### Частное учреждение образования

### «Колледж бизнеса и права»

### СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО АВТОМАТИЗАЦИИ РАБОТЫ СОТРУДНИКА АГЕНТСТВА НЕДВИЖИМОСТИ УП «Скиба»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

### к курсовому проектированию по дисциплине

### «Конструирование программ и языки программирования»

КП Т.196020.401 ПЗ

Руководитель проекта (А.В.Кривошеина)

Учащийся (Ю.П.Скибицкий )

Минск, 2024

**СОДЕРЖАНИЕ**

Изм.

Изм.

Лист

Лист

№ докум.

№ докум.

Подпись

Подпись

Дата

Дата

Лист

Лист

3

3

КП Т.196020.401 ПЗ

ПЗ

ДП Т.517074.401 ПЗ

Разраб.

Разраб.

Скибицкий Ю.П.

Мовчан П.И.

Провер.

Провер.

Кривошеина А.В.

Багласова Е.В.

Т. контр.

Т. контр.

*.*

*Якимович К.О.*

Н. контр.

Н. контр.

Багласова Е.В.

Утверд.

Утверд.

*Багласова Т.Г.*

### *СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО* *АВТОМАТИЗАЦИИ РАБОТЫ СОТРУДНИКА АГЕНТСТВА НЕДВИЖИМОСТИ УП «Скиба»*

*Программное средство для автоматизации учета заявок жителей Фрунзенского района г.Минска*

Лит.

Лит.

Листов

Листов

49

54

КБП

КБП

[Введение 3](#_Toc169941869)

[1. Описание задачи 4](#_Toc169941870)

[1.1. Анализ предметной области 4](#_Toc169941871)

[1.2. Постановка задачи 5](#_Toc169941872)

[2. Проектирование системы 6](#_Toc169941873)

[2.1. Требования к приложению 6](#_Toc169941874)

[2.2. Проектирование модели 6](#_Toc169941875)

[2.3. Организация данных 9](#_Toc169941876)

[2.4. Концептуальный прототип 11](#_Toc169941877)

[3. Описание реализации программного средства 14](#_Toc169941878)

[3.1. Инструменты разработки и применения технологии 14](#_Toc169941879)

[3.2. Порядок авторизации пользователей 15](#_Toc169941880)

[3.3. Организация данных 16](#_Toc169941881)

[3.4. Функции: логическая и физическая организация 18](#_Toc169941882)

[3.5. Входные и выходные данные 22](#_Toc169941883)

[3.6. Функциональное тестирование 23](#_Toc169941884)

[3.7. Описание справочной системы 27](#_Toc169941885)

[4. Применение 28](#_Toc169941886)

[4.1. Назначение программного средства 28](#_Toc169941887)

[4.2. Условия применения 28](#_Toc169941888)

[Заключение 30](#_Toc169941889)

[Список использованных источников 31](#_Toc169941890)

[Приложение А 33](#_Toc169941891)

[Приложение Б 42](#_Toc169941892)

[Приложение В 43](#_Toc169941893)

# **Введение**

В современном мире, где технологии проникают во все сферы жизни, отрасль транспортной логистики не является исключением. Автоматизация процессов становится все более актуальной и важной задачей, позволяющей минимизировать потери рабочего времени персонала, экономить на его численности и повышать уровень сервиса.

Целью курсового проекта «Автоматизированное рабочее место менеджера по работе с клиентами в агентстве недвижимости» является создание программного средства, которое позволит автоматизировать работу агента в агентстве недвижимости.

Пояснительная записка к курсовому проекту содержит пять разделов:

Описание задачи: В этом разделе раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть решены. Описываются существующие аналоги.

Проектирование системы: В этом разделе перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения. Описываются требования к приложению, строится концептуальный прототип, описывается логическая и физическая организация данных, проектируется справочная система.

Описание реализации программного средства: В этом разделе представлены общие сведения о программном средстве и его функциональном назначении, входные и выходные данные.

Применение предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы.

Заключение будет проанализировано созданное программное средство, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А будет содержать текст программы.

Графическая часть содержит в себе все необходимые диаграммы для проектирования данного программного средства, такие как:

* диаграмма вариантов использования;
* диаграмма деятельности;
* диаграмма классов;
* диаграмма последовательности.

# **Описание задачи**

# **Анализ предметной области**

Агентство недвижимости – индивидуальный предприниматель или юридическое лицо, специализирующееся на комплексном решении вопроса рыночного оборота недвижимости, путем организации поиска потенциального продавца и/или покупателя, обеспечения юридической «чистоты» сделок, организации документооборота, организации процесса заключения сделок купли-продажи, аренды недвижимости.

Недвижимость – вид имущества, признаваемого в законодательном порядке недвижимым.

Важными элементами агентства недвижимости являются:

* **управление недвижимостью**: это включает в себя прогнозирование спроса, определение оптимальной недвижимости и управление остатками недвижимости;
* **управление информацией**: это включает в себя сбор, обработку и анализ данных о рынке недвижимости для поддержки принятия решений в области покупок/продаж;
* **управление рисками**: это включает в себя идентификацию, оценку и управление рисками, связанными с сделками, юридическими аспектами.

Внутреннее агентство недвижимости занимается решением множества задач, которые контролируют юридические процессы. Особенности недвижимости, такие как площадь и кол-во комнат, напрямую влияют на выбор квартиры клиента.

Агент выполняет следующие задачи:

* подсчитать стоимость аренды или продажи;
* выбор оптимального для клиента недвижимости;
* оповещать об освобождающихся недвижимостях;
* и другое.

Итоговые показатели, такие как площадь, стоимость недвижимости, месторасположение, могут быть рассчитаны на основе собранных данных.

Автоматизация процессов с помощью компьютерных систем позволяет значительно увеличить эффективность работы, минимизировать вероятность ошибок, связанных с человеческим фактором, и сократить время на выполнение рутинных задач.

Компьютерная обработка информации обеспечивает быстрый доступ к актуальным данным, возможность их анализа и принятия обоснованных решений на основе полученной информации. Это особенно важно в области агентства недвижимости, где требуется оперативное реагирование на изменяющиеся условия и быстрое принятие решений.

Потенциальной аудиторией данного приложения являются агентства недвижимости, их сотрудники (агенты) и клиенты, которым необходима оперативная и точная информация о недвижимости и их сделках.

# **Постановка задачи**

Исходя из анализа предметной области, можно выделить следующие задачи, подлежащие автоматизации:

* осуществить ведение базы данных, содержащей информацию о объектах недвижимости, сотрудниках, клиентах и договорах;
* предоставить возможность сортировки данных по критериям;
* автоматизировать подбор недвижимости для клиента по его желанию;
* автоматизировать подсчет стоимости аренды или продажи;
* автоматизировать генерацию отчетов;

В сравнении с другими приложениями для агентства недвижимости, такими как «1С:Предприятие», «SAP», «Relog», «Axelot», «Fretron», «Оracle Transportation Management» и другими, наше приложение Скиба имеет следующие преимущества:

* автоматическое управление объектами: Скиба автоматически обновляет информацию о наличии недвижимости на основе данных о покупках и продажах, что помогает агентам определить и принять нужное решение;
* гибкость и масштабируемость: Скиба спроектирован так, чтобы быть гибким и масштабируемым, позволяя легко добавлять новые функции и поддерживать большое количество пользователей;
* пользовательский Макет: Скиба имеет интуитивно понятный и простой в использовании пользовательский Макет, что облегчит работу с ним для всех пользователей, независимо от их уровня технической грамотности.

Таким образом, Скиба предлагает уникальный набор функций, которые помогут улучшить эффективность и производительность в области транспортной логистики, делая его важным инструментом для любого бизнеса, который хочет оставаться конкурентоспособным в современном быстро меняющемся бизнес-окружении.

# **Проектирование системы**

# **Требования к приложению**

В приложении Скиба будет реализована система аутентификации для обеспечения безопасности данных пользователя. Это означает, что доступ к функционалу приложения будет возможен только после ввода уникальных учетных данных (имя пользователя и пароль). Это помогает предотвратить несанкционированный доступ и защитить конфиденциальность данных пользователя.

Макет приложения Скиба будет разработан таким образом, чтобы обеспечить простоту и удобство использования. Однако, некоторые функции могут быть доступны только авторизованным пользователям. Кроме того, некоторые функции могут быть ограничены в зависимости от роли пользователя в системе.

Макет приложения Скиба будет иметь четкую и последовательную структуру. Все элементы управления будут иметь стандартный размер, форму и расположение, чтобы облегчить навигацию пользователям. Шрифт будет выбран таким образом, чтобы обеспечить хорошую читаемость на всех устройствах. Кроме того, цветовая схема и стиль дизайна будут выбраны так, чтобы обеспечить приятное визуальное восприятие и удобство использования.

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению персонального компьютера, являются:

* процессор 800 МГц и выше;
* оперативная память 128 Мбайт и более;
* свободное место на диске 100 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* монитор;
* мышь, клавиатура;
* принтер.

# **Проектирование модели**

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями.

Наиболее распространенные средства моделирования являются: диаграмма вариантов использования, диаграмма классов, диаграмма последовательности, диаграмма деятельности.

Диаграмма вариантов использования – это тип диаграммы, используемой в области разработки программного обеспечения для визуализации системы с точки зрения её акторов и вариантов использования.

Основные элементы диаграммы вариантов использования:

* актеры: ­ актеры представляют собой людей или системы, которые взаимодействуют с системой. Они находятся вне системы и могут быть как людьми (например, пользователи или администраторы), так и другими системами или устройствами;
* варианты использования: варианты использования – это функции или действия, которые система может выполнять в ответ на запросы от актеров. Они представляют собой цели, которые ­ актеры хотят достичь с помощью системы;
* отношения: отношения между актёрами и вариантами использования показывают, как ­ актеры и варианты использования взаимодействуют друг с другом. Они могут быть обозначены стрелками или линиями.

