Содержание

[Введение 4](#_Toc152549760)

[1 Цель и задачи УЧЕБНОЙ практики. 5](#_Toc152549761)

[1.1 Основные требования по технике безопасности во время прохождения учебной практики 6](#_Toc152549762)

[1.2 Нормативно-правовые документы 7](#_Toc152549763)

[2 Предметная область. Анализ предметной области. Постановка задачи. 9](#_Toc152549764)

[2.1 Описание предметной области 9](#_Toc152549765)

[2.2 Группы пользователей информационной системы 9](#_Toc152549766)

[2.3 Основные требования, предъявляемые к информационной системе 10](#_Toc152549767)

[3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «Охрана квартир» 11](#_Toc152549768)

[3.1 Моделирование некоторых динамических аспектов системы 11](#_Toc152549769)

[3.1.1 Создание диаграммы вариантов использования 12](#_Toc152549770)

[3.1.2 Создание диаграммы деятельности 14](#_Toc152549771)

[3.2 Разработка функциональной структуры информационной системы 15](#_Toc152549772)

[3.3 Разработка макета информационной системы 15](#_Toc152549773)

[4 РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ «Охрана квартир» 21](#_Toc152549774)

[4.1 Моделирование и разработка базы данных 21](#_Toc152549775)

[4.2. Разработка программных модулей информационной системы «Охрана квартир» 25](#_Toc152549776)

[5 РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ 99](#_Toc152549777)

[5.1 Назначение эксперимента. Выбор и обоснование методики проведения тестирования 99](#_Toc152549778)

[5.2 Технология тестирования 100](#_Toc152549779)

[5.3 Результаты проведения тестирования 100](#_Toc152549780)

[ЗАКЛЮЧЕНИЕ 101](#_Toc152549781)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 102](#_Toc152549782)

[ПРИЛОЖЕНИЕ A 104](#_Toc152549783)

[ПРИЛОЖЕНИЕ Б 114](#_Toc152549784)

# Введение

Каждому человеку нужна охрана своего имущества и своей квартиры. Современные вызовы в области безопасности жилья подчеркивает необходимость эффективных методов и технологий для защиты жилищных помещений. Один из таких подходов — разработка прикладного программного обеспечения для отделов вневедомственной охраны квартир. Это программное обеспечение будет предоставлять инновационные решения, направленные на повышение эффективности мер по обеспечению безопасности от злоумышленников. Разработка прикладного программного обеспечения для отдела вневедомственной охраны квартир представляет собой важный шаг в направлении повышения общей безопасности общества.

Для освоения основного вида профессиональной деятельности: «Разработка программного обеспечения для компьютерных систем» и соответствующих профессиональных компетенций:

ПК 1.1 - Выполнять разработку спецификаций отдельных компонент

ПК 1.2 - Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля

ПК 1.3 - Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 1.4 - Выполнять тестирование программных модулей

ПК 1.5 - Осуществлять оптимизацию программного модуля

ПК 1.6 - Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций

# Цель и задачи УЧЕБНОЙ практики.

Целью учебной практики является овладение видом профессиональной деятельности Разработка модулей программного обеспечения для компьютерных систем, общими и профессиональными компетенциями по специальности.

Основные задачи учебной практики:

* систематизация, обобщение, закрепление и углубление знаний и умений;
* формирование у студентов знаний, умений и навыков, профессиональных компетенций, профессионально значимых личностных качеств;
* развитие профессионального интереса, формирование мотивационно-целостного отношения к профессиональной деятельности, готовности к выполнению профессиональных задач в соответствии с нормами морали, профессиональной этики и служебного этикета.

Учебная практика профессионального модуля ПМ 01 Разработка модулей программного обеспечения для компьютерных систем проводится в кабинетах информационно - вычислительного центра техникума преподавателями общепрофессиональных дисциплин профессионального цикла.

В период выполнения заданий учебной практики профессиональных модулей на студентов распространяются требования охраны труда и правила внутреннего распорядка, действующие в ГБПОУ ИО «АПЭТ»

Электронная охрана квартир позволяет добиться безопасности от злоумышленников в районе. Программный продукт представляет собой универсальное программное обеспечение, предназначенное для деятельности отдела вневедомственной охраны квартир, для удобной эксплуатации в целях предоставления информации.

Программный продукт предназначен для индивидуального применения, то есть предполагает использование одним пользователем в целях получения информации. В программном продукте можно будет заключать договор на охрану квартир.

## Основные требования по технике безопасности во время прохождения учебной практики

К работе в компьютерном классе допускаются лица, ознакомленные с данной инструкцией по технике безопасности и правилам поведения. Необходимо неукоснительно соблюдать правила по технике безопасности, т.к. нарушение этих правил может привести к поражению электрическим током, вызвать возгорание и навредить вашему здоровью.

* Работа обучающихся в компьютерном классе разрешается только в присутствии преподавателя (лаборанта).
* Во время занятий посторонние лица могут находиться в классе только с разрешения преподавателя.
* Во время перемен между уроками проводится обязательное проветривание компьютерного кабинета с обязательным выходом обучающихся из класса.

Каждый обучающийся в ответе за состояние своего рабочего места и сохранность размещенного на нем оборудования.

Обучающийся, допустивший нарушение по технике безопасности и правилам поведения в компьютерном классе, может быть привлечен к дисциплинарной ответственности. Если нарушение техники безопасности связано с причинением имущественного ущерба, обучающийся несет и материальную ответственность в установленном законом порядке.

## 1.2 Нормативно-правовые документы

Оформление и содержание технического задания должно соответствовать требованиям стандарта «ГОСТ 19.201-78. ЕСПД. Общие положения.

Техническое задание оформляют в соответствии с ГОСТ 19.106-78 на листах формата 11 и 12 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставляются в верхней части листа над текстом.

Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78.

Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

Для внесения изменений или дополнений в техническое задание на последующих стадиях разработки про1раммы или программного изделия выпускают дополнение к нему.

Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

Техническое задание должно содержать следующие разделы:

* введение;
* основания для разработки;
* назначение разработки;
* требования к программе или программному изделию;
* требования к программной документации;
* технико-экономические показатели;
* стадии и этапы разработки;
* порядок контроля и приемки;
* в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

Вывод по главе: были описаны, техника безопасности во время прохождения учебной практики; нормативно-правовые документы, которые отвечают за стандарты разработки; цели и задачи учебной практики, включая вводную часть, где описана тематика отрасли и её тенденции развития.

# Предметная область. Анализ предметной области. Постановка задачи.

Предметная область информационной системы — это материальная система или система, характеризующая элементы материального мира, информация о которой хранится и обрабатывается. Предметная область рассматривается как некоторая совокупность реальных объектов и связей между ними

## 2.1 Описание предметной области

В рамках учебной практики было выдано следующее задание: разработать прикладное программное обеспечение деятельности отдела вневедомственной охраны квартир. Этот отдел обеспечивает электронную охрану квартир граждан в одном районе города. Для установки охранной сигнализации требуется наличие квартирного телефона. Один гражданин может заключить договор на охрану нескольких квартир. Из-за ложных срабатываний сигнализации возможно несколько выездов патрульных экипажей по одной квартире. На владельца квартиры, вовремя не отключившего сигнализацию после своего прихода домой, налагается штраф, величина которого оговаривается при заключении договора охраны. Если отдел вневедомственной охраны не уберег имущество владельца квартиры, то он выплачивает пострадавшему заранее оговоренную сумму. От величины этой суммы зависит размер ежемесячной оплаты за охрану квартиры.

Необходимо создать базу данных, обеспечивающая деятельность отдела вневедомственной охраны квартир.

В приложении «А» на рисунке 1 представлены набор данных к информационной системе «Охрана квартир».

## 2.2 Группы пользователей информационной системы

В данной информационная системе можно выделить несколько пользователей:

* Случайный пользователь (Гость) – данный пользователь не может взаимодействовать с информационной системой.
* Конечный пользователь (потребитель информационной системы) – лицо или группа лиц, в интересах которых работает ИС. Данный пользователь был зарегистрирован в информационной системе и, как правило, не является программистом
* Коллектив специалистов (персонал информационной системы) – включающий администратора, командир экипажа и оператора.

## 2.3 Основные требования, предъявляемые к информационной системе

Данная информационная система должна соответствовать следующим требованиям:

* Гибкость – данная информационная система должна иметь возможность добавления новых таблиц.
* Надежность – данная информационная система должна выполнять свои функции.
* Эффективность – данная информационная система должна максимально быстро выполнять свои функции.
* Безопасность – данная информационная система должна быть защищен от взломов и иметь возможность восстановления и создания резерва.

Вывод по главе: была описана предметная область, также были описаны группы пользователей и основные требования, предъявляемые к информационной системе «Охрана квартир»,

# ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «Охрана квартир»

Проектированием информационных систем называется многоступенчатый процесс их создания и/или модернизации путём применения упорядоченной совокупности методологий и инструментария. Проектирование (в отличие от моделирования) предполагает работу с пока несуществующим объектом и направлено на создание информационной системы в области:

* обработки объектов будущей базы данных,
* написания программ (в том числе – отчётных и экранных форм), обеспечивающих выполнение запросов к данным,
* выполнения учёта функционирования конкретной среды (технологии).

Если выделять стадию проектирования информационных систем в качестве отдельного этапа, то его можно разместить между этапами анализа и разработки. Однако на практике чёткое разделение на этапы, как правило, затруднено или невозможно, поскольку проектирование, формально начинаясь с определения цели проекта, часто продолжается на стадиях тестирования и реализации.

## 3.1 Моделирование некоторых динамических аспектов системы

Диаграммы деятельности применяются для моделирования динамических аспектов поведения системы. Эти динамические аспекты могут включать деятельность на любом уровне абстракции в любом представлении системной архитектуры, включая классы (в том числе активные), интерфейсы, компоненты и узлы.

Использовать диаграммы деятельности для моделирования некоторых динамических аспектов системы вы можете в контексте почти любого моделируемого элемента. Однако чаще вы будете прибегать к таким диаграммам в контексте всей системы, подсистемы, операции или класса. Диаграмму деятельности можно присоединить к варианту использования, чтобы моделировать сценарий, и к кооперации, чтобы моделировать динамические аспекты поведения совокупности объектов.

При разработке программного обеспечения заблаговременное планирование и моделирование значительно упрощают программирование. На этапе создания концептуальной модели для описания функционального назначения используется диаграмма вариантов использования, а для описания логики процедур и бизнес-процессов используется диаграмма деятельности.

### 3.1.1 Создание диаграммы вариантов использования

Перед началом разработки информационной системы необходимо создать визуальное представление о вариантах использования разрабатываемой системы.

Диаграмма вариантов использования является концептуальным представлением системы в процессе ее проектирования и разработки. Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования.

Цели построения диаграммы вариантов использования:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования;
* сформулировать общие требования к функциональному проектированию системы;
* разработать исходную концептуальную модель системы для ее последующей реализации;
* подготовить документацию для взаимодействия разработчика системы с ее заказчиком и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых, вариантов использования. При этом актером или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

В свою очередь, вариант использования служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером.

В Приложении «А» на рисунке 2 представлена диаграмма вариантов использования для информационной системы «Охрана квартир». На этой диаграмме представлены следующие актеры:

1. Случайный пользователь (Гость) – может добавлять записи в таблицы: Client.
2. Конечный пользователь – может просматривать и читать таблицы: Capture, Contract, Flat, House, Prolonging, Calling. Также может добавлять записи в таблицы: Prolonging, Flat, House, Contract.
3. Командир экипажа – может просматривать и читать таблицы: House, Contract, Flat, Client, Flat-House, Settlement. И добавлять, изменять и удалять записи в таблице Capture, Calling.
4. Оператор – может изменять, удалять и добавлять данные во всех таблицах.
5. Администратор – может добавлять, удалять и изменять все таблицы в информационной системе. Также он может добавлять новых пользователей и разграничивать права доступа к базе данных. Это необходимо для того, чтобы сотрудники, не имеющие определенных полномочий, не могли получить доступ к различным объектам и данным.

### 3.1.2 Создание диаграммы деятельности

При моделировании поведения системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для этого применяется диаграмма деятельности.

Диаграмма деятельности — диаграмма, описывающая динамические аспекты системы. Диаграмма деятельности UML позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему.

В Приложении «А» на рисунке 3 находится диаграмма деятельности «Гость», которая показывает последовательность действий гостя, необходимых для достижения той или иной цели.

В Приложении «А» на рисунке 4 находится диаграмма деятельности «Гость», которая показывает последовательность действий гостя, необходимых для достижения той или иной цели.

В Приложении «А» на рисунке 5 находится диаграмма деятельности «Гость», которая показывает последовательность действий гостя, необходимых для достижения той или иной цели.

В Приложении «А» на рисунке 6 находится диаграмма деятельности «Гость», которая показывает последовательность действий гостя, необходимых для достижения той или иной цели.

В Приложении «А» на рисунке 7 находится диаграмма деятельности «Гость», которая показывает последовательность действий гостя, необходимых для достижения той или иной цели.

## 3.2 Разработка функциональной структуры информационной системы

Принципы модульного программирования во многом схожи с нисходящим проектированием, сначала определяется состав и подчиненность функции, а затем набор программных модулей, реализующих эти функции. Функции верхнего уровня обеспечиваются главным модулем, он управляет выполнением нижестоящих функций, которым соответствуют подчиненные модули. При определении набора модулей необходимо учитывать:

* каждый модуль вызывается на выполнение вышестоящим;
* принятие основных решений в алгоритме выносится на максимально высокий уровень по иерархии уровней;
* для использования одной и той же функции создается один модуль, который вызывается на выполнение по мере необходимости.

Карта навигации по информационной системе представлена в приложении «А» на рисунке 8 представлена карта переходов по информационной системе «Охрана квартир».

## 3.3 Разработка макета информационной системы

Прежде чем приступить к разработке информационной системы, сначала необходимо сориентироваться в элементах будущего интерфейса. Для этого уже определена задача, исследована предметная область, аудитория и то, как она будет пользоваться программным продуктом.

Пользовательский интерфейс — это внешний вид продукта, способ общения между пользователем и программой. Задача разработки интерфейса: сделать его полезным, интуитивно понятным и максимально простым.

Качественный дизайн повышает внешнюю привлекательность информационной системы за счет стратегической реализации таких элементов, как шрифты, цвета и изображения.

Так как разрабатываемая информационная система предназначена для широкого круга пользователей и предполагает разграничение прав доступа, необходима авторизация и регистрация.

На рисунке 9 представлен макет окна авторизации.

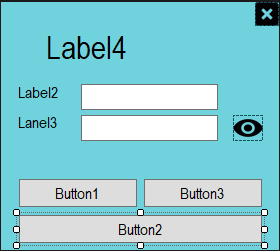


Рисунок 9 – Макет окна Авторизации

На этапе авторизации пользователь может ввести логин и пароль, чтобы продолжить работу в информационной системе.

В случае если пользователь еще не зарегистрирован в системе, он может сделать это, нажав на кнопку «Зарегистрироваться». В этом случае откроется окно регистрации, представленное на рисунке 10.

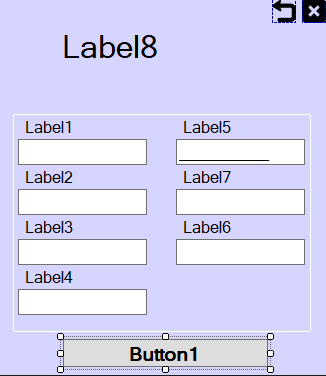


Рисунок 10 – Макет окна регистрации

На этапе регистрации пользователю необходимо придумать свой логин и пароль, а также ввести персональные данные, такие как фамилия, имя, отчество, номер телефона и адрес.

После того, как пользователь успешно прошел авторизацию, он может продолжить работу с таблицами на главной форме. На рисунке 11 - 14 представлена главная форма, которая будет отображена, в зависимости от прав пользователя в системе.



Рисунок 11 – Окно Администратора.



Рисунок 12 – Окно оператора

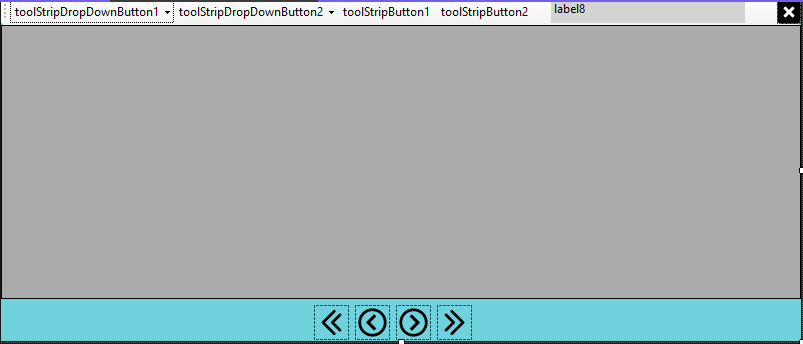


Рисунок 13 – Окно командира



Рисунок 14 – Окно Конечного пользователя

У конечного пользователя есть возможность зарегистрировать свою квартиру и это делается на отдельной форме. На рисунке 15 представлено окно регистрации квартиры.

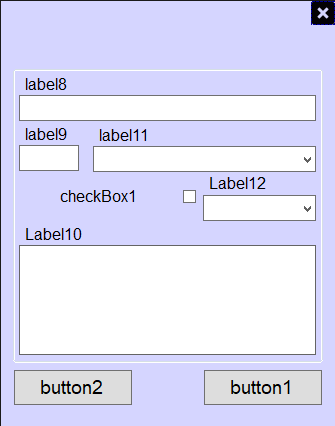


Рисунок 15 – Окно «Регистрация квартиры»

Также конечный пользователь может поменять свои персональные данные и продливать договор. На рисунках 16 и 17 представлены формы «Профиль» и «Продление договора».

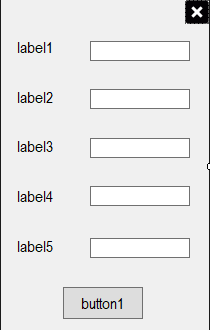


Рисунок 16 – форма Профиля клиента

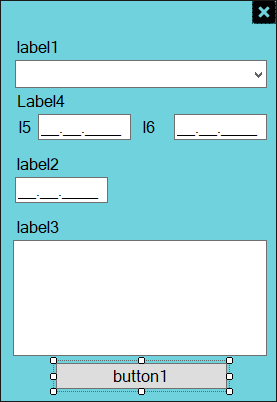


Рисунок 17 – Форма «Продление договора»

Аналогичным образом в зависимости от роли пользователя выглядит главная форма, на которой расположены все необходимые таблицы. Например, командир может изменять только определённые таблицы.

