КОЛЛЕДЖ ПРЕЗИДЕНТСКОЙ АКАДЕМИИ

КУРСОВОЙ ПРОЕКТ

По дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка системы классов для приложения «Риэлторская компания»

Специальность 09.02.07 «Информационные системы и программирование»

Выполнил студент(ка) группы 301ИС-22 _____ **Д.В. Ротницкий**

Руководитель _____ **Л.Б. Гусятинер**

Москва 2024

КОЛЛЕДЖ ПРЕЗИДЕНТСКОЙ АКАДЕМИИ

УТВЕРЖДАЮ

Зам. директора КМПО

_____С.Ф. Гасанов

«_____» _____ 2024 г.

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

по дисциплине: МДК 01.01 Разработка программных модулей

Специальность 09.02.07 «Информационные системы и программирование»

Студент группы 301ИС-22 Ротницкий Даниил

ТЕМА: «Разработка системы классов для приложения «Риэлторская компания»»

Дата выдачи задания «_____» _____ 2024 г.

Срок сдачи проекта «_____» _____ 2024 г.

Москва 2024

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ

Задачей данного курсового проекта является разработка системы классов для приложения «Риэлторская компания», которое автоматизирует процессы ручного труда и позволит ускорить работу сотрудников компании и позволит им избавиться от лишней бумажной рутины. Курсовой проект позволит эффективно и быстро управлять объектами недвижимости, клиентами и самими сделками

В первой части будет рассмотрена предметная область, включая основные аспекты работы риэлтора, виды сделок, необходимые пакеты документов и данные, необходимые для регистрации сделки, а также разбор некоторых существующих программных продуктов, а также постановка задачи и её реализация

Во второй части будет построение модели программы, выбор инструментов, её дизайн, диаграмма классов, проектирования сценария, описание модулей программы и подсчёт сложности алгоритмов.

В третьей части представлена реализация модулей разработанной программы, её дизайн и тестирование

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Информационное обеспечение задачи

Риэлтор может представлять интересы одной из сторон сделки (продавца или покупателя). Он занимается и помогает покупателям в поиске объектов для приобретения, продавцам в продаже или сдаче недвижимости, проводит показы и учувствует в подготовке документов для сделок.

Риэлторская компания - это коммерческая организация, которая предоставляет услуги клиентам по профессиональному оформлению и сопровождению всех законных операций с недвижимостью. Во-первых, это конечно продажа, покупка и аренда жилой, загородной, коммерческой, зарубежной и другой недвижимости.[1] Основная цель компании обеспечить высокий уровень сервиса, профессиональное сопровождение сделок и достижение выгодных условий для клиентов

Руководитель компании отвечает за стратегическое управление, координацию процессов и контроль за выполнением задач. Он определяет цели компании, разрабатывает планы развития и обеспечивает их реализацию. Руководитель также взаимодействует с партнерами и клиентами, чтобы поддерживать репутацию компании и укреплять ее позиции на рынке. Его задача создать комфортную и продуктивную рабочую среду для всех сотрудников.

Риэлторы это главные сотрудники, которые непосредственно взаимодействуют с покупателями и продавцами. Они консультируют клиентов, помогая им разобраться в особенностях рынка недвижимости, подбирают подходящие объекты, организуют просмотры и проводят переговоры. Риэлторы также сопровождают сделки на всех этапах, начиная от первого контакта с клиентом и заканчивая оформлением и печатью

документов. Их задача обеспечить высокий уровень сервиса и удовлетворение потребностей как покупателей, так и продавцов.

В большинстве риелторских компаний предпочитают использовать программы для управления объектами недвижимости и клиентами. Такие системы позволяют не только упорядочить все процессы, но и обеспечивают надежное хранение информации о клиентах и сделках. Это открывает новые горизонты для роста и развития бизнеса.

Современная риелторская компания представляет собой сложный механизм, в котором важны точность, скорость и удобство работы. Объекты недвижимости постоянно меняют свой статус: новые предложения появляются на рынке, сделки заключаются и отменяются, а клиенты ищут лучшие варианты. Если не обеспечить эффективный учет этих процессов, работа компании может стать неэффективной.

Специализированные программы для риелторов позволяют автоматизировать рутинные задачи, такие как ведение базы данных объектов, управление контактами клиентов и отслеживание сделок.

1.2. Обзор и анализ существующих программных решений

В это разделе рассмотрены некоторые уже имеющиеся программы для хранения и создания риэлторских сделок. Описание существующих разработок для хранения и создания риэлторских сделок.

В современном рынке недвижимости существует множество программ и онлайн-сервисов, которые помогают риэлторам и агентствам эффективно управлять сделками, клиентами и объектами недвижимости. Рассмотрим несколько популярных платформ, таких как «**Домклик**» и **ЦИАН**, а также их цели и требования.

1.2.1. «Домклик»

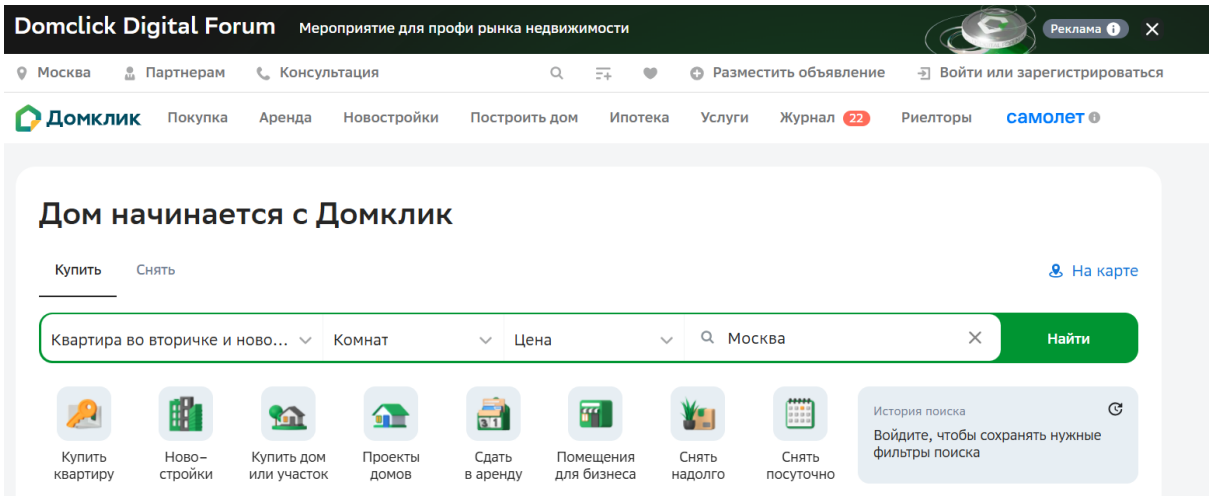

Рисунок 1.Главный интерфейс «Домклик»

«Домклик» — это онлайн-сервис, разработанный Сбербанком, который позволяет пользователям находить, покупать, продавать и арендовать недвижимость. Он предлагает услуги по ипотечному кредитованию и обеспечивает безопасное проведение сделок. Главный экран «Домлкик» (см. Рисунок 1)

Цели включают упрощение приобретения жилья, повышение безопасности сделок, интеграцию с финансовыми услугами и предоставление актуальной информации о рынке. Функционал платформы включает удобный поиск объектов по параметрам (цене, площади, местоположению), размещение объявлений для продавцов и застройщиков, ипотечный калькулятор для оценки условий кредитования, юридическую экспертизу объектов и возможность онлайн-оформления сделок

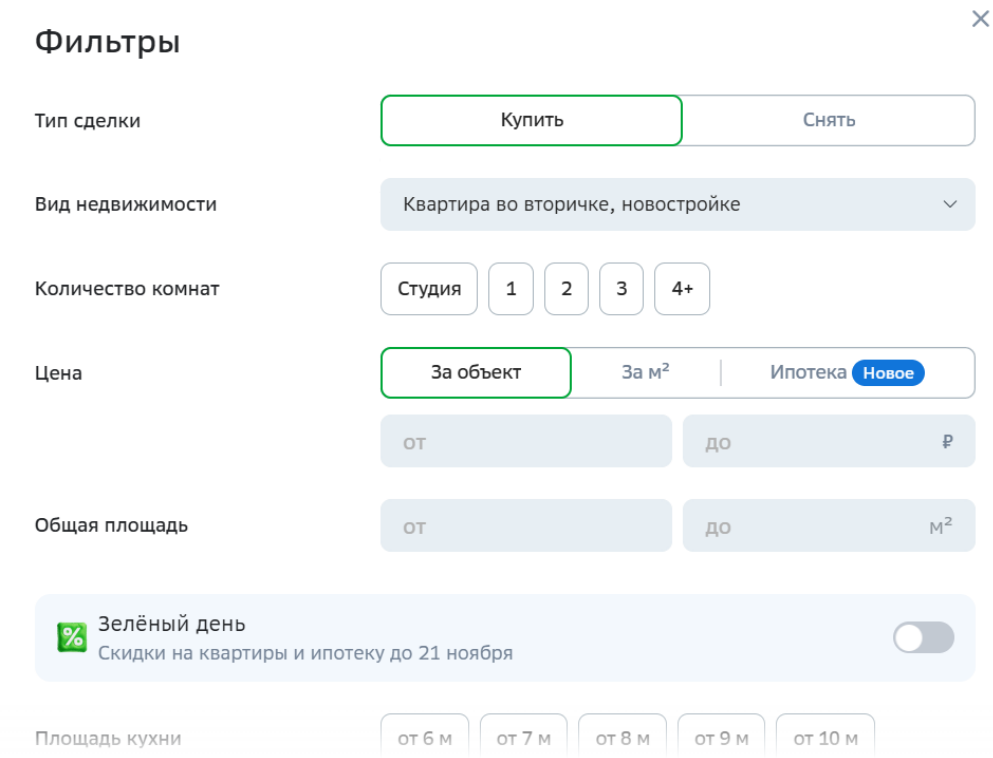

Рисунок 2. Примеры фильтров на «Домклик»