Диаграмма вариантов использования помогает разработчикам и стейкхолдерам лучше понять, как система будет работать и какие функции она будет выполнять. Она также помогает определить требования к системе и спланировать её разработку. Это важный инструмент в процессе проектирования системы.

Диаграмма вариантов использования представлена на листе 1 графической части.

На этой диаграмме можно выделить трёх пользователей программным продуктом. Клиент, который может обратиться к агенту с просьбой о покупке/продаже недвижимости. Агент, который добавляет клиента в систему и может его редактировать, а также создавать заявки на покупку/продажу недвижимости.

Диаграмма классов – это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их атрибуты (полей), методы, Макеты и взаимосвязи (отношений) между ними. Она используется для визуализации структуры классов в системе и их взаимосвязей.

Вот основные элементы диаграммы классов:

* Классы: Классы представляют собой ключевые элементы в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента: имя класса, поля (атрибуты) класса и методы класса.
* Отношения: Отношения между классами на диаграмме классов могут представлять отношения, такие как ассоциация, наследование и реализация.

Диаграмма классов соответствует принципам объектно-ориентированного программирования (ООП) и является одним из базовых инструментов проектирования ООП-систем. Она помогает лучше понимать структуру системы и ее компоненты.

В данном приложении присутствуют следующие классы:

* клиенты;
* договора;
* объекты.

Диаграмма классов представлена на листе 2 графической части.

Диаграмма деятельности – это графическое представление процессов и задач, выполняемых в рамках определенного проекта или деятельности. Эта диаграмма используется для анализа и оптимизации бизнес-процессов, планирования работы, принятия решений и управления проектами.

Основные элементы диаграммы деятельности:

* начальный узел: Начальный узел активности является первым элементом диаграммы деятельности и представляет начальную точку процесса;
* конечный узел активности: Конечный узел активности является последним элементом диаграммы деятельности и обозначает завершение процесса;
* действие: Действия представляют собой отдельные шаги или операции, которые выполняются в процессе;
* поток управления: Эти потоки показывают последовательность действий и передачу данных между действиями;
* узел принятия решений: Эти узлы используются для представления ветвлений и слияний в потоке управления.

Диаграмма деятельности позволяет визуализировать последовательность задач и операций, которые нужно выполнить, чтобы достичь желаемого результата. Она является мощным инструментом для анализа, планирования и управления проектами и бизнес-процессами.

В рамках проекта рассмотрим функцию добавления заказа на перевозку.

Диаграмма деятельности для создания посылки представлена на листе 3 графической части.

Диаграмма последовательности – это UML-диаграмма, которая показывает взаимодействие между объектами в системе в виде последовательности сообщений, действий и операций. Она отображает порядок выполнения действий и обмена информацией между объектами во времени.

Вот основные элементы диаграммы последовательности:

* объекты: представлены горизонтальной осью диаграммы. Каждый объект имеет свою колонку;
* сообщения: передаются горизонтально между объектами и располагаются вертикально в хронологическом порядке. Они представлены стрелками;
* время: представлено вертикальной осью диаграммы. Жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) информационной системы в рамках прецедента показаны на единой временной оси.

Диаграмма последовательности помогает анализировать и понимать взаимодействие между объектами в системе, а также порядок, в котором происходит взаимодействие. Она является важным инструментом для анализа и проектирования систем.

Диаграмма последовательности для создания посылки представлена на листе 4 графической части.

# **Организация данных**

Наиболее распространенным средством моделирования таких типов документации являются схемы-сети (ERD), которые применяются для графического представления многих элементов разработанных программ систем и представляют некоторые габаритные размеры обозначений для определения данных и отношений между ними. С помощью такой диаграммы можно описать отдельные компоненты концептуальной модели или выразить свою точку зрения взаимосвязей между ними, иногда какое значение или реализацию системы. Основными потенциалом данной нотации является понятная сущность в каждом случае каждый рассматриваемый объект может явиться экземпляром одного и только одного сущности, должен иметь уникальное имя или идентификатор, а также отличается от других экземпляров данной сущности. Следовательно: как фотонные линии короткого ассортимента между отделами сущности.

Графическая модель данных строится таким образом, чтобы связь между отделениями сущности органа не только естественный характер соответствующего отношения, но и дополнительные аспекты обязательности связей.

Информационная модель представлена на диаграмме «Сущность-связь» на рисунки 2.2

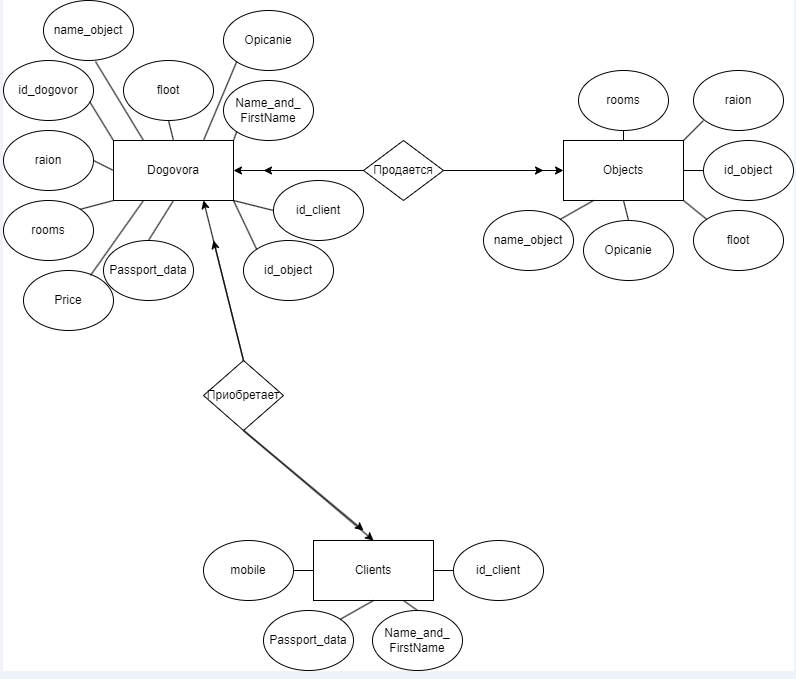


Рисунок 2.2 – Диаграмма «Сущность-связь»

Исследовав предметную область, можно выделить следующие сущности: «Клиенты», «Объекты», «Договора».

Для сущности «Объекты» атрибутами будут являться:

* наименование объекта;
* кол-во комнат;
* район;
* этаж;
* описание.

Для сущности «Клиенты» атрибутами будут являться:

* ФИО;
* паспортный номер;
* номер телефона.

Для сущности «Договора» атрибутами будут являться:

* ФИО;
* паспортный номер;
* наименование объекта;
* кол-во комнат;
* район;
* этаж;
* описание.

Типы связей между сущностями:

* Клиенты-Договора: многие к многим;
* Объекты-Договора: многие к многим.

# **Концептуальный прототип**

Концептуальный прототип состоит из описания внешнего вида пользовательского Макета, а именно, элементов управления.

Разрабатываемое приложение будет содержать две основные формы: «Главная» и «Авторизация».

В интерфейсе формы «Авторизация» есть два поля для ввода текста, подписанные как “Фамилия” и “Пароль”. Под полями для ввода находится кнопка с надписью “Войти”, которую пользователи могут нажать для входа после ввода своих учетных данных. Внизу интерфейса есть еще одна кнопка с надписью “Отмена”, если пользователь желает покинуть приложение. Макет формы «Авторизация» представлен в приложении В на рисунке В.1.

* Интерфейс главной формы для работников агентства недвижимости состоит из:
* **кнопка** «Объекты»: данная кнопка позволяет взаимодействовать пользователю с объектами;
* кнопка «Клиенты»: данная кнопка позволяет пользователю взаимодействовать с клиентами;
* **кнопка** «Договоры»: данная кнопка позволяет пользователю взаимодействовать с клиентами;
* **кнопка «Выход»**: Пользователи могут использовать эту кнопку для выхода из системы.

Макет формы «Главной» представлен в приложении В на рисунке В.3.

Макет «Объекты» состоит из:

* **кнопка** «Найти» – позволяет пользователю найти ему интересующий объект по индексу;
* **кнопка** «Добавить объект» – позволяет пользователю добавить нужный для него объект;
* **кнопка** «Удаление объекта» – позволяет пользователю удалить нужный для него объект;
* **кнопка** «На главную» – позволяет пользователю вернуться на главную форму;
* **кнопка** «Выход» – позволяет пользователю покинуть данное приложение;
* **поле «Поиск по индексу»**: пользователь вводит индекс объекта.

Макет формы «Объекты» представлен в приложении В на рисунке В.4.

Макет формы «Клиенты» состоит из:

* **кнопка** «Найти» – позволяет пользователю найти ему интересующего клиента по индексу;
* **кнопка** «Добавить клиента» – позволяет пользователю добавить нужного для него клиента;
* **кнопка** «Удалить клиента» – позволяет пользователю удалить нужного для него клиента;
* **кнопка** «На главную» – позволяет пользователю вернуться на главную форму;
* **кнопка** «Выход» – позволяет пользователю покинуть данное приложение;
* **поле «Поиск по индексу»**: пользователь вводит индекс клиента.

Макет формы «Клиенты» представлен в приложении В на рисунке В.7.

Макет формы «Договоры» состоит из:

* **кнопка** «Найти» – позволяет пользователю найти ему интересующего клиента по индексу;
* **кнопка** «Добавить договор» – позволяет пользователю добавить нужного для него клиента;
* **кнопка** «Удалить клиента» – позволяет пользователю удалить нужного для него клиента;
* **кнопка** «На главную» – позволяет пользователю вернуться на главную форму;
* **кнопка** «Выход» – позволяет пользователю покинуть данное приложение;
* **поле «Поиск по индексу»**: пользователь вводит индекс договора.

Макет формы «Договоры» представлен в приложении В на рисунке В.8.

