**Оглавление**

[Аннотация 4](#_Toc72446649)

[Введение 5](#_Toc72446650)

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ 6](#_Toc72446651)

[1.1 Основное функциональное назначение программного средства 6](#_Toc72446652)

[1.2 Полное наименование программного средства 6](#_Toc72446653)

[1.3 Условное обозначение программного средства 6](#_Toc72446654)

[1.4 Разработчики программного средства 6](#_Toc72446655)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc72446656)

[2.1 Основание для разработки 7](#_Toc72446657)

[2.2 Назначение разработки 7](#_Toc72446658)

[2.3 Требование к программному средству 7](#_Toc72446659)

[2.4 Требования к программной документации 8](#_Toc72446660)

[2.5 Требования к эргономике и технической эстетике 8](#_Toc72446661)

[2.6 Стадии и этапы разработки 9](#_Toc72446662)

[2.7 Порядок контроля и приемки 10](#_Toc72446663)

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 11](#_Toc72446664)

[3.1 Декомпозиция поставленной задачи 11](#_Toc72446665)

[3.2 Общая архитектура программного средства 12](#_Toc72446666)

[3.3 Разработка алгоритма решения задачи 15](#_Toc72446667)

[3.4 Реализация функционального назначения программного средства 16](#_Toc72446668)

[3.5 Структурная организация данных 16](#_Toc72446669)

[3.6 Разработка интерфейса ПС 24](#_Toc72446670)

[3.7 Описание структуры выходной информации 38](#_Toc72446671)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 39](#_Toc72446672)

[4.1 Назначение программного средства 39](#_Toc72446673)

[4.2 Условия выполнения программного средства 39](#_Toc72446674)

[4.3 Эксплуатация программного средства 39](#_Toc72446675)

[4.4 Сообщения пользователю 50](#_Toc72446676)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 51](#_Toc72446677)

[5.1 Определение верхней границы количества тестов 51](#_Toc72446678)

[5.2 Разработка тестовых вариантов 52](#_Toc72446679)

[5.3 Составление отчета о тестировании 54](#_Toc72446680)

[ЗАКЛЮЧЕНИЕ 60](#_Toc72446681)

[Оценка качества программного средства с помощью метрик 60](#_Toc72446682)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 63](#_Toc72446683)

[Приложение А Текст кода программного средства 67](#_Toc72446684)

[Приложение Б Спецификация 84](#_Toc72446685)

[Приложение B Диск с программным средством 87](#_Toc72446686)

**Аннотация**

На дипломную работу студента Смирнова М.В. на тему: Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан». Дипломная работа представлена пояснительной запиской на 87 листов машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве АИС MaxДрайв. В главе 2 приведено техническое задание на проектирование БД согласно ГОСТ 19.201. Глава 3 содержит пояснительную записку к программному продукту согласно ГОСТ 19.404-79. В главе 4 содержится руководство пользователя согласно ГОСТ 19.504-79. Глава 5 содержит обоснование количества необходимых тестов и тестовые пакеты для оценки работоспособности программного продукта. В главе 6 приведены результаты ручного и автоматического тестирования программного продукта и выполнена оценка качества с помощью метрик.

В работе содержится общее описание программного средства, предназначенного для объекта проектирования – автоматизированной иинформационной системы «Учет привлекаемых сил и средств местных пожарно – спасательных гарнизонов Республики Татарстан», изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**Введение**

В дипломной работе была поставлена задача разработки автоматизированной информационной системы «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан», которая позволяет автоматизировать деятельность диспетчера ПСЧ.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов был выбран язык программирования C#. Выбор обусловлен тем, что возможностей этого языка достаточно для достижения поставленной цели моей дипломной работы.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа, а также тип пользователей, которые с ней будут работать.

**1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена для автоматизации деятельности диспетчера ПСЧ. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан».

**1.3 Условное обозначение программного средства**

«FireDepartmentApp»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 195 Смирнов М.В.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

Работа ведётся на основании задания на дипломную работу по специальности 09.02.03 — «Программирование в компьютерных системах».

**2.2 Назначение разработки**

Программа «FireDepartmentApp» работает под управление операционной системы Windows 7, Windows 8, Windows 10. Программа предназначена для автоматизации деятельности диспетчера ПСЧ. Цель создания программы – учет выездов пожарных подразделений и сил местного звена ТП РСЧС. Программа позволяет:

* создавать и редактировать выезд подразделений;
* создавать отчеты;
* создавать и редактировать базу справочников.

**2.3 Требование к программному средству**

«FireDepartmentApp» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 4 Гб;

Процессор: Intel Core i / Phenom II X4 / AMD Athlon64 X2 4600 + 2.4 ГГц;

Видео карта: nVidia GeForce 8600 / ATI Radeon HD 2400;

Звуковая карта: Совместимая с DirectX;

Программный продукт «FireDepartmentApp» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«FireDepartmentApp» написана на объектно-ориентированном языке программировании C# (си-шарп) в интегрированной среде разработки Visual Studio 2022. В качестве СУБД используется MS SQL. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1. техническое задание;

2. сроки выполнения работ;

3. программу и методику испытаний;

4. эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя;

- взаимодействие пользователя с системой осуществляется на двух языках по выбору (русский и английский), для наибольшего удобства

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;

2. рабочее проектирование;

3. внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на нее;

5. выбор языков программирования;

6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. разработка, согласование и утверждение порядка и методики испытаний;

2. проведение испытаний;

3. корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ**

**3.1 Декомпозиция поставленной задачи**

Для достижения цели работы была поставлена следующая задача:

Создать модуль “CodeDatabase” отвечающий за сохранение БД. Результат работы модуля - сохранение БД.

Поставленные задачи наглядно отображаются на диаграмме деятельности.

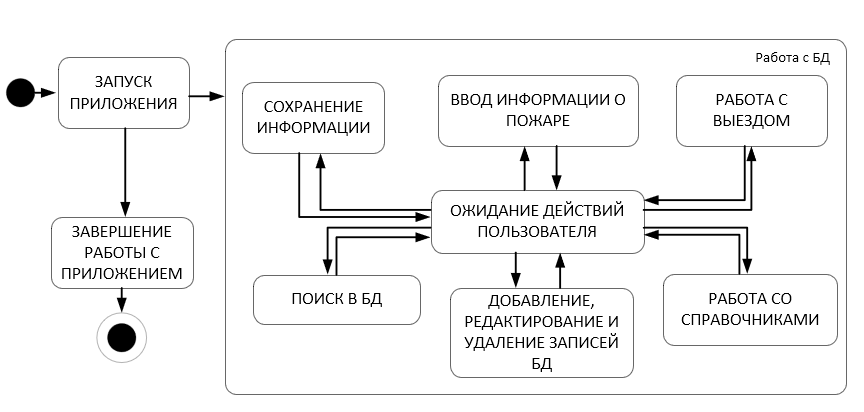


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан».

Рассматриваемые процессы: учёт выездов подразделений.

Объекты моделирования представлены на диаграмме классов, рис.2.

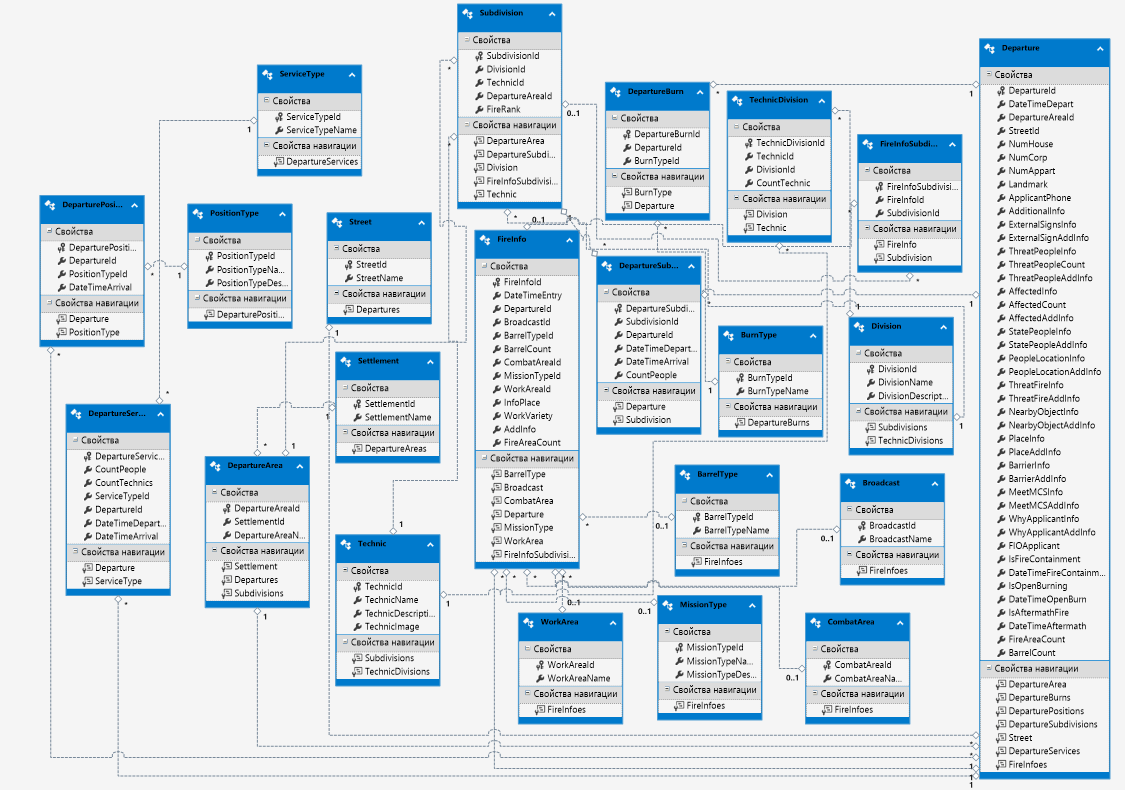


Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

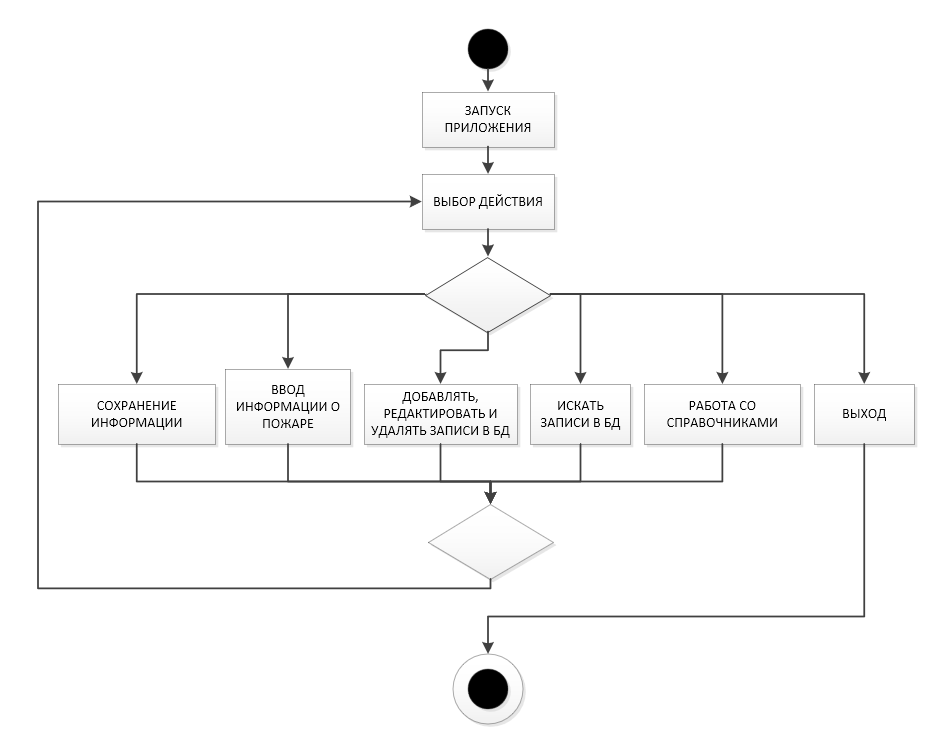


Рис.3 Схема событийно-ориентированной архитектуры

В «FireDepartmentApp» используется событийно-ориентированная архитектура.(Рис.3) Роль агента (источник событий) в программе выполняют: пользователи программы, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие: добавить, удалить, редактировать, вывести в Excel и т.д. система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована, либо отчет выведен в Excel. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.



Рис.4 Диаграмма вариантов использования

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

****Рис.5 Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «FireDepartmentApp» имеет следующий набор входных данных, такие как: список выездов, добавление выезда.

Данные вводятся пользователем в соответствующие поля ввода, снабженные всплывающими подсказками.

Выходными данными являются: отчет о выезде, путевой лист.

Данные вводятся пользователем в соответствующие поля вывода, снабженные всплывающими подсказками.

Выходные данные редактировать вручную невозможно, т.к. они служат в качестве отчетной информации для пользователя.

**3.5 Структурная организация данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы. Все эти данные указаны в реляционной модели представленной на рисунке. Любая реляционная база данных и называется реляционной, что характеризуется отношениями (relation) между таблицами. На рисунке изображены таблицы моей базы данных. При этом одна таблица является родительской (главной), а вторая – дочерней (подчиненной). Главной таблицей является «Выезд». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)



Рис.6 Реляционная модель

Первичный ключ в базе уникален, используется для организации отношений между таблицами, который не может иметь пустых и повторяющихся значений. Первичными ключами в базе являются поля: «Код выезда» (таблица «Выезд»), «Код прибытия» (Таблица «Прибытие служб»), «Код службы» (Таблица «Служба»), «Код улицы» (Таблица «Улица»), «Код должностного лица» (Таблица «Должностное лицо»), «Код прибытия должностного лица» (Таблица «Прибытие должностных лиц»), «Код прибытие подразделения» (Таблица «Прибытие подразделений»), «Код населенного пункта» (Таблица «Населенный пункт»), «Код приданных сил» (Таблица «Приданные силы») и т.д. Остальные ID-ключи являются внешними ключами.

Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:

* БД находится в форме -1НФ потому, что
  + Таблица не имеет повторяющихся записей;
  + Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
  + Таблица не имеет повторяющихся групп полей.
    - Вторая нормальная форма(2НФ):
      * Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора, т.е. ID.
        + Третья нормальная форма(3НФ):

Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

На основе реляционной модели базы данных мною в MS SQL была построена следующая база данных. (Рис.7) В ней первичным ключом является поле таблицы “FireInfo” – “FireInfoId”. Связь таблиц “FireInfo” и “BarrelTypes” по полю “BarrelTypeId” один ко многим (1:М). Связь таблиц “FireInfo” и “Broadcasts ” по полю “BroadcastId” один ко многим (1:1). Связь таблиц “FireInfo” и “WorkAreas” по полю “WorkAreaId” один ко многим (1:М). Связь таблиц “FireInfo” и “CombatAreas” по полю “CombatAreaId” один ко многим (1:М). Связь таблиц “FireInfo” и “MissionTypes” по полю “MissionTypeId” один ко многим (1:М). Связь таблиц “Departures” и “DepartureAreas” по полю “DepartureAreaId” один ко многим (1:М). Связь таблиц “Departures” и “Streets” по полю “StreetId” один ко многим (1:М). Связь таблиц “TechnicDivisions” и “Divisions” по полю “DivisionId” один ко многим (1:М) и т.д. Все связи между таблицами базы данных типа один ко многим.