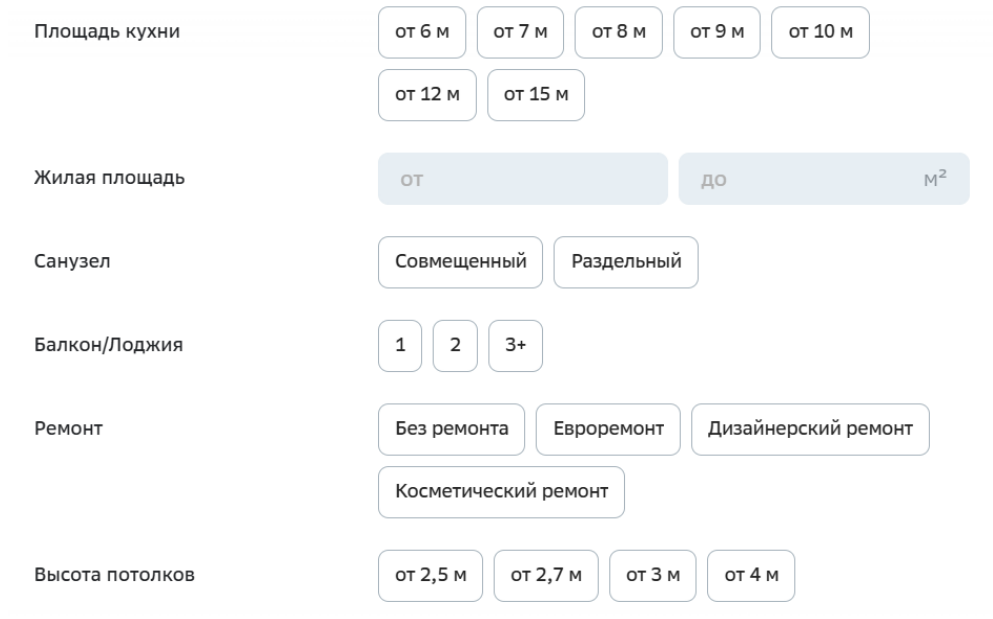

Рисунок 3. Примеры фильтров на «Домклик»

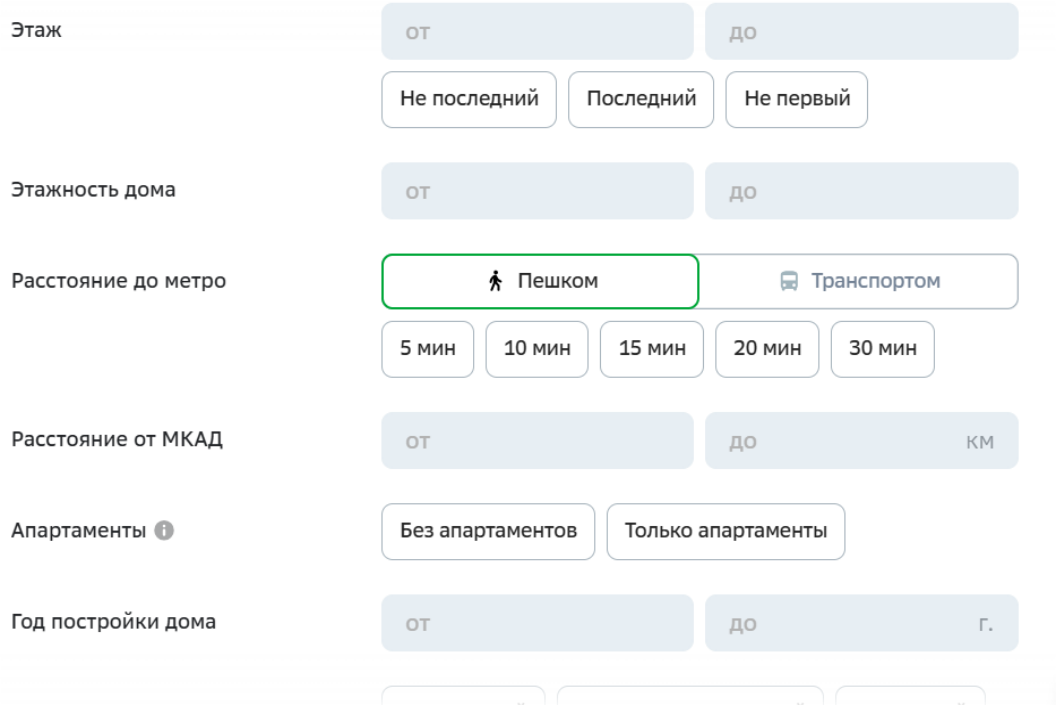

Рисунок 4. Примеры фильтров на «Домклик»

«Домклик» предоставляет разнообразные фильтры, которые помогают пользователям точно настроить поиск недвижимости. (см. Рисунки 2,3,4)

Особенности «Домклик»

1. Удобный интерфейс: Платформа имеет интуитивно понятный интерфейс, что облегчает процесс поиска недвижимости.
2. Широкий выбор объектов: Пользователи могут выбирать из большого количества объявлений о продаже и аренде недвижимости по всей России.
3. Безопасность сделок: «Домклик» предлагает услуги по юридической проверке объектов и безопасным расчетам, что снижает риски мошенничества.
4. Аналитические инструменты: Платформа предоставляет данные о ценах на недвижимость и тенденциях рынка, что помогает пользователям принимать обоснованные решения.

1.2.2 ЦИАН

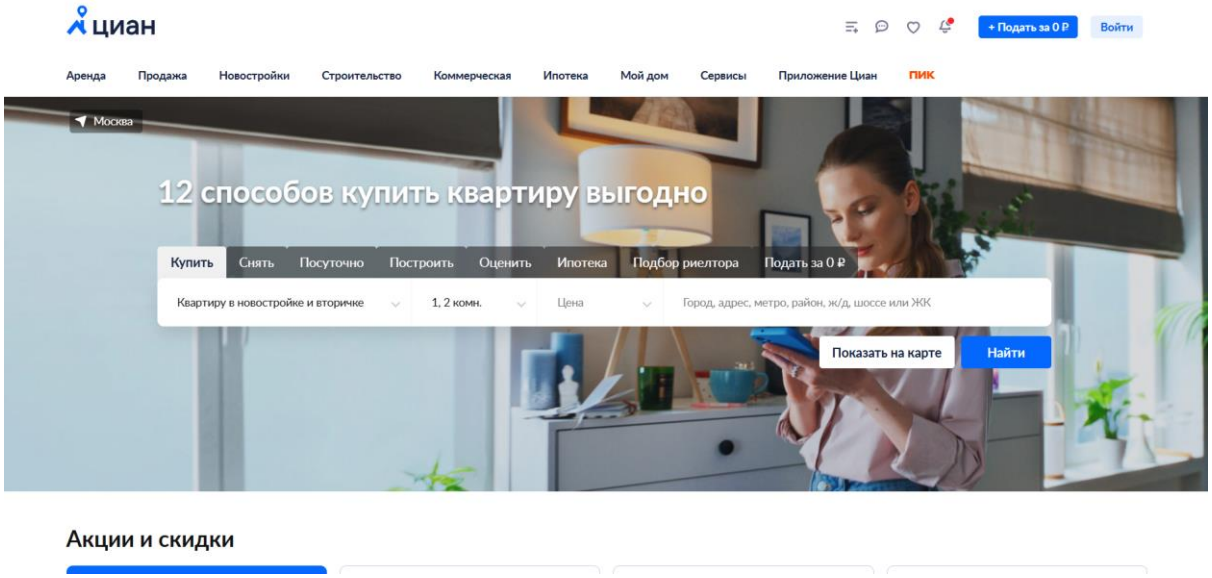

Рисунок 5. Главный интерфейс ЦИАН

ЦИАН — это один из крупнейших онлайн-сервисов для поиска и размещения объявлений о недвижимости в России. Платформа предлагает пользователям широкий функционал для аренды, покупки и продажи объектов недвижимости, а также включает инструменты для оценки стоимости жилья. Главный экран ЦИАН (см. Рисунок 5)

Основные цели ЦИАН заключаются в упрощении поиска недвижимости, создании безопасной среды для сделок с юридической проверкой объектов, внедрении новых технологий для улучшения пользовательского опыта и обеспечении прозрачности рынка через предоставление актуальной информации о ценах на недвижимость.