Макет формы добавления объекта состоит из:

* **поля** «Местонахождение», «Цена», «Владелец» – вносятся данные об недвижимости.
* **кнопка** «Добавить объект» – позволяет пользователю добавить объект;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы добавления объекта представлен в приложении В на рисунке В.9.

Макет формы добавления клиента состоит из:

* **поля** «ФИО клиента», «Адрес», «Номер телефона» – вносятся данные об недвижимости.
* **кнопка** «Добавить клиента» – позволяет пользователю добавить клиента;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы добавления клиента представлен в приложении В на рисунке В.10.

Макет добавления договора состоит из:

* **поля** «ФИО клиента», «Данные агентства недвижимости», «Наименование недвижимости» – вносятся данные об недвижимости.
* **кнопка** «Добавить договор» – позволяет пользователю добавить договор;
* **кнопка «Отмена»**: если пользователь решит не сохранять введенную информацию, он может нажать эту кнопку, чтобы отменить действие и вернуться к предыдущему экрану.

Макет формы добавления договора представлен в приложении В на рисунке В.11.

# **Описание реализации программного средства**

# **Инструменты разработки и применения технологии**

Инструментами разработки для будущего программного приложения будут являться:

* операционная система MS Widows 11;
* программная среда разработки Microsoft Visual Studio 2022;
* язык программирования C#;
* система управления реляционными базами данных (СУБД) Microsoft SQL Server 2024;
* программа для построения блок-схем Microsoft Visio
* язык определения данных DDL;
* справочная система Dr.Explain;
* офисный пакет приложений Microsoft Office.

MS Windows 11 – операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства MS Windows. Доступна единая платформа разработки и единый магазин универсальных приложений, совместимых со всеми поддерживаемыми устройствами [19].

Microsoft Visual Studio 2022 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского Макета, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию [14].

MS Visual Studio 2022 включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня [14].

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка [15].

DDL состоит из набора SQL-команд, используемых для создания, изменения и удаления структур баз данных, но не данных. Он просто имеет дело с описаниями схемы базы данных и используется для создания и изменения структуры объектов, присутствующих в базе данных. Все DDL-команды автоматически фиксируются, что означает, что все изменения навсегда сохраняются в базе данных. DDL подразделяется на пять команд, которые широко используются в SQL-запросах[15].

Microsoft Visio – векторный графический редактор, редактор диаграмм и блок-схем для Windows. Выпускается в трёх редакциях: Standard, Professional и Pro for Office 365.Аналогично с Adobe Reader, в стандартный набор программ Microsoft Office входит только средство для просмотра и печати диаграмм Microsoft Visio Viewer. Полнофункциональная версия Microsoft Visio Professional для создания и редактирования монограмм и диаграмм. Первоначально Visio разрабатывался и выпускался компанией Shapeware, затем переименованной в Visio Corporation. Microsoft приобрела компанию в 2000 году, тогда продукт назывался Visio 2000 [18].

C# – объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота[6] как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [16].

Dr.Explain – это профессиональная программа для создания электронных справочных руководств, пользовательских инструкций и баз знаний. Она автоматизирует создание скриншотов, аннотацию изображений, индексацию контента и форматирование документации.

Microsoft Office – это офисный пакет, разработанный американской корпорацией Microsoft. В его состав входят приложения для работы с текстовыми документами, электронными таблицами, базами данных и другими типами файлов. Он предоставляет инструменты, такие как Word, Excel, PowerPoint, Outlook, Teams и многое другое [17].

# **Порядок авторизации пользователей**

В приложении можно выделить следующие роли и функции:

Агент:

* Аутентификация: Логист регистрируется в системе, предоставляя необходимые учетные данные;
* Авторизация: после успешной аутентификации логист получает доступ к функциям, связанным с его ролью;
* Функции: Создание профилей клиентов, объектов и договоров. Это может включать ввод информации о клиенте и информации о недвижимости.

# **Организация данных**

Организация данных подразумевает создание модели данных, главными элементами, которые являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

Структура базы данных разрабатываемого программного средства включает шесть таблиц.

Таблица «Clients» содержит информацию о клиентах, структура таблицы представлена в таблице 3.1

Таблица 3.1 – Clients

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| Id\_client | int | 4 | Номер клиента |
| Name\_and\_FirstName | nvarchar(50) | 100 | ФИО клиента |
| Passport\_data | nvarchar(50) | 100 | Паспортные данные |
| mobile | nvarchar(50) | 100 | Номер телефона |

Таблица «register» содержит информацию о зарегистрированныъ пользователях, структура таблицы представлена в таблице 3.2

Таблица 3.2 – register

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id\_user | int | 4 | Номер пользователя |
| login\_user | varchar(50) | 250 | Логин пользователя |
| password\_user | varchar(50) | 250 | Пароль пользователя |

Таблица «Objects» содержит информацию о объектах, структура таблицы представлена в таблице 3.3

Таблица 3.3 – Objects

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id\_object | int | 4 | Номер объекта |
| raion | nvarchar(50) | 100 | Район |
| rooms | int | 4 | Кол-во комнат |
| floot | int | 4 | Этаж |
| Opicanie | nvarchar(MAX) | N\*2 | Подробное описание |
| name\_object | nvarchar(50) | 100 | Наименование недвижимости |

Таблица «Dogovora» содержит информацию о договорах, структура таблицы представлена в таблице 3.4

Таблица 3.4 – Dogovora

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id\_dogovor | int | 4 | Номер договора |
| name\_object | nvarchar(50) | 100 | Наименование недвижимости |
| raion | nvarchar(50) | 100 | Район |
| rooms | int | 4 | Кол-во комнат |
| floot | int | 4 | Этаж |
| Opicanie | nvarchar(MAX) | N\*2 | Подробное описание |
| Name\_and\_FirstName | nvarchar(50) | 100 | ФИО клиента |
| Passport\_data | nvarchar(50) | 100 | Паспортные данные |
| Price | float | 8 | Цена |
| id\_client | int | 4 | Номер клиента |
| id\_object | int | 4 | Номер объекта |

Все таблицы представлены на рисунке 3.1

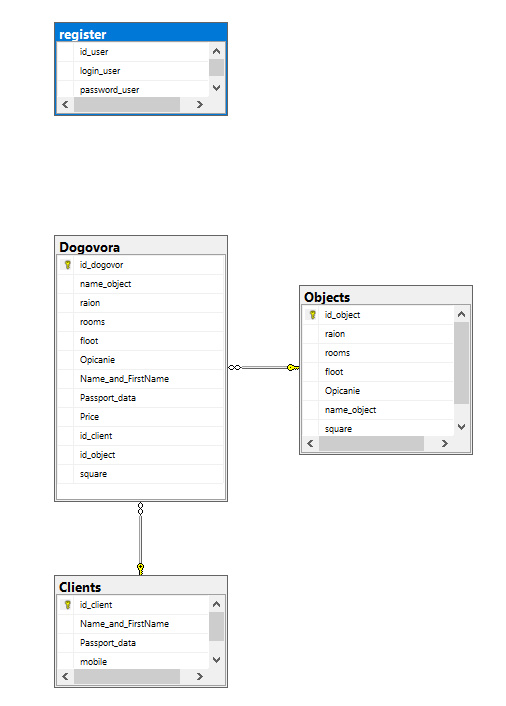


Рисунок 3.1 – Схема данных

# **Функции: логическая и физическая организация**

В данном проекте реализованы функции добавления, редактирования и удаления информации о договорах, об клиентах, об объектах; формирование договора на заключение купли/продажи и информация об посылках заказчика; формирование поиска.

Функция добавления информации о посылке находиться на форме «Добавление договора». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события button2. Все данные, вводимые в поля формы, будут сохранены в таблице «Dogovora». Код функции добавления договора:

public void SaveOrder(DoGoVoRa dogovor)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Dogovora (name\_object, raion, rooms, floot, Opicanie, Name\_and\_FirstName, Passport\_data, Price, id\_client, id\_object, square) VALUES (@name\_object, @raion, @rooms, @floot, @Opicanie, @Name\_and\_FirstName, @Passport\_data, @Price, @id\_client, @id\_object, @square)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@name\_object", dogovor.name\_object);

command.Parameters.AddWithValue("@raion", dogovor.raion);

command.Parameters.AddWithValue("@rooms", dogovor.rooms);

command.Parameters.AddWithValue("@floot", dogovor.floot);

command.Parameters.AddWithValue("@Opicanie", dogovor.Opicanie);

command.Parameters.AddWithValue("@Name\_and\_FirstName", dogovor.Name\_and\_FirstName);

command.Parameters.AddWithValue("@Passport\_data", dogovor.Passport\_data);

command.Parameters.AddWithValue("@Price", dogovor.Price);

command.Parameters.AddWithValue("@id\_client", dogovor.id\_client);

command.Parameters.AddWithValue("@id\_object", dogovor.id\_object);

command.Parameters.AddWithValue("@square", dogovor.square);

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void AddDogovor(DoGoVoRa dogovor)

{

try

{

DataRow newRow = dt.NewRow();

newRow["id\_dogovor"] = dogovor.id\_dogovor;

newRow["name\_object"] = dogovor.name\_object;

newRow["raion"] = dogovor.raion;

newRow["rooms"] = dogovor.rooms;

newRow["floot"] = dogovor.floot;

newRow["Opicanie"] = dogovor.Opicanie;

newRow["Name\_and\_FirstName"] = dogovor.Name\_and\_FirstName;

newRow["Passport\_data"] = dogovor.Passport\_data;

newRow["Price"] = dogovor.Price;

newRow["id\_client"] = dogovor.id\_client;

newRow["id\_object"] = dogovor.id\_object;

newRow["square"] = dogovor.square;

dt.Rows.Add(newRow);

SaveDogovor(dogovor);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void SaveDogovor(DoGoVoRa dogovor)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Dogovora (name\_object, raion, rooms, floot, Opicanie, Name\_and\_FirstName, Passport\_data, Price, id\_client, id\_object, square) VALUES (@name\_object, @raion, @rooms, @floot, @Opicanie, @Name\_and\_FirstName, @Passport\_data, @Price, @id\_client, @id\_object, @square)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@name\_object", dogovor.name\_object);

command.Parameters.AddWithValue("@raion", dogovor.raion);

command.Parameters.AddWithValue("@rooms", dogovor.rooms);

command.Parameters.AddWithValue("@floot", dogovor.floot);

command.Parameters.AddWithValue("@Opicanie", dogovor.Opicanie);

command.Parameters.AddWithValue("@Name\_and\_FirstName", dogovor.Name\_and\_FirstName);

command.Parameters.AddWithValue("@Passport\_data", dogovor.Passport\_data);

command.Parameters.AddWithValue("@Price", dogovor.Price);

command.Parameters.AddWithValue("@id\_client", dogovor.id\_client);

command.Parameters.AddWithValue("@id\_object", dogovor.id\_object);

command.Parameters.AddWithValue("@square", dogovor.square);

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

Функция редактирования информации о посылке находится на форме «Договоры». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Click.Все данные, вводимые в поля формы, будут сохранены в таблице «Box».