Рис.7 Схема базы данных

Таблица “AlignmentFoce”(Расстановка сил и средств)

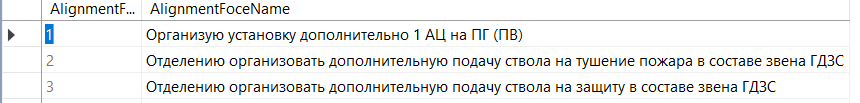


Рис.8 Таблица AlignmentFoce

Таблица “BarrelTypes”(Типы стволов)

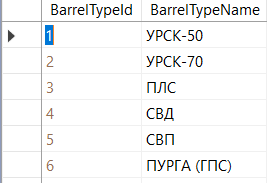


Рис.9 Таблица BarrelTypes

Таблица “Broadcasts”(Тип передач)

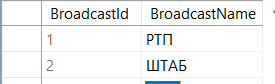


Рис.10 Таблица Broadcasts

Таблица “CombatAreas”(Боевые участки)

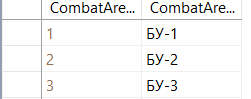


Рис.11 Таблица CombatAreas

Таблица “CombatDepForces”(Боевое развертывание сил и средств)

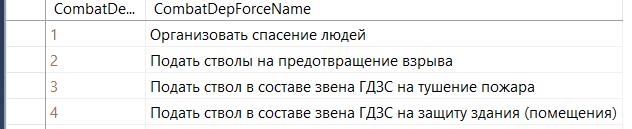


Рис.12 Таблица CombatDepForces

Таблица “DepartureAreas”(Районы выездов)

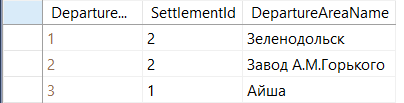


Рис.13 Таблица DepartureAreas

Таблица “DepartureBurns”(Список объектов горения)

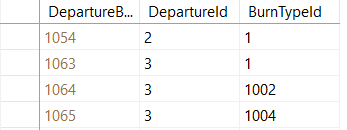


Рис.14 Таблица DepartureBurns

Таблица “DeparturePositions”(Прибывшие должностные лица)

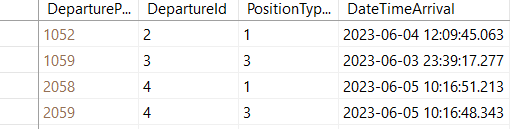


Рис.15 Таблица DeparturePositions

Таблица “Departures”(Выезды)

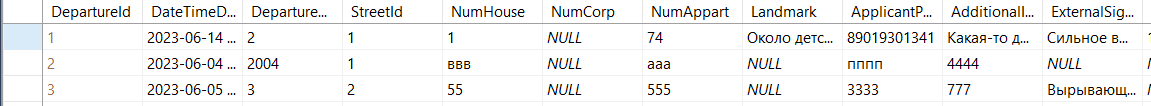


Рис.16 Таблица Departures

Таблица “DepartureServices”(Прибывашие службы)

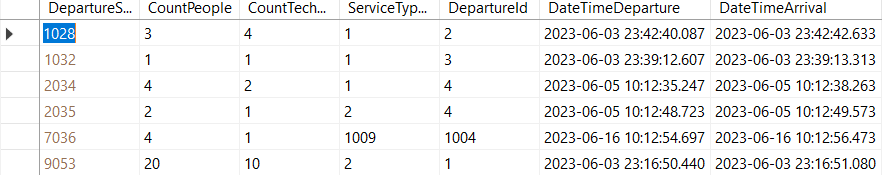


Рис.17 Таблица DepartureServices

Таблица “DepartureSubdivisions”(Прибывшие подразделения)

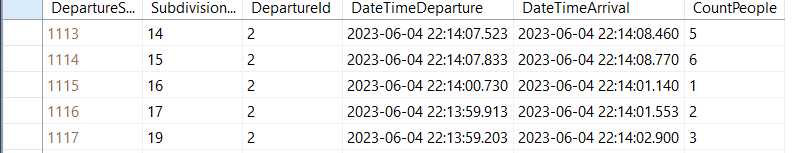


Рис.18 Таблица DepartureSubdivisions

Таблица “Divisions”(ПСЧ)

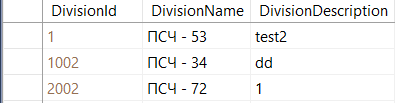


Рис.19 Таблица Divisions

Таблица “FireInfo”(Информация с места пожара)

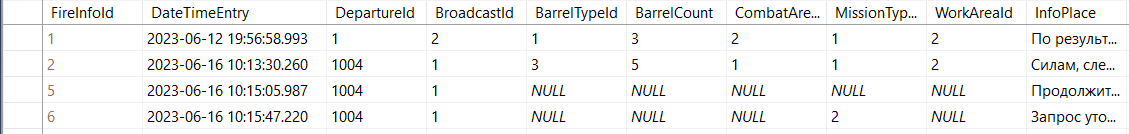


Рис.20 Таблица FireInfo

Таблица “FireInfoSubdivisions”(Приданные силы)

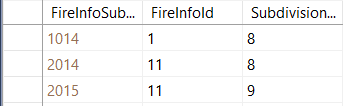


Рис.21 Таблица FireInfoSubdivisions

Таблица “MissionTypes”(Задача)

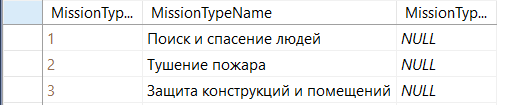


Рис.22 Таблица MissionTypes

Таблица “PositionTypes”(Должностные лица)

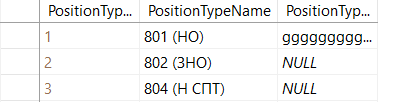


Рис.23 Таблица PositionTypes

Таблица “ServiceTypes”(Службы)

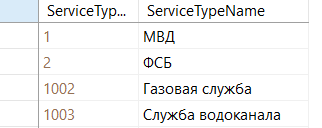


Рис.24 Таблица ServiceTypes

Таблица “Settlements”(Населенные пункты)

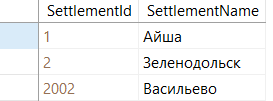


Рис.25 Таблица Settlements

Таблица “Streets”(Улицы)

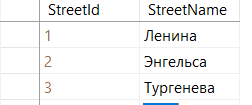


Рис.26 Таблица Streets

Таблица “Subdivisions”(Подразделения)

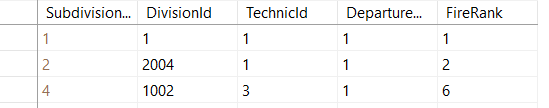


Рис.27 Таблица Subdivisions

Таблица “TechnicDivisions”(Техника ПСЧ)

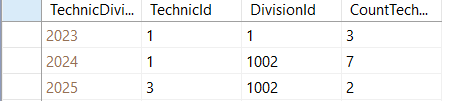


Рис.28 Таблица TechnicDivisions

Таблица “Technics”(Техника)

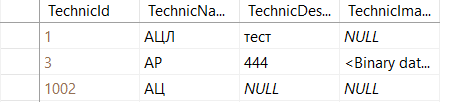


Рис.29 Таблица Technics

Таблица “WorkAreas”(Участок работы)

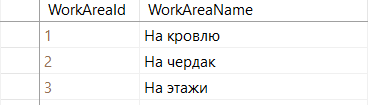


Рис.30 Таблица WorkAreas

**3.6 Разработка интерфейса ПС**

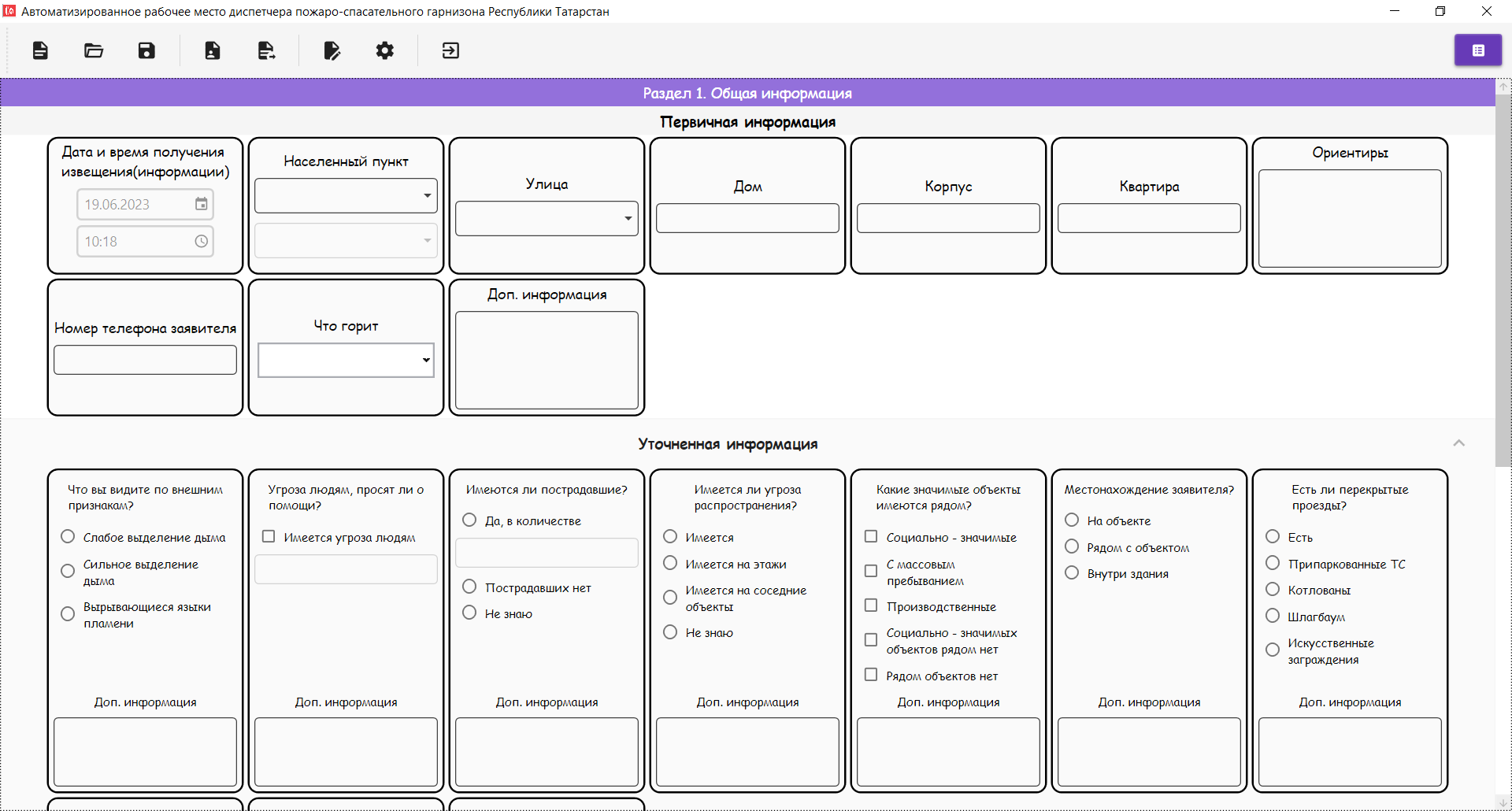
Приложение предназначено для диспетчера ПСЧ. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После запуска программы пользователь приступит к работе с главной формой программы, на которой отбражается форма добавления выезда. «FireDepartmentApp» обладает удобным интерфейсом, так что даже неопытному в обращение с пользовательскими компьютерными программами пользователю не составит труда быстро привыкнуть к работе с данным программным средством.

На рабочем экране расположены несколько кнопок для комфортной работы с приложением.

**Форма Автоматизированная инфомационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан»»**. Основная форма приложения (Рис.31). При разработке интерфейса этой формы было использовано множество различныx компонентов:

1. ComboBox – выпадающие список с выбором элементов;
2. DatePicker – для выбора даты выезда;
3. TimePicker – для выбора времени выезда;
4. Buttons – кнопки для перехода на другие формы;
5. TextBox – поля для ввода



Button

ComboBox

DatePicker

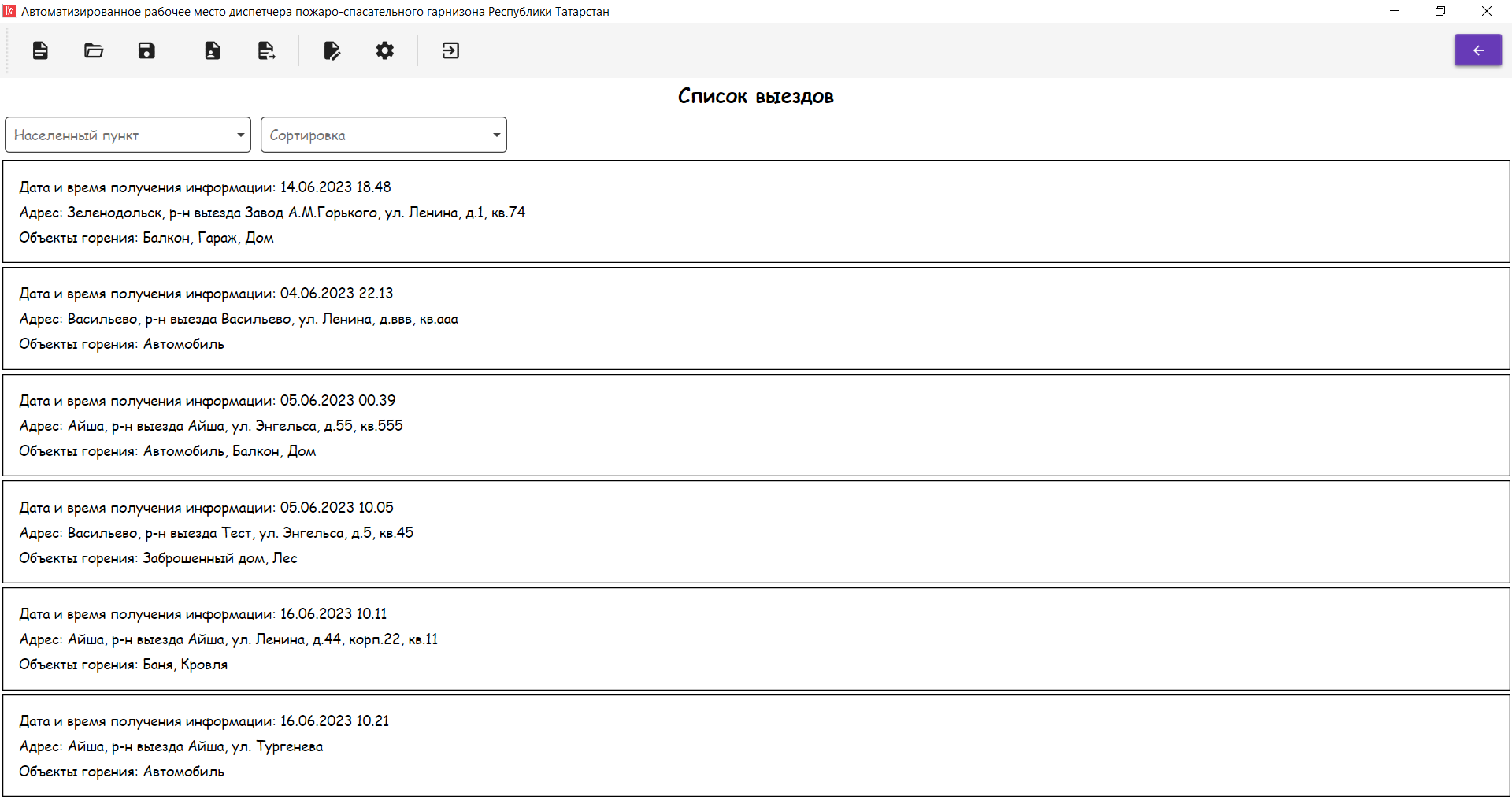
TextBox

TimePicker

Рис.31 Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан»