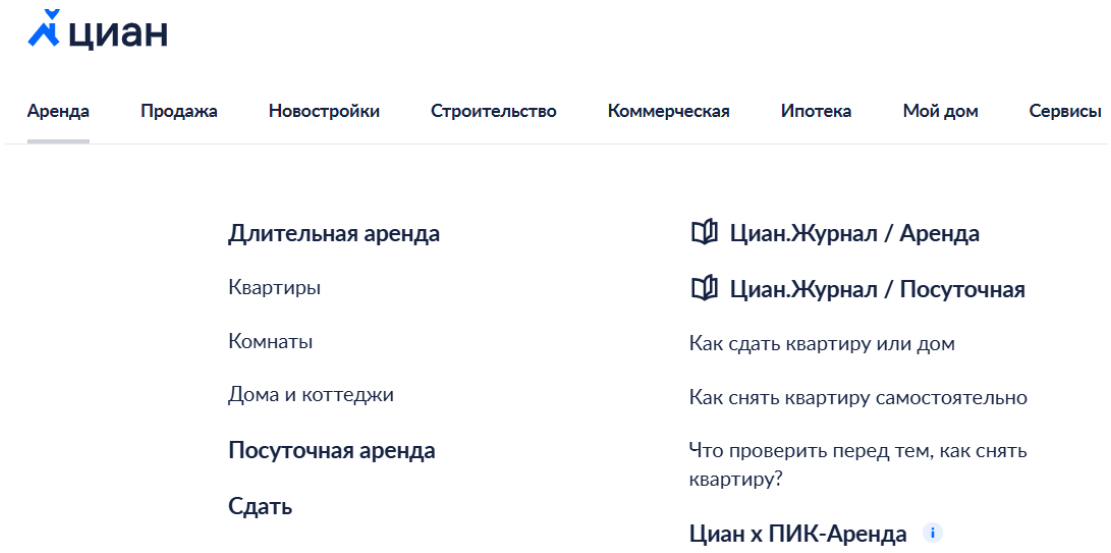

Рисунок 6. Пример фильтров на ЦИАН

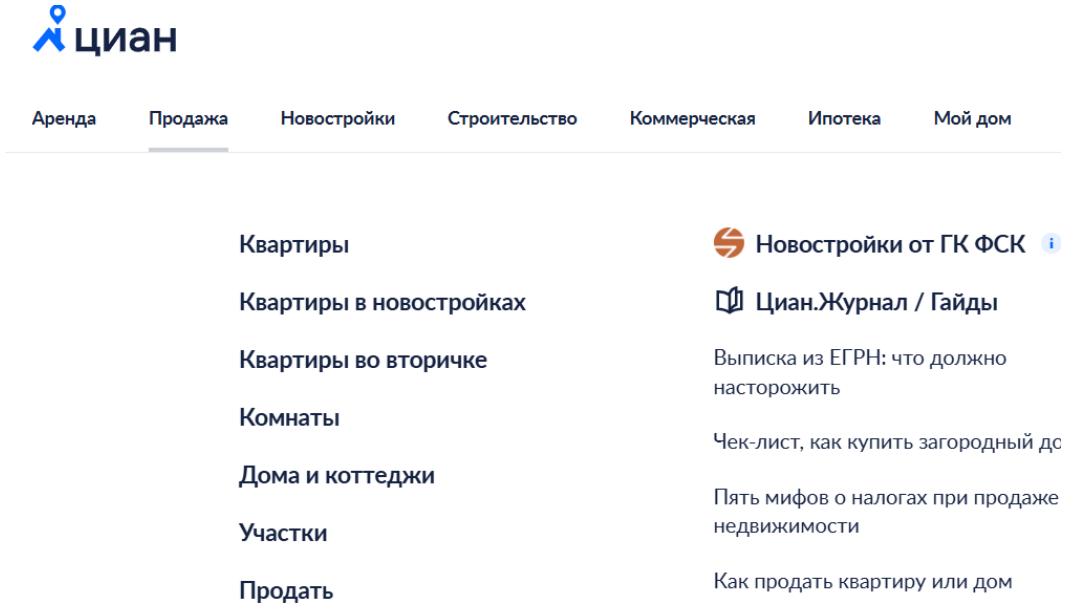

Рисунок 7. Пример фильтров на ЦИАН

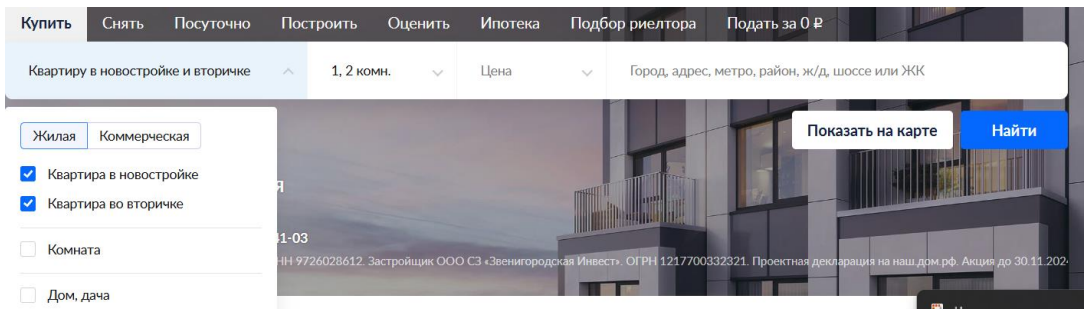

Рисунок 8. Пример фильтров на ЦИАН

ЦИАН также обладает большим количеством фильтров для сортировки и распределения сделок. (см. Рисунки 6,7,8)

Особенности ЦИАН

1. Широкая база данных: Доступ к миллионам объявлений о продаже и аренде недвижимости.
2. Удобный интерфейс: Интуитивно понятный интерфейс упрощает поиск объектов.
3. Аналитические инструменты: Оценка рыночной стоимости объектов и анализ ценовых тенденций.
4. Безопасность сделок: Услуги по юридической проверке объектов снижают риски мошенничества.

1.3 Постановка задачи. Структура входной и выходной информации

Проект должен выполнять определённый перечень требований. Среди них создание, просмотр, удаление сделок, а также генерация договоров купля-продажи. Для написания проекта следует использовать такую парадигму программирования как ООП для более правильного и чёткого структурирования информации. Надо использовать классы для представления клиентов, недвижимости и сделок. Программа должна поддерживать способность загрузки и выгрузки данных из файла и создания договора в формате .docx.

Входные данные представляют собой личные данные клиентов: имя, фамилия, отчество (если нужно), номер телефона.

Выходные данные 1) Список всех существующих сделок 2) Создание договора о купля-продаже

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1. Построение модели системы

Система классов приложения «Риэлторская компания» предназначена для облегчения и автоматизации бумажной волокиты у представителей компании, таких как риэлторы. Программа обеспечит минимальные физические затраты, что полностью исключит бумажные носители для хранения информации, повысит скорость и качество взаимодействия с клиентами. Основной целью можно обозначить хранение всей информации о сделках с последующими их изменениями. Программа прекрасно подойдет как для больших компаний, для и частных риэлторов, за счёт своей простоты и нетребовательности в использовании. За счёт использования подобного приложения качество исполнения задач будет на высшем уровне.

На основе данной информации была построена диаграмма прецедентов. На диаграмме показаны отношения риэлтора, покупателя и продавца по отношению к сделке (см. Приложение 10)

Для проекта также был разработан дизайн в приложение Figma. Представление результата курсового проекта в будущем (см. Приложение 12)

Анализируя нескончаемое количество языков, было выбрано два из наилучших языка программирования. Были выбраны критерии и ним расставлены баллы (см. Таблицу 1 и см. Таблицу 2)

Таблица 1. Критерии оценки и их важность при сравнении языков программирования

Критерий	Важность критерия
Русскоязычная документация	Средняя
Простота сопровождения	Средняя
Наличие библиотек	Высокая
Понятный синтаксис	Высокая

Скорость разработки	Средняя

Таблица 2. Оценка языков программирования по критериям

Критерий/Язык программирования	C++	Python
Русскоязычная документация	8	9
Простота сопровождения	6	9
Наличие библиотек	7	10
Понятный синтаксис	6	9
Скорость разработки	7	10
Итого баллов	34	47

- **C++**: хорошо подходит для крупных проектов, но может быть сложнее в сопровождении из-за своей сложности.
- **Python**: отличается понятным синтаксисом и большим количеством библиотек, что делает его идеальным для поставленного проекта и его сопровождения

Таким образом, был выбран язык программирования, Python, поскольку обладает большим количеством библиотек, необходимых для реализации, поставленной задачи для проекта.