Функция удаления информации о посылке находится на форме «Главная». После нажатия на кнопку «Удалить» на данной форме происходит обработка событий Button\_Click\_delet. Выбранные данные будут удалены из таблицы «Dogovora».

Функция добавления информации об клиентах находиться на форме «Clients». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события Button\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Clients».

Функция редактирования информации об клиентах находится на форме «Добавление клиентов». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Click.Все данные, вводимые в поля формы, будут сохранены в таблице «Clients».

Функция удаления информации об автомобилях находится на форме «Клиенты». После нажатия на кнопку «Удалить» на данной форме происходит обработка событий Button\_Click\_delet. Выбранные данные будут удалены из таблицы «Clients».

Функция добавления информации об объектах находиться на форме «Добавление объекта». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события Button\_Click\_addPeople. Все данные, вводимые в поля формы, будут сохранены в таблице «Objects».

Функция редактирования информации об заказчиках находится на форме «Объекты». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Click\_addPeople. Все данные, вводимые в поля формы, будут сохранены в таблице «Objects».

Функция удаления информации об заказчиках находится на форме «Объекты». После нажатия на кнопку «Удалить» на данной форме происходит обработка событий Button\_Click\_delet. Выбранные данные будут удалены из таблицы «Objects».

Функция формирования созданного договора происходит при нажатии на кнопку «Создать» на форме «Документы». Происходит обработка события Button\_Click\_1, в процессе которого будет создан документ Word со всеми данными в договоре. Код функции создания документа word:

public static void GenerateWordReport(Customer selectedCustomer, Box box, Drivers driver)

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create("C:\\Users\\kupit\\OneDrive\\Desktop\\Отчет.docx", WordprocessingDocumentType.Document))

{

MainDocumentPart mainPart = wordDocument.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = mainPart.Document.AppendChild(new Body());

AppendParagraph(body, "ДОГОВОР ПЕРЕВОЗКИ ГРУЗА", true, JustificationValues.Center);

AppendParagraph(body, $"{DateTime.Now:dd.MM.yyyy}.", false, JustificationValues.Right);

AppendParagraph(body, "Наименование компании: Sub\_Zero", false, JustificationValues.Right);

AppendParagraph(body, "1. СТОРОНЫ ДОГОВОРА", true);

AppendParagraph(body, $"1.1. Перевозчик: {driver.FullName}");

AppendParagraph(body, $"1.2. Грузополучатель: {selectedCustomer.Fio}");

AppendParagraph(body, $"1.3. Грузоотправитель: {selectedCustomer.Fio}");

AppendParagraph(body, "2. ХАРАКТЕРИСТИКИ ГРУЗА", true);

AppendParagraph(body, $"2.1. Вес: {box.Weight}");

AppendParagraph(body, $"2.2. Объем: {box.Value}");

AppendParagraph(body, "3. ПУНКТ ОТПРАВЛЕНИЯ И НАЗНАЧЕНИЯ", true);

AppendParagraph(body, $"3.1. Пункт отправления: {GetCityIdByName(box.SityOtprav)}");

AppendParagraph(body, $"3.2. Пункт назначения: {GetCityIdByName(box.SityDostav)}");

AppendParagraph(body, "4. СТОИМОСТЬ ДОСТАВКИ И ПОРЯДОК РАСЧЕТОВ", true);

AppendParagraph(body, $"4.1. Стоимость доставки: {box.PriceBox – box.Price}");

AppendParagraph(body, "5. СРОКИ ПЕРЕВОЗКИ", true);

AppendParagraph(body, $"5.1. Дата отправки: {box.dateOtprav}");

AppendParagraph(body, $"5.2. Дата получения: {box.datePolych}");

AppendParagraph(body, "6. ПРАВА И ОБЯЗАННОСТИ СТОРОН", true);

AppendParagraph(body, "6.1. Перевозчик обязуется доставить груз в указанное место в указанные сроки. Грузоотправитель обязуется оплатить услуги перевозки в соответствии с договором. Перевозчик не несет ответственности за потерю или повреждение груза.");

AppendParagraph(body, "7. ОТВЕТСТВЕННОСТЬ ЗА НАРУШЕНИЕ УСЛОВИЙ СДЕЛКИ", true);

AppendParagraph(body, "7.1. В случае нарушения условий договора стороны несут ответственность в соответствии с действующим законодательством.");

AppendParagraph(body, "8. ФОРС-МАЖОР", true);

AppendParagraph(body, "8.1. В случае наступления обстоятельств непреодолимой силы, стороны освобождаются от ответственности за неисполнение обязательств по договору.");

if (selectedCustomer.IsUrFace == 1)

{

AppendParagraph(body, "9. ЮРИДИЧЕСКИЕ АДРЕСА И БАНКОВСКИЕ РЕКВИЗИТЫ СТОРОН", true);

AppendParagraph(body, $"9.1. Юридический адрес: {selectedCustomer.AdressCompany}");

AppendParagraph(body, $"9.2. Банковские реквизиты: Банк – {selectedCustomer.NameBank}, Код банка – {selectedCustomer.CodeBank}");

}

AppendParagraph(body, "10. ПОДПИСИ СТОРОН", true);

AppendParagraph(body, "Грузоотправитель (Подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_");

AppendParagraph(body, "Директор (Подпись)\_\_\_\_\_\_\_\_\_\_\_\_\_");

mainPart.Document.Save();

}

}

Функция формирования созданного документа информации об посылках заказчика происходит при нажатии на кнопку «Создать» на форме «Документы». Происходит обработка события Button\_Click\_1, в процессе которого будет создан документ Excel со всеми данными о посылках.

Полный код программных модулей представлен в приложении А.

# **Входные и выходные данные**

Входными данными являются данные, заносимые программой в таблицы: «Clients», «Dogovora», «Objects». Также входными данными являются данные, вводимые в поля для поиска.

Для добавления данных в таблицу «Clients» необходимо ввести следующие данные:

* ФИО клиента;
* номер паспорта;
* номер телефона.

Для добавления данных в таблицу «Objects» необходимо ввести следующие данные:

* наименование объекта;
* кол-во комнат;
* район;
* этаж;
* описание.

Для добавления данных в таблицу «Dogovora» необходимо ввести следующие данные:

* ФИО клиента;
* номер паспорта;
* цена;
* наименование объекта;
* кол-во комнат;
* район;
* этаж;
* описание.

Выходные данными являются данные, которые экспортируются в файлы Microsoft Office, а именно договоры. А также отчёт:

* отчёт по договорам;
* отчёт по недвижимости.

Договор генерируется по нажатию на кнопку «Создать». Выходные документы представлен в приложении В.

# **Функциональное тестирование**

Функциональное тестирование – это тестирование функций приложения на соответствия требованиям. Оценка производится в соответствии с ожидаемым и фактическим результатом.

Тестирование программы будет производится последовательно, переходя из одной программы в другую. Во время теста будут проверятся все действия с программой, навигация пунктов меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

В таблицах 3.7-3.14 представлены тест-кейсы, подготовленные для проведения функционального тестирования.

Первый тест будет проведён в окне авторизации программы.

Тест-кейс для тестирования функции авторизации представлен в таблице 3.7

Таблица 3.7 – Тест-кейс функции авторизации

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Функция | Шаги выполнения | Результат |
| 1 | Авторизация в приложении | 1. Заполнить поля данными:  - Фамилия: “123”;  - Пароль: “123”.  2. Нажать на кнопку “Войти” | Ожидаемый результат: переход на форму “Главная” |
| Фактический результат: успешный переход на форму “Главная” |

Второй тест будет проведён на форме «Договора».

Тест-кейс для тестирования функции добавления договора будет представлен в таблицы 3.8

Таблица 3.8 – Тест-кейс функции добавления договора

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Функция | Шаги выполнения | Результат |
| 2 | Добавление договора | 1. На главной форме нажать на кнопку “Договоры”;  2. Нажать на кнопку “Добавить договор”; 3. Заполнить поля данными: - Наименование объекта:“Недвижимость1”;  - Район: “Фрунзенский”; - Комнаты: “10”; - Этаж: “3”; - Описание: “Описание”; - ФИО Клиента: “Скибицкий Юра”; - Паспортные данные: “12414DFHDH23452”; - Цена: “1499”; - ID Клиента: “1406926793”; - ID Объекта: “1”; 4. Нажать на кнопку “Добавить договор” | Ожидаемый результат: Добавление договора в таблицу. |
| Фактический результат: успешное добавление в таблице. |

Третий тест будет проведён на форме «Заказчики».