**Форма Список выездов (**Рис.32**)**. При разработке интерфейса этой формы были использованы компоненты:

1. ListView – отображение списка выездов в виде таблицы;
2. ComboBox – для фильтрации выездов;
3. Buttons – кнопки для перехода к справочникам, назад и т.д.;
4. TextBlock – текстовая метка;



Button

ListView

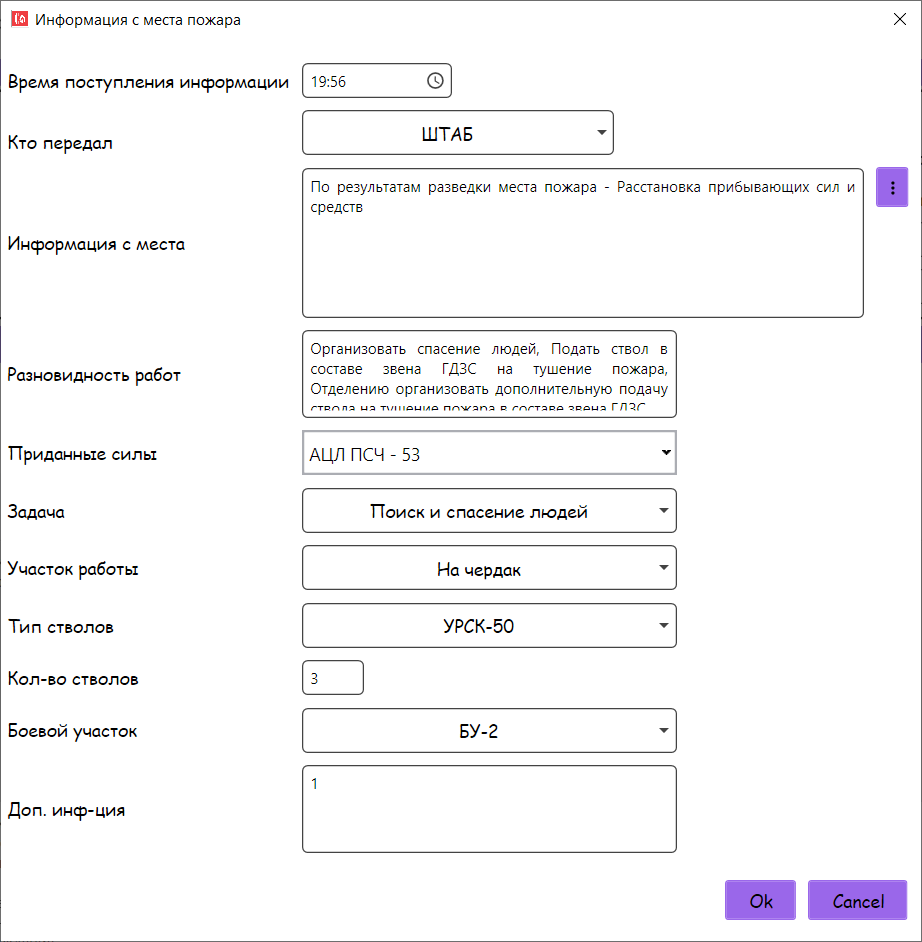
TextBlock

ComboBox

Рис.32 Форма Список выездов

**Форма Информация с места пожара(**Рис.33**)**. При разработке интерфейса этой формы были использованы компоненты:

1. TextBox – для ввода информации;
2. ComboBox – для выбора элементов;
3. Buttons – кнопки для перехода к справочникам, сохранить, отменить;
4. TextBlock – текстовая метка;



Button

TextBox

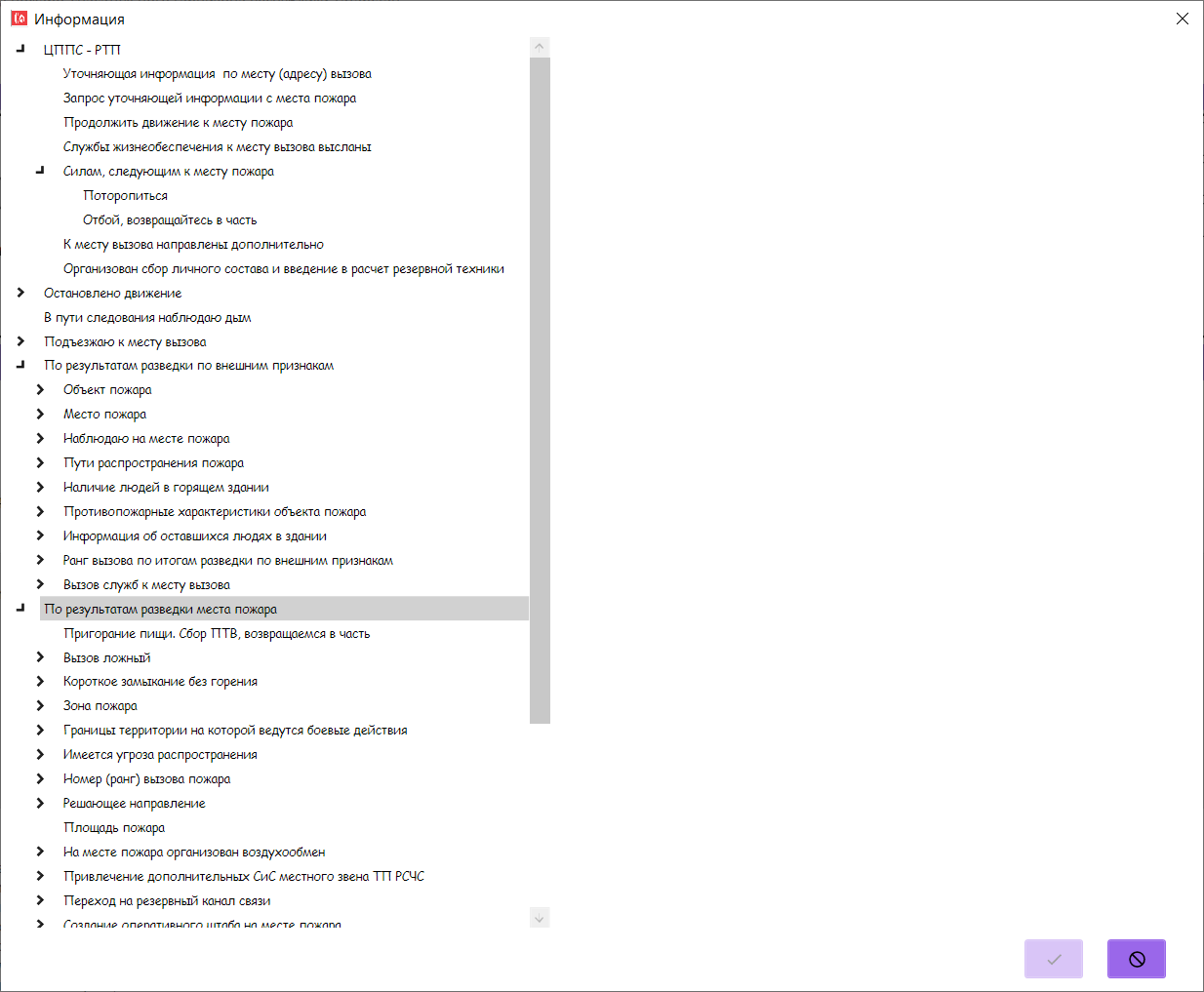
TextBlock

ComboBox

Рис.33 Форма Информация с места пожара

**Форма Информация (**Рис.34**)**. При разработке интерфейса этой формы были использованы компоненты:

1. TextBox – для ввода информации;
2. Buttons – кнопки сохранить, отменить;



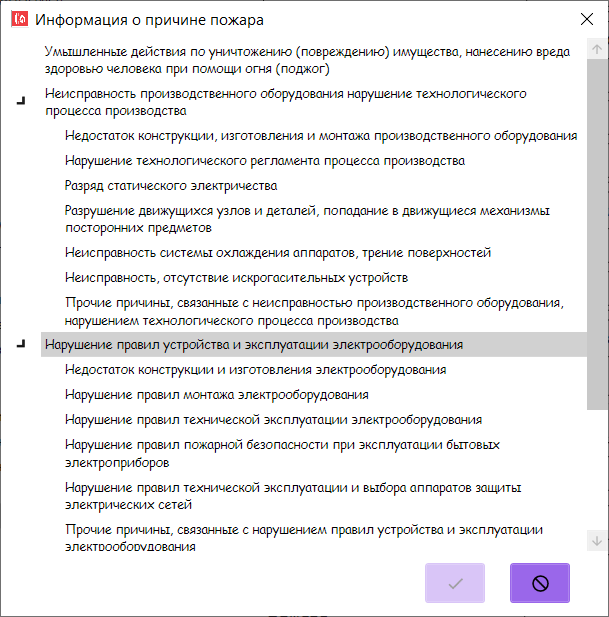
Button

ListViewItem

Рис.34 Форма Информация

**Форма Информация о причине пожара(**Рис.35**)**. При разработке интерфейса этой формы были использованы компоненты:

1. TextBox – для ввода информации;
2. Buttons – кнопки сохранить, отменить;



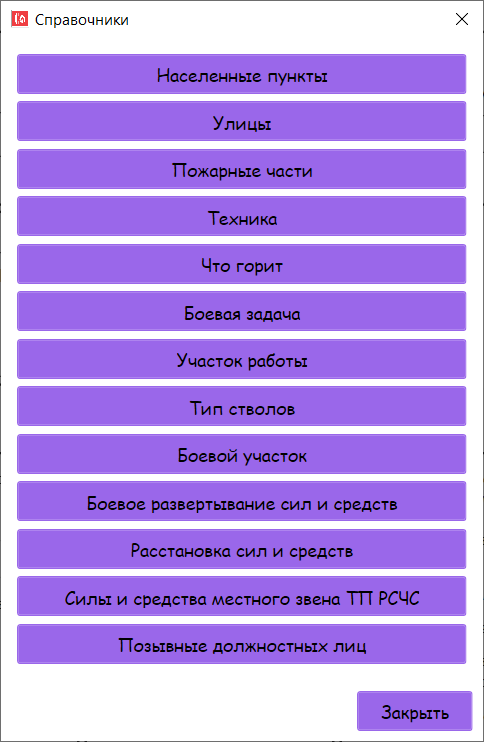
Button

ListViewItem

Рис.35 Форма Информация о причине пожара

**Форма Справочники(**Рис.36**)**. При разработке интерфейса этой формы были использованы компоненты:

Buttons – кнопки закрыть, а также кнопки переходов к другим формам;

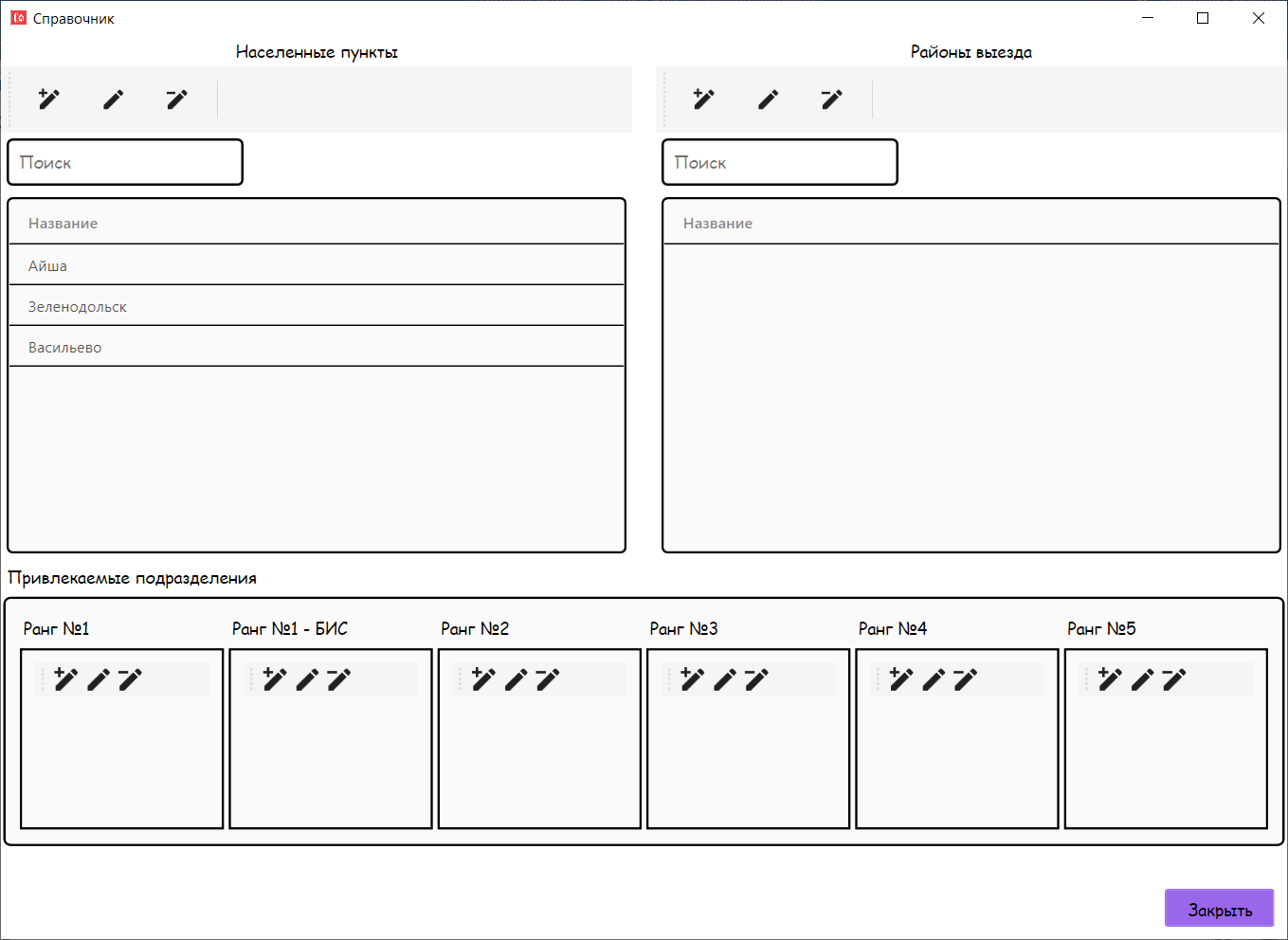


Button

Рис.36 Форма Справочники

**Форма Справочник(**Рис.37**)**. При разработке интерфейса этой формы были использованы компоненты:

1. Buttons – кнопки закрыть, а также кнопки переходов к другим формам;
2. TextBox – для поиска населенного пункта или района выезда;
3. DataGrid – для отображения информации;