Дизайн выполнен в минималистичном стиле для того, чтобы пользователь с любым уровнем владения компьютером мог без препятствий работать с программным обеспечением.

Вывод по главе: была спроектирована диаграмма вариантов использования, диаграмма деятельности, карта переходов и макеты будущей информационной системы.

# РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ «Охрана квартир»

Прототипирование — это один из этапов разработки, который заключается в продумывании содержания и расположения важных элементов интерфейса. Прототип — это моделирование конечного продукта. Это интерактивный макет, который может иметь любую степень точности. Основная цель создания прототипов – проверить, насколько последователен путь пользователя, и выявить препятствия, которые могут возникнуть в процессе его взаимодействия с продуктом.

Прототипы не только позволяют проверить удобство разрабатываемого продукта до начала написания кода, они также приводят к неожиданным открытиям и новым идеям, которые могут вывести проектируемый продукт на новый уровень.

## 4.1 Моделирование и разработка базы данных

Исходя из анализа предметной области и набора, данных предоставленные вариантом учебной практики из Приложения «А» (рис. 1), можно выделить 9 сущностей (таблиц): Client, Flat, House, Contract, Prolonging, Capture, Settlement, Flat-House, Calling.

Описание сущностей:

Один клиент, может иметь несколько квартир, поэтому между объектами Client и Settlement имеется связь «один-ко-многим». В одном доме может быть несколько квартир, поэтому между объектами Flat-House и House имеется связь «один-ко-многим». На одну квартиру можно заключить несколько договоров, поэтому между объектами Settlement лицо и Contract имеется связь «один-ко-многим». На одну квартиру может выехать несколько бригад на захват, поэтому между объектами Capture и Contract имеется связь «один-ко-многим». Договор можно несколько раз продлить, поэтому между объектами Contract и Prolonging имеется связь «один-ко-многим».

* к объекту Client относятся такие характеристики как: SecondName, FirstName, ThirdName, Phone, Address и Registr, который является полем первичного ключа;
* к объекту Flat относится: AddressFlat, Floor, TypeDoor, TypeBalcony, Balcony, Plan и FlatID, который является полем первичного ключа;
* к объекту House относится: Floors, Key, TypeHouse и HouseID, который является первичным ключом;
* к объекту Settlement относится: Registr, который является полем внешнего ключа и соединен с полем Registr в таблице Client, FlatID, который является полем внешнего ключа и соединен с полем FlatID в таблице Flat, SettlementID, который является полем первичного ключа;
* к объекту Flat-House относится: FlatID, который является полем внешнего ключа и соединен с полем FlatID в таблице Flat, HouseID, который является полем внешнего ключа и соединен с полем HouseID в таблице House, Flat-HouseID, который является полем первичного ключа;
* к объекту Contract относится: ContractName, DataStart, StopDate, Cost, SettlementID, который является полем внешнего ключа и соединен с полем SettlementID в таблице Settlement, TreatyID, который является полем первичного ключа;
* к объекту Prolonging относится: Prolong, Comment, TreatyID, который является полем внешнего ключа и соединен с полем TreatyID в таблице Contract, ProlongingID, который является полем первичного ключа;
* к объекту Capture относится: PatrollID, Chief, Brand, Document, ActionID, который является полем первичного ключа;
* к объекту Calling относится: DateTime, False, TreatyID, который является полем внешнего ключа и соединен с полем TreatyID в таблице Contract, ActionID, который является полем внешнего ключа и соединен с полем ActionID в таблице Capture, CallingID, который является полем первичного ключа;

В приложении «А» на рисунке 18 представлена логическая модель данных, созданная по этому набору данных.

С помощью раздела «Создание» и конструктора таблиц MS Visual Studio созданы все таблицы и поля, представленные на логической модели данных.

Названия полей и их типы данных представлены в таблицах 1-10.

Таблица 1 – Поля таблицы «Client»

| Название поля | Тип данных |
| --- | --- |
| Registr | Числовой |
| SecondName | Текстовый |
| FirstName | Текстовый |
| ThirdName | Текстовый |
| Address | Текстовый |
| Phone | Текстовый |

Таблица 2 – Поля таблицы «Flat»

| Название поля | Тип данных |
| --- | --- |
| FlatID | Числовой |
| AddressFlat | Текстовый |
| Floor | Текстовый |
| TypeDoor | Текстовый |
| Balcony | Логический |
| TypeBalcony | Текстовый |
| Plane | Image |

Таблица 3 – Поля таблицы «House»

| Название поля | Тип данных |
| --- | --- |
| HouseID | Числовой |
| Floors | Текстовый |
| Key | Логический |
| TypeHouse | Текстовый |

Таблица 4 – Поля таблицы «Flat-House»

| Название поля | Тип данных |
| --- | --- |
| Flat-HouseID | Числовой |
| HouseID | Числовой |
| FlatID | Числовой |

Таблица 5 – Поля таблицы «Settlement»

| Название поля | Тип данных |
| --- | --- |
| SettlementID | Числовой |
| Registr | Числовой |
| FlatID | Числовой |

Таблица 6 – Поля таблицы «Contract»

| Название поля | Тип данных |
| --- | --- |
| TreatyID | Числовой |
| SettlementID | Числовой |
| DateStart | Дата |
| DateStop | Дата |
| Cost | Денежный |

Таблица 7 – Поля таблицы «Prolonging»

| Название поля | Тип данных |
| --- | --- |
| ProlongingID | Числовой |
| TreatyID | Числовой |
| Prolong | Дата |
| Comment | Текстовый |

Таблица 8 – Поля таблицы «Capture»

| Название поля | Тип данных |
| --- | --- |
| ActionID | Числовой |
| PatrolID | Числовой |
| Chief | Текстовый |
| Brand | Текстовый |
| Document | Текстовый |

Таблица 9 – Поля таблицы «Calling»

| Название поля | Тип данных |
| --- | --- |
| CallingID | Числовой |
| TreatyID | Числовой |
| DateTime | Дата |
| False | Логический |
| Tax | Числовой |
| Compensation | Числовой |

Таблица 10 – Поля таблицы «User»

| Название поля | Тип данных |
| --- | --- |
| Registr | Числовой |
| Логин | Текстовый |
| Пароль | Текстовый |
| Права доступа | Текстовый |

На рисунке 19 представлены все созданные таблицы в обозревателе объектов базы данных.

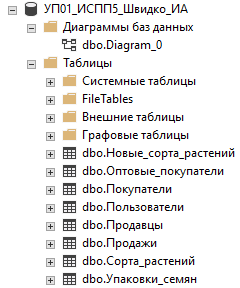
****

Рисунок 19 – Созданные таблицы

После того, как таблицы созданы, необходимо создать между ними связи.

В приложении «А» На рисунке 20 представлена физическая модель данных, отражающая все таблицы и их поля, а также связи между таблицами.

## 4.2. Разработка программных модулей информационной системы «Охрана квартир»

Модульное программирование — это организация программы как совокупности небольших блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Принцип модульности является средством упрощения задачи проектирования программного обеспечения и распределения процесса разработки. При разбиении ПО на модули для каждого модуля указывается реализуемая им функциональность, а также связи с другими модулями. Удобство использования модульной архитектуры заключается в возможности обновления или замены модуля, без необходимости изменения остальной системы.

Разработанные в MS Visual Studio формы продемонстрированы на рисунках 21 – 28.

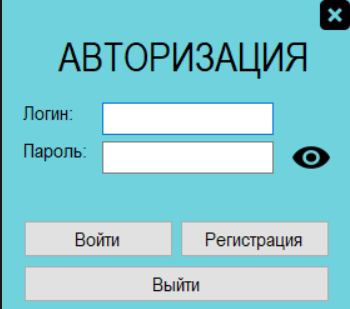


Рисунок 21 – Форма «Авторизации»

Свойства элементов формы «Авторизация» представлены в таблице 11.

Таблица 11 – Свойства элементов формы «Авторизация».

| Объект | Свойство | Значение |
| --- | --- | --- |
| Авторизация | FormBorderStyle | None |
| BackColor | 111; 210; 221 |
| Text | «Авторизация» |
| Size | 371; 305 |
| StartPosition | CenterScreen |
| Надпись1\_Label1 | Text | «» |
| Font | Arial Narrow; 12pt |
| Логин\_Label2 | Text | «Логин:» |
| Font | Arial Narrow; 12pt |
| Пароль\_Label3 | Text | «Пароль:» |
| Font | Arial Narrow; 12pt |
| НадписьАвторизация\_Label4 | Text | «АВТОРИЗАЦИЯ» |
| Font | Arial Narrow; 12pt |
| ПолеЛогин\_TextBox1 | MaxLength | 50 |
| ShortcutsEnabled | false |
| Font | Arial Narrow; 12pt |
| ПолеПароль\_TextBox2 | PasswordChar | «\*» |
| ShortcutsEnabled | false |
| Font | Arial Narrow; 12pt |
| Закрыть\_PictureBox1 | BackColor | 111; 210; 221 |
| Size | 24; 24 |
| СкрытьПароль\_PictureBox2 | BackColor | 111; 210; 221 |
| Size | 30; 26 |
| Войти\_Button1 | Text | «Войти» |
| Font | Arial Narrow; 12pt |
| Выйти\_Button2 | Text | «Выйти» |
| Font | Arial Narrow; 12pt |
| Регистрация\_Button3 | Text | «Регистрация» |
| Font | Arial Narrow; 12pt |

Событийно-управляемые процедуры в форме «Авторизация» представлены в таблице 12

Таблица 12 – Событийно-управляемые процедуры в форме «Авторизация»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Авторизация | public int ogr = 5;  int click = 0;  Point mouse;  public Авторизация()  {  InitializeComponent();  } |
| Войти\_Click | private void Войти\_Click(object sender, EventArgs e)  {  if (ПолеЛогин.Text == "" && ПолеПароль.Text == "")  {  MessageBox.Show("Введите и Логин и Пароль", "Ошибка");  }  else if (ogr == 0)  {  MessageBox.Show("Количество ваших попыток было исчерпанно\nОбратитись к администратору и закройте форму авторизации");  Войти.Visible = false;  Регистрация.Visible = false;  Логин.Visible = false;  Надпись1.Visible = false;  Пароль.Visible = false;  ПолеЛогин.Visible = false;  ПолеПароль.Visible = false;  СкрытьПароль.Visible = false;  }  else  {  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog =уП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  SqlDataAdapter da = new SqlDataAdapter("select \* from [User]", sqlConnect);  DataTable dt = new DataTable();  da.Fill(dt);  Boolean логин = false;  int g = 0;  Boolean пароль = false;  for (int i = 0; i < dt.Rows.Count; i++)  {  if ((dt.Rows[i]["Логин"].ToString() == ПолеЛогин.Text) && (dt.Rows[i]["Пароль"].ToString() == ПолеПароль.Text))  {  логин = true; пароль = true; g = i;  }  if ((dt.Rows[i]["Логин"].ToString() != ПолеЛогин.Text) && (dt.Rows[i]["Пароль"].ToString() == ПолеПароль.Text))  {  логин = false; пароль = true; g = i;  }  if ((dt.Rows[i]["Логин"].ToString() == ПолеЛогин.Text) && (dt.Rows[i]["Пароль"].ToString() != ПолеПароль.Text))  {  логин = true; пароль = false; g = i;  }  }  if (логин == true && пароль == true)  {  switch (dt.Rows[g]["Права доступа"].ToString())  {  case "КП":  this.Hide();  КП f4 = new КП();  f4.Show();  f4.registr(dt.Rows[g]["Registr"].ToString());  break;  case "Администратор":  this.Hide();  Администратор f57 = new Администратор();  f57.Show();  f57.Права("Админ");  break;  case "Командир":  Командир f2 = new Командир();  this.Hide();  f2.Show();  break;  case "Оператор":  this.Hide();  Администратор f5 = new Администратор();  f5.Show();  f5.Права("Опер");  break;  }  }  else if (логин == false || пароль == false)  {  MessageBox.Show("Неправельные Логин или Пароль, попробуйте введите еще раз");  ogr = ogr - 1;  Надпись1.Text = $"Для входа осталось попыток: {ogr}";  }  }  }  } |
| СкрытьПароль\_Click | private void СкрытьПароль\_Click(object sender, EventArgs e)  {  click += 1;  if (click == 1)  {  СкрытьПароль.ImageLocation = $"//apetfs/ИСПП-5/Неверов\_ДС/УП ПМ 01/Охрана квартир/Охрана квартир/5618419.png";  ПолеПароль.PasswordChar = '\0';  }  else if (click == 2)  {  СкрытьПароль.ImageLocation = $"//apetfs/ИСПП-5/Неверов\_ДС/УП ПМ 01/Охрана квартир/Охрана квартир/free-icon-visibility-button-60809.png";  ПолеПароль.PasswordChar = '\*';  click = 0;  }  } |
| ПолеЛогин\_KeyPress | private void ПолеЛогин\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеПароль.Focus();//Фокус на ПолеПароль  }  } |
| ПолеПароль\_KeyPress | private void ПолеПароль\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  Войти.Focus();//Фокус на кнопку Войти  }  } |
| Закрыть\_Click | private void Закрыть\_Click(object sender, EventArgs e)  {  //Выйти из приложения  Application.Exit();  } |
| Регистрация\_Click | private void Регистрация\_Click(object sender, EventArgs e)  {  //Переход на форму Регистрация  Регистрация а1 = new Регистрация();  Hide();  а1.Show();  } |
| Авторизация\_Load | private void Авторизация\_Load(object sender, EventArgs e)  {  ToolTip tt = new ToolTip();  tt.SetToolTip(Регистрация, "Нажмите, чтобы оформить документ и зарегестрироваться");  tt.SetToolTip(СкрытьПароль, "Нажмите, чтобы просмотреть пароль");  } |
| Выйти\_Click | private void Выйти\_Click(object sender, EventArgs e)  {  //Выйти из приложения  Application.Exit();  } |
| Авторизация\_MouseDown | private void Авторизация\_MouseDown(object sender, MouseEventArgs e)  {  //отпеделяет точку где была эажата клавиша мыши на форме  mouse = new Point(e.X, e.Y);  } |
| Авторизация\_MouseMove | private void Авторизация\_MouseMove(object sender, MouseEventArgs e)  {  //передвигает форму по тому как движется мышка по экрану  if (e.Button == MouseButtons.Left)  {  Left += e.X - mouse.X;  Top += e.Y - mouse.Y;  }  } |

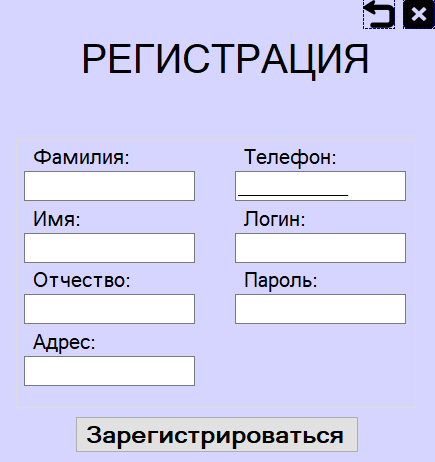


Рисунок 22 – Форма «Регистрация»

Свойства элементов формы «Регистрация» представлены в таблице 13.

Таблица 13 – Свойства элементов формы «Регистрация».

| Объект | Свойство | Значение |
| --- | --- | --- |
| Регистрация\_Form | FormBorderStyle | None |
| BackColor | 213; 213; 255 |
| Text | «Регистрация» |
| Size | 435; 462 |
| StartPosition | CenterScreen |
| Фамилия\_Label1 | Text | «Фамилия:» |
| Font | Microsoft Sans Serif; 12pt |
| Имя\_Label2 | Text | «Имя:» |
| Font | Microsoft Sans Serif; 12pt |
| Отчество\_Label3 | Text | «Отчество:» |
| Font | Microsoft Sans Serif; 12pt |
| Адрес\_Label4 | Text | «Адрес:» |
| Font | Microsoft Sans Serif; 12pt |
| Телефон\_Label5 | Text | «Телефон:» |
| Font | Microsoft Sans Serif; 12pt |
| Логин\_Label6 | Text | «Логин:» |
| Font | Microsoft Sans Serif; 12pt |
| Пароль\_Label7 | Text | «Пароль:» |
| Font | Microsoft Sans Serif; 12pt |
| НадписьРегистрация\_Label8 | Text | «РЕГИСТРАЦИЯ» |
| Font | Microsoft Sans Serif; 24pt |
| Надпись\_Label9 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| ДанныеКлиента\_GroupBox1 | Text | «» |
| Size | 400; 277 |
| ПолеФамилия\_TextBox1 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеИмя\_TextBox2 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 60 |
| ПолеОтчество\_TextBox3 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеАдрес\_TextBox4 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеЛогин\_TextBox5 | MaxLength | 40 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеПароль\_TextBox6 | MaxLength | 40 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеТелефон\_MaskedTextBox1 | Mask | 9990000000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Выйти\_PictureBox1 | BackColor | 213; 213; 255 |
| Вернуться\_PictureBox2 | BackColor | 213; 213; 255 |
| Зарегистрироваться\_Button1 | Text | «Зарегистрироваться» |
| Font | Arial Narrow; 12pt |

