk.Entry(add_window)
    time_entry.pack()

    price_label = tk.Label(add_window, text="Цена:")
    price_label.pack()
    price_entry = tk.Entry(add_window)
    price_entry.pack()

    owner_label = tk.Label(add_window, text="код сотрудника:")
    owner_label.pack()
    owner_entry = tk.Entry(add_window)
    owner_entry.pack()

    # Функция для добавления данных в базу данных
    def add_realty():
        type = type_entry.get()
        city = city_entry.get()
        street = street_entry.get()
        home = home_entry.get()
        flat = flat_entry.get()
```

```
90          space = space_entry.get()
91          rooms = rooms_entry.get()
92          time = time_entry.get()
93          price = price_entry.get()
94          owner= owner_entry.get()
95
96          try:
97              home = int(home_entry.get())
98              # continue with adding the realty
99          except ValueError:
100             messagebox.showerror("Error", "Ошибка, некорректный ввод данных в
                    поле номер дома")
101         try:
102             flat = int(flat_entry.get())
103             # continue with adding the realty
104         except ValueError:
105             messagebox.showerror("Error", "Ошибка, некорректный ввод данных в
                    поле номер квартиры")
106         try:
107             rooms = int(rooms_entry.get())
108             # continue with adding the realty
109         except ValueError:
110             messagebox.showerror("Error", "Ошибка, некорректный ввод данных в
                    поле кол-во комнат ")
111
112         # Check if all fields are filled
113         if type == "" or city == "" or street == "" or home == "" or flat ==
                "" or space == "" or rooms == "" or price == "" or owner == "":
114             # Display error message if any field is empty
115             messagebox.showerror("Error", "Ошибка, не все нобходимые поля
                    заполнены")
116         else:
117             # Add the realty details to the database
118             # Code to add the realty details to the database
119             pass
120
121         c.execute("INSERT INTO realty (type, city, street, home, flat, space,
                rooms, time, price, owner) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
                ,
122             (type, city, street, home, flat, space, rooms, time, price, owner
                    ))
123         conn.commit()
124     # Кнопка для добавления данных
125     add_button = tk.Button(add_window, text="Добавить", command=add_realty)
126     add_button.pack()
127
128 # Функция для открытия окна просмотра данных
129 def open_view_window():
130     view_window = tk.Toplevel(root)
131     view_window.title("Просмотр объектов недвижимости")
132     view_window.geometry("800x600")
133
134     # Получение данных из базы данных
135     c.execute("SELECT * FROM realty")
```

```
    data = c.fetchall()

    scrollbar = Scrollbar(view_window)
    scrollbar.pack(side="right", fill="y")

    realty_text = Listbox(view_window, yscrollcommand=scrollbar.set)
    realty_text.pack( fill="both", expand=True)

    # Отображение данных в текстовом виджете
    #data_text = tk.Text(view_window)
    for row in data:
        realty_text.insert(tk.END, f"ID: {row[0]}")
        realty_text.insert(tk.END, f"тип сделки: {row[1]}")
        realty_text.insert(tk.END, f"город: {row[2]}")
        realty_text.insert(tk.END, f"улица: {row[3]}")
        realty_text.insert(tk.END, f"номер дома: {row[4]}")
        realty_text.insert(tk.END, f"номер квартиры: {row[5]}")
        realty_text.insert(tk.END, f"площадь кв.м: {row[6]}")
        realty_text.insert(tk.END, f"кол-во комнат: {row[7]}")
        realty_text.insert(tk.END, f"срок сдачи: {row[8]}")
        realty_text.insert(tk.END, f"цена: {row[9]}")
        realty_text.insert(tk.END, f"код владельца: {row[10]}")
        realty_text.pack()

    scrollbar.config(command=realty_text.yview)




#раздел о сотруднике
# Функция для открытия окна добавления данных
def open_work_window():
    work_window = tk.Toplevel(root)
    work_window.title("Добавить сотрудника")
    work_window.geometry("800x600")