Тест-кейс для тестирования функции добавления заказчика будет представлен в таблицы 3.9

Таблица 3.9 – Тест-кейс функции добавления клиента

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Функция | Шаги выполнения | Результат |
| 3 | Добавление клиента | 1. На главной форме нажать на кнопку “Клиенты”;  2. Нажать на кнопку “Добавить клиента”; 3. Заполнить поля данными: -ФИО:“Скибицкий Юра”;  - Паспортные данные: “12414DFHDH23452”; - Номер телефона: “+37212234343”; 4. Нажать на кнопку “Добавить клиента”. | Ожидаемый результат: Добавление клиента в таблицу. |
| Фактический результат: успешное добавление в таблице. |

Четвертый тест будет проведён на форме «Объекты».

Тест-кейс для тестирования функции добавления объекта будет представлен в таблицы 3.9

Таблица 3.9 – Тест-кейс функции добавления посылки

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Функция | Шаги выполнения | Результат |
| 4 | Добавление объекта недвижимости | 1. На главной форме нажать на кнопку “Объекты”;  2. Нажать на кнопку “Добавить объект”; 3. Заполнить поля данными: -Район:“Московский”;  - Количество комнат: “5”; - Этаж: “5”; - Описание: “Описание”; - Наименование объекта: “Недвижимость2” 4. Нажать на кнопку “Добавить объект”. | Ожидаемый результат: Добавление объекта в таблицу. |
| Фактический результат: успешное добавление в таблице. |

Пятый тест будет проведён на форме «Договоры».

Тест-кейс для тестирования функции поиска недвижимости будет представлен в таблицы 3.10

Таблица 3.10– Тест-кейс функции добавления заказчика

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Функция | Шаги выполнения | Результат |
| 5 | Поиск недвижимости | 1. На главной форме нажать на кнопку “Объекты” или “Договоры”;  2. Заполнить поле данными: “Недвижимость1”;  3. Нажать на кнопку “Поиск ”. | Ожидаемый результат: Показ результата поиска. |
| Фактический результат: успешный поиск и вывод. |

Шестой тест будет проведён на форме «Клиенты».

Тест-кейс для тестирования функции поиска клиента будет представлен в таблицы 3.11

Таблица 3.11 – Тест-кейс функции создания отчета

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Функция | Шаги выполнения | Результат |
| 6 | Поиск клиента | 1. На главной форме нажать на кнопку “Клиенты”  2. Заполнить поле данными: “Ски”;  3. Нажать на кнопку “Поиск ”. | Ожидаемый результат: Показ результата поиска. |
| Фактический результат: успешный поиск и вывод. |

# **Описание справочной системы**

Справочная система была разработана с использованием программного обеспечения Dr.Explain.

Справочная система включает в себя несколько ключевых разделов, которые были определены в предоставленном изображении:

* вход;
* работа с клиентами;
* работа с объектами;
* работа с договорами;
* создание документации;

Справочная система может быть вызвана из главной формы при нажатии на кнопку «Справка». Справка представлена на рисунке 3.2

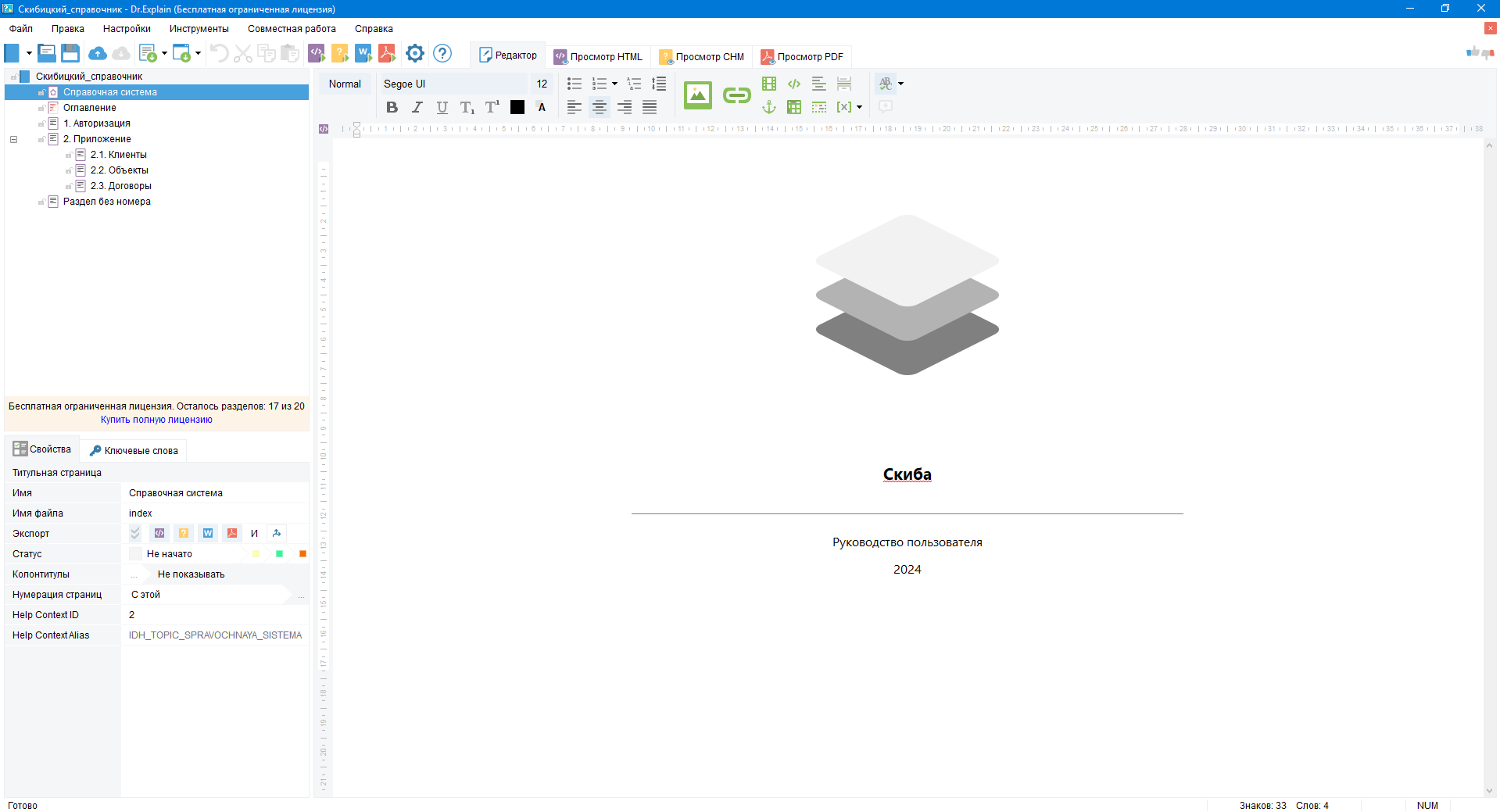


Рисунок 3.2 – Справка

# **Применение**

# **Назначение программного средства**

Приложение «Скиба.exe» предназначено для автоматизации агентства недвижимости. Программное средство представляет пользователю возможности для поиска, добавления, отображения, и редактирования имеющихся в базе объектов: объектов, клиентов, договоров, уменьшает затраты времени используемого при поиске недвижимости, ведет учет недвижимостей. Под эту роль был создан специальный макет, который позволит агенту быстро и удобно заключать договоры.

Основные разработанные функции:

* авторизация и регистрация работников в системе;
* возможность добавлять и редактировать клиентов;
* возможность добавлять и редактировать объекты;
* возможность добавлять и редактировать договоры;
* возможность экспорта данных в документы Microsoft Office.

Приложение «Скиба.exe» разработано специально для агентства недвижимости и не предназначено для использования в других отраслях, таких как производство или розничная торговля. Оно оптимизировано для работы с данными, связанными с недвижимостью, и может не подходить для управления данными, не относящимися к логистике. Кроме того, функциональность экспорта ограничена поддержкой документов Microsoft Office, что может не соответствовать потребностям компаний, использующих другие форматы для документооборота.

# **Условия применения**

Для применения данного программного средства необходимы следующие технические требования:

* процессор Intel Core i5 или выше;
* минимальный объём оперативной памяти 500 Мбайт;
* оперативная система Windows 11;
* платформа .Net Framework v4.8;
* видеокарта с объёмом памяти не менее 256 Мбайт;
* клавиатура, мышь, принтер.

Процесс установки программного средства «Скиба.exe» включает следующие шаги:

* загрузка программы: скачайте установочный файл «Скиба.exe» с github или носителя, предоставленного разработчиком;
* установка: запустите скачанный установочный файл и следуйте инструкциям мастера установки.

# **Заключение**

Проект “Скиба.exe” представляет собой программное средство, разработанное для автоматизации рабочего места агента по работе с клиентами и недвижимостями в агентстве недвижимости. Организационная сущность задачи заключается в создании инструмента, который облегчит работу агентов в области недвижимости. Предметная область проекта охватывает клиентов, управление объектами, выбор недвижимости, создание договоров и генерацию отчетов.

Технические требования для программы включают процессор Intel Core i5 или выше, минимальный объем оперативной памяти 500 Мбайт, операционную систему Windows 11, платформу .Net Framework v4.8 и видеокарту с объемом памяти не менее 256 Мбайт. Логическая организация данных включает описание структуры данных и связей между объектами (посылки, клиенты, водители, транспорт). Физическая организация данных включает выбор СУБД и оптимизацию запросов. Справочная система описывает функциональность и предоставляет помощь пользователям.

Программное средство Скиба предоставляет возможности поиска, добавления, отображения и редактирования объектов в базе данных. Входные данные включают информацию о заказчиках, посылках, водителях и транспорте. Выходные данные могут быть отчетами, статусами доставки и другой информацией.

Среди преимуществ разработанного ПС можно выделить автоматическое управление запасами, гибкость и масштабируемость, а также интуитивный пользовательский Макет. Однако следует учитывать ограниченность функциональности экспорта данных в документы Microsoft Office и то, что программа не подходит для управления данными, не связанными с транспортной-логистикой.

Скиба представляет собой важный инструмент для транспортно-логистических компаний, помогая им оставаться конкурентоспособными в современном бизнес-окружении.