Событийно-управляемые процедуры в форме «Регистрация» представлены в таблице 14

Таблица 14 – Событийно-управляемые процедуры в форме «Регистрация»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Регистрация | public partial class Регистрация : Form  {  Point mouse;  int index;  public Регистрация()  {  InitializeComponent();  } |
| Регистрация\_Load | private void Регистрация\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet1.Client". При необходимости она может быть перемещена или удалена.  this.clientTableAdapter1.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet1.Client);  Надпись.Text = "Введите свои персональные данные";  this.Size = new Size(325, 375);  ToolTip tt = new ToolTip();//Данные строки позволяют показывать сплывающую подсказку  tt.SetToolTip(Выйти, "Нажмите, чтобы закрыть приложение");  tt.SetToolTip(Вернуться, "Нажмите, чтобы вернуться к форме авторизации");  for (int i = 0; i < ТаблицаКлиент.RowCount; i++)  {  index = int.Parse(ТаблицаКлиент[0, i].Value.ToString());  }  } |
| Зарегистрироваться\_Click | private void Зарегистрироваться\_Click(object sender, EventArgs e)  {  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  SqlDataAdapter da = new SqlDataAdapter("select \* from [User]", sqlConnect);  DataTable dt = new DataTable();  da.Fill(dt);  Boolean логин = true;  for (int i = 0; i < dt.Rows.Count; i++)  {  if (dt.Rows[i]["Логин"].ToString() == ПолеЛогин.Text)  {  логин = false;  }  }  if (ПолеФамилия.Text == "" || ПолеИмя.Text == "" || ПолеАдрес.Text == "" || ПолеТелефон.Text == "" || ПолеЛогин.Text == "" || ПолеПароль.Text == "")  {  MessageBox.Show("Введите все данные!");  }  else  {  if (логин == false)  {  MessageBox.Show("Пользователь уже существует!!!");  }  else  {  index = index + 1;  //создание новых записей в таблицах  SqlDataAdapter info1 = new SqlDataAdapter($"INSERT INTO [Client] (Registr,SecondName,FirstName,ThirdName,Address,Phone)VALUES('{index}','{ПолеФамилия.Text}','{ПолеИмя.Text}','{ПолеОтчество.Text}','{ПолеАдрес.Text}','{ПолеТелефон.Text}');", sqlConnect);  SqlDataAdapter info2 = new SqlDataAdapter($"INSERT INTO [User] (Registr,Логин,Пароль)VALUES('{index}','{ПолеЛогин.Text}','{ПолеПароль.Text}');", sqlConnect);  DataTable dt1 = new DataTable();  DataTable dt2 = new DataTable();  info1.Fill(dt1);  info2.Fill(dt2);  }  MessageBox.Show($"Пользователь {ПолеИмя.Text} {ПолеФамилия.Text} {ПолеОтчество.Text} зарегестрирован.\nСейчас вас перекинет на форму авторизации, где вам необходимо ввести логин и пароль, который вы вводили.");  Авторизация fd = new Авторизация();  Hide();  fd.Show();  }  }  } |
| Выйти\_Click | private void Выйти\_Click(object sender, EventArgs e)  {  //закрытие приложения  Application.Exit();  } |
| Вернуться\_Click | private void Вернуться\_Click(object sender, EventArgs e)  {  //переход на форму авторизации  Авторизация fd = new Авторизация();  Hide();  fd.Show();  } |
| ПолеФамилия\_KeyPress | private void ПолеФамилия\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеИмя.Focus();  }  } |
| ПолеИмя\_KeyPress | private void ПолеИмя\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеОтчество.Focus();  }  } |
| ПолеОтчество\_KeyPress | private void ПолеОтчество\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеАдрес.Focus();  }  } |
| ПолеАдрес\_KeyPress | private void ПолеАдрес\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеТелефон.Focus();  }  } |
| ПолеТелефон\_KeyPress | private void ПолеТелефон\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеЛогин.Focus();  }  } |
| ПолеЛогин\_KeyPress | private void ПолеЛогин\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеПароль.Focus();  }  } |
| ПолеПароль\_KeyPress | private void ПолеПароль\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  Зарегистрироваться.Focus();  }  } |
| Регистрация\_MouseMove | private void Регистрация\_MouseMove(object sender, MouseEventArgs e)  {  //передвигает форму по тому как движется мышка по экрану  if (e.Button == MouseButtons.Left)  {  Left += e.X - mouse.X;  Top += e.Y - mouse.Y;  }  } |
| Регистрация\_MouseDown | private void Регистрация\_MouseDown(object sender, MouseEventArgs e)  {  //отпеделяет точку где была эажата клавиша мыши на форме  mouse = new Point(e.X, e.Y);  } |

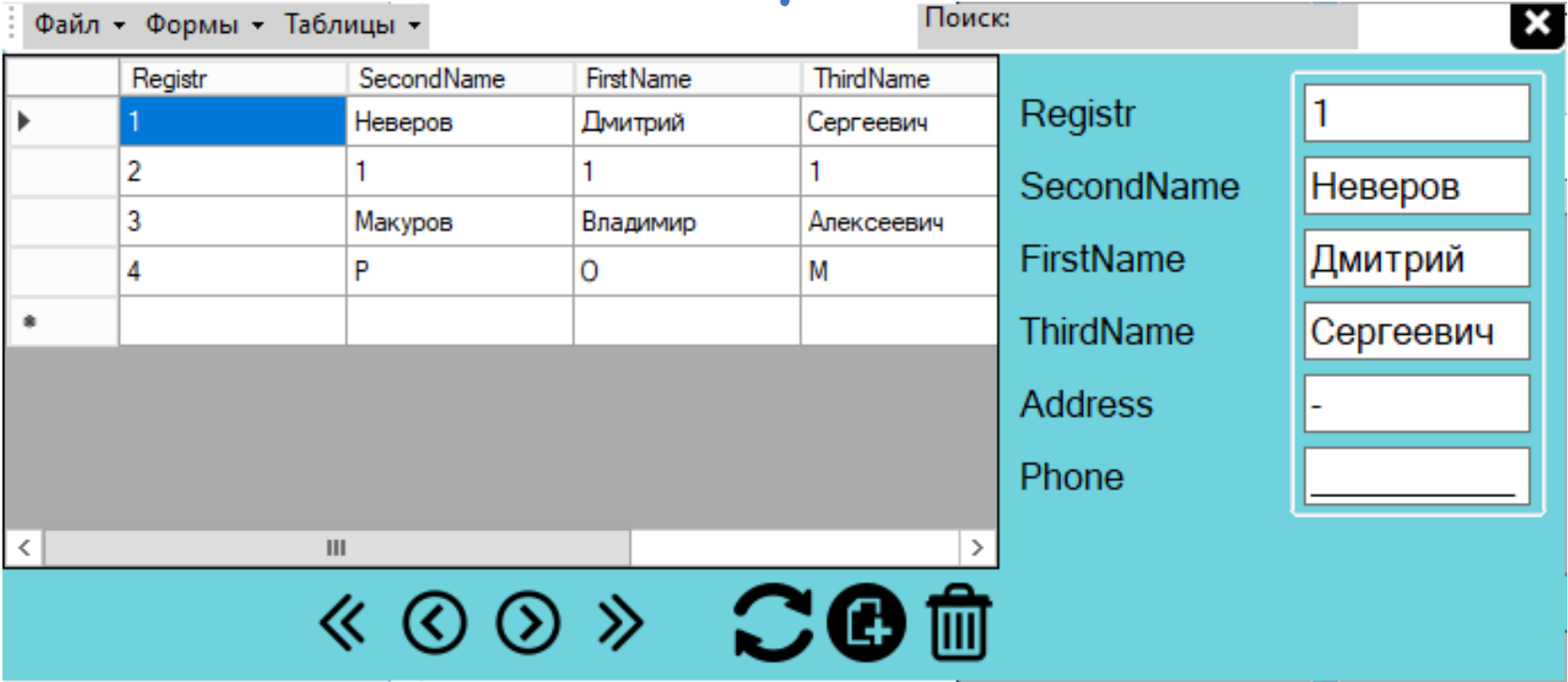


Рисунок 23 – Форма «Администратора»

Свойства элементов формы «Администратор» представлены в таблице 15.

Таблица 15 – Свойства элементов формы «Администратор».

| Объект | Свойство | Значение |
| --- | --- | --- |
| Администратор\_Form | FormBorderStyle | None |
| BackColor | 111; 210; 221 |
| Text | «Форма Администратор» |
| Size | 688, 301 |
| StartPosition | CenterScreen |
| Меню\_ToolStrip1 | Items | Файл\_toolStripDropDownButton1 |
| Формы\_toolStripDropDownButton2 |
| Таблицы\_toolStripDropDownButton3 |
| ВставкаКартинки\_OpenFileDialog1 | Filter | «Image files (\*.BMP, \*.JPG, \*.GIF,\*.PNG)|\*.bmp;\*.jpg;\*.gif;\*.png» |
| ТаблицаАдминистратор\_  DataGridView1 | Size | 585; 279 |
| ПолеТипБалкона\_ComboBox1 | Items | «Отдельный», «Совмещенный» |
| Font | Microsoft Sans Serif; 12pt |
| ПолеТипДвери\_ComboBox2 | Items | «Металическая», «Деревянная», «Две двери» |
| Font | Microsoft Sans Serif; 12pt |
| ПолеТипДома\_ComboBox3 | Items | «Кирпичный», «Панельный» |
| Font | Microsoft Sans Serif; 12pt |
| ПолеПраваДоступа\_ComboBox4 | Items | «Администратор», «Оператор», «Командир», «КП» |
| Font | Microsoft Sans Serif; 12pt |
| ЕстьБалкон\_CheckBox1 | Text | «Есть балкон» |
| Font | Arial Narrow; 12pt |
| RightToLeft | Yes |
| ЕстьЗамок\_CheckBox2 | Text | «Есть замок» |
| Font | Arial Narrow; 12pt |
| RightToLeft | Yes |
| ЛожныйВызов\_CheckBox3 | Text | «?Ложный» |
| Font | Arial Narrow; 12pt |
| RightToLeft | Yes |
| Клиент\_GroupBox1 | Text | «» |
| Size | 151; 250 |
| Квартира\_GroupBox2 | Text | «» |
| Size | 151; 250 |
| Дом\_GroupBox3 | Text | «» |
| Size | 151; 250 |
| КвартираДом\_GroupBox4 | Text | «» |
| Size | 151; 250 |
| Заселение\_GroupBox5 | Text | «» |
| Size | 151; 250 |
| Договор\_GroupBox6 | Text | «» |
| Size | 151; 250 |
| Продление\_GroupBox7 | Text | «» |
| Size | 151; 250 |
| ГруппаЗаквата\_GroupBox8 | Text | «» |
| Size | 151; 250 |
| Вызов\_GroupBox9 | Text | «» |
| Size | 164; 289 |
| Пользователь\_GroupBox10 | Text | «» |
| Size | 151; 250 |
| Надпись1\_Label1 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись2\_Label2 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись3\_Label3 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись4\_Label4 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись5\_Label5 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись6\_Label6 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись7\_Label7 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Поиск\_Label8 | Text | «Поиск:» |
| Font | Segoe UI; 9pt |
| Доступ\_Label9 | Text | «» |
| Font | Microsoft Sans Serif; 7,8pt |
| PlanFlat\_Label10 | Text | «» |
| Font | Microsoft Sans Serif; 7,8pt |
| Plan\_Label11 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| НадписьТипБалкона\_Label12 | Text | «» |
| Font | Microsoft Sans Serif; 7,8pt |
| НадписьТипДвери\_Label13 | Text | «» |
| Font | Microsoft Sans Serif; 7,8pt |
| НадписьТипДома\_Label14 | Text | «» |
| Font | Microsoft Sans Serif; 7,8pt |
| НадписьПраваДоступа \_Label15 | Text | «» |
| Font | Microsoft Sans Serif; 7,8pt |
| ПолеКодКвартиры\_TextBox1 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодКлиента\_TextBox2 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 32767 |
| ПолеФамилия\_TextBox3 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеИмя\_TextBox4 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеОтчество\_TextBox5 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеАдрес\_TextBox6 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеАдресКвартиры\_TextBox7 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодДома\_TextBox8 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 32767 |
| ПолеКодКлиента3\_TextBox9 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеЭтажКвартиры\_TextBox10 | MaxLength | 2 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеЭтажейВДоме\_TextBox11 | MaxLength | 2 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодКвартираДом\_TextBox12 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодКвартиры2\_TextBox13 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодДома2\_TextBox14 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 32767 |
| ПолеКодЗаселения\_TextBox15 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодКвартиры3\_TextBox16 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодКлиента\_TextBox17 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодДоговора\_TextBox18 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодЗаселения2\_TextBox19 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеПлата\_TextBox20 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 15 |
| ПолеКодПродление\_TextBox21 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодДоговора2\_TextBox22 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКоммент\_TextBox23 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодГруппыЗахвата\_TextBox24 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеНомерЭкипажа\_TextBox25 | MaxLength | 4 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеДокумент\_TextBox26 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 40 |
| ПолеКомандирЭкипажа\_TextBox27 | MaxLength | 20 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеАвтомобиль\_TextBox28 | MaxLength | 15 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодВызова\_TextBox29 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодДоговора3\_TextBox30 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодГруппыЗахвата2\_TextBox31 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеШтраф\_TextBox32 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 15 |
| ПолеКомпенсация\_TextBox33 | MaxLength | 15 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеЛогин\_TextBox34 | MaxLength | 40 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеПароль\_TextBox35 | MaxLength | 40 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодКлиента3\_TextBox36 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеТелефон\_MaskedTextBox1 | Mask | 9990000000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеДатаНачала\_MaskedTextBox2 | Mask | 00/00/0000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеДатаОкончание\_MaskedTextBox3 | Mask | 00/00/0000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеДатаПродление\_MaskedTextBox4 | Mask | 00/00/0000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеДатаВызова\_MaskedTextBox5 | Mask | 00/00/0000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Выход\_PictureBox1 | BackColor | White |
| SizeMode | StretchImage |
| ПерваяЗапись\_PictureBox2 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПредыдущаяЗапись\_PictureBox3 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| СледующаяЗапись\_PictureBox4 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПоследняяЗапись\_PictureBox5 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ОбновлениеЗаписей\_PictureBox6 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ДобавитьЗапись\_PictureBox7 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| УдалитьЗапись\_PictureBox8 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПолеПланКвартиры\_PictureBox9 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |

Событийно-управляемые процедуры в форме «Администратор» представлены в таблице 16.