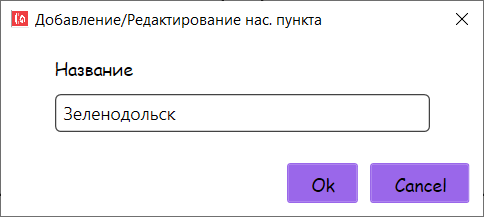
TextBox

DataGrid

Button

Рис.37 Форма Справочник

**Окно Добавление/Редактирование населенного пункта**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.38)



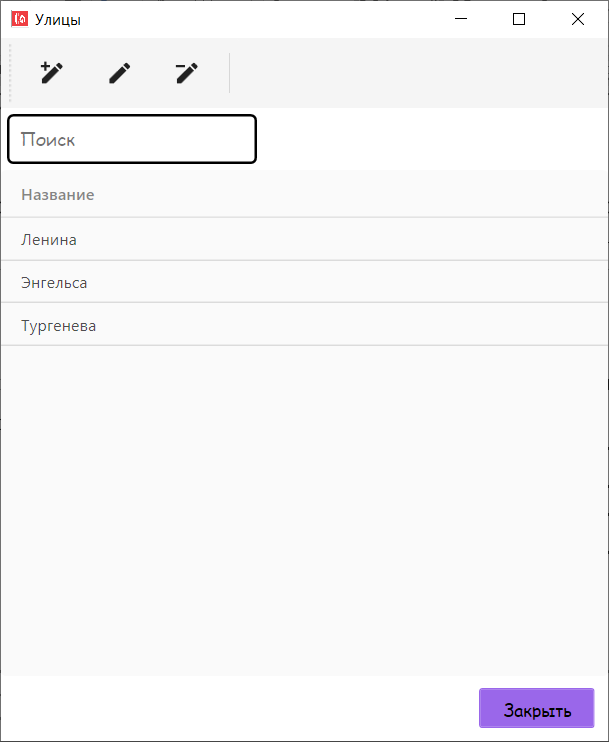
Button

TextBox

Рис.38 Окно Добавление/Редактирование населенного пункта

**Форма Улицы(**Рис.39**)**. При разработке интерфейса этой формы были использованы компоненты:

1. Button – кнопки закрыть;
2. TextBox – для поиска улицы;
3. DataGrid – для отображения информации;



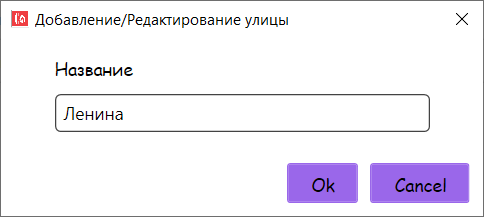
TextBox

DataGrid

Button

Рис.39 Форма Улицы

**Окно Добавление/Редактирование улицы**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.40)



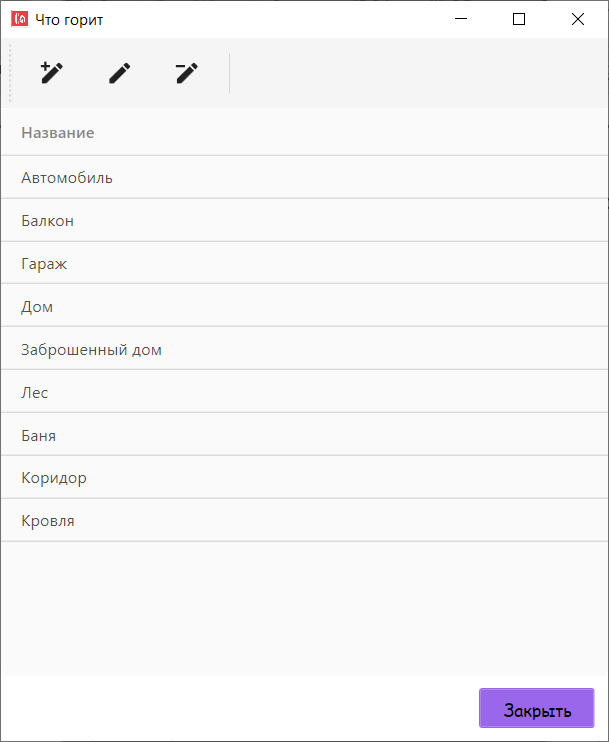
Button

TextBox

Рис.40 Окно Добавление/Редактирование улицы

**Форма Что горит(**Рис.41**)**. При разработке интерфейса этой формы были использованы компоненты:

1. Button – кнопки закрыть;
2. DataGrid – для отображения информации;

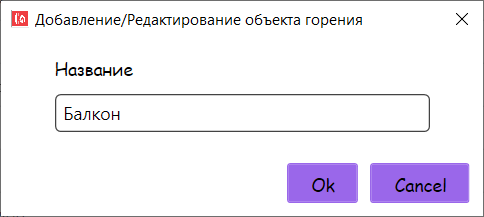


DataGrid

Button

Рис.41 Форма Что горит

**Окно Добавление/Редактирование объекта горения**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.42)



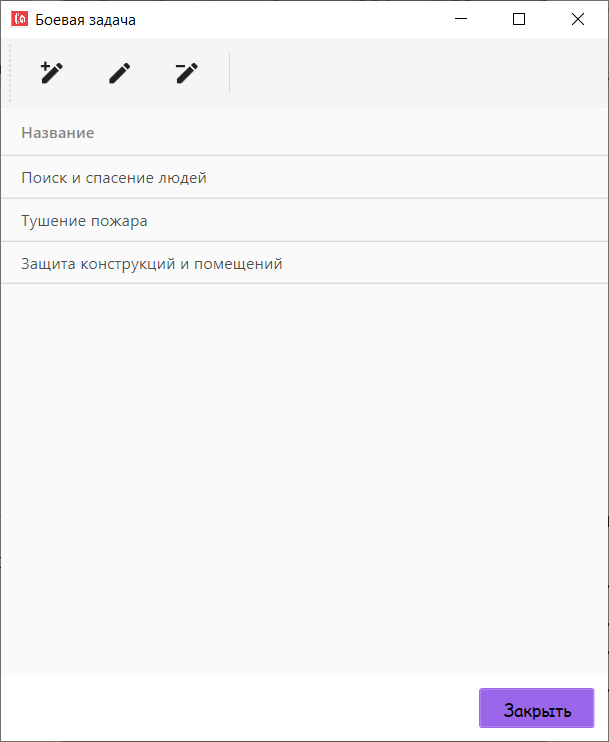
Button

TextBox

Рис.42 Окно Добавление/Редактирование объекта горения

**Форма Боевая задача(**Рис.43**)**. При разработке интерфейса этой формы были использованы компоненты:

1. Button – кнопки закрыть;
2. DataGrid – для отображения информации;

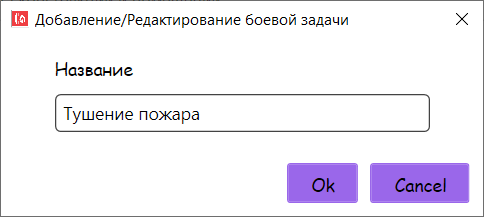


DataGrid

Button

Рис.43 Форма Боевая задача

**Окно Добавление/Редактирование боевой задачи**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.44)



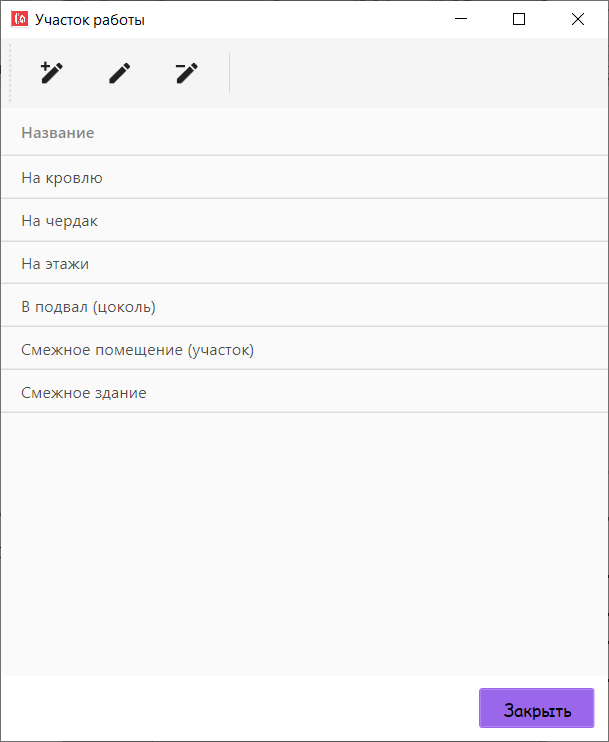
Button

TextBox

Рис.44 Окно Добавление/Редактирование боевой задачи

**Форма Участок работы(**Рис.45**)**. При разработке интерфейса этой формы были использованы компоненты:

1. Button – кнопки закрыть;
2. DataGrid – для отображения информации;

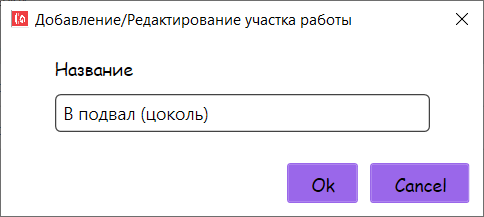


DataGrid

Button

Рис.45 Форма Участок работы

**Окно Добавление/Редактирование участка работы**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.46)



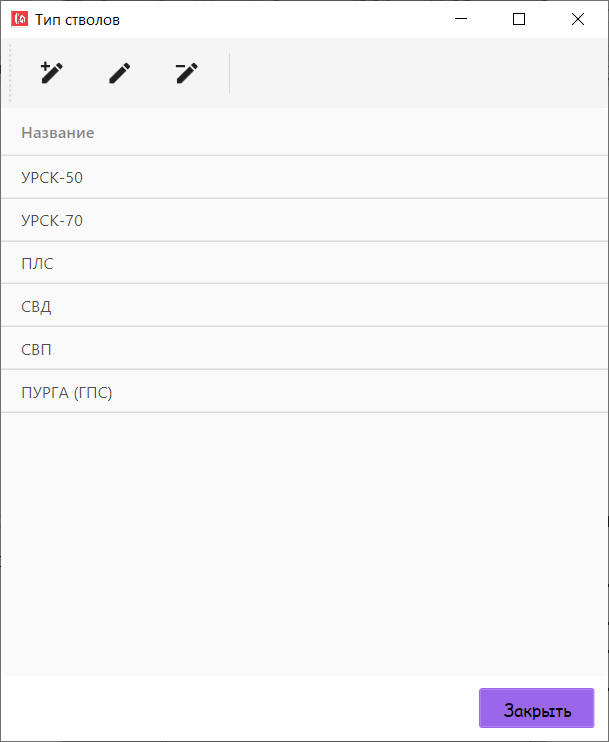
Button

TextBox

Рис.46 Окно Добавление/Редактирование участка работы

**Форма Тип стволов(**Рис.47**)**. При разработке интерфейса этой формы были использованы компоненты:

1. Button – кнопки закрыть;
2. DataGrid – для отображения информации;



DataGrid

Button

Рис.47 Форма Тип стволов

**Окно Добавление/Редактирование типа ствола**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.48)

Button

TextBox

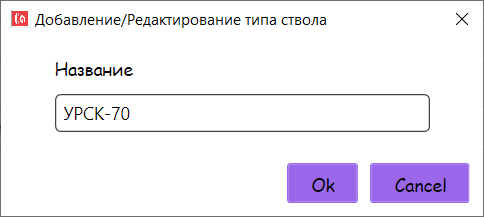
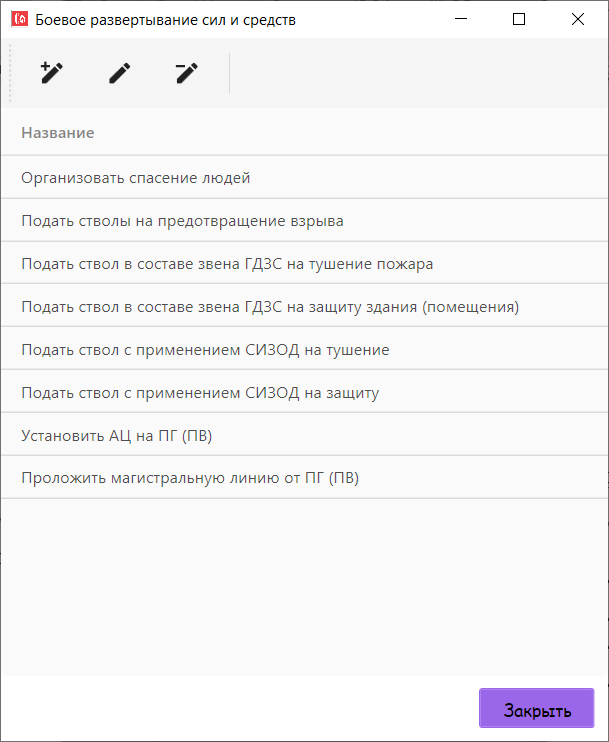


Рис.48 Окно Добавление/Редактирование типа ствола

**Форма Боевое развертывание сил и средств(**Рис.49**)**. При разработке интерфейса этой формы были использованы компоненты:

1. Button – кнопки закрыть;
2. DataGrid – для отображения информации;

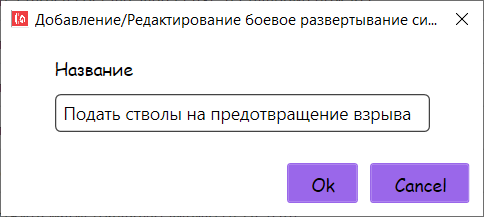


DataGrid

Button

Рис.49 Форма Боевое развертывание сил и средств

**Окно Добавление/Редактирование боевое развертывание сил**. Интерфейс данной формы состоит из компонентов: TextBox – текстовое поле ввода и кнопок Button. (Рис.50)



Button

TextBox

Рис.50 Окно Добавление/Редактирование боевое развертывание сил

**3.7 Описание структуры выходной информации**

Выходными данными являются отчет выезда и путевой лист.