    # Добавление элементов интерфейса для ввода данных
    surname_label = tk.Label(work_window, text="Фамилия:")
    surname_label.pack()
    surname_entry = tk.Entry(work_window)
    surname_entry.pack()

    name_label = tk.Label(work_window, text="Имя:")
    name_label.pack()
    name_entry = tk.Entry(work_window)
    name_entry.pack()

    middlename_label = tk.Label(work_window, text="Отчество:")
    middlename_label.pack()
    middlename_entry = tk.Entry(work_window)
    middlename_entry.pack()

    telephone_label = tk.Label(work_window, text="Контакты:")
    telephone_label.pack()
```

```
    telephone_entry = tk.Entry(work_window)
    telephone_entry.pack()

    # Функция для добавления данных в базу данных
    def add_employee():
        surname = surname_entry.get()
        name = name_entry.get()
        middlename = middlename_entry.get()
        telephone = telephone_entry.get()

        c.execute("INSERT INTO employee (surname, name, middlename, telephone
            ) VALUES (?, ?, ?, ?)",
             (surname, name, middlename, telephone))
        conn.commit()
    # Кнопка для добавления данных
    add_button = tk.Button(work_window, text="Добавить", command=add_employee
        )
    add_button.pack()

# Функция для открытия окна просмотра данных
def open_worker_window():
    worker_window = tk.Toplevel(root)
    worker_window.title("Просмотр сотрудников")
    worker_window.geometry("800x600")

    # Получение данных из базы данных
    c.execute("SELECT * FROM employee")
    employee = c.fetchall()

    scrollbar = Scrollbar(worker_window)
    scrollbar.pack(side="right", fill="y")

    employee_text = Listbox(worker_window, yscrollcommand=scrollbar.set)
    employee_text.pack( fill="both", expand=True)

    # Отображение данных в текстовом виджете
    #data_text = tk.Text(view_window)
    for row in employee:
        employee_text.insert(tk.END, f"ID: {row[0]}")
        employee_text.insert(tk.END, f"Фамилия: {row[1]}")
        employee_text.insert(tk.END, f"Имя: {row[2]}")
        employee_text.insert(tk.END, f"Отчество: {row[3]}")
        employee_text.insert(tk.END, f"Контакты: {row[4]}")
        employee_text.pack()

    scrollbar.config(command=employee_text.yview)




#раздел о покупателе
# Функция для открытия окна добавления данных
def open_buy_window():
```

```
    buy_window = tk.Toplevel(root)
    buy_window.title("Добавить покупателя/арендатора")
    buy_window.geometry("800x600")

    # Добавление элементов интерфейса для ввода данных
    surname_label = tk.Label(buy_window, text="Фамилия:")
    surname_label.pack()
    surname_entry = tk.Entry(buy_window)
    surname_entry.pack()

    name_label = tk.Label(buy_window, text="Имя:")
    name_label.pack()
    name_entry = tk.Entry(buy_window)
    name_entry.pack()

    middlename_label = tk.Label(buy_window, text="Отчество:")
    middlename_label.pack()
    middlename_entry = tk.Entry(buy_window)
    middlename_entry.pack()

    passport_label = tk.Label(buy_window, text="Паспортные данные:")
    passport_label.pack()
    passport_entry = tk.Entry(buy_window)
    passport_entry.pack()

    telephone_label = tk.Label(buy_window, text="Контакты:")
    telephone_label.pack()
    telephone_entry = tk.Entry(buy_window)
    telephone_entry.pack()

    # Функция для добавления данных в базу данных
    def add_client():
        surname = surname_entry.get()
        name = name_entry.get()
        middlename = middlename_entry.get()
        passport= passport_entry.get()
        telephone = telephone_entry.get()

        c.execute("INSERT INTO client (surname, name, middlename, passport,
            telephone) VALUES (?, ?, ?, ?, ?)",
             (surname, name, middlename, passport, telephone))
        conn.commit()
    # Кнопка для добавления данных
    add_button = tk.Button(buy_window, text="Добавить", command=add_client)
    add_button.pack()

# Функция для открытия окна просмотра данных
def open_buyer_window():
    buyer_window = tk.Toplevel(root)
    buyer_window.title("Просмотр покупателей/арендаторов")
    buyer_window.geometry("800x600")