Таблица 16 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Администратор | public partial class Администратор : Form  {  public Администратор()  {  InitializeComponent();  }  public int c;  private Bitmap bmp;  int i, index,v,g,y;  Point mouse; |
| Администратор\_Load | private void Администратор\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat". При необходимости она может быть перемещена или удалена.  this.flatTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement". При необходимости она может быть перемещена или удалена.  this.settlementTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.User". При необходимости она может быть перемещена или удалена.  this.userTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.User);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Prolonging". При необходимости она может быть перемещена или удалена.  this.prolongingTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Prolonging);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.House". При необходимости она может быть перемещена или удалена.  this.houseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House". При необходимости она может быть перемещена или удалена.  this.flat\_HouseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling". При необходимости она может быть перемещена или удалена.  this.callingTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract". При необходимости она может быть перемещена или удалена.  this.contractTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Client". При необходимости она может быть перемещена или удалена.  this.clientTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Client);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture". При необходимости она может быть перемещена или удалена.  this.captureTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture);  ToolTip tt = new ToolTip();//Данные строки позволяют показывать сплывающую подсказку  tt.SetToolTip(ПолеПоиск, "Введите значение, которое хотите найти в таблице");  tt.SetToolTip(УдалитьЗапись, "Удаление выделенной строки");  tt.SetToolTip(ДобавитьЗапись, "Добавить строку");  tt.SetToolTip(СледующаяЗапись, "Следующая строка");  tt.SetToolTip(ОбновлениеЗаписей, "Обновить данные");  tt.SetToolTip(ПредыдущаяЗапись, "Предыдущая строка");  tt.SetToolTip(Выход, "Закрыть");  c = 0;  Table(0);  Надпись1.Visible = true;  Надпись2.Visible = true;  Надпись3.Visible = true;  Надпись4.Visible = true;  Надпись5.Visible = true;  Надпись6.Visible = true;  Надпись7.Visible = false;  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Надпись5.Text = ТаблицаАдминистратор.Columns[4].HeaderText;  Надпись6.Text = ТаблицаАдминистратор.Columns[5].HeaderText;  Клиент.Visible = true;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  //dataGridView1.AllowUserToAddRows = false;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = false;  this.Size = new Size(688, 340);  } |
| Метод Права | public void Права(string b)  {  //Данный метод определяет кто защел на форму Администратор или Оператор  Доступ.Text = b;  if (Доступ.Text == "Админ")  {  user.Visible = true;  Формы.Visible = true;  }  else  {  user.Visible = false;  Формы.Visible = false;  }  } |
| Метод Table | private void Table (int c)  {  //Данный метод нужен для присваивания к dataGridView1 таблицы из базы данных  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  switch (c)  {  case 0:  ТаблицаАдминистратор.DataSource = clientBindingSource;  break;  case 1:  ТаблицаАдминистратор.DataSource = flatBindingSource;  break;  case 2:  ТаблицаАдминистратор.DataSource = houseBindingSource;  break;  case 3:  ТаблицаАдминистратор.DataSource = flatHouseBindingSource;  break;  case 4:  ТаблицаАдминистратор.DataSource = settlementBindingSource;  break;  case 5:  ТаблицаАдминистратор.DataSource = contractBindingSource;  break;  case 6:  ТаблицаАдминистратор.DataSource = prolongingBindingSource;  break;  case 7:  ТаблицаАдминистратор.DataSource = captureBindingSource;  break;  case 8:  ТаблицаАдминистратор.DataSource = callingBindingSource;  break;  case 11:  ТаблицаАдминистратор.DataSource = userBindingSource;  break;  }  }  } |
| выйтиИзПриложения\_Click | private void выйтиИзПриложенияToolStripMenuItem\_Click\_1(object sender, EventArgs e)  {  Application.Exit();  } |
| user\_Click | private void user\_Click(object sender, EventArgs e)  {  c = 11;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible =true;  this.Size = new Size(688, 340);  Пользователь.Location = new Point(567, 26);  Надпись4.Visible = true;  Надпись6.Visible = false;  Надпись5.Visible = false;  Надпись7.Visible = false;  ПолеПраваДоступа.Text = НадписьПраваДоступа.Text;  } |
| формаГостя\_Click | private void формаГостя\_Click(object sender, EventArgs e)  {  //открытие формы регистрация  Регистрация us = new Регистрация();  us.Show();  } |
| формаКомандира\_Click | private void формаКомандира\_Click(object sender, EventArgs e)  {  //открытие формы командира  Командир com = new Командир();  com.Show();  } |
| client\_Click | private void client\_Click(object sender, EventArgs e)  {  c = 0;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Надпись5.Text = ТаблицаАдминистратор.Columns[4].HeaderText;  Надпись6.Text = ТаблицаАдминистратор.Columns[5].HeaderText;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = false;  Клиент.Visible = true;  Клиент.Location = new Point(567, 26);  Квартира.Visible = false;  Дом.Visible = false;  this.Size = new Size(688, 340);  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  Надпись6.Visible = true;  Надпись5.Visible = true;  Надпись4.Visible = true;  Надпись7.Visible = false;  } |
| flat\_Click | private void flat\_Click(object sender, EventArgs e)  {  c = 1;  Table(c);  PlanFlat.Text = ТаблицаАдминистратор.Columns[6].HeaderText;  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Надпись5.Text = ТаблицаАдминистратор.Columns[4].HeaderText;  Надпись6.Text = ТаблицаАдминистратор.Columns[5].HeaderText;  Выход.Location = new Point(839, -1);  ПолеПланКвартиры.Visible = true;  Клиент.Visible = false;  Квартира.Visible = true;  ПолеТипБалкона.Visible = false;  Квартира.Location = new Point(567, 26);  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  Надпись4.Visible = true;  Надпись5.Visible = true;  Надпись6.Visible = false;  Надпись7.Visible = false;  this.Size = new Size(863, 340);  ПолеТипБалкона.Text = НадписьТипБалкона.Text;  ПолеТипДвери.Text = НадписьТипДвери.Text;  } |
| house\_Click | private void house\_Click(object sender, EventArgs e)  {  c = 2;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = false;  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = true;  Дом.Location = new Point(567, 26);  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  Надпись4.Visible = true;  Надпись6.Visible = false;  Надпись5.Visible = false;  Надпись7.Visible = false;  this.Size = new Size(688, 340);  ПолеТипДома.Text = НадписьТипДома.Text;  } |
| flatHouse\_Click | private void flatHouse\_Click(object sender, EventArgs e)  {  c = 3;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись6.Visible = false;  Надпись5.Visible = false;  Надпись4.Visible = false;  Надпись7.Visible = false;  Выход.Location = new Point(664, -1);  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = true;  КвартираДом.Location = new Point(567, 26);  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  ПолеПланКвартиры.Visible = false;  this.Size = new Size(688, 340);  } |
| settlement\_Click | private void settlement\_Click(object sender, EventArgs e)  {  c = 4;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = false;  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = true;  Заселение.Location = new Point(567, 26);  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  Надпись6.Visible = false;  Надпись5.Visible = false;  Надпись4.Visible = false;  Надпись7.Visible = false;  this.Size = new Size(688, 340);  } |
| contract\_Click | private void contract\_Click(object sender, EventArgs e)  {  c = 5;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Надпись5.Text = ТаблицаАдминистратор.Columns[4].HeaderText;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = true;  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = true;  Договор.Location = new Point(567, 26);  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  Надпись6.Visible = false;  Надпись5.Visible = true;  Надпись4.Visible = true;  Надпись7.Visible = false;  this.Size = new Size(688, 340);  } |
| prolonging\_Click | private void prolonging\_Click(object sender, EventArgs e)  {  c = 6;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = false;  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = true;  Продление.Location = new Point(567, 26);  ГруппаЗаквата.Visible = false;  Вызов.Visible = false;  Пользователь.Visible = false;  Надпись4.Visible = true;  Надпись6.Visible = false;  Надпись5.Visible = false;  Надпись7.Visible = false;  this.Size = new Size(688, 340);  } |
| capture\_Click | private void capture\_Click(object sender, EventArgs e)  {  c = 7;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Надпись5.Text = ТаблицаАдминистратор.Columns[4].HeaderText;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = false;  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = true;  ГруппаЗаквата.Location = new Point(567, 26);  Вызов.Visible = false;  Пользователь.Visible = false;  Надпись6.Visible = false;  Надпись5.Visible = true;  Надпись4.Visible = true;  Надпись7.Visible = false;  this.Size = new Size(688, 340);  } |
| Выход\_Click | private void Выход\_Click(object sender, EventArgs e)  {  //Выход из прилдожения  Application.Exit();  } |
| вернутьсяКАвторизации\_Click | private void вернутьсяКАвторизации\_Click(object sender, EventArgs e)  {  //Переход на форму авторизации  Авторизация ав = new Авторизация();  Hide();  ав.Show();  } |
| обновить\_Click | private void обновить\_Click(object sender, EventArgs e)  {//обновлени тыблиц базы данных  switch (c)  {  case 0:clientTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Client); break;  case 1:flatTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat); break;  case 2:houseTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.House); break;  case 3:flat\_HouseTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House); break;  case 4:settlementTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement); break;  case 5:contractTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract); break;  case 6:prolongingTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Prolonging); break;  case 7:captureTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture);break;  case 8:callingTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling);break;  case 11:userTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.User); break;  }  } |
| calling\_Click | private void calling\_Click(object sender, EventArgs e)  {  c = 8;  Table(c);  Надпись1.Text = ТаблицаАдминистратор.Columns[0].HeaderText;  Надпись2.Text = ТаблицаАдминистратор.Columns[1].HeaderText;  Надпись3.Text = ТаблицаАдминистратор.Columns[2].HeaderText;  Надпись4.Text = ТаблицаАдминистратор.Columns[3].HeaderText;  Надпись5.Text = ТаблицаАдминистратор.Columns[4].HeaderText;  Надпись6.Text = ТаблицаАдминистратор.Columns[5].HeaderText;  Надпись7.Text = ТаблицаАдминистратор.Columns[6].HeaderText;  Выход.Location = new Point(664, -1);  ПолеПланКвартиры.Visible = false;  Клиент.Visible = false;  Квартира.Visible = false;  Дом.Visible = false;  КвартираДом.Visible = false;  Заселение.Visible = false;  Договор.Visible = false;  Продление.Visible = false;  ГруппаЗаквата.Visible = false;  Вызов.Visible = true;  Вызов.Location = new Point(567, 26);  Пользователь.Visible = false;  Надпись7.Visible = true;  Надпись6.Visible = true;  Надпись5.Visible = true;  Надпись4.Visible = true;  Надпись7.Visible = false;  this.Size = new Size(688, 340);  } |
| ПредыдущаяЗапись\_Click | private void ПредыдущаяЗапись\_Click(object sender, EventArgs e)  {  //Переход на предыдущую строку  i = ТаблицаАдминистратор.RowCount;  index = ТаблицаАдминистратор.CurrentRow.Index;  if (index<=0)  {  ТаблицаАдминистратор.CurrentCell = ТаблицаАдминистратор[0, i-1];  }  else  {  ТаблицаАдминистратор.Rows[index].Selected = true;  ТаблицаАдминистратор.CurrentCell = ТаблицаАдминистратор[0, index - 1];  }  ПолеПраваДоступа.Text = НадписьПраваДоступа.Text;  } |
| СледующаяЗапись\_Click | private void СледующаяЗапись\_Click(object sender, EventArgs e)  {  //Переход на следующую строку  i = ТаблицаАдминистратор.RowCount;  index = ТаблицаАдминистратор.CurrentRow.Index;  if (index >= (i - 1))  {  ТаблицаАдминистратор.CurrentCell = ТаблицаАдминистратор[0,0];  }  else  {  ТаблицаАдминистратор.Rows[index].Selected = true;  ТаблицаАдминистратор.CurrentCell = ТаблицаАдминистратор[0, index + 1];  }  ПолеПраваДоступа.Text = НадписьПраваДоступа.Text;  } |
| ДобавитьЗапись\_Click | private void ДобавитьЗапись\_Click(object sender, EventArgs e)  {  //Добавление строки  switch (c)  {  case 0:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Client] VALUES ('{ПолеКодКлиента.Text}','{ПолеФамилия.Text}','{ПолеИмя.Text}','{ПолеОтчество.Text}','{ПолеАдрес.Text}','{ПолеТелефон.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодКлиента.Text = "";  ПолеФамилия.Text = "";  ПолеИмя.Text = "";  ПолеОтчество.Text = "";  ПолеАдрес.Text = "";  ПолеТелефон.Text = "";  }  break;  case 1:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Flat] VALUES ('{ПолеКодКвартиры.Text}','{ПолеАдресКвартиры.Text}','{ПолеЭтажКвартиры.Text}','{ПолеТипДвери.Text}','{ЕстьБалкон.Text}','{ПолеТипБалкона.Text}','{Plan.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодКлиента.Text = "";  ПолеФамилия.Text = "";  ПолеИмя.Text = "";  ПолеОтчество.Text = "";  ПолеАдрес.Text = "";  ПолеТелефон.Text = "";  }  break;  case 2:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [House] VALUES ('{ПолеКодДома.Text}','{ПолеЭтажейВДоме.Text}','{y}','{ПолеТипДома.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодДома.Text = "";  ПолеЭтажейВДоме.Text = "";  ЕстьЗамок.Checked = false;  ПолеТипДома.Text = "";  }  break;  case 3:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Flat-House] VALUES ('{ПолеКодКвартираДом.Text}','{ПолеКодКвартиры2.Text}','{ПолеКодДома2.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодКвартираДом.Text = "";  ПолеКодКвартиры2.Text = "";  ПолеКодДома2.Text = "";  }  break;  case 4:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Settlement] VALUES ('{ПолеКодЗаселения.Text}','{ПолеКодКвартиры3.Text}','{ПолеКодКлиента2.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодЗаселения.Text = "";  ПолеКодКвартиры3.Text = "";  ПолеКодКлиента2.Text = "";  }  break;  case 5:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Contract] VALUES ('{ПолеКодДоговора.Text}','{ПолеКодЗаселения2.Text}','{ПолеДатаНачала.Text}','{ПолеДатаОкончание.Text}','{ПолеПлата.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодДоговора.Text = "";  ПолеКодЗаселения2.Text = "";  ПолеДатаНачала.Text = "";  ПолеДатаОкончание.Text = "";  ПолеПлата.Text = "";  }  break;  case 6:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Prolonging] VALUES ('{ПолеКодПродление.Text}','{ПолеКодДоговора2.Text}','{ПолеДатаПродление.Text}','{ПолеКоммент.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодПродление.Text = "";  ПолеКодДоговора2.Text = "";  ПолеДатаПродление.Text = "";  ПолеКоммент.Text = "";  }  break;  case 7:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Capture] VALUES ('{ПолеКодГруппыЗахвата.Text}','{ПолеНомерЭкипажа.Text}','{ПолеКомандирЭкипажа.Text}','{ПолеАвтомобиль.Text}','{ПолеДокумент.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодГруппыЗахвата.Text = "";  ПолеНомерЭкипажа.Text = "";  ПолеКомандирЭкипажа.Text = "";  ПолеАвтомобиль.Text = "";  ПолеДокумент.Text = "";  }  break;  case 8:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Calling] VALUES ('{ПолеКодВызова.Text}','{ПолеКодДоговора3.Text}','{ПолеКодГруппыЗахвата2.Text}','{ПолеДатаВызова.Text}','{v}','{ПолеШтраф.Text}','{ПолеКомпенсация.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодВызова.Text = "";  ПолеКодДоговора3.Text = "";  ПолеКодГруппыЗахвата2.Text = "";  ПолеДатаВызова.Text = "";  ЛожныйВызов.Checked = false;  ПолеШтраф.Text = "";  ПолеКомпенсация.Text = "";  }  break;  case 11:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [User] VALUES ('{ПолеЛогин.Text}','{ПолеПароль.Text}','{ПолеПраваДоступа.Text}','{ПолеКодКлиента3.Text}');", sqlConnect);  da.Fill(dt);  ПолеЛогин.Text = "";  ПолеПароль.Text = "";  ПолеКодКлиента3.Text = "";  ПолеПраваДоступа.Text = "";  }  break;  }  } |
| УдалитьЗапись\_Click | private void УдалитьЗапись\_Click(object sender, EventArgs e)  {  //удаление выделенный строки  for (int i = 0; i < ТаблицаАдминистратор.RowCount; i++)  {  if (ТаблицаАдминистратор.Rows[i].Cells[0].Selected == true)  {  ТаблицаАдминистратор.Rows.RemoveAt(i);  }  }  } |
| ПолеПланКвартиры\_Click | private void ПолеПланКвартиры\_Click(object sender, EventArgs e)  {  //открытие проводника для вставки картинки в pictureBox6 по нажатию на pictureBox6  ВставкаКартинки.Filter = "Image files (\*.BMP, \*.JPG, " + "\*.GIF, \*.PNG)|\*.bmp;\*.jpg;\*.gif;\*.png";  if (ВставкаКартинки.ShowDialog() == DialogResult.OK)  {  Image image = Image.FromFile(ВставкаКартинки.FileName);  int width = ПолеПланКвартиры.Width;  int height = ПолеПланКвартиры.Height;  bmp = new Bitmap(image, width, height);  ПолеПланКвартиры.Image = bmp;  Plan.Text = ВставкаКартинки.FileName;  }  } |
| ПолеПоиск\_TextChanged | private void ПолеПоиск\_TextChanged(object sender, EventArgs e)  {  //Поиск строк  for (int i = 0; i < ТаблицаАдминистратор.RowCount; i++)  {  ТаблицаАдминистратор.Rows[i].Selected = false;  for (int j = 0; j < ТаблицаАдминистратор.ColumnCount; j++)  {  if (ТаблицаАдминистратор.Rows[i].Cells[j].Value != null)  {  if (ТаблицаАдминистратор.Rows[i].Cells[j].Value.ToString().Contains(ПолеПоиск.Text))  {  ТаблицаАдминистратор.Rows[i].Selected = true;  break;  }  }  }  }  } |
| ТаблицаАдминистратор\_RowEnter | private void ТаблицаАдминистратор\_RowEnter(object sender, DataGridViewCellEventArgs e)  {  //вставка картинки при фокусе на строки таблицы  ПолеПланКвартиры.ImageLocation = Plan.Text;  } |
| ЕстьБалкон\_CheckedChanged | private void ЕстьБалкон\_CheckedChanged(object sender, EventArgs e)  {  if (ЕстьБалкон.Checked == true)  {  g = 1;  ПолеТипБалкона.Visible = true;  Надпись6.Visible = true;  }  else if (ЕстьБалкон.Checked == false)  {  g = 0;  ПолеТипБалкона.Visible = false;  Надпись6.Visible = false;  }  } |
| ЕстьЗамок\_CheckedChanged | private void ЕстьЗамок\_CheckedChanged(object sender, EventArgs e)  {  if (ЕстьЗамок.Checked == true)  {  y = 1;  }  else if (ЕстьЗамок.Checked == false)  {  y = 0;  }  } |
| ПолеПраваДоступа\_SelectedIndexChanged | private void ПолеПраваДоступа\_SelectedIndexChanged(object sender, EventArgs e)  {  НадписьПраваДоступа.Text = ПолеПраваДоступа.Text;  } |
| ПолеТипДома\_SelectedIndexChanged | private void ПолеТипДома\_SelectedIndexChanged(object sender, EventArgs e)  {  НадписьТипДома.Text = ПолеТипДома.Text;  } |
| Администратор\_MouseDown | private void Администратор\_MouseDown(object sender, MouseEventArgs e)  {  //отпеделяет точку где была эажата клавиша мыши на форме  mouse = new Point(e.X, e.Y);  } |
| Администратор\_MouseMove | private void Администратор\_MouseMove(object sender, MouseEventArgs e)  {  //передвигает форму по тому как движется мышка по экрану  if (e.Button == MouseButtons.Left)  {  Left += e.X - mouse.X;  Top += e.Y - mouse.Y;  }  } |
| ПолеТипДвери\_SelectedIndexChanged | private void ПолеТипДвери\_SelectedIndexChanged(object sender, EventArgs e)  {  НадписьТипДвери.Text = ПолеТипДвери.Text;  } |
| ПолеТипБалкона\_SelectedIndexChanged | private void ПолеТипБалкона\_SelectedIndexChanged(object sender, EventArgs e)  {  НадписьТипБалкона.Text = ПолеТипБалкона.Text;  } |
| ПолеКодКлиента\_KeyPress | private void ПолеКодКлиента\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеФамилия.Focus();  }  if ((e.KeyChar >='0')&&(e.KeyChar <= '9')||(e.KeyChar==(char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеФамилия\_KeyPress | private void ПолеФамилия\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеИмя.Focus();  }  } |
| ПолеИмя\_KeyPress | private void ПолеИмя\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеОтчество.Focus();  }  } |
| ПолеОтчество\_KeyPress | private void ПолеОтчество\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеАдрес.Focus();  }  } |
| ПолеАдрес\_KeyPress | private void ПолеАдрес\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеТелефон.Focus();  }  } |
| ПолеКодАдресКвартиры\_KeyPress | private void ПолеКодАдресКвартиры\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеАдресКвартиры.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеАдресКвартиры\_KeyPress | private void ПолеАдресКвартиры\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеЭтажКвартиры.Focus();  }  } |
| ПолеЭтажКвартиры\_KeyPress | private void ПолеЭтажКвартиры\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеТипДвери.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодКвартираДом\_KeyPress | private void ПолеКодКвартираДом\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодКвартиры2.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодКвартиры2\_KeyPress | private void ПолеКодКвартиры2\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодДома2.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодДома\_KeyPress | private void ПолеКодДома\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеЭтажейВДоме.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеЭтажейВДоме\_KeyPress | private void ПолеЭтажейВДоме\_KeyPress(object sender, KeyPressEventArgs e)  {  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодКлиента2\_KeyPress | private void ПолеКодКлиента2\_KeyPress(object sender, KeyPressEventArgs e)  {  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодЗаселения\_KeyPress | private void ПолеКодЗаселения\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодКвартиры3.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодКвартиры3\_KeyPress | private void ПолеКодКвартиры3\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодКлиента2.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодДоговора\_KeyPress | private void ПолеКодДоговора\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодЗаселения2.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодЗаселения2\_KeyPress | private void ПолеКодЗаселения2\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеДатаНачала.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеДатаНачала\_KeyPress | private void ПолеДатаНачала\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеДатаОкончание.Focus();  }  } |
| ПолеДатаОкончание\_KeyPress | private void ПолеДатаОкончание\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеПлата.Focus();  }  } |
| ПолеКодПродление\_KeyPress | private void ПолеКодПродление\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодДоговора2.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодДоговора2\_KeyPress | private void ПолеКодДоговора2\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеДатаПродление.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеДатаПродление\_KeyPress | private void ПолеДатаПродление\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКоммент.Focus();  }  } |
| ПолеКодГруппыЗахвата\_KeyPress | private void ПолеКодГруппыЗахвата\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеНомерЭкипажа.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеНомерЭкипажа\_KeyPress | private void ПолеНомерЭкипажа\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКомандирЭкипажа.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКомандирЭкипажа\_KeyPress | private void ПолеКомандирЭкипажа\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеАвтомобиль.Focus();  }  } |
| ПолеАвтомобиль\_KeyPress | private void ПолеАвтомобиль\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеДокумент.Focus();  }  } |
| ПолеКодВызова\_KeyPress | private void ПолеКодВызова\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодДоговора3.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодДоговора3\_KeyPress | private void ПолеКодДоговора3\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодГруппыЗахвата2.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеКодГруппыВызова\_KeyPress | private void ПолеКодГруппыВызова\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеДатаВызова.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеШтраф\_KeyPress | private void ПолеШтраф\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКомпенсация.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеЛогин\_KeyPress | private void ПолеЛогин\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеПароль.Focus();  }  } |
| ПолеПароль\_KeyPress | private void ПолеПароль\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеПраваДоступа.Focus();  }  } |
| ПолеПраваДоступа\_KeyPress | private void ПолеПраваДоступа\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеКодКлиента3.Focus();  }  } |
| ПолеКодКлиента3\_KeyPress | private void ПолеКодКлиента3\_KeyPress(object sender, KeyPressEventArgs e)  {  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ОбновлениеЗаписей\_Click | private void ОбновлениеЗаписей\_Click(object sender, EventArgs e)  {  //обновление таблиц базы данных  switch (c)  {  case 0: clientTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Client); break;  case 1: flatTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat); break;  case 2: houseTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.House); break;  case 3: flat\_HouseTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House); break;  case 4: settlementTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement); break;  case 5: contractTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract); break;  case 6: prolongingTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Prolonging); break;  case 7: captureTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture); break;  case 8: callingTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling); break;  case 11: userTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.User); break;  }  } |
| ПерваяЗапись\_Click | private void ПерваяЗапись\_Click(object sender, EventArgs e)  {  //переход на первую строку таблицы  ТаблицаАдминистратор.CurrentCell = ТаблицаАдминистратор[0, 0];  } |
| ПоследняяЗапись\_Click | private void ПоследняяЗапись\_Click(object sender, EventArgs e)  {  //переход на последную строку таблицы  ТаблицаАдминистратор.ClearSelection(); //снять выделение всех выбранных ячеек  index = ТаблицаАдминистратор.Rows.Count - 1; // индекс последней строки  ТаблицаАдминистратор.Rows[index].Selected = true; // выделить нужную строку  ТаблицаАдминистратор.CurrentCell = ТаблицаАдминистратор[0, index];  } |
| ЛожныйВызов\_CheckedChanged | private void ЛожныйВызов\_CheckedChanged(object sender, EventArgs e)  {  if (ЛожныйВызов.Checked == true)  {  v = 1;  }  else if (ЛожныйВызов.Checked == false)  {  v = 0;  }  } |