Был проведён выбор среды разработки для проекта по наборам критериев, представленных в таблице. (см. Таблицу 3). Были сравнены две среды PyCharm и VS code. (см. Таблицу 4)

Таблица 3. Критерии выбора среды разработки

Критерий	Важность критерия
Поддержка языков программирования	Средняя

Производительность	Средняя
Функциональность	Высокая
Удобство использования	Высокая
Документация	Средняя

Таблица 4. Оценка среды разработки про критерии

Критерий/Среда разработки	Microsoft Visual Studio Code	PyCharm
Поддержка языков программирования	9	10
Производительность	8	7
Функциональность	8	9
Удобство использования	8	9
Документация	8	9
Итого баллов	41	44

После выбора языка программирования был построен сценарий использования приложения со стороны риэлтора. (см. Приложение 11)

В конце, после представления общей структуры была подготовлена диаграмма для системы классов, основанная на необходимых данных и объектах, которые будут участвовать при регистрации сделки. Я выделил три класса: 1) Person (Человек) – класс, который служит для представления клиента, покупателя или продавца. Данные необходимые для данного класса, были приведены в пункте 1.3 при описании входных данных. 2) RealEstate (Недвижимость) – класс, который служит для представления объекта недвижимости. Параметрами класса были обозначены адрес, цена, цена и количество комнат. 3) Deal (Сделка) – класс, необходимый для представления

сделки с недвижимостью. Параметрами класса являются номер сделки, покупатель, продавец, объект продажи, город проведения сделки. Подробная диаграмма представлена в приложении. (см. Приложение 1). Код каждого класс представлен в приложениях (см Приложение 4, Приложение 5, Приложение 6)

2.2. Описание главного модуля

Файл main.py является главным модулем программы, в котором выполняются все функции. Модуль main.py. Содержит в себе основную функцию main(). Работает при помощи бесконечного цикла, который работает до тех пор, пока не будет введена одна из существующих команд. В начале программы идёт выгрузка всех сделок из файла с помощью load_deals_from_file из модуля file_operations.py. Затем, идёт вход в бесконечный цикл, где необходимо ввести одну из команд: new, view, contract, delete, exit.

New позволяет создать новую сделку. Для этого необходимо ввести необходимые данные на продавца, покупателя и недвижимость. После ввода всех данных идёт создание объектов классов и новая сделка добавляется в deals_list (список сделок). После чего обновлённый список сохраняется с помощью save_deal_to_file из file_operations.py. При вводе данных предусмотрены проверки. Если обязательное поле пустое, ты выходит ошибка с предупреждением. Отчество не считается обязательным полем. Проверка на отсутствие пробелов при вводе номера телефона, так как ввод с пробелом может вызвать лишние ошибки с форматирование CSV файла. А также проверка на присутствие именно цифр в номере телефона, а не посторонних данных. Аналогично у недвижимости, но здесь положительными числами должны уже являться цена, площадь и количество комнат.

View выводит список всех существующих сделок. При запуске проходит проверка, на заполненность списка. Если он пустой, то выводится

соответствующее предупреждение. В противном случае все данные выводятся в консоль.

Contract генерирует договор для сделки. При выборе функции у пользователя производится запрос номера сделки. Необходимый номер ищется в deals_list. Если нужная сделка найдена, то применяется метод из класса Deal generate_contract, который создаёт текст для договора. После чего файл сохраняется в формате docx с помощью функции модуля. Если сделка найдена не была, выводится предупреждение.

Delete используется для удаления сделки по её номеру. При запуске функции программа запрашивает номер сделки для удаления. После чего также идёт поиск сделки в списке. Если сделка найдена, то она удаляется из списка, а все остальные номера сделок корректируются. Сохранённый список сохраняется и выводится сообщение об удалении сделки. Если сделка, не найдена, то выводится предупреждение.

Exit завершает работу программы, прерывая работу бесконечного цикла через break.

В приложении представлены блок схема и код модуля. (см. Приложение 2 и Приложение 3).

2.3. Описание спецификаций к модулям

Модуль main.py предназначен для выполнения всех функций программы. Задачей модуля является взаимодействие с пользователем через консоль, а также выполнение запрашиваемых функций. Входными данными модуля являются одна из функций, выбранная с консоли пользователем, а выходными список сделок в консоли, печать результата о выполнении сделки или создание договора в формате .docx

Модуль file_operations.py служит для работы с файлами для сохранения и загрузки данных сделок. Входными данными для модуля являются список сделок и имя файла, с которым необходимо работать. Выходными данными

является структурированная информация, которая загружается в формате CSV, а также выгруженный список объектов из класса Deal.

Модуль contract_saver предназначен для создания и сохранения договора купля-продажи в формате .docx. Его задача — это генерация текста договора и последующее сохранения в Word файл. Входными данными модуля является объект класса deal и имя файла, где будет сохранён договор.

2.4. Описание модулей.

Модуль file_operations.py. Отвечает за работу с файлами, их загрузкой и выгрузкой. При помощи функций файл сохраняется в формате CSV и загружает данные из списка.

Save_deal_to_file сохраняет сделки в формате CSV. Как параметры принимаются список сделок и название файла для работы. Работа с файлом осуществляется через менеджера контекста. Все данные сделки форматируются в строку и делятся “;”. После чего такой формат строки записывается в файл.

Load_deals_from_file загружает данные из файла и создаёт объекты класса Deal. Как параметр принимается только название файла. Работа также осуществляется через менеджера контекста. Деление данных осуществляется через “;’, данные извлекаются и создаются объекты для классов. После чего происходит добавление объекта Deal в список сделок.

Код представлен в приложении (см. Приложение 7)

Модуль contract_saver.py нужен для создания договора купля-продажи в формате docx. На входе принимается объект из Deal и название файла. Для модуля импортируется библиотека docx для создания документа. Устанавливается заголовок №1 внутри, а потом при помощи расширения docx.shared устанавливается текст. Его шрифт, цвет, размер. После этого происходит генерация текста, добавление его в документ и сохранение.

Код представлен в приложении (см. Приложение 8)

2.5. Расчёт сложности алгоритма.

Сложность алгоритма следует рассматривать с двух сторон: Время и Память.

Если рассмотреть main.py, то с точки зрения времени: view – O(n), поскольку зависит от количества сделок, чем больше сделок, тем больше времени нужно на обработку. Contract – O(n), потому что поиск идёт по номеру сделки, чем больше номер, тем больше времени. New – O(1), это константное время, на которое не влияют внешние факторы. Delete – O(n), потому что она не просто удаляют сделку, но и ведёт пересчёт номеров оставшихся. Поэтому в контексте времени модуль main.py – O(n). Если смотреть в контексте памяти, то объём списка сделок (deals_list) зависит от их числа, поэтому это O(n)

Классы Person, RealEstate, Deal в обоих контекстах O(1) – константные значения, которые уже зафиксированы и на их производительность ничто не влияет.

В контексте времени в file_operations.py функции перебирают сделки в списке, а также читают строки и объекты, что напрямую зависит от количества, поэтому это O(n). В контексте памяти save_deal_to_file не требует дополнительной памяти, поэтому это O(1). А load_deals_from_file создаёт объекты для каждой из сделок, поэтому O(n)

Модуль contract_saver.py в обоих случаях работает за четкий промежуток времени и не требует к себе дополнительной памяти, поэтому оба случая это O(1).

3. ОТЛАДКА И ТЕСТИРОВАНИЕ МОДУЛЯ

3.1. Описание тестовых наборов модулей

Имеется тестовый сценарий для проверки работы функции добавления сделок в файл. Все тестовые данные представлены в таблице. (см. Таблицу 5)

Таблица 5. Тестовый сценарий для проверки работы функции сохранения

Набор теста	Входные данные	Ожидаемый результат	Фактический результат
Тест 1 (Позитивный)	Продавец: Иван Иванов Иванович, Телефон: 79453322116 Покупатель: Петр Петров Петрович, Телефон: 75643442210 Недвижимость: Москва, ул. Ленина, 10, Цена: 5000000, Площадь: 50 м², 3 комнаты Город: Москва	Сделка успешно создается, сохраняется в файл.	`Сделка успешно зарегистрирована!`

Тест 2
(Негативный)
	Продавец: Александр Евгеньевич, Телефон:
	Покупатель: Сергей Иванов Иванович, Телефон: 79876543210
	Недвижимость: Казань, ул. Победы, 25, Цена: 2500000,
		Ошибка ввода, отсутствует основное поля фамилии продавца и номера телефона	`Имя продавца: Алескандр`
			`Фамилия продавца:`
			`Ошибка: фамилия продавца не может быть пустой.`

	Площадь: 40 м², 2 комнаты Город: Казань		
Тест 3 (Негативный)	Продавец: Дмитрий Федоров Иванович, Телефон: 79981234567 Покупатель: Анастасия Кузнецова Александровна, Телефон: 72345678900 Недвижимость: Санкт-Петербург, ул. Льва Толстого, Цена: 9000000, Площадь: x80 (ошибка), 4 комнаты Город: Санкт-Петербург	Ошибка: Некорректная площадь (нечисловое значение). Сделка не сохраняется.	Цена (руб., только целое число): 9000000 Площадь (м², может быть дробным числом): x80 Ошибка: площадь должна быть числом.