# **Список использованных источников**

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение технологий» / С.В. Банцевич, Т.Г. Багласова. – Минск: КБП, 2022. – 49 c.
2. Михалевич, В.Ю. Методические указания к курсовому проектированию для обучающихся специальности 2-40 01 01 «Программное обеспечение информационных технологий – Минск: КБП, 2023. – 3 c.
3. ГОСТ Р 54694-2011. Логистика. Термины и определения. – Введ. 2013-01-01. – М.: Стандартинформ, 2012.;
4. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
5. 9 лучших программ для автоматизации рабочих процессов [2024 г.] [Электронный ресурс] – Режим доступа: https://nauchniestati.ru/spravka;
6. Автоматизация рабочего места – RPA ROBIN [Электронный ресурс] – Режим доступа: https://www.atlassian.com/ru/agile;
7. Автоматизированное рабочее место (АРМ): что это? [Электронный ресурс] – Режим доступа: https://www.liga-commersantov.ru/articles;
8. Обязанности менеджера по работе с клиентами – чем занимается и как [Электронный ресурс] – Режим доступа: https://sales-generator.ru/blog;
9. Разработка автоматизированного рабочего места менеджера [Электронный ресурс] – Режим доступа: https://spravochnick.ru/informatika;
10. Топ лучших приложений для управления персоналом для автоматизации [Электронный ресурс] – Режим доступа: https://www.guru99.com/ru/best;
11. ТОП-26 программ мониторинга и контроля работы сотрудников за [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles;
12. Удобство и эффективность: разработка автоматизированного рабочего места [Электронный ресурс] – Режим доступа: <https://appmaster.io/ru/blog>;
13. Что значит АРМ (Автоматизированное рабочее место)? [Электронный ресурс] – Режим доступа: https://alexrovich.ru/info;
14. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
15. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
16. C Sharp – Википедия [2024 г.] [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp;
17. [Microsoft Office – Wikipedia](https://en.wikipedia.org/wiki/Microsoft_Office) [2024 г.] [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Microsoft\_Office;
18. [Microsoft Visio – Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_Visio) 2024 г.] [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Visio>;
19. Windows 11 [Электронный ресурс]. – Microsoft, 2024. – Режим доступа: https://support.microsoft.com/ru-ru/meetwindows11. – Дата доступа: 16.03.2024;

# **Приложение А**

**(обязательное)**

**Текст программных модулей**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Media;

using System.Xml.Linq;

namespace project

{

public partial class Main2 : Window

{

private DataTable dt = new DataTable();

public Main2()

{

InitializeComponent();

this.Loaded += Main2\_Loaded;

FlowDocument flowDoc = new FlowDocument();

Table table1 = new Table();

flowDoc.Blocks.Add(table1);

table1.CellSpacing = 10;

table1.Background = Brushes.White;

int numberOfColumns = 7; // Increase the number of columns to include the new 'square' field

for (int x = 0; x < numberOfColumns; x++)

{

table1.Columns.Add(new TableColumn());

if (x % 2 == 0)

table1.Columns[x].Background = Brushes.Beige;

else

table1.Columns[x].Background = Brushes.LightSteelBlue;

}

}

private void Main2\_Loaded(object sender, RoutedEventArgs e)

{

string connectionString = @"Data Source=DESKTOP-EU2KAKK\SQLEXPRESS;Initial Catalog=Name;Integrated Security=True;Encrypt=False";

string sql = "SELECT id\_object, raion, rooms, floot, Opicanie, name\_object, square FROM Objects"; // Include 'square' in the SELECT statement

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

dt.Clear();

adapter.Fill(dt);

ObjectGrid.ItemsSource = dt.DefaultView;

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string searchValue = textbox1.Text;

FilterData(searchValue);

}

private void FilterData(string searchValue)

{

DataView dv = dt.DefaultView;

dv.RowFilter = string.Format("name\_object LIKE '%{0}%'", searchValue);

ObjectGrid.ItemsSource = dv;

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

Order order = new Order();

order.Show();

}

private void button4\_Click(object sender, RoutedEventArgs e)

{

window2 window2 = new window2();

window2.Show();

this.Close();

}

private void \_\_\_button5\_\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

private void \_\_\_button6\_\_Click(object sender, RoutedEventArgs e)

{

DeleteSelectedObject();

}

public void AddOrder(Object order)

{

try

{

DataRow newRow = dt.NewRow();

newRow["raion"] = order.raion;

newRow["rooms"] = order.rooms;

newRow["floot"] = order.floot;

newRow["Opicanie"] = order.Opicanie;

newRow["name\_object"] = order.name\_object;

newRow["square"] = order.square; // Include the 'square' field

dt.Rows.Add(newRow);

SaveOrder(order);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void SaveOrder(Object order)

{

try

{

string connectionString = @"Data Source=DESKTOP-EU2KAKK\SQLEXPRESS;Initial Catalog=Name;Integrated Security=True;Encrypt=False";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Objects (raion, rooms, floot, Opicanie, name\_object, square) VALUES (@raion, @rooms, @floot, @Opicanie, @name\_object, @square)"; // Include 'square' in the INSERT statement

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@raion", order.raion);

command.Parameters.AddWithValue("@rooms", order.rooms);

command.Parameters.AddWithValue("@floot", order.floot);

command.Parameters.AddWithValue("@Opicanie", order.Opicanie);

command.Parameters.AddWithValue("@name\_object", order.name\_object);

command.Parameters.AddWithValue("@square", order.square); // Add 'square' parameter

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void DeleteSelectedObject()

{

if (ObjectGrid.SelectedItem is DataRowView selectedObject)

{

string connectionString = @"Data Source=DESKTOP-EU2KAKK\SQLEXPRESS;Initial Catalog=Name;Integrated Security=True;Encrypt=False";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Objects WHERE id\_object = @id\_object";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@id\_object", selectedObject["id\_object"]);

command.ExecuteNonQuery();

}

}

selectedObject.Row.Delete();

}

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

namespace project

{

public partial class Main1 : Window

{

private DataTable dt = new DataTable();

private string connectionString = @"Data Source=DESKTOP-EU2KAKK\SQLEXPRESS;Initial Catalog=Name;Integrated Security=True";

public Main1()

{

InitializeComponent();

this.Loaded += Main1\_Loaded;

}

private void Main1\_Loaded(object sender, RoutedEventArgs e)

{

LoadData();

}

private void LoadData()

{

string sql = "SELECT id\_dogovor, name\_object, raion, rooms, floot, Opicanie, Name\_and\_FirstName, Passport\_data, Price, id\_client, id\_object, square FROM Dogovora";

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

dt.Clear();

adapter.Fill(dt);

DoGoVoRaGrid.ItemsSource = dt.DefaultView;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string searchNameValue = textbox1.Text;

string searchSquareValue = textbox2.Text;

FilterData(searchNameValue, searchSquareValue);

}

private void FilterData(string searchNameValue, string searchSquareValue)

{

DataView dv = dt.DefaultView;

string filter = string.Empty;

if (!string.IsNullOrEmpty(searchNameValue))

{

filter = string.Format("name\_object LIKE '%{0}%'", searchNameValue);

}

if (!string.IsNullOrEmpty(searchSquareValue))

{

if (int.TryParse(searchSquareValue, out int square))

{

if (!string.IsNullOrEmpty(filter))

{

filter += " AND ";

}

filter += string.Format("square = {0}", square);

}

else

{

MessageBox.Show("Введите правильное числовое значение для площади.");

return;

}

}

dv.RowFilter = filter;

DoGoVoRaGrid.ItemsSource = dv;

// Debugging output

MessageBox.Show("Фильтр: " + filter);

}

public void SaveOrder(DoGoVoRa dogovor)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Dogovora (name\_object, raion, rooms, floot, Opicanie, Name\_and\_FirstName, Passport\_data, Price, id\_client, id\_object, square) VALUES (@name\_object, @raion, @rooms, @floot, @Opicanie, @Name\_and\_FirstName, @Passport\_data, @Price, @id\_client, @id\_object, @square)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@name\_object", dogovor.name\_object);

command.Parameters.AddWithValue("@raion", dogovor.raion);

command.Parameters.AddWithValue("@rooms", dogovor.rooms);

command.Parameters.AddWithValue("@floot", dogovor.floot);

command.Parameters.AddWithValue("@Opicanie", dogovor.Opicanie);

command.Parameters.AddWithValue("@Name\_and\_FirstName", dogovor.Name\_and\_FirstName);

command.Parameters.AddWithValue("@Passport\_data", dogovor.Passport\_data);

command.Parameters.AddWithValue("@Price", dogovor.Price);

command.Parameters.AddWithValue("@id\_client", dogovor.id\_client);

command.Parameters.AddWithValue("@id\_object", dogovor.id\_object);

command.Parameters.AddWithValue("@square", dogovor.square);

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void AddDogovor(DoGoVoRa dogovor)

{

try

{

DataRow newRow = dt.NewRow();

newRow["id\_dogovor"] = dogovor.id\_dogovor;

newRow["name\_object"] = dogovor.name\_object;

newRow["raion"] = dogovor.raion;

newRow["rooms"] = dogovor.rooms;

newRow["floot"] = dogovor.floot;

newRow["Opicanie"] = dogovor.Opicanie;

newRow["Name\_and\_FirstName"] = dogovor.Name\_and\_FirstName;

newRow["Passport\_data"] = dogovor.Passport\_data;

newRow["Price"] = dogovor.Price;

newRow["id\_client"] = dogovor.id\_client;

newRow["id\_object"] = dogovor.id\_object;

newRow["square"] = dogovor.square;

dt.Rows.Add(newRow);