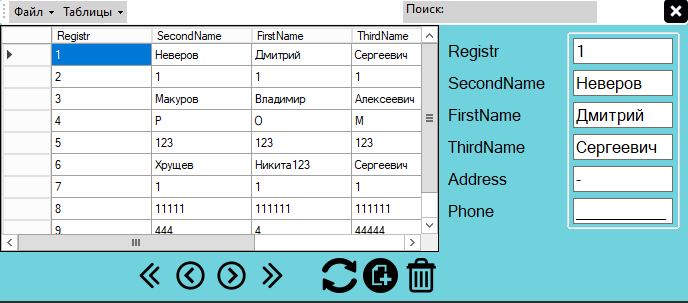


Рисунок 21 – Форма «Оператор»

Свойства элементов формы «Авторизация» представлены в таблице 17.

Таблица 17 – Свойства элементов формы «Оператор».

Событийно-управляемые процедуры в форме авторизации представлены в таблице 18.

Таблица 18 – Событийно-управляемые процедуры в форме «Оператор»

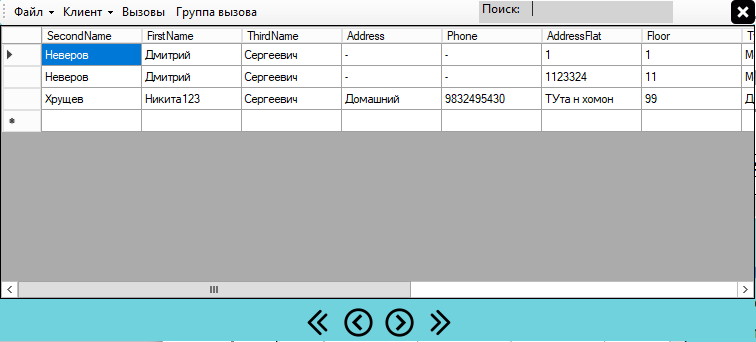


Рисунок 24 – Форма «Командир»

Свойства элементов формы «Командир» представлены в таблице 17.

Таблица 17 – Свойства элементов формы «Командир».

| Объект | Свойство | Значение |
| --- | --- | --- |
| Администратор\_Form | FormBorderStyle | None |
| BackColor | 111; 210; 221 |
| Text | «Форма Администратор» |
| Size | 688, 301 |
| StartPosition | CenterScreen |
| Меню\_ToolStrip1 | Items | toolStripDropDownButton1 |
| toolStripDropDownButton2 |
| toolStripButton1 |
| toolStripButton2 |
| ТаблицыКомандир\_  DataGridView1 | Size | 585; 279 |
| Вызовы\_GroupBox1 | Text | «» |
| Size | 164; 289 |
| Захват\_GroupBox2 | Text | «» |
| Size | 164; 289 |
| ЛожныйВызов\_  CheckBox1 | Text | ?Ложный |
| Font | Arial Narrow; 11,25pt |
| RightToLeft | Yes |
| Надпись1\_Label1 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись2\_Label2 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись3\_Label3 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись4\_Label4 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись5\_Label5 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись6\_Label6 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Надпись7\_Label7 | Text | «» |
| Font | Microsoft Sans Serif; 12pt |
| Поиск\_Label8 | Text | «Поиск:» |
| Font | Segoe UI; 9pt |
| CallingID\_TextBox1 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Segoe UI; 9pt |
| TreatyID\_TextBox2 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 32767 |
| ActionID\_TextBox3 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеКодГруппыЗахвата \_TextBox4 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Компенсация\_TextBox5 | MaxLength | 15 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Штраф\_TextBox6 | MaxLength | 15 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеПоиск\_TextBox7 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| НомерЭкипажа\_  TextBox8 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 4 |
| КомандирЭкипажа\_  TextBox9 | MaxLength | 20 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Автомобиль\_TextBox10 | MaxLength | 15 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Документ\_TextBox11 | MaxLength | 40 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ДатаВызова\_  MaskedTextBox1 | Mask | 00/00/0000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Выйти\_PictureBox2 | BackColor | White |
| SizeMode | StretchImage |
| ПерваяЗапись\_  PictureBox1 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПредыдущаяЗапись\_  PictureBox3 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| СледующаяЗапись\_  PictureBox4 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПоследняяЗапись\_  PictureBox5 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ОбновитьТаблицу\_  PictureBox6 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ДодбавитьЗапись\_  PictureBox7 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| УдалитьЗапись\_  PictureBox8 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |

Событийно-управляемые процедуры в форме «Командир» представлены в таблице 18.

Таблица 18 – Событийно-управляемые процедуры в форме «Командир»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Командир | public partial class Командир : Form  {  public Командир()  {  InitializeComponent();  }  int i, c, index,a;  Point Mouse; |
| Метод Tables | private void Tables (int c)  {  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter[] sa = new SqlDataAdapter[5];  sa[1] = new SqlDataAdapter("SELECT [SecondName], [FirstName], [ThirdName], [Address], [Phone], [AddressFlat], [Floor], [TypeDoor], [Balcony], [TypeBalcony], [FlatPlan], [Floors], [Key] FROM [Flat-House] INNER JOIN Flat ON [Flat-House].FlatID = Flat.FlatID INNER JOIN [Settlement] ON Flat.FlatID = Settlement.FlatID INNER JOIN [Client] ON Settlement.Registr = Client.Registr INNER JOIN [House] ON [Flat-House].HouseID = House.HouseID", sqlConnect);  sa[2] = new SqlDataAdapter("SELECT [SecondName], [FirstName], [ThirdName], [Address], [Phone], [DateStart], [StopDate], [Cost] FROM [Client] INNER JOIN [Settlement] ON Client.Registr = Settlement.Registr INNER JOIN [Contract] ON Settlement.SettlementID = Contract.SettlementID", sqlConnect);  sa[3] = new SqlDataAdapter("SELECT \* from [Calling]", sqlConnect);  sa[4] = new SqlDataAdapter("SELECT \* from [Capture]", sqlConnect);  DataSet ds = new DataSet();  sa[c].Fill(ds);  ТаблицыКомандир.DataSource = ds.Tables[0];  } |
| Командир\_Load | private void Командир\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement". При необходимости она может быть перемещена или удалена.  this.settlementTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.House". При необходимости она может быть перемещена или удалена.  this.houseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House". При необходимости она может быть перемещена или удалена.  this.flat\_HouseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat". При необходимости она может быть перемещена или удалена.  this.flatTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract". При необходимости она может быть перемещена или удалена.  this.contractTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Client". При необходимости она может быть перемещена или удалена.  this.clientTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Client);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture". При необходимости она может быть перемещена или удалена.  this.captureTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling". При необходимости она может быть перемещена или удалена.  this.callingTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling);  Надпись6.Visible = false;  Надпись7.Visible = false;  Надпись5.Visible = false;  Надпись4.Visible = false;  Надпись3.Visible = false;  Надпись2.Visible = false;  Надпись1.Visible = false;  УдалитьЗапись.Visible = false;  ОбновитьТаблицу.Visible = false;  ДодбавитьЗапись.Visible = false;  Вызовы.Visible = false;  Захват.Visible = false;  ТаблицыКомандир.Size = new Size(755, 274);  this.Size = new Size(755, 341);  c = 1;  Tables(c);  } |
| КвартираКлиента\_Click | private void КвартираКлиента\_Click(object sender, EventArgs e)  {  ТаблицыКомандир.Size = new Size(755, 274);  c = 1;  Tables(c);  УдалитьЗапись.Visible = false;  ОбновитьТаблицу.Visible = false;  ДодбавитьЗапись.Visible = false;  Захват.Visible=false;  Вызовы.Visible=false;  Надпись1.Visible = false;  Надпись2.Visible = false;  Надпись3.Visible = false;  Надпись4.Visible = false;  Надпись5.Visible = false;  Надпись6.Visible = false;  Надпись7.Visible = false;  } |
| Договоры\_Click | private void Договоры\_Click(object sender, EventArgs e)  {  ТаблицыКомандир.Size = new Size(755, 274);  c = 2;  Tables(c);  УдалитьЗапись.Visible = false;  ОбновитьТаблицу.Visible = false;  ДодбавитьЗапись.Visible = false;  Захват.Visible = false;  Вызовы.Visible = false;  Надпись1.Visible = false;  Надпись2.Visible = false;  Надпись3.Visible = false;  Надпись4.Visible = false;  Надпись5.Visible = false;  Надпись6.Visible = false;  Надпись7.Visible = false;  } |
| ВернутьсяКАвторизации\_Click | private void ВернутьсяКАвторизации\_Click(object sender, EventArgs e)  {  //переход на форму авторизация  Авторизация fd = new Авторизация();  fd.Show();  Hide();  } |
| ВыйтиИзПриложения\_Click | private void ВыйтиИзПриложения\_Click(object sender, EventArgs e)  {  //Выход из приложения  Application.Exit();  } |
| ГруппаЗахвата\_Click | private void ГруппаЗахвата\_Click(object sender, EventArgs e)  {  ТаблицыКомандир.Size = new Size(480, 274);  c = 4;  Tables(c);  УдалитьЗапись.Visible = true;  ОбновитьТаблицу.Visible = true;  ДодбавитьЗапись.Visible = true;  Надпись6.Visible = false;  Надпись7.Visible = false;  Надпись5.Visible = true;  Надпись4.Visible = true;  Надпись3.Visible = true;  Надпись2.Visible = true;  Надпись1.Visible = true;  Надпись1.Text = ТаблицыКомандир.Columns[0].HeaderText;  Надпись2.Text = ТаблицыКомандир.Columns[1].HeaderText;  Надпись3.Text = ТаблицыКомандир.Columns[2].HeaderText;  Надпись4.Text = ТаблицыКомандир.Columns[3].HeaderText;  Надпись5.Text = ТаблицыКомандир.Columns[4].HeaderText;  Вызовы.Visible = false;  Захват.Visible = true;  Захват.Location = new Point(621, 27);  } |
| Вызовы\_Click | private void Вызовы\_Click(object sender, EventArgs e)  {  ТаблицыКомандир.Size = new Size(480, 274);  c = 3;  Tables(c);  УдалитьЗапись.Visible = true;  ОбновитьТаблицу.Visible = true;  ДодбавитьЗапись.Visible = true;  Надпись6.Visible = true;  Надпись7.Visible = true;  Надпись5.Visible = true;  Надпись4.Visible = true;  Надпись3.Visible = true;  Надпись2.Visible = true;  Надпись1.Visible = true;  Надпись1.Text = ТаблицыКомандир.Columns[0].HeaderText;  Надпись2.Text = ТаблицыКомандир.Columns[1].HeaderText;  Надпись3.Text = ТаблицыКомандир.Columns[2].HeaderText;  Надпись4.Text = ТаблицыКомандир.Columns[3].HeaderText;  Надпись5.Text = ТаблицыКомандир.Columns[4].HeaderText;  Надпись6.Text = ТаблицыКомандир.Columns[5].HeaderText;  Надпись7.Text = ТаблицыКомандир.Columns[6].HeaderText;  Захват.Visible = false;  Вызовы.Visible = true;  } |
| Выйти\_Click | private void Выйти\_Click(object sender, EventArgs e)  {  //выход из придожения  Application.Exit();  } |
| ПерваяЗапись\_Click | private void ПерваяЗапись\_Click(object sender, EventArgs e)  {  //переход на первую строку таблицы  ТаблицыКомандир.CurrentCell = ТаблицыКомандир[0, 0];  } |
| ПредыдущаяЗапись\_Click | private void ПредыдущаяЗапись\_Click(object sender, EventArgs e)  {  //переход на предыдущую строку таблицы  i = ТаблицыКомандир.RowCount;  index = ТаблицыКомандир.CurrentRow.Index;  if (index <= 0)  {  ТаблицыКомандир.CurrentCell = ТаблицыКомандир[0, i - 1];  }  else  {  ТаблицыКомандир.Rows[index].Selected = true;  ТаблицыКомандир.CurrentCell = ТаблицыКомандир[0, index - 1];  }  } |
| СледующаяЗапись\_Click | private void СледующаяЗапись\_Click(object sender, EventArgs e)  {  //переход на следующую строку таблицы  i = ТаблицыКомандир.RowCount;  index = ТаблицыКомандир.CurrentRow.Index;  if (index >= (i - 1))  {  ТаблицыКомандир.CurrentCell = ТаблицыКомандир[0, 0];  }  else  {  ТаблицыКомандир.Rows[index].Selected = true;  ТаблицыКомандир.CurrentCell = ТаблицыКомандир[0, index + 1];  }  } |
| УдалитьЗапись\_Click | private void УдалитьЗапись\_Click(object sender, EventArgs e)  {  //удаление строки выделеной строки таблицы  for (int i = 0; i < ТаблицыКомандир.RowCount; i++)  {  for (int j = 0; j < ТаблицыКомандир.ColumnCount; j++)  {  if (ТаблицыКомандир.Rows[i].Cells[j].Selected == true)  {  ТаблицыКомандир.Rows.Remove(ТаблицыКомандир.Rows[i]);  switch (c)  {  case 4: captureTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture); break;  case 3: callingTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling); break;  }  }  }  }  } |
| ДобавитьЗапись\_Click | private void ДобавитьЗапись\_Click(object sender, EventArgs e)  {  //Добавление новой строки в таблицу  switch (c)  {  case 3:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Calling] VALUES ('{CallingID.Text}','{TreatyID.Text}','{ActionID.Text}','{ДатаВызова.Text}','{a}','{Штраф.Text}','{Компенсация.Text}');", sqlConnect);  da.Fill(dt);  CallingID.Text = "";  TreatyID.Text = "";  ActionID.Text = "";  Штраф.Text = "";  Компенсация.Text = "";  Вызов.Checked = false;  ДатаВызова.Text = "";  }  break;  case 4:  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  DataSet dt = new DataSet();  SqlDataAdapter da = new SqlDataAdapter();  da = new SqlDataAdapter($"INSERT INTO [Capture] VALUES ('{ПолеКодГруппыЗахвата.Text}','{НомерЭкипажа.Text}','{КомандирЭкипажа.Text}','{Автомобиль.Text}','{Документ.Text}');", sqlConnect);  da.Fill(dt);  ПолеКодГруппыЗахвата.Text = "";  НомерЭкипажа.Text = "";  КомандирЭкипажа.Text = "";  Автомобиль.Text = "";  Документ.Text = "";  }  break;  }  } |
| ОбновитьТаблицу\_Click | private void ОбновитьТаблицу\_Click(object sender, EventArgs e)  {  //обновлpictureBox7\_Clickение данных таблиц баз данных  switch (c)  {  case 4: captureTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture); break;  case 3: callingTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling); break;  }  } |
| Вызов\_CheckedChanged | private void Вызов\_CheckedChanged(object sender, EventArgs e)  {  if (Вызов.Checked == true)  {  a = 1;  }  else if (Вызов.Checked == false)  {  a = 0;  }  } |
| ПоследняяЗапись\_Click | private void ПоследняяЗапись\_Click(object sender, EventArgs e)  {  //переход на последнюю строку таблицы  ТаблицыКомандир.ClearSelection(); //снять выделение всех выбранных ячеек  index = ТаблицыКомандир.Rows.Count - 1; // индекс последней строки  ТаблицыКомандир.Rows[index].Selected = true; // выделить нужную строку  ТаблицыКомандир.FirstDisplayedScrollingRowIndex = index; // фокус в нужную строку  } |
| CallingID\_KeyPress | private void CallingID\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  TreatyID.Focus();  }  } |
| TreatyID\_KeyPress | private void TreatyID\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ActionID.Focus();  }  } |
| ActionID\_KeyPress | private void ActionID\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ДатаВызова.Focus();  }  } |
| Штраф\_KeyPress | private void Штраф\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  Компенсация.Focus();  }  } |
| ПолеКодГруппыЗахвата\_KeyPress | private void ПолеКодГруппыЗахвата\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  НомерЭкипажа.Focus();  }  } |
| НомерЭкипажа\_KeyPress | private void НомерЭкипажа\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  КомандирЭкипажа.Focus();  }  } |
| КомандирЭкипажа\_KeyPress | private void КомандирЭкипажа\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  Автомобиль.Focus();  }  } |
| Автомобиль\_KeyPress | private void Автомобиль\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  Документ.Focus();  }  } |
| ПолеПоиск\_TextChanged | private void ПолеПоиск\_TextChanged(object sender, EventArgs e)  {  //поиск в таблице  for (int i = 0; i < ТаблицыКомандир.RowCount; i++)  {  ТаблицыКомандир.Rows[i].Selected = false;  for (int j = 0; j < ТаблицыКомандир.ColumnCount; j++)  {  if (ТаблицыКомандир.Rows[i].Cells[j].Value != null)  {  if (ТаблицыКомандир.Rows[i].Cells[j].Value.ToString().Contains(ПолеПоиск.Text))  {  ТаблицыКомандир.Rows[i].Selected = true;  break;  }  }  }  }  } |
| Командир\_MouseDown | private void Командир\_MouseDown(object sender, MouseEventArgs e)  {  //отпеделяет точку где была эажата клавиша мыши на форме  mouse = new Point(e.X, e.Y);  } |
| Командир\_MouseMove | private void Командир\_MouseMove(object sender, MouseEventArgs e)  {  //передвигает форму по тому как движется мышка по экрану  if (e.Button == MouseButtons.Left)  {  Left += e.X - mouse.X;  Top += e.Y - mouse.Y;  }  } |

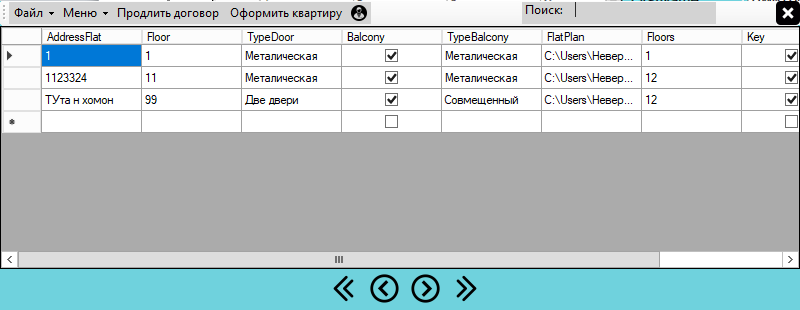


Рисунок 25 – Форма «Конечный пользователь»

Свойства элементов формы «Конечный пользователь» представлены в таблице 19.