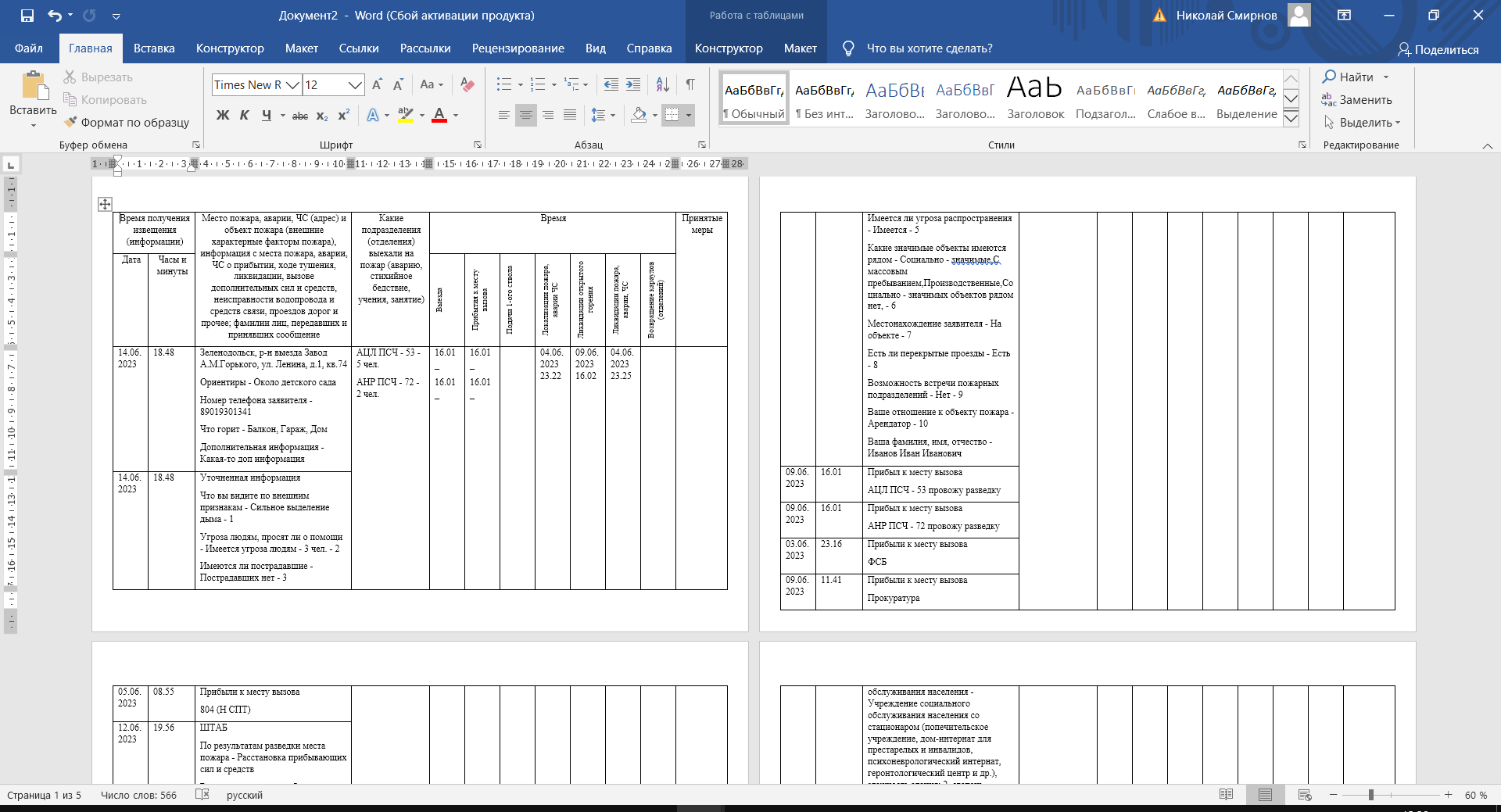


Рис.51 Отчет выезда

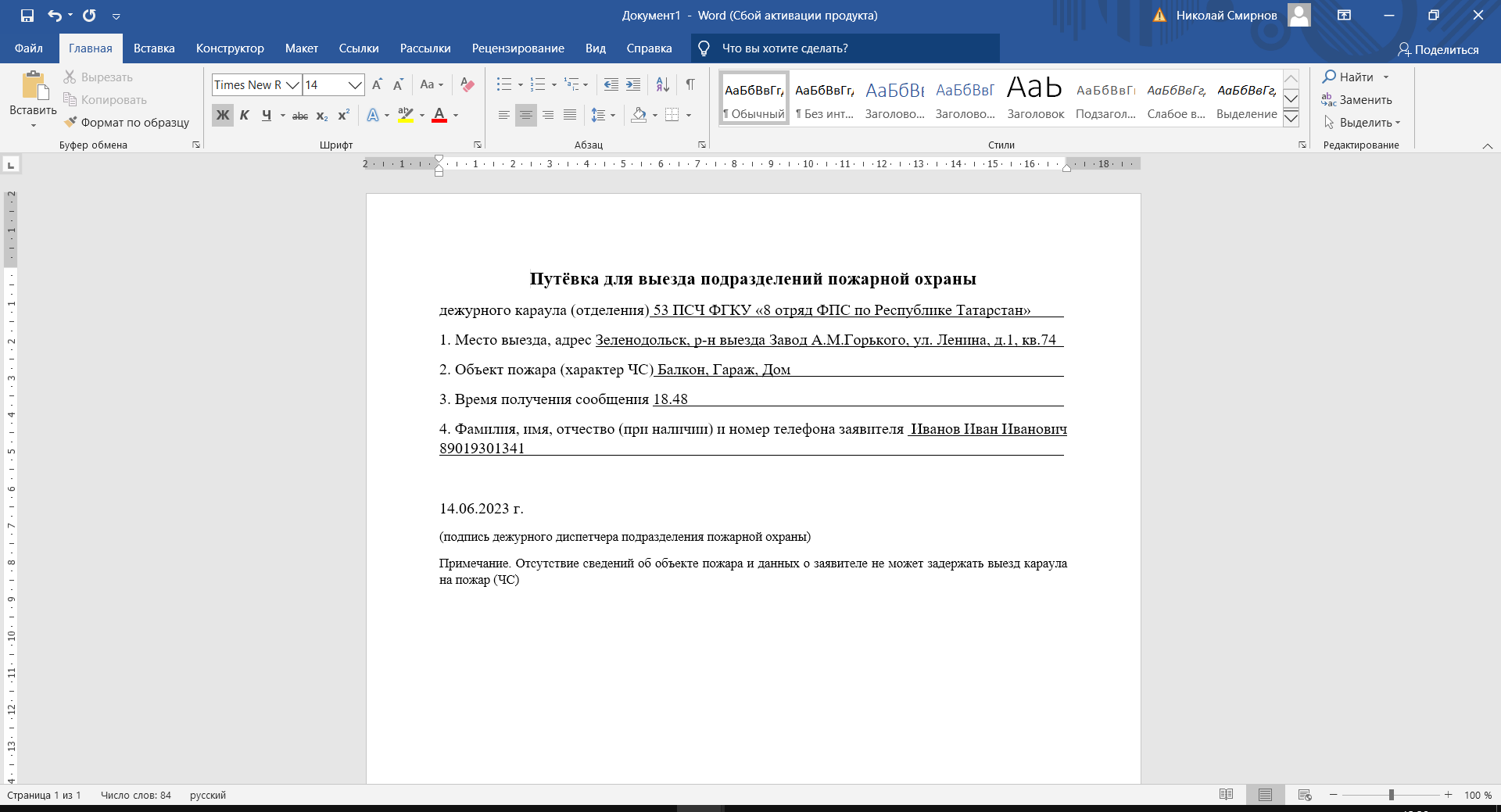


Рис.52 Путевой лист

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация деятельности диспетчера ПСЧ.

**4.2 Условия выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

-Компьютер с установленной на нём «FireDepartmentApp».

Подготовительные действия:

-На иконке «FireDepartmentApp» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «FireDepartmentApp» и ее основных параметров.

Активировать копию программы FireDepartmentApp можно сразу после её установки, когда появится окно подтверждения подлинности Вашей версии. Ключ активации расположен внутри упаковки программного продукта. Для успешного выполнения активации ключ активации необходимо вводить именно в том виде, в котором он предоставлен.

Вводные сведения об интерфейсе пользователя

После запуска приложения отображается стартовое окно приложения (Рис.53). Главная форма представляет собой форму заполнения информации о выезде.

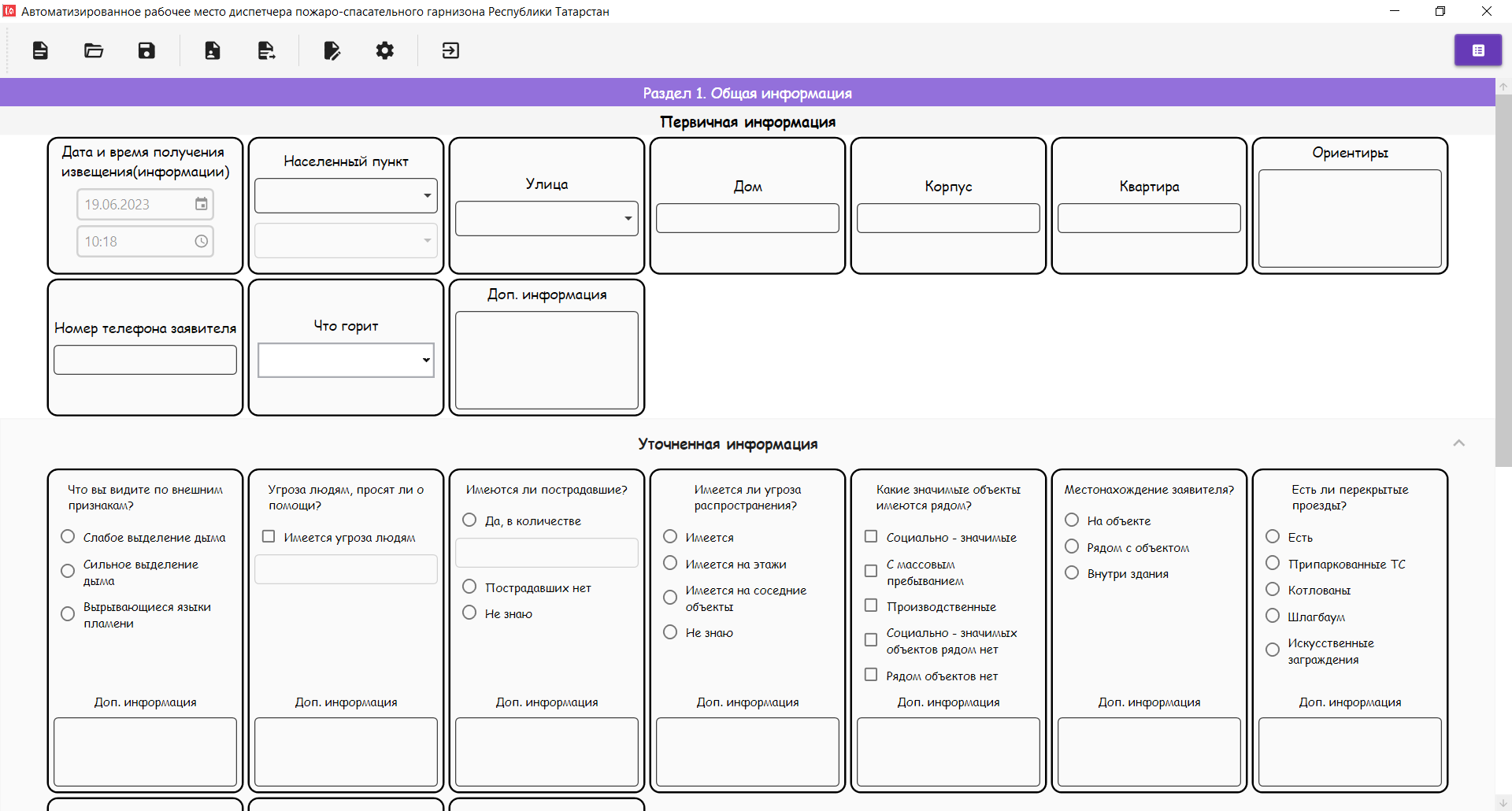


Рис.53 Окно Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан»

Кликнув по кнопке «Закрыть» программа потребует подтверждение. (Рис.54)

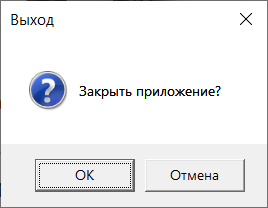


Рис.54 Сообщение программы

Ниже описаны пункты меню управления и информации.

Страница «Список выездов». Данная страница предназначена для отображения информации о выездах в табличном виде, открытие страницы редактирования выделенной записи на двойное нажатие по элементу списка. (Рис.55)

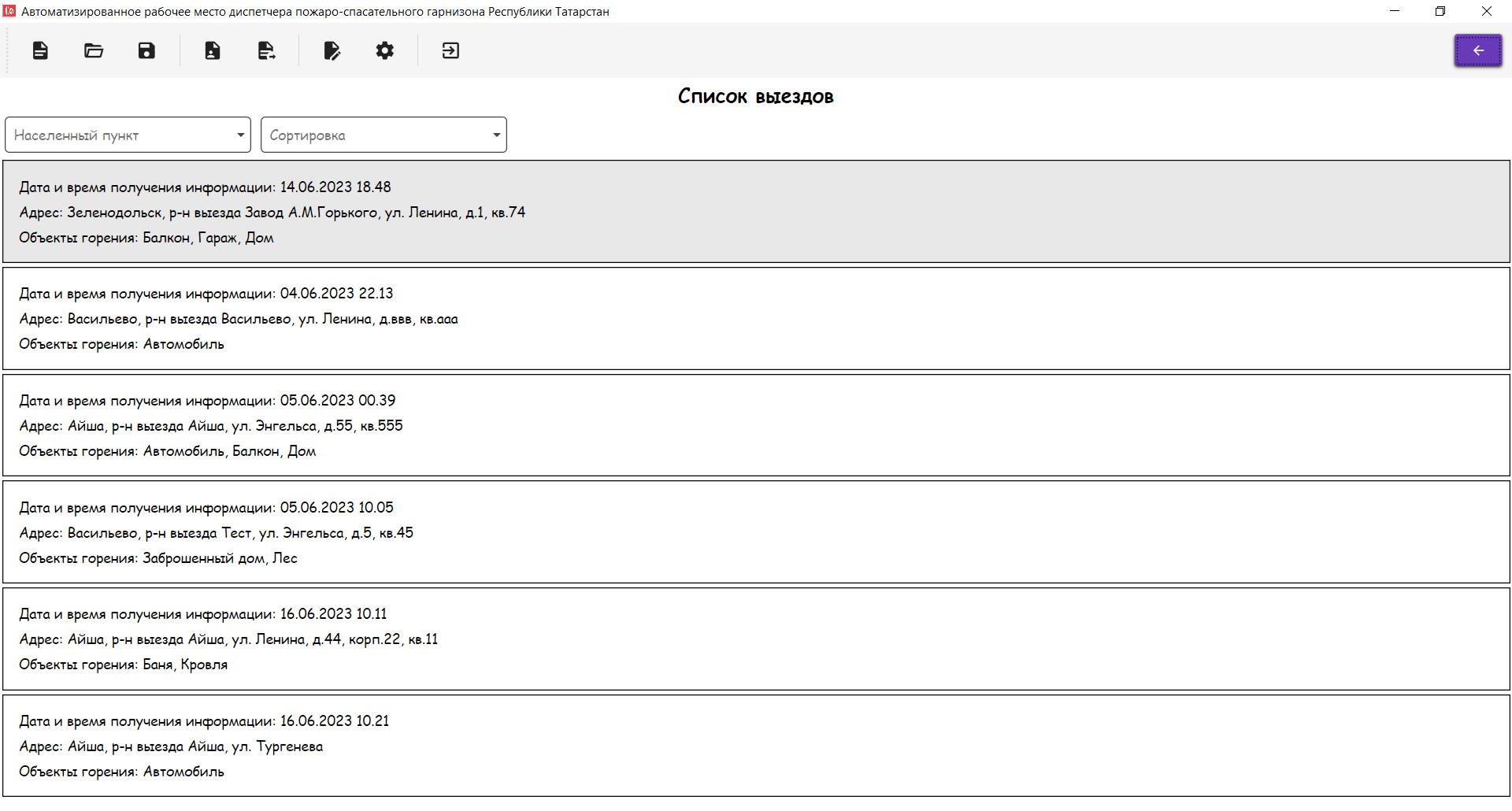


Рис.55 Страница Список выездов

Форма «Информация с места пожара» **(**Рис.56**)**. Данная форма предназначена для добавления, редактирования информации с места пожара

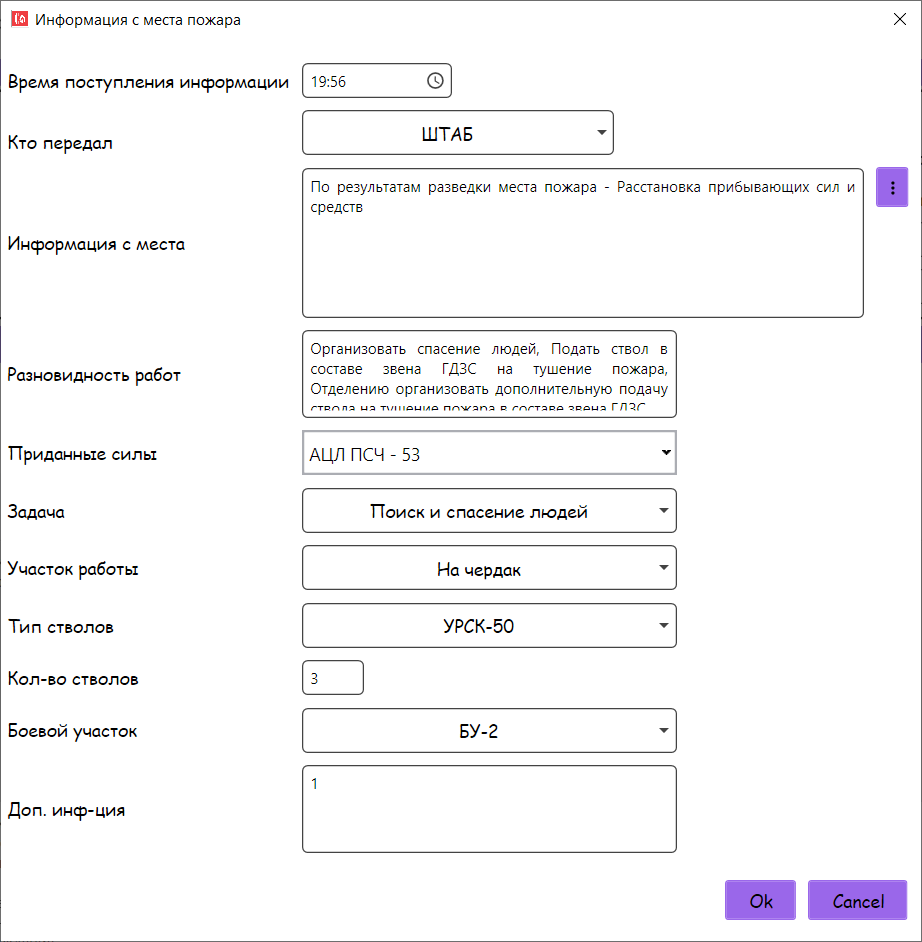


Рис.56 Форма Информация с места пожара

Форма «Информация» **(**Рис.57**)**. Данная форма предназначена для формирования информации с места пожара, а также для добавления информации о разновидностях работ.