    # Получение данных из базы данных
    c.execute("SELECT * FROM client")
```

```
    client = c.fetchall()

    scrollbar = Scrollbar(buyer_window)
    scrollbar.pack(side="right", fill="y")

    client_text = Listbox(buyer_window, yscrollcommand=scrollbar.set)
    client_text.pack( fill="both", expand=True)

    # Отображение данных в текстовом виджете
    #data_text = tk.Text(view_window)
    for row in client:
        client_text.insert(tk.END, f"ID: {row[0]}")
        client_text.insert(tk.END, f"Фамилия: {row[1]}")
        client_text.insert(tk.END, f"Имя: {row[2]}")
        client_text.insert(tk.END, f"Отчество: {row[3]}")
        client_text.insert(tk.END, f"Паспортные данные: {row[4]}")
        client_text.insert(tk.END, f"Контакты: {row[5]}")
        client_text.pack()

    scrollbar.config(command=client_text.yview)




#раздел о владельце
# Функция для открытия окна добавления данных
def open_sell_window():
    sell_window = tk.Toplevel(root)
    sell_window.title("Добавить продавца/арендодателя")
    sell_window.geometry("800x600")

    # Добавление элементов интерфейса для ввода данных
    surname_label = tk.Label(sell_window, text="Фамилия:")
    surname_label.pack()
    surname_entry = tk.Entry(sell_window)
    surname_entry.pack()

    name_label = tk.Label(sell_window, text="Имя:")
    name_label.pack()
    name_entry = tk.Entry(sell_window)
    name_entry.pack()

    middlename_label = tk.Label(sell_window, text="Отчество:")
    middlename_label.pack()
    middlename_entry = tk.Entry(sell_window)
    middlename_entry.pack()

    passport_label = tk.Label(sell_window, text="Паспортные данные:")
    passport_label.pack()
    passport_entry = tk.Entry(sell_window)
    passport_entry.pack()

    telephone_label = tk.Label(sell_window, text="Контакты:")
    telephone_label.pack()
```

```
    telephone_entry = tk.Entry(sell_window)
    telephone_entry.pack()

    # Функция для добавления данных в базу данных
    def add_owner():
        surname = surname_entry.get()
        name = name_entry.get()
        middlename = middlename_entry.get()
        passport= passport_entry.get()
        telephone = telephone_entry.get()

        c.execute("INSERT INTO owner (surname, name, middlename, passport,
            telephone) VALUES (?, ?, ?, ?, ?)",
             (surname, name, middlename, passport, telephone))
        conn.commit()
    # Кнопка для добавления данных
    add_button = tk.Button(sell_window, text="Добавить", command=add_owner)
    add_button.pack()

# Функция для открытия окна просмотра данных
def open_seller_window():
    seller_window = tk.Toplevel(root)
    seller_window.title("Просмотр продавцов/арендодателей")
    seller_window.geometry("800x600")

    # Получение данных из базы данных
    c.execute("SELECT * FROM owner")
    owner = c.fetchall()

    scrollbar = Scrollbar(seller_window)
    scrollbar.pack(side="right", fill="y")

    owner_text = Listbox(seller_window, yscrollcommand=scrollbar.set)
    owner_text.pack( fill="both", expand=True)

    # Отображение данных в текстовом виджете
    #data_text = tk.Text(view_window)
    for row in owner:
        owner_text.insert(tk.END, f"ID: {row[0]}")
        owner_text.insert(tk.END, f"Фамилия: {row[1]}")
        owner_text.insert(tk.END, f"Имя: {row[2]}")
        owner_text.insert(tk.END, f"Отчество: {row[3]}")
        owner_text.insert(tk.END, f"Паспортные данные: {row[4]}")
        owner_text.insert(tk.END, f"Контакты: {row[5]}")
        owner_text.pack()

    scrollbar.config(command=owner_text.yview)




#регистрация продаж
```

```
# Функция для открытия окна добавления данных
#регистрация продаж
# Функция для открытия окна добавления данных
def open_sale_window():
    sale_window = tk.Toplevel(root)
    sale_window.title("Создать договор продажи")
    sale_window.geometry("800x600")