В ходе тестирования было выявлено соответствие ожидаемому результату во всех тестах, что говорит о правильной работе программного кода. Полный список тестов представлен в Тест-Плане (см Приложение 9)

3.2. Описание применения средств отладки

При одном из запусков программы было выведено данное сообщение. (см. Рисунок 13)

Рисунок 13. Пример отображения ошибки

Запятая используется неправильно в данных, так как она разделяет части строки, которые на самом деле должны быть восприняты как одно целое (например, адрес). Программа, которая пытается обработать эти данные, воспринимает запятую как разделитель и пытается привести текст к числовому типу, что приводит к ошибке. После изменения кода ошибка была устранена (см. Рисунок 14)

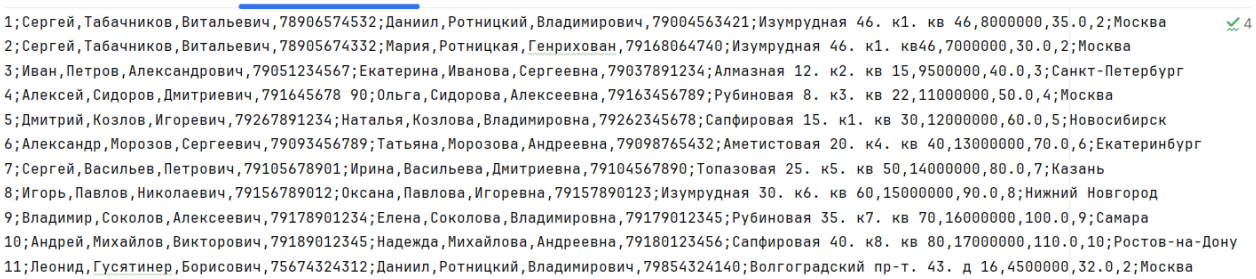

Рисунок 14. Пример ввода данных после отладки

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была разработана программа, предназначенная для автоматизации и упрощения процесса управления сделками с недвижимостью, включая регистрацию сделок, просмотр информации о них и генерацию договоров купли-продажи. Программа позволяет эффективно отслеживать информацию о продавцах, покупателях, недвижимости и сделках в целом.

В ходе работы над проектом был проведен детальный анализ предметной области, исследованы существующие решения, а также реализована функциональность для управления сделками, интеграция с файловыми операциями и создание документов в формате Word с помощью библиотеки python-docx.

Особое внимание было уделено обеспечению удобства работы с программой и расширяемости функционала. Программа продемонстрировала свои преимущества в плане организации сделок с недвижимостью, облегчая пользователю взаимодействие с данными и управлением ими.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному
2. и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.
3. ГОСТ 7.0.100-2018 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления
4. ГОСТ 7.9-95 Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования
5. ГОСТ 7.80-2000 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления
6. ГОСТ 7.90-2007 Система стандартов по информации, библиотечному и издательскому делу. Универсальная десятичная классификация. Структура, правила ведения и индексирования
7. Курс «Добрый, добрый Python с Сергеем Балакиревым». // https://stepik.org/course/100707/syllabus
8. «Классы и объекты». // https://metanit.com/python/tutorial/7.1.php
9. «Работа с библиотекой Docx». // https://python-docx.readthedocs.io/en/latest/
10. «Официальный сайт PyCharm от JetBrains». // https://www.jetbrains.com/pycharm/
11. «Сервис для покупки, аренды недвижимости Домлик » // https://domclick.ru/?utm_referrer=https%3A%2F%2Fwww.google.com%2F
12. «Сервис для покупки и аренды недвижимости ЦИАН» // https://www.cian.ru/

13.[1] Ссылка на цитату из риэлторского форума https://news.ners.ru/chto-takoe-agentstvo-nedvizhimosti-pochemu-vse-zhe-stoit-vospolzovatsya-agentstvom-pri-vybore-obekta-nedvizhimosti.html
14. Статья о расчёте сложности алгоритма https://habr.com/ru/articles/782608/