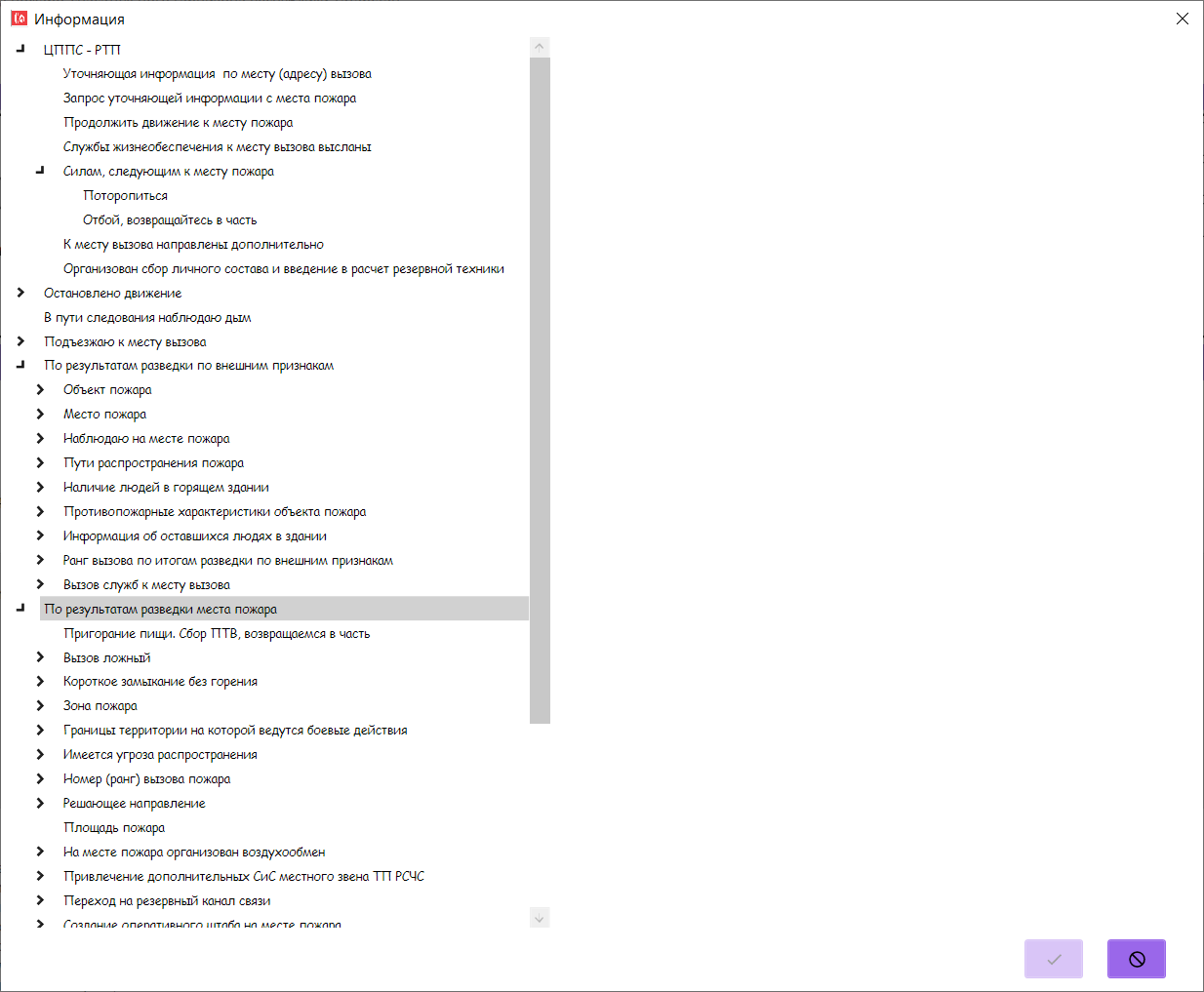


Рис.57 Форма Информация

Форма «Информация» о причине пожара **(**Рис.58**)**. Данная форма предназначена для формирования информации о причине пожара:

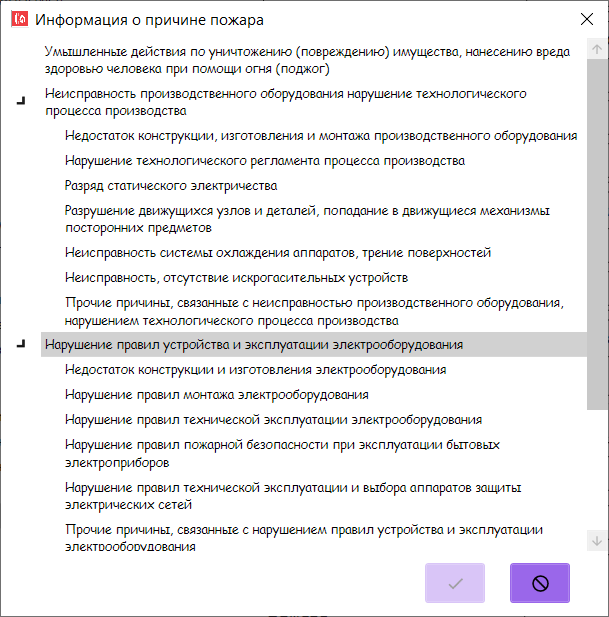


Рис.58 Форма Информация о причине пожара

Форма «Справочники» **(**Рис.59**)**. Данная форма предназначена для перехода на другие формы приложения

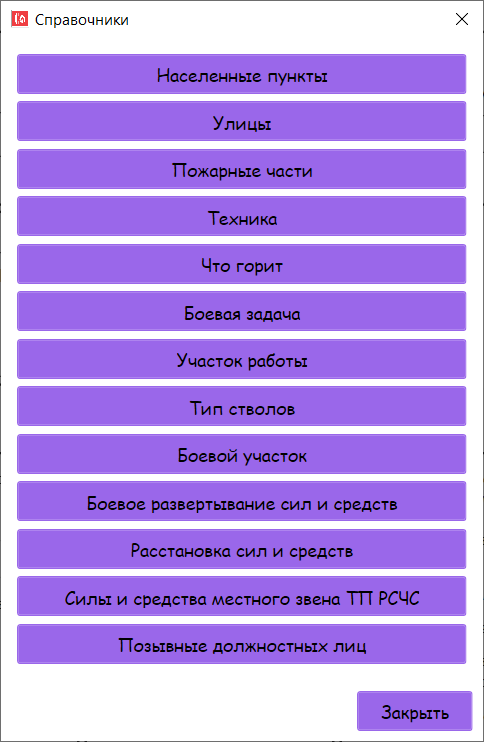


Рис.59 Форма Справочники

Форма «Справочник» **(**Рис.60**)**. Данная форма предназначена для отображения, добавления, редактирования населенных пунктов, районов выездов, а также расписания выездов.

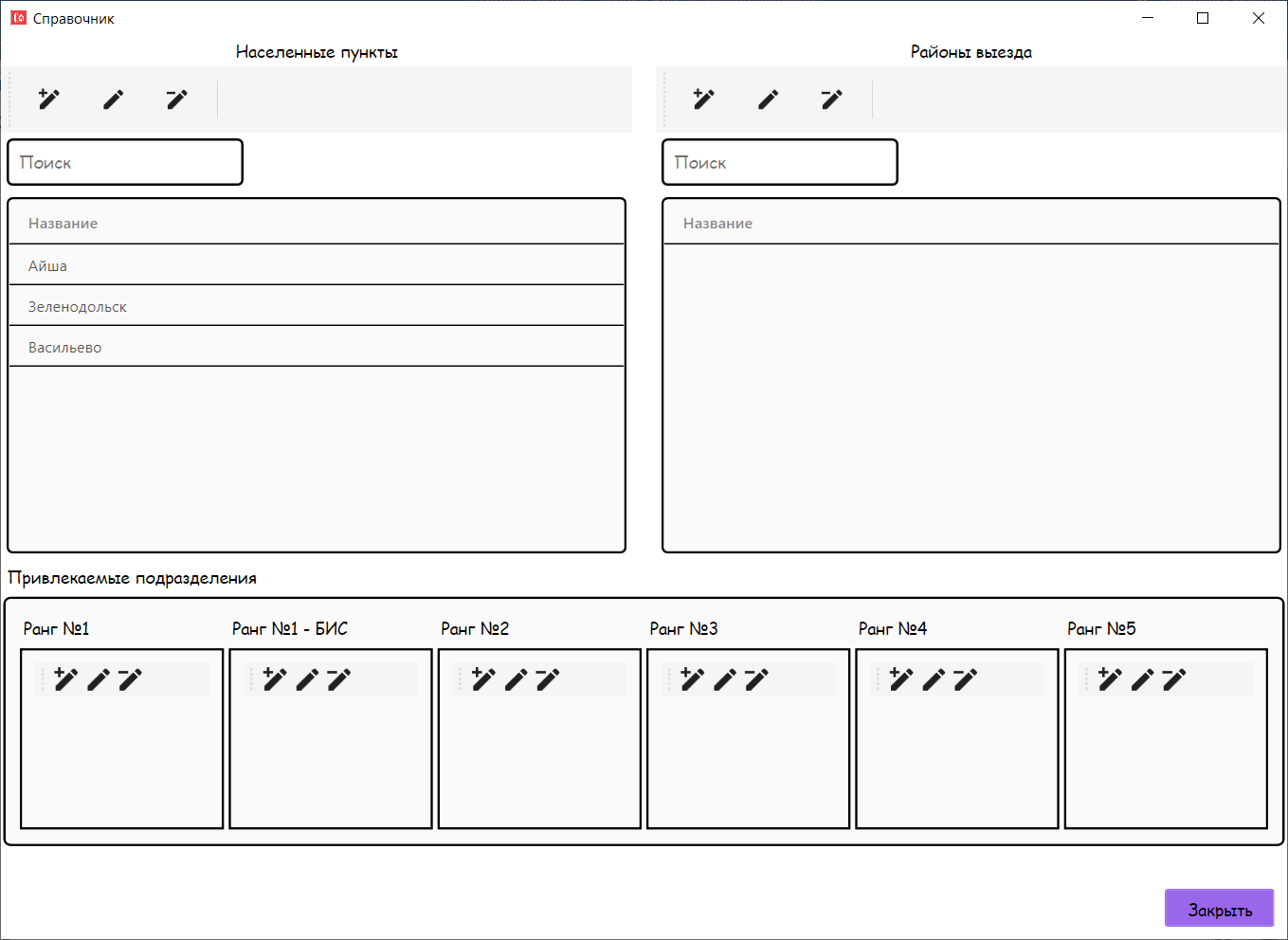


Рис.60 Форма Справочник

«Добавление/Редактирование населенного пункта»: эта форма предназначена для добавления и редактирования населенного пункта (Рис.61)

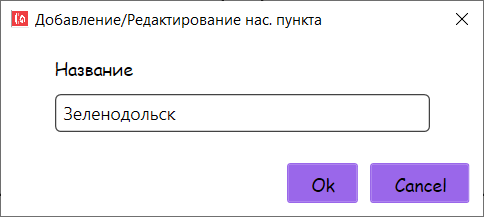


Рис.61 Добавление/Редактирование населенного пункта

Форма «Улицы» **(**Рис.62**)**. Данная форма предназначена для просмотра и удаления улиц.

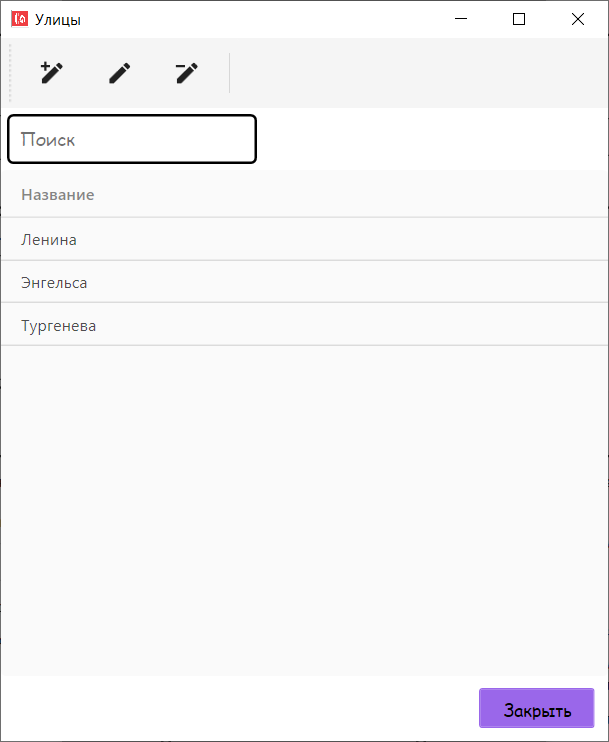


Рис.62 Форма Улицы

«Добавление/Редактирование улицы»: эта форма предназначена для добавления и редактирования улицы (Рис.63)

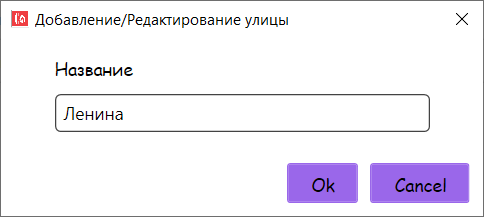


Рис.63 Добавление/Редактирование улицы

Форма «Пожарные части» **(**Рис.64**)**. Данная форма предназначена для просмотра и удаления пожарных частей.