    # Добавление элементов интерфейса для ввода данных
    employee_label = tk.Label(sale_window, text="ФИО сотрудника агентства:")
    employee_label.pack()
    employee_entry = tk.Entry(sale_window)
    employee_entry.pack()

    client_label = tk.Label(sale_window, text="ФИО покупателя:")
    client_label.pack()
    client_entry = tk.Entry(sale_window)
    client_entry.pack()

    owner_label = tk.Label(sale_window, text="ФИО владельца:")
    owner_label.pack()
    owner_entry = tk.Entry(sale_window)
    owner_entry.pack()

    realty_label = tk.Label(sale_window, text="Код объекта недвижимости:")
    realty_label.pack()
    realty_entry = tk.Entry(sale_window)
    realty_entry.pack()

    data_label = tk.Label(sale_window, text="Дата (YYYY-MM-DD):")
    data_label.pack()
    data_entry = tk.Entry(sale_window)
    data_entry.pack()

    cena_label = tk.Label(sale_window, text="Цена:")
    cena_label.pack()
    cena_entry = tk.Entry(sale_window)
    cena_entry.pack()

    # Функция для добавления данных в базу данных
    def add_sale():
        employee = employee_entry.get()
        client = client_entry.get()
        owner = owner_entry.get()
        realty = realty_entry.get()
        data = data_entry.get()
        cena = cena_entry.get()

        try:
            # Валидация данных (пример)
            if not all([employee, client, owner, realty, data, cena]):
                raise ValueError("Заполните все поля!")
            float(cena)  # Проверка, что цена - число
            datetime.strptime(data, "%Y-%m-%d")  # Проверка формата даты
```

```
456             except ValueError as e:
457                 messagebox.showerror("Ошибка ввода", str(e))
458                 return
459 
460 
461             try:
462                 c.execute("INSERT INTO sale (employee, client, owner, realty,
                        data, cena) VALUES (?, ?, ?, ?, ?, ?)",
463                     (employee, client, owner, realty, data, cena))
464                 conn.commit()
465                 messagebox.showinfo("Успех", "Запись добавлена!")
466                 sale_window.destroy()  # Закрываем окно после успешного
                        добавления
467             except sqlite3.Error as e:
468                 messagebox.showerror("Ошибка БД", str(e))
469 
470     # Кнопка для добавления данных
471     add_button = tk.Button(sale_window, text="Добавить", command=add_sale)
472     add_button.pack()
473 
474 def generate_sale_agreement(row):
475     """Генерирует текст договора продажи на основе данных из строки базы
            данных."""
476     try:
477         employee = row[1]
478         buyer = row[2]
479         owner = row[3]
480         property_id = row[4]
481         date = row[5]  # Уже a string in the database
482         price = row[6]
483 
484         # Форматируем дату, если она не в нужном формате
485         try:
486           date_object = datetime.strptime(date, "%Y-%m-%d") # Предполагаем
                    формат YYYY-MM-DD
487           formatted_date = date_object.strftime("%d.%m.%Y")
488         except ValueError:
489           formatted_date = date  # Оставляем как есть, если не удалось
                    преобразовать
490 
491         agreement_text = f"""
492                                 ДОГОВОР КУПЛИ-ПРОДАЖИ
493 
494     Настоящий договор заключен между:
495 
496     Продавец: {owner}
497     Покупатель: {buyer}
498     Агент: {employee}
499 
500     Объект недвижимости: Номер {property_id}
501 
502     Дата заключения договора: {formatted_date}
503 
504     Цена объекта недвижимости: {price} рублей
```