ПРИЛОЖЕНИЯ

Приложение 1

Диаграмма классов

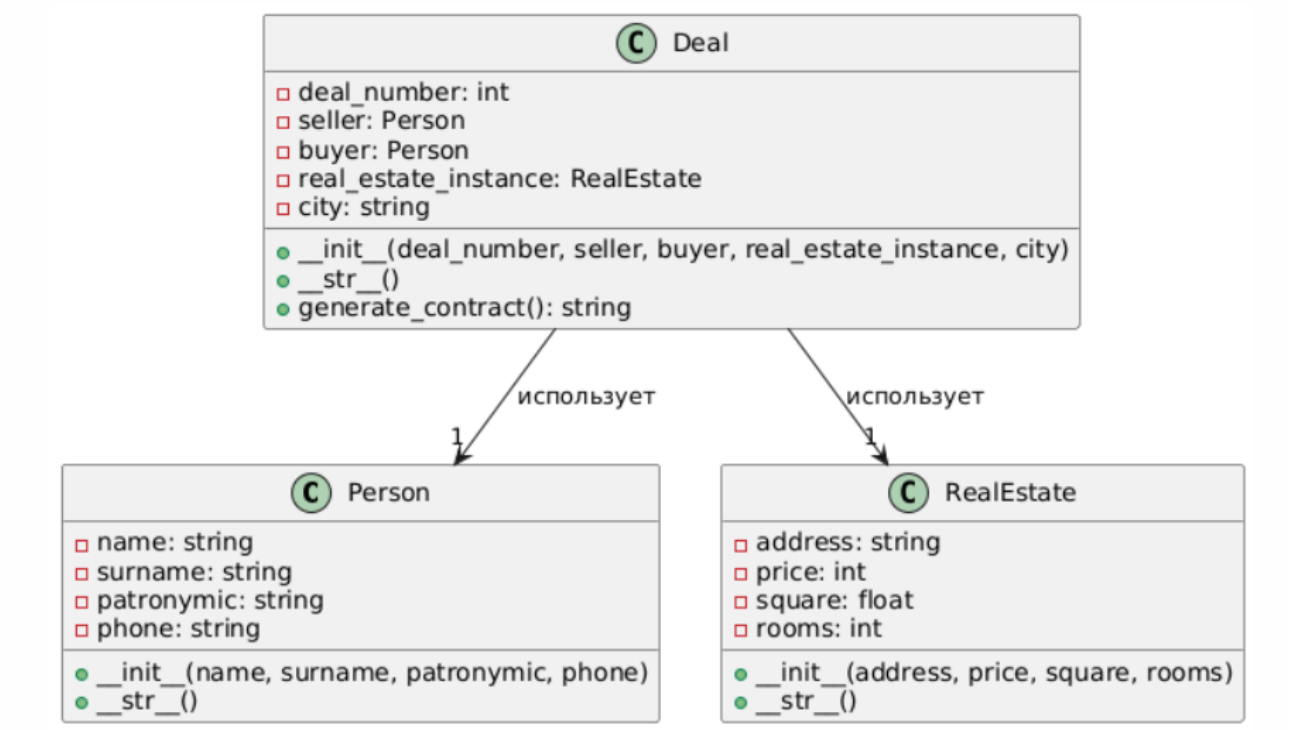

Рисунок 11. Диаграмма классов

Приложение 2

Блок-схема модуля main.py

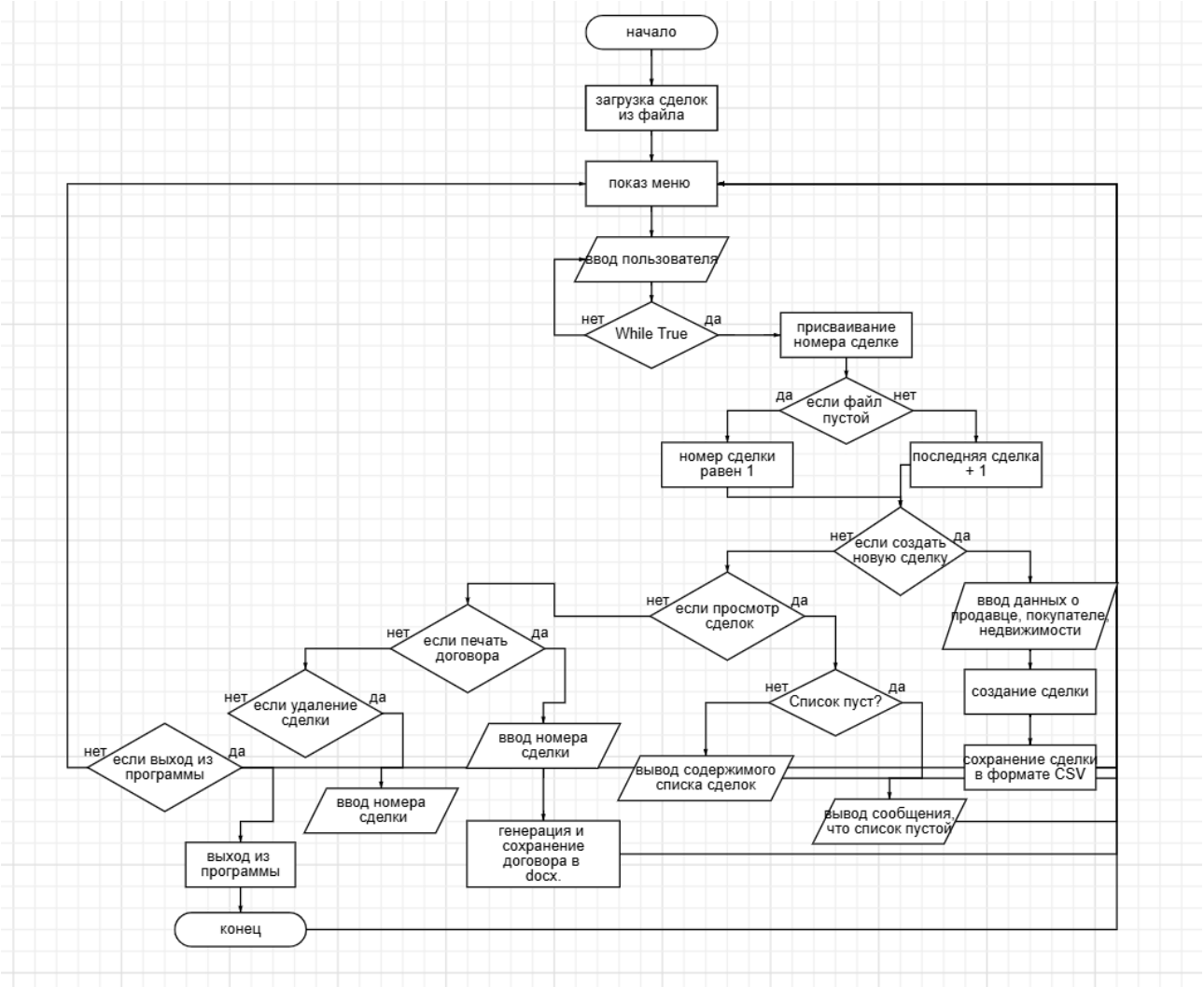

Рисунок 12. Блок-схема модуля main.py

Приложение 3

Листинг кода main.py

```
from person import Person
from real_estate import RealEstate
from deal import Deal
from file_operations import load_deals_from_file,
save_deal_to_file
from contract_saver import save_contract_to_docx

def main():
    """Основная программа для управления сделками."""
    deals_list = load_deals_from_file('deals_output.txt')

    while True:

        if deals_list:
            next_deal_number = max(deal.deal_number for deal in
deals_list) + 1
        else:
            next_deal_number = 1  # Если список пуст, начинаем с
1

        action = input(
                f"1.Введите 'new' для создания новой сделки\n"
                f"2.Введите 'view' для просмотра существующих
сделок\n"
                f"3.Введите 'contract' для создания договора для
сделки\n"
                f"4.Введите 'delete' для удаления сделки из
файла\n"
                f"5.Введите 'exit' для выхода из программы:\n"
        ).strip()

        if action == 'new':
            # Ввод данных для новой сделки
            seller_name = input("Имя продавца: ").strip()
            if not seller_name:
                print("Ошибка: имя продавца не может быть
пустым.")
                continue

            seller_surname = input("Фамилия продавца: ").strip()
            if not seller_surname:
                print("Ошибка: фамилия продавца не может быть
пустой.")
                continue
```

```
            seller_patronymic = input("Отчество продавца
(необязательно): ").strip()

            seller_phone = input("Телефон продавца (запрещены
пробелы): ").strip()
            if ' ' in seller_phone:  # Проверка на наличие
пробелов
                print("Ошибка: телефон не должен содержать
пробелы.")
                continue
            if not seller_phone.isdigit():
                print("Ошибка: телефон должен содержать только
цифры.")
                continue

            buyer_name = input("Имя покупателя: ").strip()
            if not buyer_name:
                print("Ошибка: имя покупателя не может быть
пустым.")
                continue

            buyer_surname = input("Фамилия покупателя:
").strip()
            if not buyer_surname:
                print("Ошибка: фамилия покупателя не может быть
пустой.")
                continue

            buyer_patronymic = input("Отчество покупателя
(необязательно): ").strip()

            buyer_phone = input("Телефон покупателя (запрещены
пробелы): ").strip()
            if ' ' in buyer_phone:  # Проверка на наличие
пробелов
                print("Ошибка: телефон не должен содержать
пробелы.")
                continue
            if not buyer_phone.isdigit():
                print("Ошибка: телефон должен содержать только
цифры.")
                continue

            real_estate_address = input("Адрес недвижимости
(запятые заменяются на точки): ").strip()
            if not real_estate_address:
                print("Ошибка: адрес недвижимости не может быть
пустым.")
```

```
                continue
            real_estate_address =
real_estate_address.replace(',', '.')

            try:
                real_estate_price = int(input("Цена (руб.,
только целое число): ").strip())
                if real_estate_price <= 0:
                    print("Ошибка: цена должна быть
положительным числом.")
                    continue
            except ValueError:
                print("Ошибка: цена должна быть целым числом.")
                continue

            try:
                real_estate_square = float(input("Площадь (м²,
может быть дробным числом): ").strip())
                if real_estate_square <= 0:
                    print("Ошибка: площадь должна быть
положительным числом.")
                    continue
            except ValueError:
                print("Ошибка: площадь должна быть числом.")
                continue

            try:
                real_estate_rooms = int(input("Количество комнат
(только целое число): ").strip())
                if real_estate_rooms <= 0:
                    print("Ошибка: количество комнат должно быть
положительным числом.")
                    continue
            except ValueError:
                print("Ошибка: количество комнат должно быть
целым числом.")
                continue

            city = input("Город сделки: ").strip()
            if not city:
                print("Ошибка: город сделки не может быть
пустым.")
                continue

            print("Сделка успешно зарегистрирована!\n")

            seller = Person(seller_name, seller_surname,
seller_patronymic, seller_phone)
```

```
            buyer = Person(buyer_name, buyer_surname,
buyer_patronymic, buyer_phone)
            real_estate_instance =
RealEstate(real_estate_address, real_estate_price,
real_estate_square, real_estate_rooms)

            deal = Deal(next_deal_number, seller, buyer,
real_estate_instance, city)
            deals_list.append(deal)
            save_deal_to_file('deals_output.txt', deals_list)

        elif action == 'view':
            if not deals_list:
                print("\nСписок сделок пуст.")
            else:
                print("\nСписок сделок:")
                for deal in deals_list:
                    print(deal)
                    print()

        elif action == 'contract':
            try:
                deal_number_input = int(input("Введите номер
сделки для генерации договора (только целое число): ").strip())
            except ValueError:
                print("Ошибка: номер сделки должен быть целым
числом.")
                continue

            for deal in deals_list:
                if deal.deal_number == deal_number_input:
                    save_contract_to_docx(deal,
f"contract_{deal_number_input}.docx")
                    break
            else:
                print("Сделка не найдена.")

        elif action == 'delete':
            try:
                deal_number_input = int(input("Введите номер
сделки для удаления (только целое число): ").strip())
            except ValueError:
                print("Ошибка: номер сделки должен быть целым
числом.")
                continue

            for i, deal in enumerate(deals_list):
                if deal.deal_number == deal_number_input:
```

```
                    del deals_list[i]  # Удаляем сделку
                    print(f"Сделка #{deal_number_input}
удалена.")
                    # Корректируем номера оставшихся сделок
                    for j in range(i, len(deals_list)):
                        deals_list[j].deal_number -= 1  #
Уменьшаем номера сделок
                    save_deal_to_file('deals_output.txt',
deals_list)  # Сохраняем обновленный список сделок
                    break
            else:
                print("Сделка с таким номером не найдена.")

        elif action == 'exit':
            break

        print("\n")

if __name__ == '__main__':
    main()
```

Приложение 4

Листинг кода person.py

```
class Person:
    """Класс, представляющий человека (покупателя или
продавца)."""

    def __init__(self, name, surname, patronymic, phone):
        """Инициализирует объект Person."""

        self.name = name
        self.surname = surname
        self.patronymic = patronymic
        self.phone = phone

    def __str__(self):
        """Возвращает строковое представление объекта Person."""
        return f"{self.surname} {self.name} {self.patronymic},
Телефон: {self.phone}"
```

Приложение 5

Листинг кода real_estate.py

```
class RealEstate:
    """Класс, представляющий недвижимость."""

    def __init__(self, address, price, square, rooms):
        """Инициализирует объект RealEstate."""

        self.address = address
        self.price = price
        self.square = square
        self.rooms = rooms

    def __str__(self):
        """Возвращает строковое представление
RealEstate."""
        return f"{self.rooms} комнат(ы), {self.square}
м², Адрес: {self.address}, Цена: {self.price} рублей"
```

Приложение 6

Листинг кода deal.py

```
class Deal:
    """Класс, представляющий сделку с недвижимостью."""

    def __init__(self, deal_number, seller, buyer,
real_estate_instance, city):
        """Инициализирует объект Deal."""

        self.deal_number = deal_number
        self.seller = seller
        self.buyer = buyer
        self.real_estate_instance = real_estate_instance
        self.city = city

    def __str__(self):
        """Возвращает строковое представление сделки."""
        return (f"Сделка #{self.deal_number}\n"
                f"Город: {self.city}\n"
                f"Продавец: {self.seller}\n"
                f"Покупатель: {self.buyer}\n"
                f"Недвижимость: {self.real_estate_instance}")

    def generate_contract(self):
        """Генерирует текст договора и возвращает"""

        return f"""
г. {self.city}
«_____» _____ 2024 года

Продавец: {self.seller.surname} {self.seller.name}
{self.seller.patronymic}, Телефон: {self.seller.phone}
Покупатель: {self.buyer.surname} {self.buyer.name}
{self.buyer.patronymic}, Телефон: {self.buyer.phone}

1. Предмет Договора.
1.1. Продавец продает, а Покупатель покупает квартиру,
расположенную по адресу: {self.real_estate_instance.address}.
Квартира состоит из {self.real_estate_instance.rooms} комнат,
общей площадью {self.real_estate_instance.square} м².
Цена квартиры составляет {self.real_estate_instance.price}
рублей.

АДРЕСА СТОРОН

Продавец:
{self.seller.surname} {self.seller.name}
```

```
{self.seller.patronymic}

Покупатель:
{self.buyer.surname} {self.buyer.name} {self.buyer.patronymic}

ПОДПИСИ СТОРОН

Продавец:
_________________/_________________
(подпись)                                                    (ФИО)

Покупатель:
_________________/_________________
(подпись)                                                    (ФИО)
"""
```