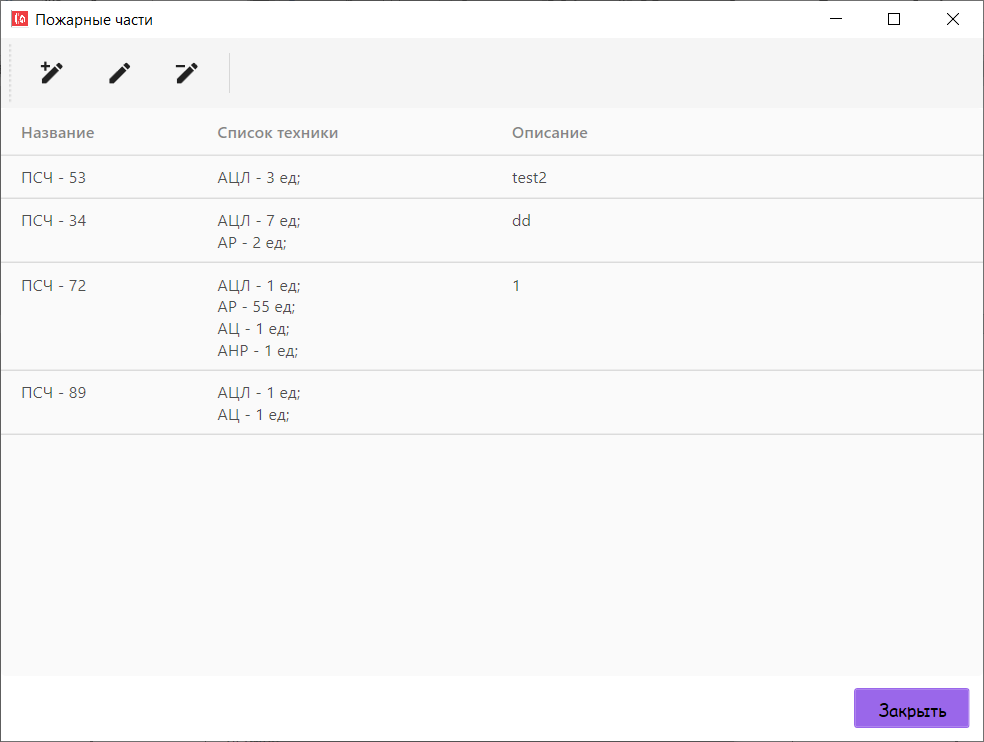


Рис.64 Форма Пожарные части

«Добавление/Редактирование пожарной части»: эта форма предназначена для добавления и редактирования пожарной части (Рис.65)

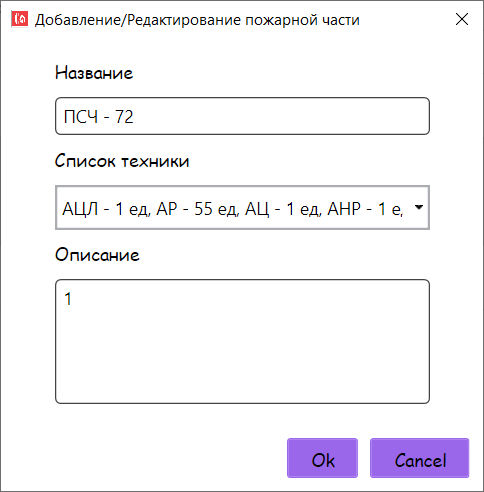


Рис.65 Добавление/Редактирование пожарной части

Форма «Техника» **(**Рис.66**)**. Данная форма предназначена для просмотра и удаления техники.

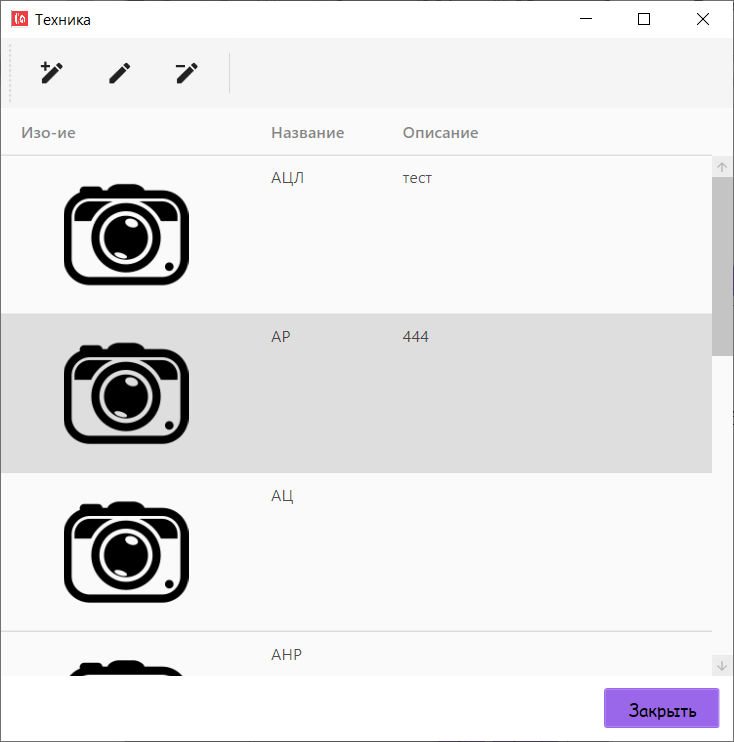


Рис.66 Форма Техника

«Добавление/Редактирование техники»: эта форма предназначена для добавления и редактирования техники (Рис.67)

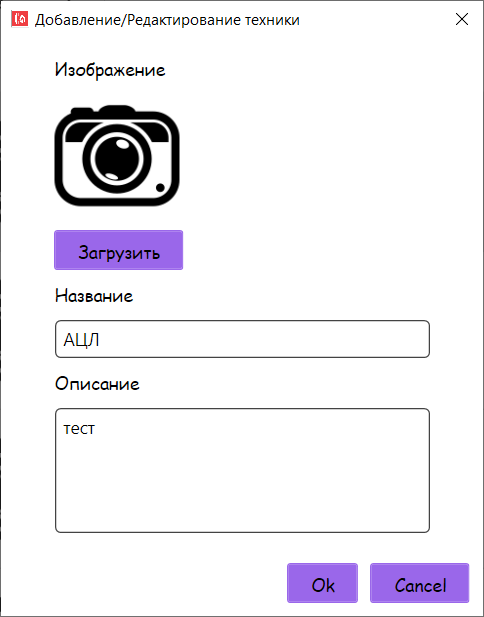


Рис.67 Добавление/Редактирование техники

Работа с FireDepartmentApp

После запуска программы Вы перейдете на главную форму программы. В форме справочники заполняем все необходимые для работы пункты.

После этого кликаем по кнопке «Ок».

Если Вы желаете удалить любую введенную ранее информацию, Вам необходимо кликнуть по нужной кнопке.

Для того чтобы изменить запись, введите в редактируемые поля данные и нажмите кнопку Ок. Данные формы редактированию и управлению со стороны пользователя не подлежат.

После того, как Вы закончите сеанс работы с программой, если Вы хотите выйти из FireDepartmentApp кликните на соответствующей кнопке «Выход».

Основные кнопки FireDepartmentApp

Кнопки управления «Добавить запись», «Удалить запись», «Сохранить» расположены на каждой конкретной форме в базе данных.

Действия этих кнопок приведены в таблице 1:

Таблица 1 – Действие кнопок

|  |  |
| --- | --- |
| Добавить запись | После нажатия на эту кнопку выйдет вспомогательная форма для добавления записей |
| Удалить запись | Кнопка «Удалить запись» позволяет удалить выделенную запись. |
| Сохранить данные | Кнопка «Сохранить данные» добавляет запись введенную пользователем в БД, либо обновляет существующую. |
| Изменить | Открывает форму в режиме редактирования информации |
|  | Позволяет Загрузить файл |
|  | Переход на предыдущую страницу |

**4.4 Сообщения пользователю**

Для уверенной и комфортной работы пользователю важно понимать поведение программы, а программе, в свою очередь, поведение пользователя. Для информирования и уточнения намерений пользователей в программах используются сообщения.

При попытке удалить запись, пользователю выводится сообщение с запросом на подтверждение или отмену действия. (Рис.68)

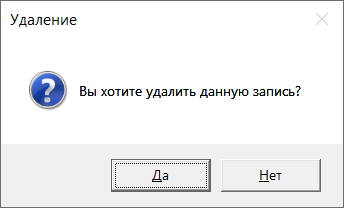


Рис.68 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.69)

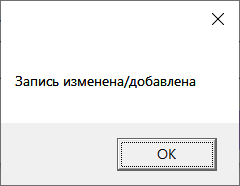


Рис.69 Сообщение Запись изменена/добавлена

После обновления записи, пользователю выводится сообщение. (Рис.70)

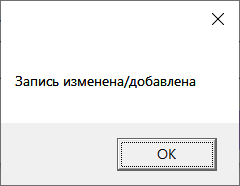


Рис.70 Сообщение Запись изменена/добавлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.71)

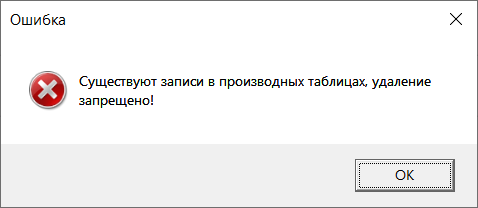


Рис.71 Сообщение Ошибка удаления

**5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Определение верхней границы количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.72 Потоковый граф

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=34, N=25. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=34-25+2=11.

Для данного программного средства необходимо разработать не менее 11 тестовых сценариев.

Чтобы определить количество тестовых прогонов для программного продукта FireDepartmentApp необходимо выделить требования. Из требований к приложению выделим поддержку 2-х операционных систем с одним основным языком локализации и выполнения 6-ти основных функций. Кроме того, приложение позволяет выполнять 5 функций по взаимодействию с системой (запуск приложения, выход из приложения, сохранение результатов в файл, работа с буфером и т.п.).

Таким образом, полное покрытие требований задаёт набор из 2\*1\*(6+5) =22 тестовых прогонов.

**5.2 Разработка тестовых вариантов**

Таблица 2 – Тестирование корректного добавления информации в Улицы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Улицы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Улицы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = Тест | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Улицы открыта | | Отображается форма Улицы |  |

Таблица 3 – Тестирование некорректного добавления информации в таблицу Улицы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Улицы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Улицы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Сообщение “Введите название улицы” | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Улицы открыта | | Отображается форма Улицы |  |

Таблица 4 – Тестирование корректного редактирования информации в таблице Улицы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Улицы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Улицы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = Название1 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Улицы  открыта | | Отображается форма Улицы |  |

Таблица 5 – Тестирование некорректного редактирования информации в таблице Улицы

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Улицы | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Улицы | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Сообщение “Введите название улицы ” | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Улицы открыта | | Отображается форма Улицы |  |

Таблица 6 – Тестирование корректного добавления информации в таблицу Пожарные части

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Пожарные части | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Пожарные части | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = ПСЧ-53  Список техники = АЦЛ – 7 ед  Описание = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Пожарные части открыта | | Отображается Пожарные части |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Пожарные части

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Пожарные части | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Пожарные части | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название =  Список техники = АЦЛ – 7 ед  Описание = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Сообщение “Введите название пожарной части” | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Пожарные части  открыта | | Отображается форма Пожарные части |  |

Таблица 8 – Тестирование корректного редактирования информации в таблице Пожарные части

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Пожарные части | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Пожарные части | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = ПСЧ-532  Список техники = АЦЛ – 7 ед  Описание = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Пожарные части открыта | | Отображается форма Пожарные части |  |

Таблица 9 – Тестирование некорректного редактирования информации в таблице Пожарные части

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Пожарные части | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Пожарные части | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название =  Список техники = АЦЛ – 7 ед  Описание = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Сообщение “Введите название пожарной части” | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Пожарные части открыта | | Отображается форма Пожарные части |  |

Таблица 10 – Тестирование корректного добавления информации в Типы стволов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Типы стволов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Типы стволов | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = Тест1 | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Типы стволов открыта | | Отображается форма Типы стволов |  |

Таблица 11 – Тестирование некорректного добавления информации в таблицу Типы стволов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Типы стволов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Типы стволов | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Название = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Сообщение “Введите тип ствола” | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Типы стволов открыта | | Отображается форма Типы стволов |  |

Таблица 12 – Тестирование корректного редактирования информации в таблице Типы стволов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Типы стволов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Типы стволов | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = Тест | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Типы стволов открыта | | Отображается форма Типы стволов |  |

Таблица 13 – Тестирование некорректного редактирования информации в таблице Типы стволов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Типы стволов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Типы стволов | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные,  Название = | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Сообщение “Введите тип ствола” | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Типы стволов открыта | | Отображается форма Типы стволов |  |

**5.3 Составление отчета о тестировании**

Для форм проекта были созданы и проведены тесты, результаты сведены в таблицу:

Таблица 14 – Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Прохождение | | |
| Форма | Число ТП | пройдено | | не пройдено |
| Главная | 4 | 4 | | 0 |
| Квест | 4 | 4 | | 0 |
| Категории | 4 | 4 | | 0 |
| Расписание | 4 | 4 | | 0 |
| Бронь | 4 | 4 | | 0 |
| Отзывы | 4 | 4 | | 0 |
| Организаторы | 4 | 4 | | 0 |
| Профиль | 4 | 4 | | 0 |
| ИТОГО | 32 | 36 | 0 | |

По результатам тестирования были выявлены и устранены следующие проблемы:

При выполнении теста таблица №9 Тестирование формы «Пожарные части», Некорректный запрос

**Ошибка**

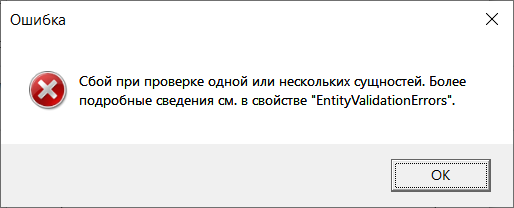


Рис.73 Ошибка теста №8

**Причина**

Поле название ПСЧ обязательно для заполнения, название ПСЧ не было введено.

**Принятые меры**

Ошибка устранена. Добавлена проверка заполненности поля.

**ЗАКЛЮЧЕНИЕ**

**Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система – Windows 10 Professional.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 4 Gb

Таблица 15 – Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 16 – Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |
| *Тестируемость:*  *-трудоемкость тестирования изменений;*  *- длительность тестирования изменений.* | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

Таблица 17 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 17 | x1 | x2 | **x3** | =51 |
| Отчет | 3 | x2 | x5 | **x8** | =24 |

Таблица 18 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество клиентских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | **Всего >8** |
|  | (<2 срв, <3 клт) | (2-3 срв, 3-5 клт) | (>3 срв, >5 клт) |
| <3 | Простой | Простой | Средний |
| 3-7 | Простой | Средний | Сложный |
| **>8** | Средний | Сложный | **Сложный** |

Таблица 19 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество Возвратских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | Всего >8 |
|  | (<2 срв, <3 клт) | (2-3 срв, 3-5 клт) | **(>3 срв, >5 клт)** |
| 0 или 1 | Простой | Простой | Средний |
| **2 или 3** | Простой | Средний | **Сложный** |
| >4 | Средний | Сложный | Сложный |

Производительность = 6000 тысяч строк кода/1 месяца=6000 тысяч строк кода/месяц

Качество = 30 ошибок /6 тысяч строк кода=5 ошибок/тысяч строк кода

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Г. Шилдт C#: Учебный курс. - Спб.: Питер, 2021. - 512с

2. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2018. - 684 с.

3. Ватсон К. Си Шарп/К.Ватсон.-М.:Лори, 2021.-862 с

4. Балена Ф., Димауро Д. Современная практика программирования на Visual Basic и Visual C#.-М.:Русская редакция, 2021,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС, 2021,464с.,СД

6. Лабор В.В. Си Шарп: СоВозврат приложений для Windows/-Мн.: Харвест, 2021-384с.

7. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2021-512с.

8. ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению

9. ГОСТ 19.102-77.ЕСПД. Стадии разработки

10. ГОСТ 19.505-79.ЕСПД. Руководство оператора. Требования к содержанию и оформлению

11. ГОСТ 19.002-80.ЕСПД. Схемы алгоритмов и программ. Правила выполнения

12. ГОСТ 19.404-79.ЕСПД. Пояснительная записка. Требования к содержанию и оформлению

13. Владимир Биллинг. Основы программирования на С#. 2018 г.