```
505
506         Подписи сторон:
507
508         _____ (Продавец)
509
510         _____ (Покупатель)
511
512         _____ (Агент)
513         """
514             return agreement_text
515         except Exception as e:
516             return f"Ошибка при создании договора: {e}"
517
518 # Функция для открытия окна просмотра данных
519 def open_salee_window():
520     salee_window = tk.Toplevel(root)
521     salee_window.title("Просмотр договоров о продаже")
522     salee_window.geometry("800x600")
523
524     try:
525         # Получение данных из базы данных
526         with sqlite3.connect('database.db') as conn:
527             c = conn.cursor()
528             c.execute("SELECT * FROM sale")
529             sale = c.fetchall()
530     except sqlite3.Error as e:
531          messagebox.showerror("Ошибка БД", f"Ошибка при подключении к базе
                данных: {e}")
532          return
533
534     scrollbar = Scrollbar(salee_window)
535     scrollbar.pack(side="right", fill="y")
536
537     sale_text = Listbox(salee_window, yscrollcommand=scrollbar.set)
538     sale_text.pack(fill="both", expand=True)
539
540     # Отображение данных в текстовом виджете
541     for row in sale:
542         sale_text.insert(tk.END, f"Сотрудник агентства: {row[1]}")
543         sale_text.insert(tk.END, f"Покупатель: {row[2]}")
544         sale_text.insert(tk.END, f"Владелец: {row[3]}")
545         sale_text.insert(tk.END, f"Объект недвижимости: {row[4]}")
546         sale_text.insert(tk.END, f"Дата: {row[5]}")
547         sale_text.insert(tk.END, f"Цена: {row[6]}")
548         sale_text.insert(tk.END, "")  # Add an empty line
549
550     scrollbar.config(command=sale_text.yview)
551
552     def create_agreement():
553         selected_index = sale_text.curselection()
554         if selected_index:
555             index = selected_index[0]
556             record_index = index // 7  # Correct index calculation
557             if 0 <= record_index < len(sale):
```

```
558             selected_row = sale[record_index]
559             agreement = generate_sale_agreement(selected_row)
560             # Display in a new window (more suitable for larger text)
561             agreement_window = Toplevel(salee_window)
562             agreement_window.title("Договор купли-продажи")
563             agreement_text = tk.Text(agreement_window, wrap="word") #wrap
                    ="word" to prevent cutting words
564             agreement_text.insert(tk.END, agreement)
565             agreement_text.pack(fill="both", expand=True)
566             agreement_text.config(state="disabled") #Make it read-only
567 
568             #Add a save button
569             def save_agreement():
570                 filename = tk.filedialog.asksaveasfilename(
                        defaultextension=".txt",
571                                                            filetypes=[("
                                                               Text files
                                                               ", "*.txt"
                                                               ),
572                                                                    ("
                                                                       All
                                                                       files
                                                                       "
                                                                       ,
                                                                       "
                                                                       *.*
                                                                       "
                                                                       )
                                                                       ])
573                 if filename:
574                     try:
575                         with open(filename, "w", encoding="utf-8") as f:
                                # Use utf-8 encoding
576                             f.write(agreement)
577                         messagebox.showinfo("Сохранено", "Договор
                                сохранен")
578                     except Exception as e:
579                         messagebox.showerror("Ошибка", f"Ошибка
                                сохранения файла: {e}")
580             save_button = Button(agreement_window, text="Сохранить",
                    command=save_agreement)
581             save_button.pack()
582 
583 
584         else:
585             messagebox.showerror("Ошибка", "Неверный индекс выбранной
                    записи.")
586     else:
587         messagebox.showinfo("Внимание", "Выберите договор для создания!")
588 
589 
```

```
590     create_button = Button(salee_window, text="Создать договор", command=
            create_agreement)
591     create_button.pack()
592
593
594
595
596
597 #регистрация аренд
598 # Функция для открытия окна добавления данных
599 def open_rent_window():
600     rent_window = tk.Toplevel(root)
601     rent_window.title("Создать договор аренды")
602     rent_window.geometry("800x600")
603
604     # Добавление элементов интерфейса для ввода данных
605     employee_label = tk.Label(rent_window, text="ФИО сотрудника агентства:")
606     employee_label.pack()
607     employee_entry = tk.Entry(rent_window)
608     employee_entry.pack()
609
610     client_label = tk.Label(rent_window, text="ФИО покупателя:")
611     client_label.pack()
612     client_entry = tk.Entry(rent_window)
613     client_entry.pack()
614
615     owner_label = tk.Label(rent_window, text="ФИО владельца:")
616     owner_label.pack()
617     owner_entry = tk.Entry(rent_window)
618     owner_entry.pack()
619
620     realty_label = tk.Label(rent_window, text="Код объекта недвижимости:")
621     realty_label.pack()
622     realty_entry = tk.Entry(rent_window)
623     realty_entry.pack()
624
625     data_label = tk.Label(rent_window, text="Дата:")
626     data_label.pack()
627     data_entry = tk.Entry(rent_window)
628     data_entry.pack()
629
630     cena_label = tk.Label(rent_window, text="Цена:")
631     cena_label.pack()
632     cena_entry = tk.Entry(rent_window)
633     cena_entry.pack()
634
635     period_label = tk.Label(rent_window, text="Срок сдачи:")
636     period_label.pack()
637     period_entry = tk.Entry(rent_window)
638     period_entry.pack()
639
640     # Функция для добавления данных в базу данных
641     def add_renta():
642         employee = employee_entry.get()
```