Приложение 7

Листинг кода file_operations.py

```
from person import Person
from real_estate import RealEstate
from deal import Deal


def save_deal_to_file(filename, deals):
    """Сохраняет список сделок в текстовый файл."""
    with open(filename, 'w', encoding='utf-8') as file:
        for deal in deals:

file.write(f"{deal.deal_number};{deal.seller.name},{deal.seller.
surname},{deal.seller.patronymic},{deal.seller.phone};"

f"{deal.buyer.name},{deal.buyer.surname},{deal.buyer.patronymic}
,{deal.buyer.phone};"

f"{deal.real_estate_instance.address},{deal.real_estate_instance
.price},"

f"{deal.real_estate_instance.square},{deal.real_estate_instance.
rooms};"
                        f"{deal.city}\n")


def load_deals_from_file(filename):
    """Загружает сделки из текстового файла."""
    deals = []
    with open(filename, 'r', encoding='utf-8') as file:
        for line in file:
            parts = line.strip().split(';')

            deal_number = int(parts[0])
            seller_info = parts[1].split(',')
            buyer_info = parts[2].split(',')
            real_estate_info = parts[3].split(',')
            city = parts[4]

            seller = Person(seller_info[0], seller_info[1],
seller_info[2], seller_info[3])
            buyer = Person(buyer_info[0], buyer_info[1],
buyer_info[2], buyer_info[3])
            real_estate_instance =
RealEstate(real_estate_info[0], int(real_estate_info[1]),

float(real_estate_info[2]), int(real_estate_info[3]))
```

```
            deals.append(Deal(deal_number, seller, buyer,
real_estate_instance, city))

    return deals
```

Приложение 8

Листинг кода contract_save.py

```
from docx import Document
from docx.shared import Pt

def save_contract_to_docx(deal, filename):
    """Сохраняет договор в формате .docx."""

    doc = Document()
    doc.add_heading('Договор купли-продажи квартиры', level=1)


    # Устанавливаем стиль шрифта
    style = doc.styles['Normal']
    style.font.name = 'Times New Roman'
    style.font.size = Pt(10)

    contract_text = deal.generate_contract()

    for line in contract_text.split("\n"):
        doc.add_paragraph(line)

    doc.save(filename)
    print(f"Договор сохранён в файл: {filename}")
```

Приложение 9

Тест-план

ID теста	Описание теста (тип)	Предусловия	Шаги для воспроизведения	Ожидаемый результат	Фактический результат
1	Создание новой сделки (позитивный)	Программа запущена, пользователь выбрал пункт "Создать новую сделку"	1. Ввести данные для новой сделки (имя и фамилия продавца и покупателя, информация о недвижимости и т.д.). 2. Сохранить сделку.	Сделка создаётся, выводится сообщение об успешной регистрации. Сделка сохраняется в файл.	Сделка создаётся, выводится сообщение об успешной регистрации. Сделка сохраняется в файл.
2	Просмотр существующих сделок (позитивный)	Программа запущена, сделки уже загружены.	1. Ввести команду "Просмотреть сделки". 2. Просмотреть список сделок.	Все загруженные сделки отображаются на экране.	Все загруженные сделки отображаются на экране.
3	Генерация договора для сделки (позитивный)	Программа запущена, сделки загружены..	1. Ввести номер сделки для генерации договора. 2. Сохранить договор в файл с именем contract_<deal_number>.docx	Создаётся файл договора в формате .docx с корректными данными о сделке.	Создаётся файл договора в формате .docx с корректными данными о сделке.
4	Удаление сделки (позитивный)	Программа запущена, сделки загружены	1. Ввести номер сделки для удаления. 2. Сделка удаляется из списка, и номер оставшихся сделок корректируется.	Сделка удаляется, номера оставшихся сделок обновляются, данные сохраняются в файл.	Сделка удаляется, номера оставшихся сделок обновляются, данные сохраняются в файл.
5	Проверка корректности сохранения сделок в файл (позитивный)	Программа запущена, сделки уже загружены.	1. Запустить программу. 2. Проверить, что данные о сделках корректно записаны в файл deals_output.txt.	Все данные о сделках должны быть сохранены в файл в	Все данные о сделках должны быть сохранены в файл в

				правильном формате	правильном формате
6	Проверка загрузки сделок из файла (позитивный)	Файл deals_output.txt существует и содержит корректные данные.	1. Запустить программу и загрузить данные из файла. 2. Проверить, что все сделки были загружены и корректно отображаются.	Все сделки загружаются и отображаются в программе.	Все сделки загружаются и отображаются в программе.
7	Проверка загрузки сделок из пустого файла (негативный)	Файл deals_output.txt пуст.	1. Запустить программу и загрузить данные из пустого файла.	Программа сообщает, что не удалось загрузить данные, так как файл пуст.	Программа сообщает, что не удалось загрузить данные, так как файл пуст.
8	Проверка правильности формата договора (позитивный)	Программа запущена, сделка с номером 1 существует.	1. Ввести номер сделки для создания договора. 2. Проверить корректность формата генерированного договора.	Договор сохраняется в формате .docx с корректным содержанием: все данные о продавце, покупателе, недвижимости и условиях.	Договор сохраняется в формате .docx с корректным содержанием: все данные о продавце, покупателе, недвижимости и условиях.
9	Проверка корректности вывода информации о сделке (позитивный)	Программа запущена, сделки уже загружены.	1. Ввести команду "Просмотреть сделки". 2. Проверить, что информация о сделке выводится корректно (например, корректный адрес, цена, покупатель).	Информация о сделках выводится в корректном формате, все данные отображаются правильно.	Информация о сделках выводится в корректном формате, все данные отображаются правильно.
10	Проверка на создание договора с несуществующ	Программа запущена, сделки загружены.	1. Ввести несуществующий номер сделки для создания договора	Выводится сообщение об ошибке:	Выводится сообщение об ошибке:

	им номером сделки (негативный)			"Сделка не найдена."	"Сделка не найдена."
11	Создание новой сделки с неполными данными (негативный)	Программа запущена, пользователь выбрал пункт "Создать новую сделку".	1. Ввести неполные данные для новой сделки, например, пропустить обязательные поля, такие как телефон продавца или цена недвижимости. 2. Попробовать сохранить сделку.	Программа должна отобразить ошибку ввода и не сохранить сделку, если обязательные данные отсутствуют.	Программа не выдаёт ошибку и продолжает работу дальше.
12	Проверка правильности адреса недвижимости (позитивный)	Программа запущена, сделка добавлена	1. Ввести корректный адрес недвижимости, который не содержит запятых. 2. Убедиться, что адрес сохраняется в правильном формате и без изменений	Адрес недвижимости сохраняется корректно, без изменения и в правильном формате (например, "Москва, ул. Ленина, 5").	Адрес недвижимости сохраняется корректно, без изменения и в правильном формате (например, "Москва, ул. Ленина, 5").
13	Проверка ввода данных с невалидными символами (негативный)	Программа запущена, пользователь выбрал пункт "Создать новую сделку".	1. Ввести невалидные символы в числовые поля, такие как цена или площадь (например, "10000a" или "50.5m"). 2. Попробовать сохранить сделку.	Программа должна отобразить ошибку и не сохранить сделку, если введены невалидные символы в числовые поля.	Программа должна отобразить ошибку и не сохранить сделку, если введены невалидные символы в числовые поля.