Таблица 19 – Свойства элементов формы «Конечный пользователь».

| Объект | Свойство | Значение |
| --- | --- | --- |
| КП\_Form | FormBorderStyle | None |
| BackColor | 111; 210; 221 |
| Text | «Форма конечного пользователя» |
| Size | 1067; 380 |
| StartPosition | CenterScreen |
| ТаблицыКП\_DataGridView1 | Size | 1067; 299 |
| Поиск\_Label1 | Text | «Поиск:» |
| Font | Segoe UI; 9pt |
| ПолеПоиск\_TextBox1 | MaxLength | 32767 |
| ShortcutsEnabled | false |
| Font | Segoe UI; 9pt |
| ПерваяЗапись\_PictureBox1 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| Выйти\_PictureBox2 | BackColor | White |
| SizeMode | StretchImage |
| ПредыдущаяЗапись\_PictureBox3 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| СледующаяЗапись\_PictureBox4 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПоследняяЗапись\_PictureBox5 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПанельИнструментов\_ToolStrip1 | Items | Профиль\_ToolStripButton1 |
| ОформитьКвартиру\_ToolStripButton2 |
| ПродлитьДоговор\_ToolStripButton4 |
| Меню\_ToolStripDropDownButton1 |
| Файл\_ToolStripDropDownButton2 |

Событийно-управляемые процедуры в форме «Конечный пользователь» представлены в таблице 20.

Таблица 20 – Событийно-управляемые процедуры в форме «Конечный пользователь»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class КП | public partial class КП : Form  {  public КП()  {  InitializeComponent();  }  int i, index;  string regist;  Point mouse; |
| registr | public void Registr (string reg)  {  regist = reg;  } |
| ОформитьКвартиру\_Click | private void ОформитьКвартиру\_Click(object sender, EventArgs e)  {  //преход на форму регистрация квартир  Регистрация\_квартир hr = new Регистрация\_квартир();  hr.Show();  hr.Referral(regist);  } |
| tables | private void Tables(int c)  {  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter[] sa = new SqlDataAdapter[5];  sa[1] = new SqlDataAdapter($"SELECT Client.Registr, [AddressFlat], [PatrolID], [Chief], [Brand], [Document], [DateTime], [False],[Tax], [Compensation] FROM Calling INNER JOIN Contract ON Calling.TreatyID = Contract.TreatyID INNER JOIN Settlement ON Contract.SettlementID = Settlement.SettlementID INNER JOIN Client ON Settlement.Registr = Client.Registr INNER JOIN Flat ON Settlement.FlatID = Flat.FlatID INNER JOIN Capture ON Calling.ActionID = Capture.ActionID WHERE (Client.Registr = '{regist}') ", sqlConnect);  sa[2] = new SqlDataAdapter("SELECT [PatrolID], [Chief], [Brand], [Document] FROM [Capture]", sqlConnect);  sa[3] = new SqlDataAdapter($"SELECT Client.Registr, [AddressFlat], [DateStart], [StopDate], [Cost] FROM Flat INNER JOIN Settlement ON Flat.FlatID = Settlement.FlatID INNER JOIN Client ON Settlement.Registr = Client.Registr INNER JOIN Contract ON Settlement.SettlementID = Contract.SettlementID WHERE (Client.Registr = '{regist}') ", sqlConnect);  sa[4] = new SqlDataAdapter($"SELECT Client.Registr, [AddressFlat], [Floor], [TypeDoor], [Balcony], [TypeBalcony], [FlatPlan], [Floors], [Key],[TypeHouse] FROM [Flat] INNER JOIN [Flat-House] ON Flat.FlatID = [Flat-House].FlatID INNER JOIN [House] ON [Flat-House].HouseID = House.HouseID INNER JOIN [Settlement] ON Flat.FlatID = Settlement.FlatID INNER JOIN [Client] ON [Settlement].Registr = [Client].Registr WHERE (Client.Registr = '{regist}') ", sqlConnect);  DataSet ds = new DataSet();  sa[c].Fill(ds);  ТаблицыКП.DataSource = ds.Tables[0];  } |
| Выйти\_Click | private void Выйти\_Click(object sender, EventArgs e)  {  //выход из приложения  Application.Exit();  } |
| вернутьсяКАвторизации\_Click | private void вернутьсяКАвторизации\_Click(object sender, EventArgs e)  {  //переход на форму авторизации  Авторизация av = new Авторизация();  av.Show();  Hide();  } |
| выйтиИзПриложения\_Click | private void выйтиИзПриложения\_Click(object sender, EventArgs e)  {  //выход из приложения  Application.Exit();  } |
| Профиль\_Click | private void Профиль\_Click(object sender, EventArgs e)  {  //переход на форму профиль  Профиль gh = new Профиль();  gh.Show();  gh.Index(regist);  } |
| ПредыдущаяЗапись\_Click | private void ПредыдущаяЗапись\_Click(object sender, EventArgs e)  {  //переход на предыдущую строку таблицы  i = ТаблицыКП.RowCount;  index = ТаблицыКП.CurrentRow.Index;  if (index <= 0)  {  ТаблицыКП.CurrentCell = ТаблицыКП[0, i - 1];  }  else  {  ТаблицыКП.Rows[index].Selected = true;  ТаблицыКП.CurrentCell = ТаблицыКП[0, index - 1];  }  } |
| ПерваяЗапись\_Click | private void ПерваяЗапись\_Click(object sender, EventArgs e)  {  //переход на первую строку таблицы  ТаблицыКП.CurrentCell = ТаблицыКП[0,0];  } |
| ПоследняяЗапись\_Click | private void ПоследняяЗапись\_Click(object sender, EventArgs e)  {  //переход на последную строку таблицы  ТаблицыКП.ClearSelection(); //снять выделение всех выбранных ячеек  index = ТаблицыКП.Rows.Count - 1; // индекс последней строки  ТаблицыКП.Rows[index].Selected = true; // выделить нужную строку  ТаблицыКП.CurrentCell = ТаблицыКП[0, index];  } |
| КП\_Load | private void КП\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Client". При необходимости она может быть перемещена или удалена.  this.clientTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Client);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Prolonging". При необходимости она может быть перемещена или удалена.  this.prolongingTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Prolonging);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.House". При необходимости она может быть перемещена или удалена.  this.houseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House". При необходимости она может быть перемещена или удалена.  this.flat\_HouseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat". При необходимости она может быть перемещена или удалена.  this.flatTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract". При необходимости она может быть перемещена или удалена.  this.contractTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture". При необходимости она может быть перемещена или удалена.  this.captureTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Capture);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling". При необходимости она может быть перемещена или удалена.  this.callingTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Calling);  ТаблицыКП.ReadOnly = true;  } |
| вызовы\_Click | private void вызовы\_Click(object sender, EventArgs e)  {  tables(1);  ТаблицыКП.ReadOnly = true;  } |
| группаЗахвата\_Click | private void группаЗахвата\_Click(object sender, EventArgs e)  {  tables(2);  ТаблицыКП.ReadOnly = true;  } |
| ПолеПоиск\_TextChanged | private void ПолеПоиск\_TextChanged(object sender, EventArgs e)  {  //Поиск  for (int i = 0; i < ТаблицыКП.RowCount; i++)  {  ТаблицыКП.Rows[i].Selected = false;  for (int j = 0; j < ТаблицыКП.ColumnCount; j++)  {  if (ТаблицыКП.Rows[i].Cells[j].Value != null)  {  if (ТаблицыКП.Rows[i].Cells[j].Value.ToString().Contains(ПолеПоиск.Text))  {  ТаблицыКП.Rows[i].Selected = true;  break;  }  }  }  }  } |
| договор\_Click | private void договор\_Click(object sender, EventArgs e)  {  tables(3);  ТаблицыКП.ReadOnly = true;  } |
| квартиры\_Click | private void квартиры\_Click(object sender, EventArgs e)  {  tables(4);  ТаблицыКП.ReadOnly = true;  } |
| КП\_MouseMove | private void КП\_MouseMove(object sender, MouseEventArgs e)  {  //передвигает форму по тому как движется мышка по экрану  if (e.Button == MouseButtons.Left)  {  Left += e.X - mouse.X;  Top+= e.Y - mouse.Y;  }  } |
| КП\_MouseDown | private void КП\_MouseDown(object sender, MouseEventArgs e)  {  //отпеделяет точку где была эажата клавиша мыши на форме  mouse = new Point(e.X, e.Y);  } |
| СледующаяЗапись\_Click | private void СледующаяЗапись\_Click(object sender, EventArgs e)  {  //следующая строка  i = ТаблицыКП.RowCount;  index = ТаблицыКП.CurrentRow.Index;  if (index >= (i - 1))  {  ТаблицыКП.CurrentCell = ТаблицыКП[0, 0];  }  else  {  ТаблицыКП.Rows[index].Selected = true;  ТаблицыКП.CurrentCell = ТаблицыКП[0, index + 1];  }  } |
| ПродлитьДоговор\_Click | private void ПродлитьДоговор\_Click(object sender, EventArgs e)  {  //переход на форму Продление  Продление ghj = new Продление();  ghj.Show();  ghj.Id(regist);  } |

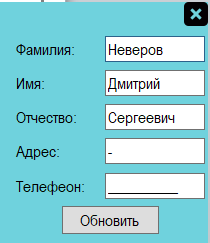


Рисунок 26 – Форма «Профиль»

Свойства элементов формы «Профиль» представлены в таблице 21.

Таблица 21 – Свойства элементов формы «Профиль».

| Объект | Свойство | Значение |
| --- | --- | --- |
| Профиль\_Form | FormBorderStyle | None |
| BackColor | 111; 210; 221 |
| Text | «Профиль» |
| Size | 208, 242 |
| StartPosition | CenterScreen |
| ТаблицаКлиент\_DataGridView1 | Visible | false |
| Фамилия\_Label1 | Text | «Фамилия:» |
| Font | Microsoft Sans Serif; 12pt |
| Имя\_Label2 | Text | «Имя:» |
| Font | Microsoft Sans Serif; 12pt |
| Отчество\_Label3 | Text | «Отчество:» |
| Font | Microsoft Sans Serif; 12pt |
| Адрес\_Label4 | Text | «Адрес:» |
| Font | Microsoft Sans Serif; 12pt |
| Телефон\_Label5 | Text | «Телефон:» |
| Font | Microsoft Sans Serif; 12pt |
| ПолеФамилия TextBox1 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеИмя TextBox2 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 60 |
| ПолеОтчество TextBox3 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеАдрес\_TextBox4 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| ПолеТелефон\_MaskedTextBox1 | Mask | 9990000000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| Выйти\_PictureBox2 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| Обновить\_Button1 | Text | «Обновить» |
| Font | Arial Narrow; 12pt |
| Size | 132; 37 |

Событийно-управляемые процедуры в форме «Профиль» представлены в таблице 22.

Таблица 22 – Событийно-управляемые процедуры в форме «Профиль»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Профиль | public partial class Профиль : Form  {  public Профиль()  {  InitializeComponent();  }  string reg;  Point mouse; |
| Метод Index | public void Index(string i)  {  reg = i;  for (int j = 0; j < ТаблицаКлиент.RowCount; j++)  {  if (ТаблицаКлиент.Rows[j].Cells[0].Value.ToString().Contains(i))  {  ТаблицаКлиент.Rows[j].Selected = true;  ТаблицаКлиент.CurrentCell = ТаблицаКлиент[0, j];  break;  }  }  } |
| Выйти\_Click | private void Выйти\_Click(object sender, EventArgs e)  {  //закрытие формы  this.Close();  } |
| Обновить\_Click | private void Обновить\_Click(object sender, EventArgs e)  {  ТаблицаКлиент.CurrentCell = ТаблицаКлиент[0, 0];  Index(reg);  clientTableAdapter.Update(уП\_ПМ\_01\_Неверов\_ДСDataSet.Client);  MessageBox.Show("Ваши данные были обновлены");  } |
| Профиль\_Load | private void Профиль\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Client". При необходимости она может быть перемещена или удалена.  this.clientTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Client);  ToolTip tt = new ToolTip();//Данные строки позволяют показывать сплывающую подсказку  tt.SetToolTip(Обновить, "Нажмите, чтобы подтвердить изменение");  this.Size = new Size(208, 242);  } |
| Профиль\_MouseDown | private void Профиль\_MouseDown(object sender, MouseEventArgs e)  {  //отпеделяет точку где была эажата клавиша мыши на форме  mouse = new Point(e.X, e.Y);  } |
| Профиль\_MouseMove | private void Профиль\_MouseMove(object sender, MouseEventArgs e)  {  //передвигает форму по тому как движется мышка по экрану  if (e.Button == MouseButtons.Left)  {  Left += e.X - mouse.X;  Top += e.Y - mouse.Y;  }  } |

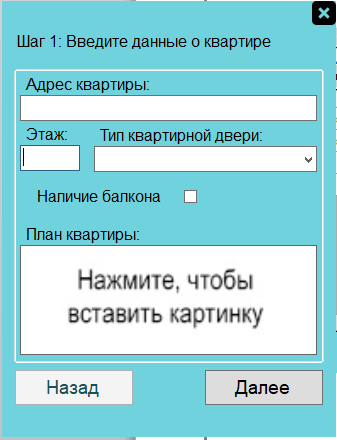


Рисунок 27 – Форма «Регистрация квартир»

Свойства элементов формы «Регистрация квартир» представлены в таблице 23.

Таблица 23 – Свойства элементов формы «Регистрация квартир».

| Объект | Свойство | Значение |
| --- | --- | --- |
| Регистрация\_квартир\_Form | FormBorderStyle | None |
| BackColor | 111; 210; 221 |
| Text | «Регистрация\_квартир» |
| Size | 334; 436 |
| StartPosition | CenterScreen |
| ТаблицаFlatHouse\_DataGridView1 | Visible | false |
| ТаблицаКвартира\_DataGridView2 | Visible | false |
| ТаблицаSettlement\_DataGridView3 | Visible | false |
| ТаблицаДоговор\_DataGridView4 | Visible | false |
| ТаблицаДом\_DataGridView5 | Visible | false |
| Квартир\_GroupBox1 | Size | 413; 369 |
| Text | «» |
| Дом\_GroupBox2 | Size | 413; 369 |
| Text | «» |
| Договор\_GroupBox3 | Size | 413; 369 |
| Text | «» |
| НаличиеБалкона\_CheckBox1 | Text | «Наличие балкона» |
| Font | Arial; 12pt |
| RightToLeft | Yes |
| КодовыйЗамок\_CheckBox2 | Text | «Наличие кодового замка на подъезде» |
| Font | Arial; 12pt |
| RightToLeft | Yes |
| ПолеТипБалкона\_ComboBox1 | Items | «Отдельный», «Совмещенный» |
| Font | Arial; 12pt |
| ПолеТипКвартирнойДвери\_  ComboBox2 | Items | «Металическая», «Деревянная», «Две двери» |
| Font | Arial; 12pt |
| ПолеТипДома\_ComboBox3 | Items | «Кирпичный», «Панельный» |
| Font | Arial; 12pt |
| АдресКвартиры\_Label1 | Text | «Адрес квартиры:» |
| Font | Arial; 12pt |
| ЭтажКвартиры\_Label2 | Text | «Этаж:» |
| Font | Arial; 12pt |
| ТипДвери\_Label3 | Text | «Тип квартирной двери:» |
| Font | Arial; 12pt |
| ТипБалкона\_Label4 | Text | «Тип балкона:» |
| Font | Arial; 12pt |
| ПланКвартиры\_Label5 | Text | «План квартиры:» |
| Font | Arial; 12pt |
| ТипДома\_Label6 | Text | «Тип дома:» |
| Font | Arial; 12pt |
| ЭтажейВДоме\_Label7 | Text | «Этажей в доме:» |
| Font | Arial; 12pt |
| ДатаНачало\_Label8 | Text | «Дата начало действия договора:» |
| Font | Arial; 12pt |
| ДатаОкончания\_Label9 | Text | «Дата окончание действия договора:» |
| Font | Arial; 12pt |
| Плата\_Label10 | Text | «Стоимость ежемесячная плата:» |
| Font | Arial; 12pt |
| Надпись1\_Label11 | Text | «» |
| Font | Arial; 12pt |
| Надпись2\_Label12 | Text | «Телефон:» |
| Font | Microsoft Sans Serif; 7,8pt |
| ПолеАдресКвартиры\_TextBox1 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Arial; 12pt |
| ПолеЭтажКвартиры\_TextBox2 | ShortcutsEnabled | false |
| Font | Arial; 12pt |
| MaxLength | 2 |
| ПолеЭтажейВДоме\_TextBox3 | MaxLength | 2 |
| ShortcutsEnabled | false |
| Font | Arial; 12pt |
| ПолеПлата\_TextBox4 | MaxLength | 15 |
| ShortcutsEnabled | false |
| Font | Arial; 12pt |
| ПолеДатаНачалаДействия\_  MaskedTextBox1 | Mask | 00/00/0000 |
| ShortcutsEnabled | false |
| Font | Arial; 12pt |
| ПолеДатаОкончаниеДействия\_  MaskedTextBox2 | Mask | 00/00/0000 |
| ShortcutsEnabled | false |
| Font | Arial; 12pt |
| Выйти\_PictureBox1 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| ПолеПланКвартиры\_PictureBox2 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| Назад\_Button1 | Text | «Назад» |
| Font | Arial Narrow; 12pt |
| Size | 160; 46 |
| Далее\_Button2 | Text | «Далее» |
| Font | Arial Narrow; 12pt |
| Size | 160; 46 |

Событийно-управляемые процедуры в форме «Регистрация квартир» представлены в таблице 24.