SaveDogovor(dogovor);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void SaveDogovor(DoGoVoRa dogovor)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Dogovora (name\_object, raion, rooms, floot, Opicanie, Name\_and\_FirstName, Passport\_data, Price, id\_client, id\_object, square) VALUES (@name\_object, @raion, @rooms, @floot, @Opicanie, @Name\_and\_FirstName, @Passport\_data, @Price, @id\_client, @id\_object, @square)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@name\_object", dogovor.name\_object);

command.Parameters.AddWithValue("@raion", dogovor.raion);

command.Parameters.AddWithValue("@rooms", dogovor.rooms);

command.Parameters.AddWithValue("@floot", dogovor.floot);

command.Parameters.AddWithValue("@Opicanie", dogovor.Opicanie);

command.Parameters.AddWithValue("@Name\_and\_FirstName", dogovor.Name\_and\_FirstName);

command.Parameters.AddWithValue("@Passport\_data", dogovor.Passport\_data);

command.Parameters.AddWithValue("@Price", dogovor.Price);

command.Parameters.AddWithValue("@id\_client", dogovor.id\_client);

command.Parameters.AddWithValue("@id\_object", dogovor.id\_object);

command.Parameters.AddWithValue("@square", dogovor.square);

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void DeleteSelectedOrder()

{

if (DoGoVoRaGrid.SelectedItem is DataRowView selectedOrder)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Dogovora WHERE id\_dogovor = @id\_dogovor";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@id\_dogovor", selectedOrder["id\_dogovor"]);

command.ExecuteNonQuery();

}

}

selectedOrder.Row.Delete();

}

}

private void button3\_Click(object sender, RoutedEventArgs e)

{

DeleteSelectedOrder();

}

private void button4\_Click(object sender, RoutedEventArgs e)

{

window2 window2 = new window2();

window2.Show();

this.Close();

}

private void button5\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

public void button2\_Click(object sender, RoutedEventArgs e)

{

Product product = new Product();

product.Show();

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

namespace project

{

public partial class Window1 : Window

{

private DataTable dt = new DataTable();

private string connectionString = @"Data Source=DESKTOP-EU2KAKK\SQLEXPRESS;Initial Catalog=Name;Integrated Security=True";

public Window1()

{

InitializeComponent();

this.Loaded += Window1\_Loaded;

}

void Window1\_Loaded(object sender, RoutedEventArgs e)

{

LoadData();

}

public void LoadData()

{

string sql = "SELECT id\_client, Name\_and\_FirstName, Passport\_data, mobile FROM Clients";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

dt.Clear();

adapter.Fill(dt);

ClientGrid.ItemsSource = dt.DefaultView;

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

string searchValue = textbox1.Text.Trim();

if (string.IsNullOrEmpty(searchValue))

{

MessageBox.Show("Пожалуйста, введите имя для поиска.");

return;

}

DataRow[] foundRows = dt.Select($"Name\_and\_FirstName LIKE '%{searchValue}%'");

if (foundRows.Length > 0)

{

DataTable searchResult = foundRows.CopyToDataTable();

ClientGrid.ItemsSource = searchResult.DefaultView;

}

else

{

MessageBox.Show("Клиент не найден с таким именем.");

ClientGrid.ItemsSource = null;

}

}

private void button2\_Click(object sender, RoutedEventArgs e)

{

Window3 window3 = new Window3();

window3.Show();

}

private void button3\_Click(object sender, RoutedEventArgs e)

{

DeleteSelectedClient();

}

private void button4\_Click(object sender, RoutedEventArgs e)

{

window2 window2 = new window2();

window2.Show();

this.Close();

}

private void button5\_Click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

public void AddClient(Client client)

{

try

{

DataRow newRow = dt.NewRow();

newRow["Name\_and\_FirstName"] = client.Name\_and\_FirstName;

newRow["Passport\_data"] = client.Passport\_data;

newRow["mobile"] = client.mobile;

dt.Rows.Add(newRow);

SaveClient(client);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void SaveClient(Client client)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Clients (Name\_and\_FirstName, Passport\_data, mobile) VALUES (@Name\_and\_FirstName, @Passport\_data, @mobile)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name\_and\_FirstName", client.Name\_and\_FirstName);

command.Parameters.AddWithValue("@Passport\_data", client.Passport\_data);

command.Parameters.AddWithValue("@mobile", client.mobile);

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

public void DeleteSelectedClient()

{

if (ClientGrid.SelectedItem is DataRowView selectedClient)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Clients WHERE id\_client = @id\_client";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@id\_client", selectedClient["id\_client"]);

command.ExecuteNonQuery();

}

}

selectedClient.Row.Delete();