Приложение 10

Диаграмма прецендентов

Рисунок 9. Диаграмма прецедентов

Приложение 11

Сценарий использования

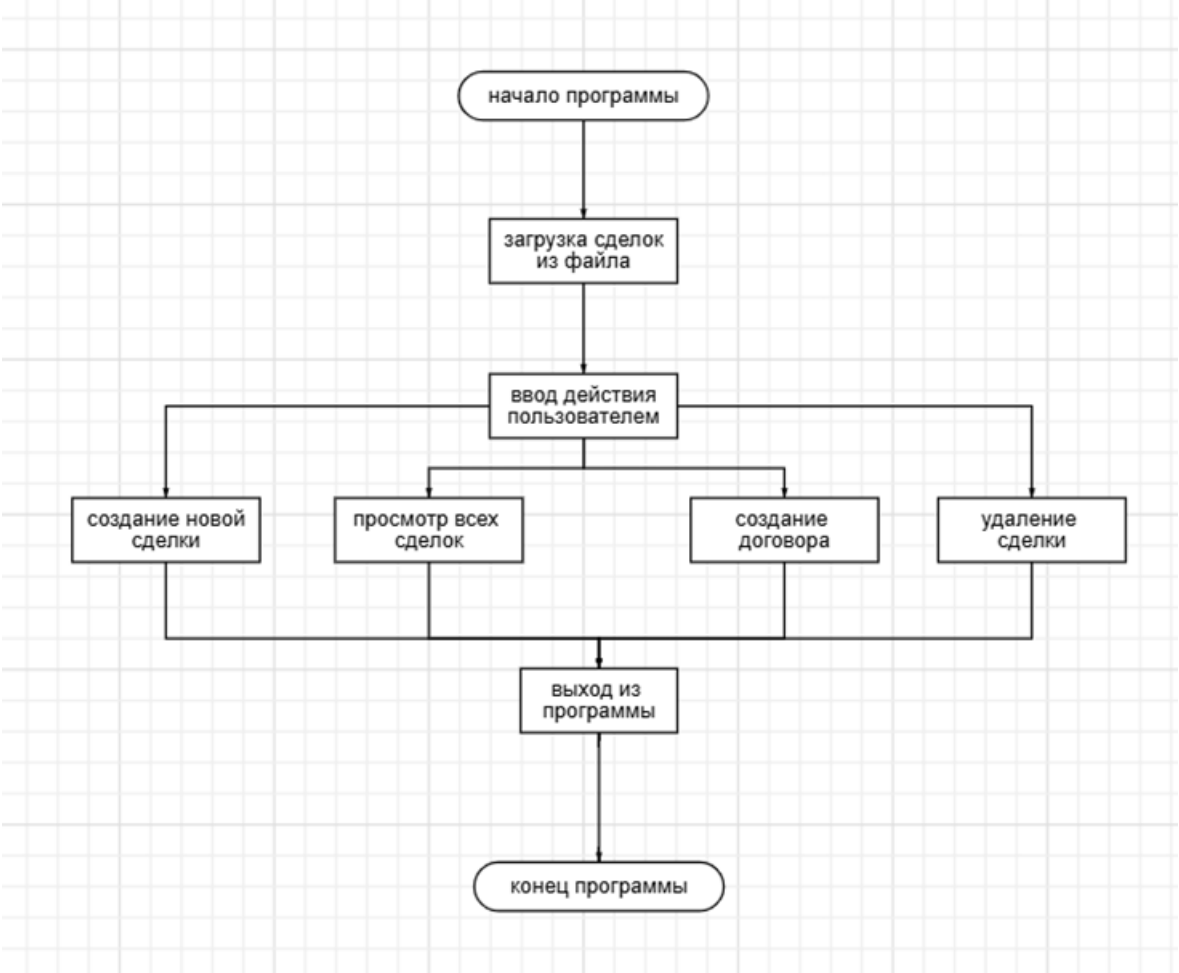

Рисунок 10. Сценарий использования

Приложение 12

Дизайн приложения

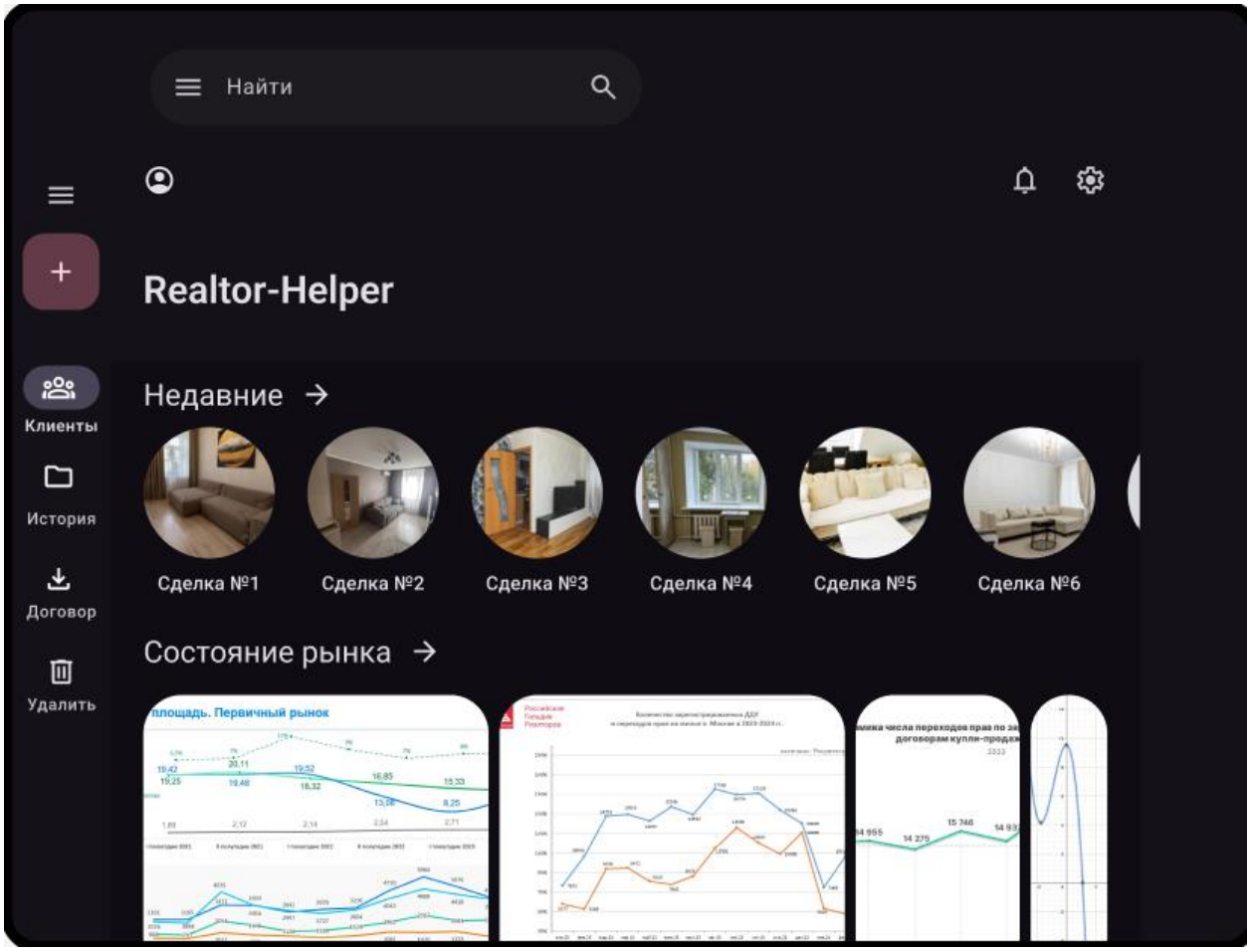

Рисунок 15. Дизайн главного меню для приложения

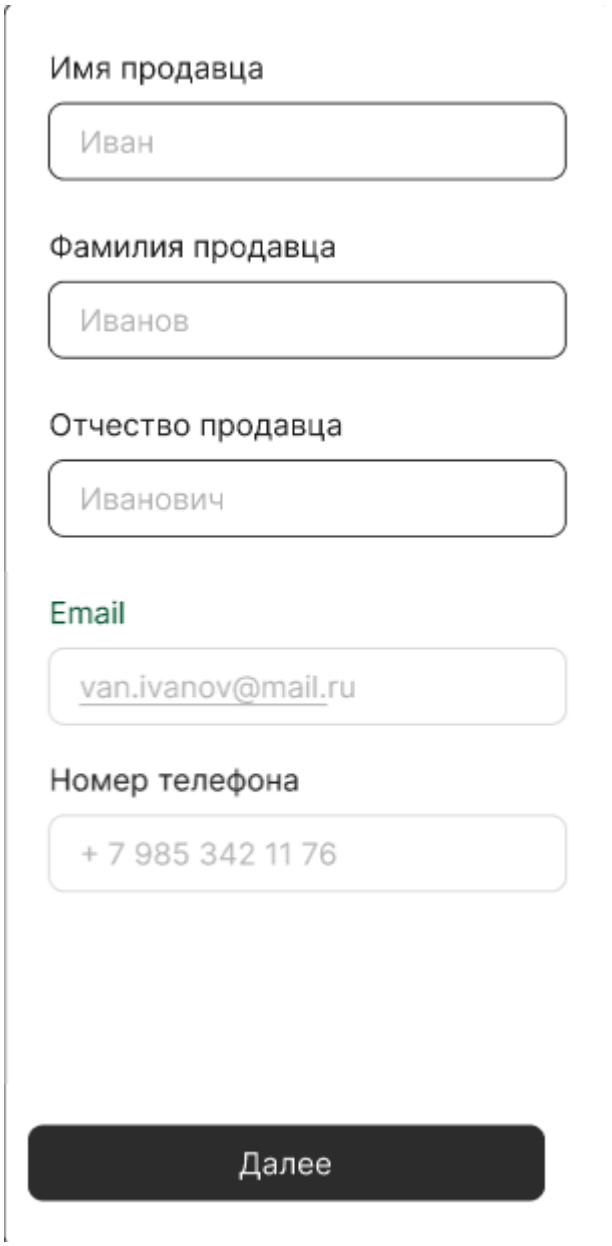

Рисунок 16. Дизайн полей для ввода данных