Таблица 24 – Событийно-управляемые процедуры в форме «Регистрация квартир»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Регистрация\_квартир | public partial class Регистрация\_квартир : Form  {  public Регистрация\_квартир()  {  InitializeComponent();  }  Bitmap bmp;  int check, check1;string iD;  private int a = 1, n, m, l,j,b;  Point mouse; |
| НаличиеБалкона\_CheckedChanged | private void НаличиеБалкона\_CheckedChanged(object sender, EventArgs e)  {  if (НаличиеБалкона.Checked == true)  {  check = 1;  ТипБалкона.Visible = true;  ПолеТипБалкона.Visible = true;  }  else if (НаличиеБалкона.Checked == false)  {  check = 0;  ТипБалкона.Visible = false;  ПолеТипБалкона.Visible = false;  }  } |
| Метод Referral | public void Referral(string id)  {  iD = id;  } |
| ПолеПланКвартиры\_Click | private void ПолеПланКвартиры\_Click(object sender, EventArgs e)  {  //Вставить картинку из проводника  ВставкаКартинки.Filter = "Image files (\*.BMP, \*.JPG, " + "\*.GIF, \*.PNG)|\*.bmp;\*.jpg;\*.gif;\*.png";  if (ВставкаКартинки.ShowDialog() == DialogResult.OK)  {  Image image = Image.FromFile(ВставкаКартинки.FileName);  int width = ПолеПланКвартиры.Width;  int height = ПолеПланКвартиры.Height;  bmp = new Bitmap(image, width, height);  ПолеПланКвартиры.Image = bmp;  Надпись2.Text = ВставкаКартинки.FileName;  }  } |
| Регистрация\_квартир\_Load | private void Регистрация\_квартир\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House". При необходимости она может быть перемещена или удалена.  this.flat\_HouseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract". При необходимости она может быть перемещена или удалена.  this.contractTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement". При необходимости она может быть перемещена или удалена.  this.settlementTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.House". При необходимости она может быть перемещена или удалена.  this.houseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat". При необходимости она может быть перемещена или удалена.  this.flatTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat);  Надпись1.Text = "Шаг 1: Введите данные о квартире";  ToolTip tt = new ToolTip();  tt.SetToolTip(Выйти, "Закрыть");  this.Size = new Size(334, 436);  a = 1;  Page(a);  НаличиеБалкона.Checked = false;  ТипБалкона.Visible = false;  ПолеТипБалкона.Visible = false;  for (int i = 0; i < ТаблицаКвартира.RowCount; i++)  {  n = int.Parse(ТаблицаКвартира[0, i].Value.ToString());  }  for (int i = 0; i < ТаблицаДом.RowCount; i++)  {  m = int.Parse(ТаблицаДом[0, i].Value.ToString());  }  for (int i = 0; i < ТаблицаSettlement.RowCount; i++)  {  l = int.Parse(ТаблицаSettlement[0, i].Value.ToString());  }  for (int i = 0; i < ТаблицаДоговор.RowCount; i++)  {  j = int.Parse(ТаблицаДоговор[0, i].Value.ToString());  }  for (int i = 0; i < ТаблицаFlatHouse.RowCount; i++)  {  b = int.Parse(ТаблицаFlatHouse[0, i].Value.ToString());  }  } |
| Выйти\_Click | private void Выйти\_Click(object sender, EventArgs e)  {  //Закрыть форму  this.Close();  } |
| Метод Page | private void Page(int a)  {  if (a == 1)  {  Назад.Enabled = false;  Квартира.Visible = true;  Дом.Visible = false;  Договор.Visible = false;  Квартира.Location = new Point(12, 62);  Надпись1.Text = "Шаг 1: Введите данные о квартире";  Далее.Text = "Далее";  }  if (a == 2)  {  Назад.Enabled = true;  Квартира.Visible = false;  Дом.Visible = true;  Договор.Visible = false;  Дом.Location = new Point(12, 62);  Надпись1.Text = "Шаг 2: Введите данные о доме, \nв которой находится квартира";  Далее.Text = "Далее";  }  if (a == 3)  {  Назад.Enabled = true;  Квартира.Visible = false;  Дом.Visible = false;  Договор.Visible= true;  Договор.Location = new Point(12, 62);  Надпись1.Text = "Шаг 3: Введите время договора";  Далее.Text = "Оформить";  }  } |
| Далее\_Click | private void Далее\_Click(object sender, EventArgs e)  {  if (Далее.Text == "Оформить")  {  if (ПолеАдресКвартиры.Text == "" || ПолеЭтажКвартиры.Text == "" || ПолеТипКвартирнойДвери.Text == "" || ПолеПланКвартиры.ImageLocation == "" || ПолеТипДома.Text == "" || ПолеЭтажейВДоме.Text == "" || ПолеДатаОкончаниеДействия.Text == "" || ПолеДатаНачалаДействия.Text == "")  {  }  else  {  n = n+1;  m = m+1;  l = l+1;  j = j+1;  b = ТаблицаFlatHouse.RowCount;  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  try  {  SqlDataAdapter info1 = new SqlDataAdapter($"INSERT INTO [Flat] (FlatID,AddressFlat,Floor,TypeDoor,Balcony,TypeBalcony,FlatPlan)VALUES('{n}','{ПолеАдресКвартиры.Text}','{ПолеЭтажКвартиры.Text}','{ПолеТипКвартирнойДвери.Text}','{check}','{ПолеТипБалкона.Text}','{Надпись2.Text}');", sqlConnect);  SqlDataAdapter info2 = new SqlDataAdapter($"INSERT INTO [House] (HouseID,Floors,[Key],TypeHouse)VALUES('{m}','{ПолеЭтажейВДоме.Text}','{check1}','{ПолеТипДома.Text}');", sqlConnect);  SqlDataAdapter info3 = new SqlDataAdapter($"INSERT INTO [Settlement] (SettlementID,FlatID,Registr)VALUES('{l}','{n}','{iD}');", sqlConnect);  SqlDataAdapter info4 = new SqlDataAdapter($"INSERT INTO [Flat-House] ([Flat-HouseID],FlatID,HouseID)VALUES('{b}','{n}','{m}');", sqlConnect);  SqlDataAdapter info5 = new SqlDataAdapter($"INSERT INTO [Contract] (TreatyID,SettlementID,DateStart,StopDate,Cost)VALUES('{j}','{l}','{ПолеДатаНачалаДействия.Text}','{ПолеДатаОкончаниеДействия.Text}','{30000}');", sqlConnect);  DataTable dt1 = new DataTable();  DataTable dt2 = new DataTable();  DataTable dt3 = new DataTable();  DataTable dt4 = new DataTable();  DataTable dt5 = new DataTable();  info1.Fill(dt1);  info2.Fill(dt2);  info3.Fill(dt3);  info4.Fill(dt4);  info5.Fill(dt5);  }  catch { }  }  }  MessageBox.Show("Квартира была оформленна");  }  else  {  a++;  Page(a);  }  } |
| Назад\_Click | private void Назад\_Click(object sender, EventArgs e)  {  a--;  Page(a);  if (a < 1)  {  a = 1;  }  } |
| ПолеЭтажейВДоме\_KeyPress | private void ПолеЭтажейВДоме\_KeyPress(object sender, KeyPressEventArgs e)  {  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеПлата\_KeyPress | private void ПолеПлата\_KeyPress(object sender, KeyPressEventArgs e)  {  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| Регистрация\_квартир\_MouseMove | private void Регистрация\_квартир\_MouseMove(object sender, MouseEventArgs e)  {  //передвигает форму по тому как движется мышка по экрану  if (e.Button == MouseButtons.Left)  {  Left += e.X - mouse.X;  Top += e.Y - mouse.Y;  }  } |
| ПолеДатаНачалаДействия\_KeyPress | private void ПолеДатаНачалаДействия\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеДатаОкончаниеДействия.Focus();  }  } |
| ПолеЭтажКвартиры\_KeyPress | private void ПолеЭтажКвартиры\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеТипКвартирнойДвери.Focus();  }  if ((e.KeyChar >= '0') && (e.KeyChar <= '9') || (e.KeyChar == (char)Keys.Back))  {  }  else  {  e.KeyChar -= e.KeyChar;  }  } |
| ПолеАдресКвартиры\_KeyPress | private void ПолеАдресКвартиры\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеЭтажКвартиры.Focus();  }  } |
| Регистрация\_квартир\_MouseDown | private void Регистрация\_квартир\_MouseDown(object sender, MouseEventArgs e)  {  //отпеделяет точку где была эажата клавиша мыши на форме  mouse = new Point(e.X, e.Y);  } |
| КодовыйЗамок\_CheckedChanged | private void КодовыйЗамок\_CheckedChanged(object sender, EventArgs e)  {  if (НаличиеБалкона.Checked == true)  {  check1 = 1;  }  else if (НаличиеБалкона.Checked == false)  {  check1 = 0;  }  } |

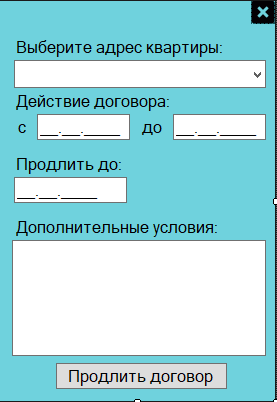


Рисунок 28 – Форма «Продление»

Свойства элементов формы «Авторизация» представлены в таблице 25.

Таблица 25 – Свойства элементов формы «Регистрация квартир».

| Объект | Свойство | Значение |
| --- | --- | --- |
| Профиль | FormBorderStyle | None |
| BackColor | 111; 210; 221 |
| Text | «Профиль» |
| Size | 208, 242 |
| StartPosition | CenterScreen |
| DataGridView1 | Visible | false |
| Label1 | Text | «Фамилия:» |
| Font | Microsoft Sans Serif; 12pt |
| Label2 | Text | «Имя:» |
| Font | Microsoft Sans Serif; 12pt |
| Label3 | Text | «Отчество:» |
| Font | Microsoft Sans Serif; 12pt |
| Label4 | Text | «Адрес:» |
| Font | Microsoft Sans Serif; 12pt |
| Label5 | Text | «Телефон:» |
| Font | Microsoft Sans Serif; 12pt |
| TextBox1 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| TextBox2 | ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaxLength | 60 |
| TextBox3 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| TextBox4 | MaxLength | 60 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| MaskedTextBox1 | Mask | 9990000000 |
| ShortcutsEnabled | false |
| Font | Microsoft Sans Serif; 12pt |
| PictureBox2 | BackColor | 111; 210; 221 |
| SizeMode | StretchImage |
| Button1 | Text | «Обновить» |
| Font | Arial Narrow; 12pt |
| Size | 132; 37 |

Событийно-управляемые процедуры в форме авторизации представлены в таблице 26.

Таблица 26 – Событийно-управляемые процедуры в форме «Регистрация квартир»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Регистрация\_квартир | public partial class Регистрация\_квартир : Form  {  public Регистрация\_квартир()  {  InitializeComponent();  }  Bitmap bmp;  int check, check1;string ID;  private int a = 1, n, m, i,j,b;  Point Mouse; |
| НаличиеБалкона\_CheckedChanged | private void НаличиеБалкона\_CheckedChanged(object sender, EventArgs e)  {  if (НаличиеБалкона.Checked == true)  {  check = 1;  ТипБалкона.Visible = true;  ПолеТипБалкона.Visible = true;  }  else if (НаличиеБалкона.Checked == false)  {  check = 0;  ТипБалкона.Visible = false;  ПолеТипБалкона.Visible = false;  }  } |
| Метод r | public void r(string id)  {  ID = id;  } |
| ПолеПланКвартиры\_Click | private void ПолеПланКвартиры\_Click(object sender, EventArgs e)  {  //Вставить картинку из проводника  ВставкаКартинки.Filter = "Image files (\*.BMP, \*.JPG, " + "\*.GIF, \*.PNG)|\*.bmp;\*.jpg;\*.gif;\*.png";  if (ВставкаКартинки.ShowDialog() == DialogResult.OK)  {  Image image = Image.FromFile(ВставкаКартинки.FileName);  int width = ПолеПланКвартиры.Width;  int height = ПолеПланКвартиры.Height;  bmp = new Bitmap(image, width, height);  ПолеПланКвартиры.Image = bmp;  Надпись2.Text = ВставкаКартинки.FileName;  }  } |
| Регистрация\_квартир\_Load | private void Регистрация\_квартир\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House". При необходимости она может быть перемещена или удалена.  this.flat\_HouseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.\_Flat\_House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract". При необходимости она может быть перемещена или удалена.  this.contractTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Contract);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement". При необходимости она может быть перемещена или удалена.  this.settlementTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Settlement);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.House". При необходимости она может быть перемещена или удалена.  this.houseTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.House);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat". При необходимости она может быть перемещена или удалена.  this.flatTableAdapter.Fill(this.уП\_ПМ\_01\_Неверов\_ДСDataSet.Flat);  Надпись1.Text = "Шаг 1: Введите данные о квартире"+ID;  ToolTip tt = new ToolTip();  tt.SetToolTip(Выйти, "Закрыть");  this.Size = new Size(334, 436);  a = 1;  page(a);  НаличиеБалкона.Checked = false;  ТипБалкона.Visible = false;  ПолеТипБалкона.Visible = false;  } |
| Выйти\_Click | private void Выйти\_Click(object sender, EventArgs e)  {  //Закрыть форму  this.Close();  } |
| Метод page | private void page(int a)  {  if (a == 1)  {  Назад.Enabled = false;  Квартира.Visible = true;  Дом.Visible = false;  Договор.Visible = false;  Квартира.Location = new Point(12, 62);  Надпись1.Text = "Шаг 1: Введите данные о квартире";  Далее.Text = "Далее";  }  if (a == 2)  {  Назад.Enabled = true;  Квартира.Visible = false;  Дом.Visible = true;  Договор.Visible = false;  Дом.Location = new Point(12, 62);  Надпись1.Text = "Шаг 2: Введите данные о доме, \nв которой находится квартира";  Далее.Text = "Далее";  }  if (a == 3)  {  Назад.Enabled = true;  Квартира.Visible = false;  Дом.Visible = false;  Договор.Visible= true;  Договор.Location = new Point(12, 62);  Надпись1.Text = "Шаг 3: Введите время договора";  Далее.Text = "Оформить";  }  } |
| Далее\_Click | private void Далее\_Click(object sender, EventArgs e)  {  if (Далее.Text == "Оформить")  {  if (ПолеАдресКвартиры.Text == "" || ПолеЭтажКвартиры.Text == "" || ПолеТипКвартирнойДвери.Text == "" || ПолеПланКвартиры.ImageLocation == "" || ПолеТипДома.Text == "" || ПолеЭтажейВДоме.Text == "" || ПолеДатаОкончаниеДействия.Text == "" || ПолеДатаНачалаДействия.Text == "")  {  }  else  {  n = ТаблицаКвартира.RowCount;  m = ТаблицаДом.RowCount;  i = ТаблицаSettlement.RowCount;  j = ТаблицаДоговор.RowCount;  b = ТаблицаFlatHouse.RowCount;  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = УП\_ПМ\_01\_Неверов\_ДС; Integrated Security = True"))  {  try  {  SqlDataAdapter info1 = new SqlDataAdapter($"INSERT INTO [Flat] (FlatID,AddressFlat,Floor,TypeDoor,Balcony,TypeBalcony,FlatPlan)VALUES('{n}','{ПолеАдресКвартиры.Text}','{ПолеЭтажКвартиры.Text}','{ПолеТипКвартирнойДвери.Text}','{check}','{ПолеТипБалкона.Text}','{Надпись2.Text}');", sqlConnect);  SqlDataAdapter info2 = new SqlDataAdapter($"INSERT INTO [House] (HouseID,Floors,[Key],TypeHouse)VALUES('{m}','{ПолеЭтажейВДоме.Text}','{check1}','{ПолеТипДома.Text}');", sqlConnect);  SqlDataAdapter info3 = new SqlDataAdapter($"INSERT INTO [Settlement] (SettlementID,FlatID,Registr)VALUES('{i}','{n}','{ID}');", sqlConnect);  SqlDataAdapter info4 = new SqlDataAdapter($"INSERT INTO [Flat-House] ([Flat-HouseID],FlatID,HouseID)VALUES('{b}','{n}','{m}');", sqlConnect);  SqlDataAdapter info5 = new SqlDataAdapter($"INSERT INTO [Contract] (TreatyID,SettlementID,DateStart,StopDate,Cost)VALUES('{j}','{i}','{ПолеДатаНачалаДействия.Text}','{ПолеДатаОкончаниеДействия.Text}','{30000}');", sqlConnect);  DataTable dt1 = new DataTable();  DataTable dt2 = new DataTable();  DataTable dt3 = new DataTable();  DataTable dt4 = new DataTable();  DataTable dt5 = new DataTable();  info1.Fill(dt1);  info2.Fill(dt2);  info3.Fill(dt3);  info4.Fill(dt4);  info5.Fill(dt5);  }  catch { }  }  }  MessageBox.Show("Квартира была оформленна");  }  else  {  a++;  page(a);  }  } |
| Назад\_Click | private void Назад\_Click(object sender, EventArgs e)  {  a--;  page(a);  if (a < 1)  {  a = 1;  }  } |
| Регистрация\_квартир\_MouseMove | private void Регистрация\_квартир\_MouseMove(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Left)  {  Left += e.X - Mouse.X;  Top += e.Y - Mouse.Y;  }  } |
| ПолеДатаНачалаДействия\_KeyPress | private void ПолеДатаНачалаДействия\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеДатаОкончаниеДействия.Focus();  }  } |
| ПолеЭтажКвартиры\_KeyPress | private void ПолеЭтажКвартиры\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеТипКвартирнойДвери.Focus();  }  } |
| ПолеАдресКвартиры\_KeyPress | private void ПолеАдресКвартиры\_KeyPress(object sender, KeyPressEventArgs e)  {  if (e.KeyChar == (char)Keys.Enter)  {  ПолеЭтажКвартиры.Focus();  }  } |
| Регистрация\_квартир\_MouseDown | private void Регистрация\_квартир\_MouseDown(object sender, MouseEventArgs e)  {  Mouse = new Point(e.X, e.Y);  } |
| КодовыйЗамок\_CheckedChanged | private void КодовыйЗамок\_CheckedChanged(object sender, EventArgs e)  {  if (НаличиеБалкона.Checked == true)  {  check1 = 1;  }  else if (НаличиеБалкона.Checked == false)  {  check1 = 0;  }  } |

Вывод по главе: была спроектирована и разработана система базы данных, были полностью спроектированы и разработаны все программные модули информационной системы «Охрана квартир».