}

}

}

}

# 

# **Приложение Б**

**(справочное)**

**Формы выходных документов**

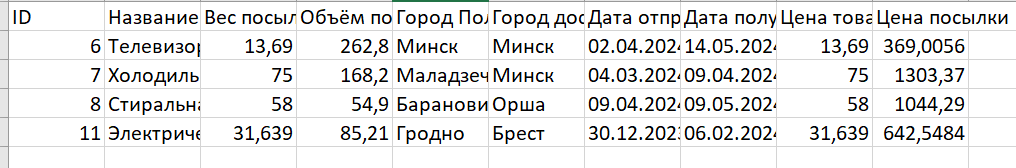


Рисунок Б.2 – Отчет о договорах клиентов

# **Приложение В**

**(справочное)**

**Результаты работы программы**

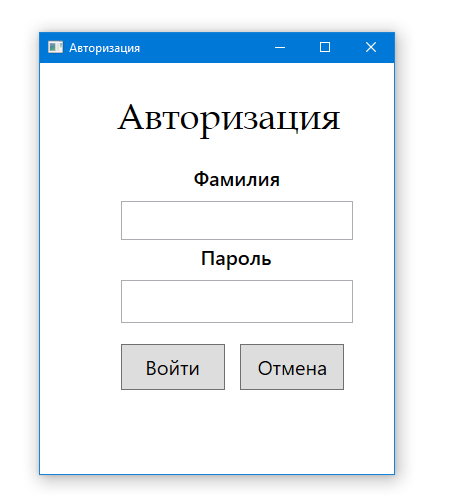


Рисунок В.1 – Форма входа

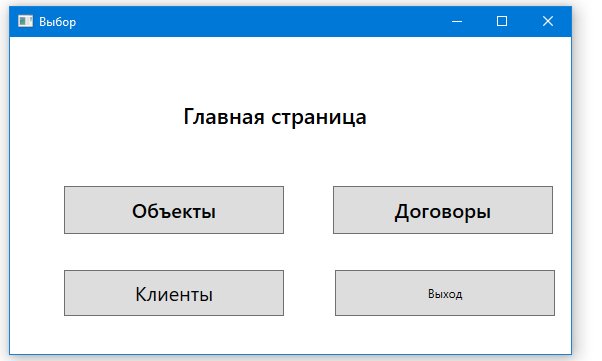


Рисунок В.3 – Форма главная для агентов

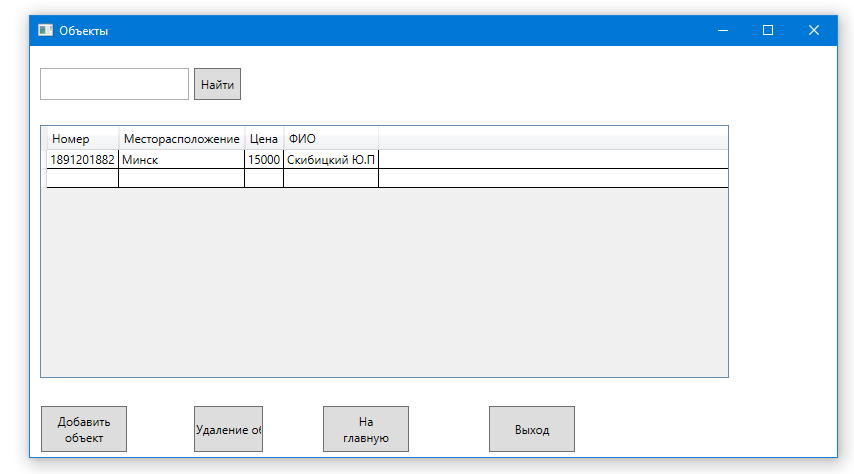


Рисунок В.4 – Форма объектов

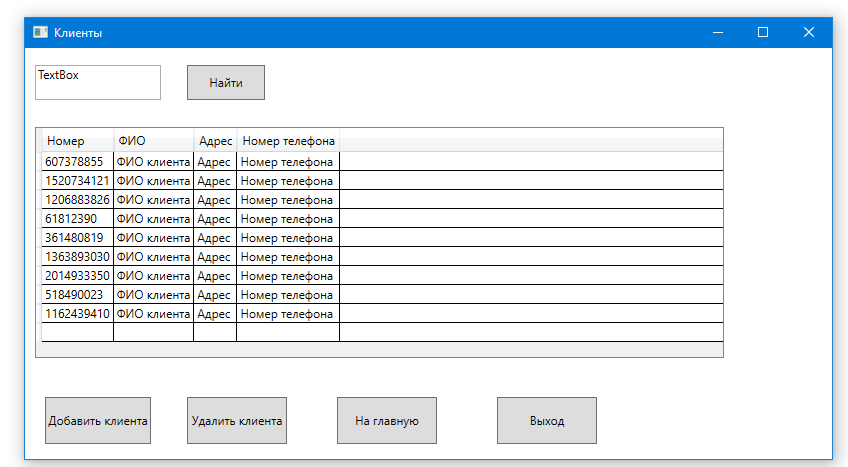


Рисунок В.5– Форма клиентов

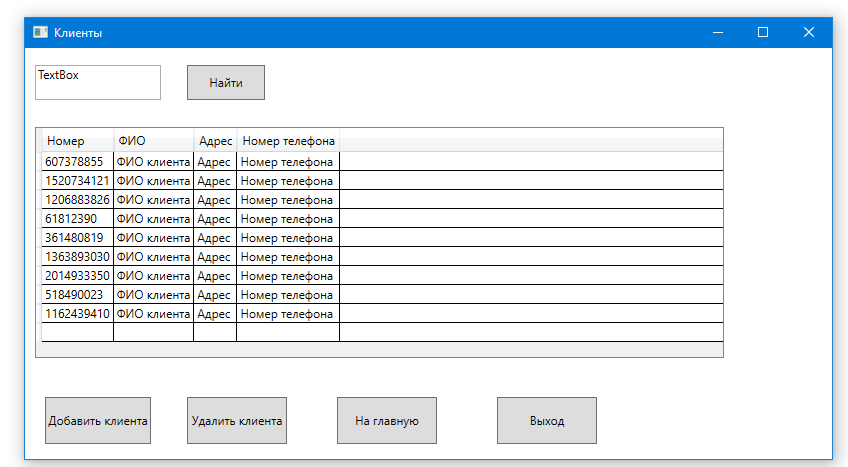


Рисунок В.7– Форма заказчиков

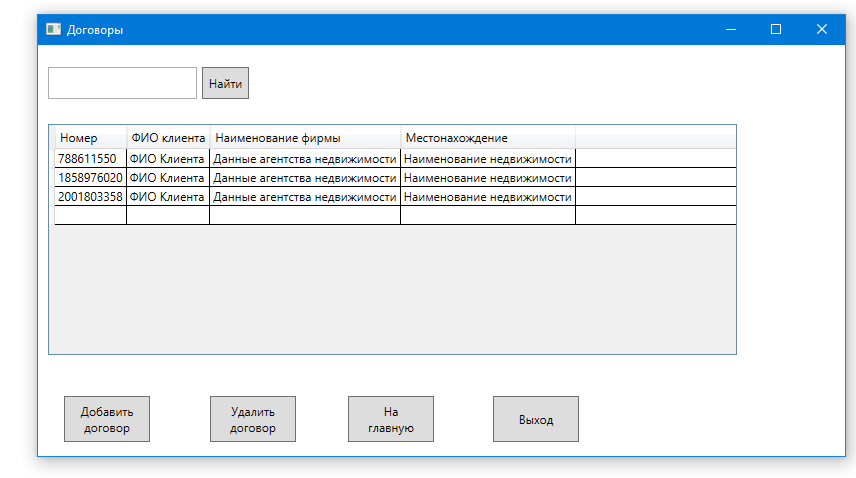


Рисунок В.8 – Форма главная для сотрудников тс организации

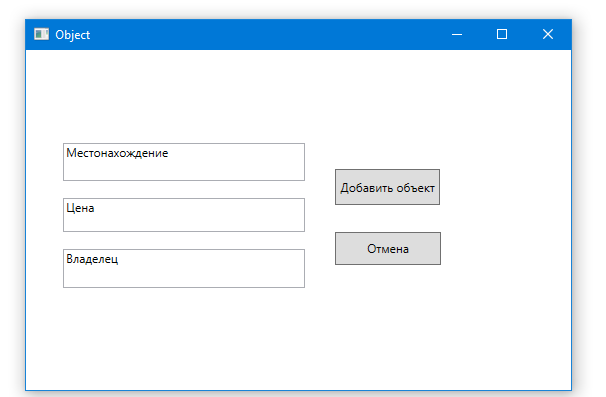


Рисунок В.9 – Форма добавления объекта

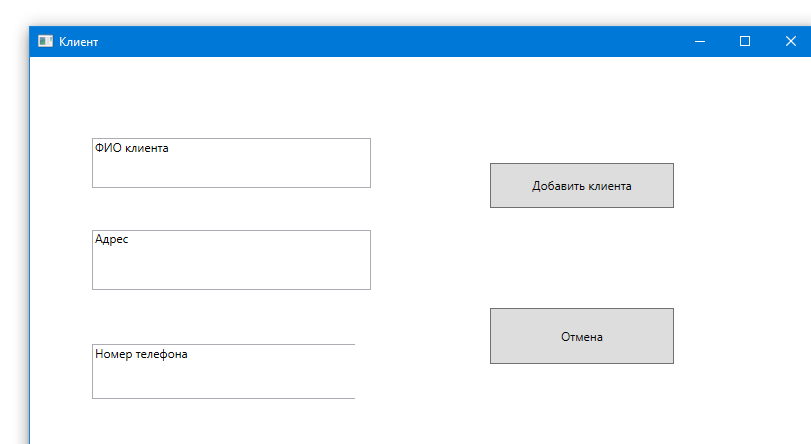


Рисунок В.10 – Форма добавления клиента

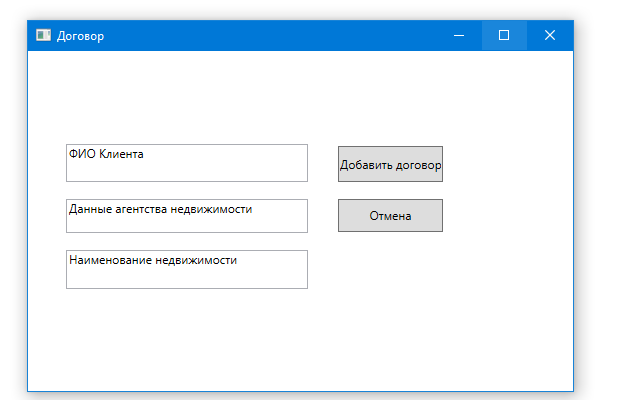


Рисунок В.11– Форма добавления договора

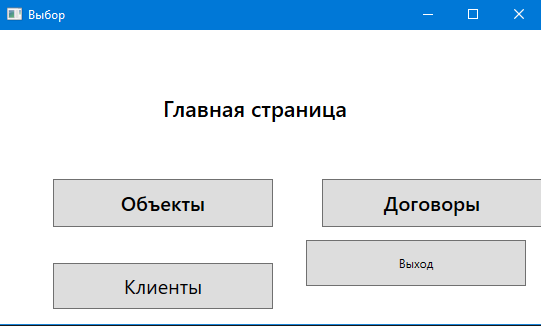


Рисунок В.14 – Результат входа агента

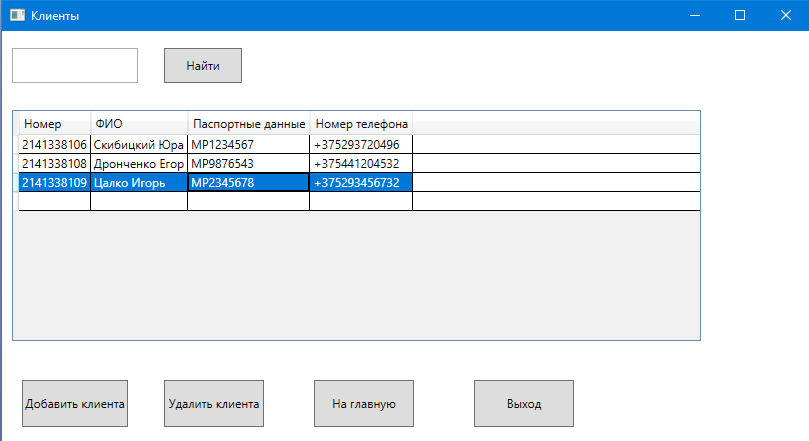


Рисунок В.16 – Результат добавления клиента

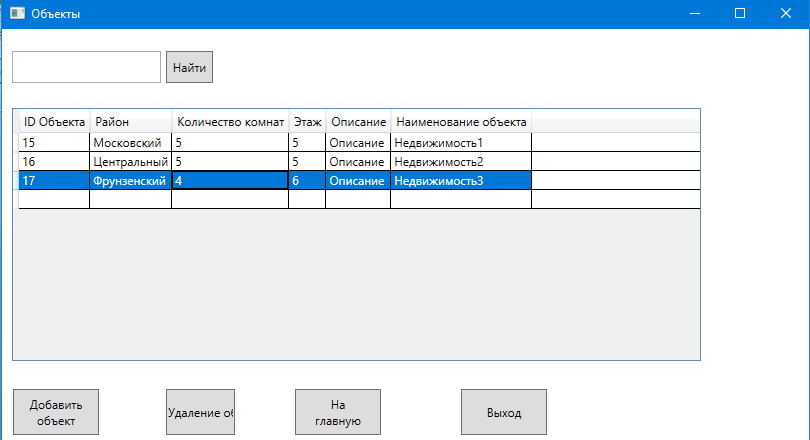


Рисунок В.17 – Результат создания объекта

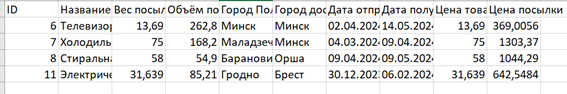


Рисунок В.19 – Результат создания отчета

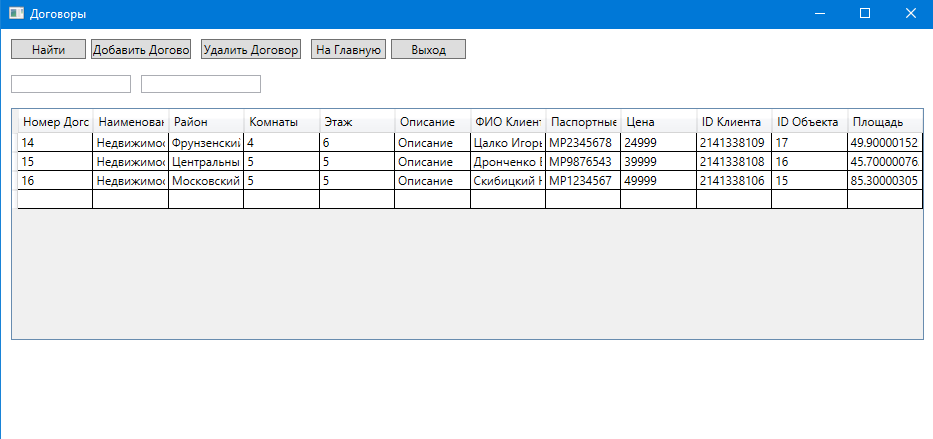


Рисунок В.20 – Результат создания договора