```
643             client = client_entry.get()
644             owner = owner_entry.get()
645             realty= realty_entry.get()
646             data = data_entry.get()
647             cena = cena_entry.get()
648             period= period_entry.get()
649 
650 
651             c.execute("INSERT INTO renta (employee, client, owner, realty, data,
                    cena, period) VALUES (?, ?, ?, ?, ?, ?, ?)",
652                  (employee, client, owner, realty, data, cena, period))
653             conn.commit()
654             messagebox.showinfo("Успех", "Запись добавлена!")
655             rent_window.destroy()
656         # Кнопка для добавления данных
657         add_button = tk.Button(rent_window, text="Добавить", command=add_renta)
658         add_button.pack()
659 def generate_rent_agreement(row):
660         """Генерирует текст договора аренды на основе данных из строки базы
                данных."""
661         try:
662             employee = row[1]  # ID уже не нужен в договоре, начинается с
                    сотрудника
663             client = row[2]
664             owner = row[3]
665             realty = row[4] # Была пропущена переменная realty
666             data = row[5]
667             cena = row[6]
668             period = row[7]
669 
670             # Форматируем дату
671             try:
672               date_object = datetime.strptime(data, "%Y-%m-%d") # Предполагаем
                      формат YYYY-MM-DD
673               formatted_date = date_object.strftime("%d.%m.%Y")
674             except ValueError:
675               formatted_date = data  # Оставляем как есть, если не удалось
                      преобразовать
676 
677             agreement_text = f"""
678                                 ДОГОВОР АРЕНДЫ
679 
680         Настоящий договор заключен между:
681 
682         Арендодатель: {owner}
683         Арендатор: {client}
684         Агент: {employee}
685 
686         Объект недвижимости: Номер {realty}
687 
688         Дата заключения договора: {formatted_date}
689 
690         Цена за месяц: {cena} рублей
691         Срок аренды: {period} месяцев
```

```
692
693         Подписи сторон:
694
695         _____ (Арендодатель)
696
697         _____ (Арендатор)
698
699         _____ (Агент)
700         """
701             return agreement_text
702         except Exception as e:
703             return f"Ошибка при создании договора: {e}"
704
705 # Функция для открытия окна просмотра данных
706 def open_rentt_window():
707         rentt_window = tk.Toplevel(root)
708         rentt_window.title("Просмотр договоров аренды")
709         rentt_window.geometry("800x600")
710
711         # Получение данных из базы данных
712         c.execute("SELECT * FROM renta")
713         renta = c.fetchall()
714
715         scrollbar = Scrollbar(rentt_window)
716         scrollbar.pack(side="right", fill="y")
717
718         renta_text = Listbox(rentt_window, yscrollcommand=scrollbar.set)
719         renta_text.pack(fill="both", expand=True)
720
721         # Отображение данных в текстовом виджете
722         #data_text = tk.Text(view_window)
723         for row in renta:
724             renta_text.insert(tk.END, f"Сотрудник агентства: {row[1]}")
725             renta_text.insert(tk.END, f"Арендатель: {row[2]}")
726             renta_text.insert(tk.END, f"Владелец: {row[3]}")
727             renta_text.insert(tk.END, f"Код объекта недвижимости: {row[4]}")
728             renta_text.insert(tk.END, f"Дата заключения сделки: {row[5]}")
729             renta_text.insert(tk.END, f"Цена за месяц: {row[6]}")
730             renta_text.insert(tk.END, f"Срок аренды: {row[7]}")
731             renta_text.pack()
732
733         scrollbar.config(command=renta_text.yview)
734
735         def create_agreement():
736             selected_index = renta_text.curselection()
737             if selected_index:
738                 index = selected_index[0]
739                 record_index = index // 9  # Now ID displayed in listbox. So
                        record occupies 9 lines
740                 if 0 <= record_index < len(renta):
741                     selected_row = renta[record_index]
742                     agreement = generate_rent_agreement(selected_row)
743
744                     agreement_window = Toplevel(rentt_window)
```