# 5 РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Тестовый сценарий – это неавтоматизированный или автоматизированный сценарий, содержащий инструкции по реализации тестового набора. Тестовый сценарий может быть написан вручную (для выполнения человеком) либо полностью или частично автоматизирован.

## 5.1 Назначение эксперимента. Выбор и обоснование методики проведения тестирования

Можно определить такие основные цели тестирования программного обеспечения:

* Предоставление информации о качестве ПО конечному заказчику.
* Повышение качества ПО.
* Предотвращение появления дефектов.

Цели тестирования могут отличаться, в зависимости от этапа разработки программного обеспечения, на котором оно проводится. К примеру, на этапе кодирования целью тестирования будет вызов как можно большего количества сбоев в работе программы, что позволит локализовать и исправить дефекты. В то же время, при приемочном тестировании необходимо показать, что система работает правильно. В период сопровождения, тестирование в основном необходимо для того, чтобы убедится в отсутствии новых багов, появившихся во время внесения изменений.

Главная же задача тестирования – поиск дефектов.

В данном разделе проведены два вида тестирования. Провести анализ тестирования и выбрать, как будет проходить тестирование ИС.

Модульное тестирование Цель: проверить, что код работает именно так, как должен (при заданных входных параметрах выдает предполагаемый результат).

Анализ покрытия кода тестами. Цель: проверить, что весь наш код отрабатывает при модульном тестировании, что нет не участвующих в тестировании участков кода.

Анализ стилистики кода. Цель: проверка кода на соблюдение стандартов разработки (отраслевых, корпоративных, принятых на проекте).

Анализ производительности. Цель: проанализировать производительность ключевых операций, потребление памяти, утечку памяти.

## 5.2 Технология тестирования

Рассмотрим, какие технологии можно применить для модульного тестирования и покрытия кода тестами.

Как в ходе выполнения проекта, так и при развитии системы, изменения в ее алгоритмы вносятся в соответствии с техническими заданиями, хотя данный документ может иметь различные названия. В технических заданиях, как правило, описываются критерии приемки: как мы и заказчик убедимся в том, что заказанный функционал работает верно.

В качестве примера можно привести такой упрощенный критерий:

* Исходные данные: в систему внесен документ реализации товаров на сумму 100 руб., со ставкой НДС 20%, НДС сверху.
* Действия пользователя: пользователь формирует отчет по продажам за период, в котором введен документ.
* В результате в отчете отображена сумма реализации 100 руб. и сумма НДС 20 руб.
* Вручную такое тестирование как правило выполняется единожды, поскольку выполнять каждый раз полный комплект тестов очень трудоемко.
* Нет гарантии того, что тесты, описанные в техническом задании, покрывают функциональность полностью. Т.е. даже если тесты описаны и проводятся регулярно, вероятно наличие не протестированного функционала.

Автоматизированное тестирование лишено данных недостатков. Конечно, возникает отдельная задача кодирования данного теста на основании критериев приемки, но это скорее дисциплинирующий фактор…

Прежде чем начинать юзабилити-тестирование необходимо собрать группу пользователей, которые будут тестировать данную информационную систему. Количество привлеченных пользователей должно быть не менее пяти человек. Пользователям будет представлен список некоторых вопросов, относящихся к графическому интерфейсу.

В таблице 27 представлены сведения о пользователях.

Таблица 27 – Информация о пользователях

|  |  |  |  |
| --- | --- | --- | --- |
| ФИО | Пол | Возраст | Образование |
| Мамакова Милослава Вадимовна | Женский | 17 | Основное общее |
| Михеева Екатерина Александровна | Женский | 18 | Полное общее |
| Иванок Иван Викторович | Мужской | 19 | Основное общее |
| Иванов Александр Валерьевич | Мужской | 30 | Среднее специальное |
| Каплан Екатерина Даниловна | Женский | 22 | Среднее специальное |

## Результаты проведения тестирования

В ходе тестирования был применен метод нагрузочного тестирования и юзабилити-тестирования, которые проверяют работу при нагрузке системы и сделать простым и удобным в пользовании информационной системы.

Суть тестирования заключается в проверке работы всех обязательных функций.

Результаты проведения нагрузочного тестирования

Для проведения нагрузочного тестирования составлена таблица 28, в которой указано количество запущенных копий программного продукта, нагрузка на процесс и оперативную память.

Таблица 28 – Результаты нагрузочного тестирования

|  |  |  |
| --- | --- | --- |
| Число копий | Нагрузка на процессор(%) | Нагрузка на память(%) |
| 1 | 0,1 | 0,2 |
| 5 | 0,1 | 1 |
| 10 | 2,1 | 2 |
| 15 | 3,5 | 3 |
| 25 | 4,8 | 5 |

Исходя из результатов тестирования, можно прийти к выводу, что нагрузка, оказываемая на центральный процессор и память, не имеет критичных значений. Это значит, что показатель производительности информационной системы на приемлемом уровне, то есть сбои и длительная загрузка не должны помешать пользователю при работе с данным программным продуктом.

Таблица 29 – Результаты проведения юзабилити-тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вопрос | Мамакова Милослава Вадимовна | Михеева Екатерина Александровна | Иванюк Иван Викторович | Иванов Александр Валерьевич | Каплан Екатерина Даниловна |
| Присутствуют ли такие дизайн-элементы, которые неприятны для зрения? | нет | нет | нет | нет | нет |
| Присутствует ли отвлекающая информация? | нет | нет | нет | нет | нет |
| Понятна ли структура приложения на интуитивном уровне? | да | да | да | да | да |
| Смогли ли вы найти интересующую вас информацию без затруднений? | да | да | да | нет | да |
| Понятен ли смысл приложения? | да | да | да | да | да |
| Общая оценка | 9/10 | 8/10 | 8/10 | 7/10 | 7/10 |

Исходя из средней оценки информационной системы, рассчитанной на основе общих оценок всех пользователей с учетом ответов на все поставленные вопросы, можно сделать вывод о том, что внешние характеристики графического интерфейса находятся на довольно высоком уровне. Также в информационной системе нет помех для пользователя в виде отвлекающей информации, так как все отображаемые данные находятся в рамках предметной области.

Вывод по главе: были проведены тесты путём проведения модульного тестирования, анализа покрытия кода тестами, анализа стилистики кода и анализом производительности. Также было проведено нагрузочное тестирование, в результате чего нагрузка на процессор и память не имела критичных значений.

# ЗАКЛЮЧЕНИЕ

В результате учебной практики был проведен анализ предметной области, спроектирована и разработана информационная система «Ремонт пассажирских вагонов», серверная часть которой была реализована в MS SQL, а клиентская часть в MS Visual Studio.

В MS SQL была создана база данных для хранения информации о вагонах, персонале, должностях, видов ремонта и о самом ремонте вагонов.

Клиентская часть, реализованная в MS Visual Studio, отражает все хранящиеся в базе данных таблицы. Также разработаны некоторые запросы, позволяющие редактировать и удалять существующие данные, добавлять новые и осуществлять поиск по таблицам.

Цели учебной практики были достигнуты путем проведения анализа предметной области, проектирования информационной системы и разработки программных модулей.

Это позволяет сделать вывод, что поставленные цели и задачи выполнены в полном объеме.

В дальнейшем данная информационная система может быть модернизирована путем добавления формы личного кабинета, создания версии для слабовидящих людей или улучшения и изменения дизайна.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. IT-GOST.RU [Электронный ресурс]: Теория и практика UML. Диаграмма деятельности, 2023. URL: <http://it-gost.ru/articles/view_articles/96> (дата обращения: 20.11.2023).
2. [Электронный ресурс]: Программа профессионального модуля ПМ 01. Разработка модулей программного обеспечения для компьютерных систем,2019. URL: <https://cs.гапоу-кгпт.рф/-/dpBw6QvcMrs97SUYUOCWug/sv/document/e2/66/37/682986/2747/РП%20ПМ%2001.pdf?1680595748> (дата обращения: 18.11.2023).
3. ГОСТ 20886-85. Организация данных в системах обработки данных. Термины и определения [Электронный доступ]. — Режим доступа: [ГОСТ 20886-85. Организация данных в системах обработки данных. Термины и определения (rostest.info)](https://rostest.info/gost/001.001.040.035/gost-20886-85/#:~:text=%D0%93%D0%9E%D0%A1%D0%A2%2020886%2D85.%20%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B2,%D0%BD%D0%B0%D1%83%D1%87%D0%BD%D0%BE%2D%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%) (Дата обращения: 01.12.2023 г.)
4. ГОСТ Р. 59793–2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания [Электронный доступ]. — Режим доступа: [ГОСТ Р. 59793-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания (internet-law.ru)](https://internet-law.ru/gosts/gost/77858/?ysclid=lifap277mg519765664) (Дата обращения: 01.12.2023 г.).
5. ГОСТ 19.201-78. ЕСПД. Общие положения. [Электронный доступ]. — Режим доступа: [ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению (swrit.ru)](https://www.swrit.ru/doc/espd/19.201-78.pdf?ysclid=lpi5u6naom706376064) (Дата обращения: 01.12.2023 г.).
6. Microsoft Visual Studio [Электронный доступ]. — Режим доступа: [Руководство по разработке модулей расширений на C# — Павел Еремеев (pvs-studio.ru)](https://pvs-studio.ru/ru/blog/posts/csharp/a0082/) (Дата обращения: 01.12.2023 г.).
7. Microsoft Visual Studios [Электронный доступ]. — Режим доступа: [Microsoft Visual Studios — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) (Дата обращения: 01.12.2023 г.).
8. Microsoft SQL Server [Электронный доступ]. — Режим доступа: [Microsoft SQL Server — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server) (Дата обращения: 01.12.2023 г.).

# ПРИЛОЖЕНИЕ A



Рисунок 1 - Набор данных к информационной системе «Охрана квартир»

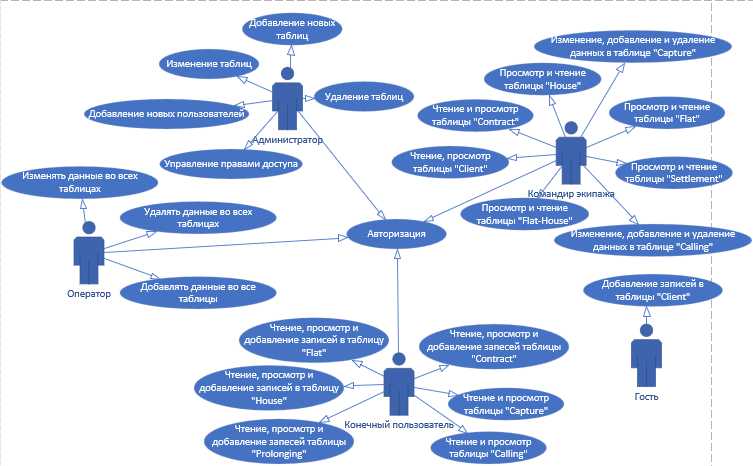


Рисунок 2 – Диаграмма вариантов использования



Рисунок 3 – Диаграмма деятельности «Гость»



Рисунок 4 – Диаграмма деятельности «Администратора»



Рисунок 5 – Диаграмма деятельности «Конечного пользователя»



Рисунок 6 – Диаграмма деятельности «Командира экипажа»



Рисунок 7 – Диаграмма деятельности «Оператора»

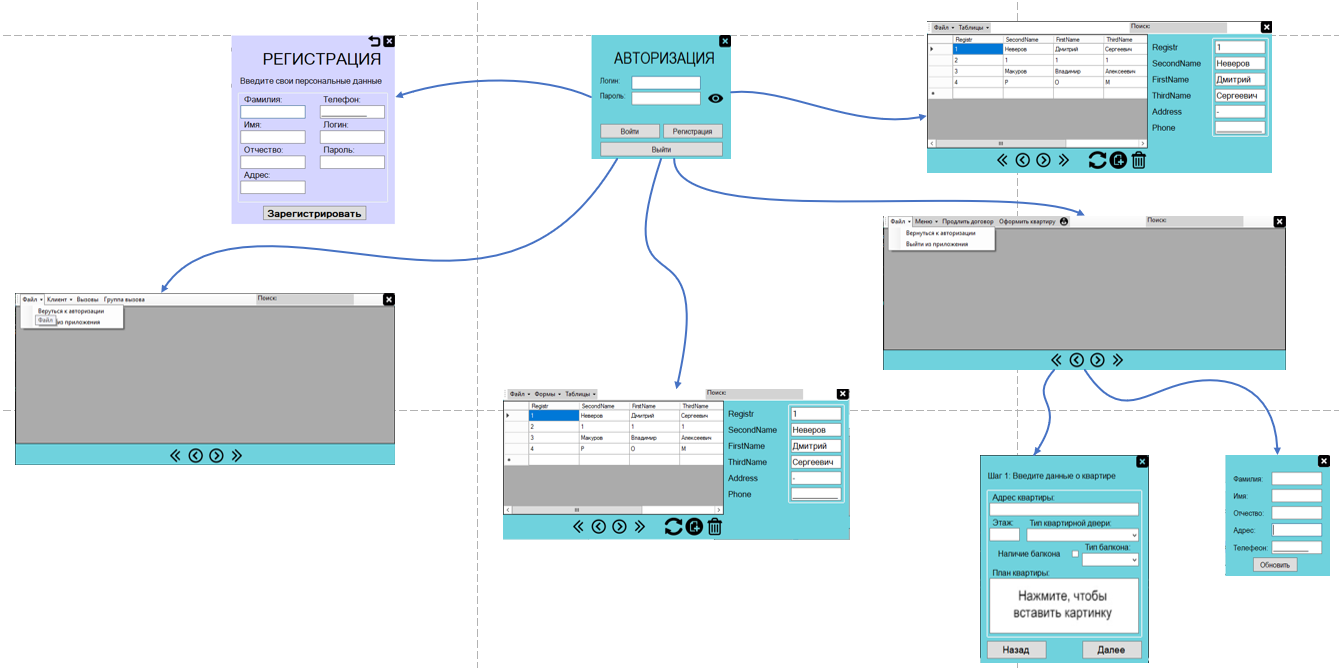


Рисунок 8 – Карта переходов

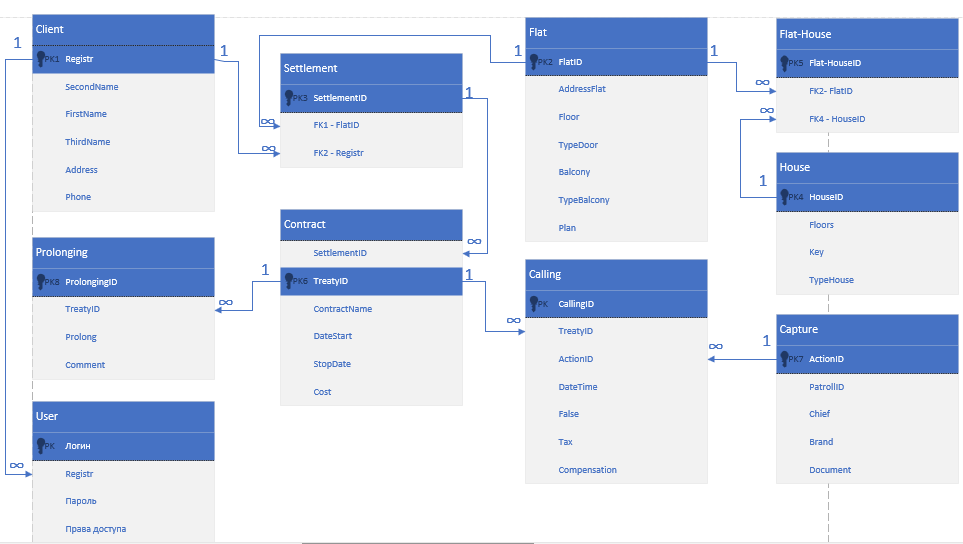


Рисунок 18 – Логическая модель данных

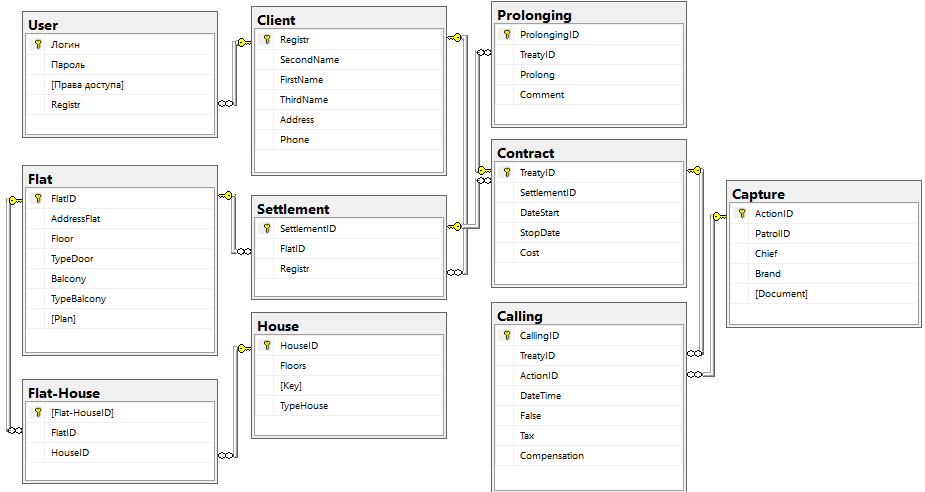


Рисунок 20 – Физическая модель данных