```
745                 agreement_window.title("Договор аренды")
746                 agreement_text = tk.Text(agreement_window, wrap="word")
747                 agreement_text.insert(tk.END, agreement)
748                 agreement_text.pack(fill="both", expand=True)
749                 agreement_text.config(state="disabled")  # Make it read-only
750 
751                 def save_agreement():
752                     filename = filedialog.asksaveasfilename(defaultextension=
                            ".txt",
753                                                             filetypes=[("Text
                                                                files", "*.
                                                               txt"),
754                                                                    ("All
                                                                       files
                                                                       ",
                                                                       "
                                                                       *.*
                                                                       ")
                                                                       ])
755                     if filename:
756                         try:
757                             with open(filename, "w", encoding="utf-8") as f:
758                                 f.write(agreement)
759                             messagebox.showinfo("Сохранено", "Договор
                                    сохранен")
760                         except Exception as e:
761                             messagebox.showerror("Ошибка", f"Ошибка
                                    сохранения файла: {e}")
762 
763                 save_button = Button(agreement_window, text="Сохранить",
                        command=save_agreement)
764                 save_button.pack()
765 
766             else:
767                 messagebox.showerror("Ошибка", "Неверный индекс выбранной
                        записи.")
768         else:
769             messagebox.showinfo("Внимание", "Выберите договор для создания!")
770 
771     create_button = Button(rentt_window, text="Создать договор", command=
            create_agreement)
772     create_button.pack()
773 
774 # Кнопки для открытия окон добавления и просмотра данных
775 add_button = tk.Button(root, text="Добавить объект недвижимости", command=
        open_add_window)
776 add_button.place (x='1', y='1')
777 view_button = tk.Button(root, text="Просмотреть объекты недвижимости",
        command=open_view_window)
778 view_button.place (x='1', y='30')
779 work_button= tk.Button(root, text="Добавить сотрудника", command=
        open_work_window)
780 work_button.place (x='500', y='1')
```

```
781 worker_button= tk.Button(root, text="Просмотреть сотрудников", command=
        open_worker_window)
782 worker_button.place (x='500', y='30')
783 buy_button= tk.Button(root, text="Добавить покупателя/арендатора", command=
        open_buy_window)
784 buy_button.place (x='1', y='200')
785 buyer_button= tk.Button(root, text="Просмотреть покупателей/арендаторов",
        command=open_buyer_window)
786 buyer_button.place (x='1', y='230')
787 sell_button= tk.Button(root, text="Добавить продавца/арендодателя", command=
        open_sell_window)
788 sell_button.place (x='1', y='100')
789 seller_button= tk.Button(root, text="Просмотреть продавцов/арендодателей",
        command=open_seller_window)
790 seller_button.place (x='1', y='130')
791 sale_button= tk.Button(root, text="Добавить договор продажи", command=
        open_sale_window)
792 sale_button.place (x='500', y='100')
793 salee_button= tk.Button(root, text="Просмотреть договора продажи", command=
        open_salee_window)
794 salee_button.place (x='500', y='130')
795 rent_button= tk.Button(root, text="Добавить договор аренды", command=
        open_rent_window)
796 rent_button.place (x='500', y='200')
797 rentt_button= tk.Button(root, text="Просмотреть договора аренды", command=
        open_rentt_window)
798 rentt_button.place (x='500', y='230')
799
800 root.mainloop()
```

Автор ВКР		А. И. Газинский
	(подпись, дата)	
Руководитель ВКР		Е. А. Петрик
	(подпись, дата)	
Нормоконтроль		А. А. Чаплыгин
	(подпись, дата)	

Место для диска