[Электронный ресурс http://mexalib.com/view/42671]

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2019. 2019 г.

[Электронный ресурс http://www.litres.ru/nikita-kultin/osnovy-programmirovaniya-v-microsoft-visual-c-2010-2]

15. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2021.

[Электронный ресурс http://www.twirpx.com/file/27852]

16. Йен Гриффитс. Программирование на С#. 2020г.

[Электронный ресурс http://rutracker.org/forum/viewtopic.php?t=4772853]

**Приложение А**

Автоматизированная информационная система

«Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан»

Текст кода программного средства

460.ЗМК.00015-01 12 01

**Текст кода программного средства**

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Automation.Peers;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using FireDepartmentApp.Entities;

using FireDepartmentApp.Windows.AddOrEditWindow;

using Microsoft.Win32;

using System.Xml.Serialization;

using System.IO;

using System.Management.Instrumentation;

namespace FireDepartmentApp.Pages

{

/// <summary>

/// Логика взаимодействия для AddDeparturePage.xaml

/// </summary>

public partial class AddDeparturePage : Page

{

// текущий вызов

public Departure \_currentDeparture = new Departure();

// выбранный район вызова

DepartureArea selectedDepartureArea = new DepartureArea();

List<BurnType> selectedBurns = new List<BurnType>();

List<ServiceType> selectedServices = new List<ServiceType>();

List<PositionType> selectedPositions = new List<PositionType>();

public AddDeparturePage(Departure departure)

{

InitializeComponent();

if (departure != null)

{

\_currentDeparture = departure;

\_currentDeparture.Settlement = departure.DepartureArea.Settlement;

//DateDepart.SelectedDate = TimeDepart.SelectedTime = departure.DateTimeDepart;

}

else

{

\_currentDeparture.DateTimeDepart = DateTime.Now;

}

DataContext = \_currentDeparture;

}

private void ComboBoxSettlement\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ColorZoneDateTime.IsEnabled = true;

ComboBoxDepArea.IsEnabled = true;

int settlementId = ((int)((ComboBoxSettlement.SelectedItem as Settlement)?.SettlementId));

ComboBoxDepArea.ItemsSource = FireDepartEntities.GetContext().DepartureAreas.Where(p => p.SettlementId == settlementId).ToList();

if (\_currentDeparture.DepartureId == 0)

{

\_currentDeparture.DateTimeDepart = DateTime.Now;

TimeDepart.SelectedTime = DateTime.Now;

DateDepart.SelectedDate = DateTime.Now;

}

}

private void ComboBoxDepArea\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ComboBoxDepArea.SelectedItem is DepartureArea departureArea)

{

selectedDepartureArea = departureArea;

LoadSubdivisionInfo();

//LoadUpdatedInfo();

//if (\_currentDeparture.DepartureId != 0)

//{

//}

//else

//{

// // 1 ранг пожара

// ListViewRank.ItemsSource = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 1 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

// // 1 Бис ранг пожара

// ListViewRank1.ItemsSource = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 2 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

// // 2 ранг пожара

// ListViewRank2.ItemsSource = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 3 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

// // 3 ранг пожара

// ListViewRank3.ItemsSource = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 4 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

// // 4 ранг пожара

// ListViewRank4.ItemsSource = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 5 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

// // 5 ранг пожара

// ListViewRank5.ItemsSource = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 6 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

//}

ColorZoneRank.Visibility = ListViewRank.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

ColorZoneRank1.Visibility = ListViewRank1.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

ColorZoneRank2.Visibility = ListViewRank2.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

ColorZoneRank3.Visibility = ListViewRank3.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

ColorZoneRank4.Visibility = ListViewRank4.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

ColorZoneRank5.Visibility = ListViewRank5.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

BtnAddFireInfo.Visibility = Visibility.Visible;

// иные службы

ListViewServiceTypes.ItemsSource = FireDepartEntities.GetContext().ServiceTypes.ToList();

ColorZoneServiceType.Visibility = ListViewServiceTypes.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

// должностные лица

ListViewPositionTypes.ItemsSource = FireDepartEntities.GetContext().PositionTypes.ToList();

ColorZonePositionType.Visibility = ListViewPositionTypes.Items.Count > 0 ? Visibility.Visible : Visibility.Collapsed;

// привлеченная техника

ListViewDepTechnics.ItemsSource = FireDepartEntities.GetContext().Technics.ToList();

// общая информация

ColorZoneGeneralInfo.Visibility = Visibility.Visible;

}

else

{

ColorZoneRank.Visibility = Visibility.Collapsed;

ColorZoneRank1.Visibility = Visibility.Collapsed;

ColorZoneRank2.Visibility = Visibility.Collapsed;

ColorZoneRank3.Visibility = Visibility.Collapsed;

ColorZoneRank4.Visibility = Visibility.Collapsed;

ColorZoneRank5.Visibility = Visibility.Collapsed;

ColorZoneServiceType.Visibility = Visibility.Collapsed;

ColorZonePositionType.Visibility = Visibility.Collapsed;

ColorZoneGeneralInfo.Visibility = Visibility.Collapsed;

}

}

public void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

try

{

Manager.IsOpenAddPage = true;

ComboBoxSettlement.ItemsSource = FireDepartEntities.GetContext().Settlements.OrderBy(p => p.SettlementName).ToList();

ComboBoxStreet.ItemsSource = FireDepartEntities.GetContext().Streets.OrderBy(p => p.StreetName).ToList();

Manager.AddDeparturePage = this;

if (Manager.IsDirectory)

{

Manager.IsDirectory = false;

ComboBoxBurns.ItemsSource = FireDepartEntities.GetContext().BurnTypes.ToList();

ListViewServiceTypes.ItemsSource = FireDepartEntities.GetContext().ServiceTypes.ToList();

ListViewPositionTypes.ItemsSource = FireDepartEntities.GetContext().PositionTypes.ToList();

}

else

{

// очистка данных привлеченной техники

FireDepartEntities.GetContext().Technics.ForEachAsync(p => p.TechnicCountGeneral = 0).Wait();

// очистка данных всей привлеченной техники и л.с.

ServiceType.TechnicCountGeneral = 0;

ServiceType.PeopleCountGeneral = 0;

Technic.AllTechnicCount = 0;

Subdivision.PeopleCountGeneral = 0;

PositionType.PeopleCountGeneral = 0;

TbGeneralCountPeople.Text = "";

TbGeneralCountTechnic.Text = "";

TbMCSCountPeople.Text = "";

TbMCSCountTechnic.Text = "";

// очистка данных от прошлого района выезда

if (\_currentDeparture.DepartureId == 0)

{

FireDepartEntities.GetContext().BurnTypes.ForEachAsync(p => p.BurnSelected = false).Wait();

ComboBoxBurns.ItemsSource = FireDepartEntities.GetContext().BurnTypes.ToList();

FireDepartEntities.GetContext().ServiceTypes.ForEachAsync(p =>

{

p.DepartureChecked = false;

p.ArrivalChecked = false;

p.TechnicCount = 0;

p.PeopleCount = 0;

}).Wait();

ListViewServiceTypes.ItemsSource = FireDepartEntities.GetContext().ServiceTypes.ToList();

FireDepartEntities.GetContext().PositionTypes.ForEachAsync(p =>

{

p.ArrivalChecked = false;

}).Wait();

ListViewPositionTypes.ItemsSource = FireDepartEntities.GetContext().PositionTypes.ToList();

FireDepartEntities.GetContext().Technics.ForEachAsync(p => p.TechnicCountGeneral = 0).Wait();

ListViewDepTechnics.Items.Refresh();

}

LoadInfo();

LoadSubdivisionInfo();

LoadUpdatedInfo();

}

}

catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); }

}

private void CheckBoxArrival\_Checked(object sender, RoutedEventArgs e)

{

Subdivision subdivision = ((CheckBox)sender).DataContext as Subdivision;

subdivision.TimeArrival = DateTime.Now;

switch (subdivision.FireRank)

{

case 1:

ListViewRank.Items.Refresh();

break;

case 2:

ListViewRank1.Items.Refresh();

break;

case 3:

ListViewRank2.Items.Refresh();

break;

case 4:

ListViewRank3.Items.Refresh();

break;

case 5:

ListViewRank4.Items.Refresh();

break;

case 6:

ListViewRank5.Items.Refresh();

break;

default:

break;

}

subdivision.Technic.TechnicCountGeneral += 1;

Technic.AllTechnicCount += 1;

TbMCSCountTechnic.Text = Technic.AllTechnicCount.ToString();

TbGeneralCountTechnic.Text = (ServiceType.TechnicCountGeneral + Technic.AllTechnicCount).ToString();

ListViewDepTechnics.Items.Refresh();

}

private void CheckBoxArrival\_Unchecked(object sender, RoutedEventArgs e)

{

Subdivision subdivision = ((CheckBox)sender).DataContext as Subdivision;

subdivision.Technic.TechnicCountGeneral -= 1;

Technic.AllTechnicCount -= 1;

TbMCSCountTechnic.Text = Technic.AllTechnicCount.ToString();

TbGeneralCountTechnic.Text = (ServiceType.TechnicCountGeneral + Technic.AllTechnicCount).ToString();

ListViewDepTechnics.Items.Refresh();

}

private void CheckBoxDepart\_Checked(object sender, RoutedEventArgs e)

{

Subdivision subdivision = ((CheckBox)sender).DataContext as Subdivision;

subdivision.TimeDeparture = DateTime.Now;

switch (subdivision.FireRank)

{

case 1:

ListViewRank.Items.Refresh();

break;

case 2:

ListViewRank1.Items.Refresh();

break;

case 3:

ListViewRank2.Items.Refresh();

break;

case 4:

ListViewRank3.Items.Refresh();

break;

case 5:

ListViewRank4.Items.Refresh();

break;

case 6:

ListViewRank5.Items.Refresh();

break;

default:

break;

}

}

private void TbPeopleCount\_TextChanged(object sender, TextChangedEventArgs e)

{

TextBox textBox = (TextBox)sender;

string peopleText = textBox.Text.Replace(" ", "").Replace("-", "");

if (peopleText != null && peopleText.Length > 0)

{

short peopleCount = short.Parse(peopleText);

if (textBox.DataContext is Subdivision subdivision)

{

Subdivision.PeopleCountGeneral = (Subdivision.PeopleCountGeneral - subdivision.PeopleCount + peopleCount);

subdivision.PeopleCount = peopleCount;

TbMCSCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral).ToString();

TbGeneralCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral + ServiceType.PeopleCountGeneral).ToString();

}

else if (textBox.DataContext is ServiceType serviceType)

{

ServiceType.PeopleCountGeneral = ServiceType.PeopleCountGeneral - serviceType.PeopleCount + peopleCount;

serviceType.PeopleCount = peopleCount;

TbGeneralCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral + ServiceType.PeopleCountGeneral).ToString();

}

}

}

private void TbPeopleCount\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

e.Handled = !char.IsNumber(e.Text, 0);

//if (!e.Handled)

//{

// TextBox textBox = (TextBox)sender;

// string peopleText = e.Text.Replace(" ", "").Replace("-", "");

// if (peopleText != null && peopleText.Length > 0)

// {

// short peopleCount = short.Parse(peopleText);

// if (textBox.DataContext is Subdivision subdivision)

// {

// Subdivision.PeopleCountGeneral = (Subdivision.PeopleCountGeneral - subdivision.PeopleCount + peopleCount);

// subdivision.PeopleCount = peopleCount;

// TbMCSCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral).ToString();

// TbGeneralCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral + ServiceType.PeopleCountGeneral).ToString();

// }

// else if (textBox.DataContext is ServiceType serviceType)

// {

// ServiceType.PeopleCountGeneral = ServiceType.PeopleCountGeneral - serviceType.PeopleCount + peopleCount;

// serviceType.PeopleCount = peopleCount;

// TbGeneralCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral + ServiceType.PeopleCountGeneral).ToString();

// }

// }

//}

}

private void TbTechnicCount\_TextChanged(object sender, TextChangedEventArgs e)

{

TextBox textBox = (TextBox)sender;

string technicText = textBox.Text.Replace(" ", "").Replace("-", "");

if (technicText != null && technicText.Length > 0)

{

short technicCount = short.Parse(technicText);

if (textBox.DataContext is ServiceType serviceType)

{

ServiceType.TechnicCountGeneral = ServiceType.TechnicCountGeneral - serviceType.TechnicCount + technicCount;

serviceType.TechnicCount = technicCount;

TbGeneralCountTechnic.Text = (ServiceType.TechnicCountGeneral + Technic.AllTechnicCount).ToString();

}

}

}

private void TbTechnicCount\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

e.Handled = !char.IsNumber(e.Text, 0);

//if (!e.Handled)

//{

// TextBox textBox = (TextBox)sender;

// string technicText = e.Text.Replace(" ", "").Replace("-", "");

// if (technicText != null && technicText.Length > 0)

// {

// short technicCount = short.Parse(technicText);

// if (textBox.DataContext is ServiceType serviceType)

// {

// ServiceType.TechnicCountGeneral = ServiceType.TechnicCountGeneral - serviceType.TechnicCount + technicCount;

// serviceType.TechnicCount = technicCount;

// TbGeneralCountTechnic.Text = (ServiceType.TechnicCountGeneral + Technic.AllTechnicCount).ToString();

// }

// }

//}

}

private void CheckBoxDepartService\_Checked(object sender, RoutedEventArgs e)

{

ServiceType serviceType = ((CheckBox)sender).DataContext as ServiceType;

serviceType.TimeDeparture = DateTime.Now;

ListViewServiceTypes.Items.Refresh();

}

private void CheckBoxArrivalService\_Checked(object sender, RoutedEventArgs e)

{

ServiceType serviceType = ((CheckBox)sender).DataContext as ServiceType;

serviceType.TimeArrival = DateTime.Now;

ListViewServiceTypes.Items.Refresh();

}

private void CheckBoxArrivalPosition\_Checked(object sender, RoutedEventArgs e)

{

PositionType positionType = ((CheckBox)sender).DataContext as PositionType;

positionType.TimeArrival = DateTime.Now;

PositionType.PeopleCountGeneral += 1;

TbGeneralCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral + ServiceType.PeopleCountGeneral).ToString();

TbMCSCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral).ToString();

ListViewPositionTypes.Items.Refresh();

}

private void CheckBoxArrivalPosition\_Unchecked(object sender, RoutedEventArgs e)

{

PositionType positionType = ((CheckBox)sender).DataContext as PositionType;

PositionType.PeopleCountGeneral -= 1;

TbGeneralCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral + ServiceType.PeopleCountGeneral).ToString();

TbMCSCountPeople.Text = (PositionType.PeopleCountGeneral + Subdivision.PeopleCountGeneral).ToString();

}

private void CheckBoxFireEnd\_Checked(object sender, RoutedEventArgs e)

{

DatePickerFireEnd.SelectedDate = TimePickerFireEnd.SelectedTime = DateTime.Now;

}

private void CheckBoxOpenFire\_Checked(object sender, RoutedEventArgs e)

{

DatePickerOpenFire.SelectedDate = TimePickerOpenFire.SelectedTime = DateTime.Now;

}

private void CheckBoxAftermathFire\_Checked(object sender, RoutedEventArgs e)

{

DatePickerAftermathFire.SelectedDate = TimePickerAftermathFire.SelectedTime = DateTime.Now;

}

public void SaveInfo()

{

try

{

if (ComboBoxSettlement.SelectedItem == null)

throw new Exception("Выберите населенный пункт");

if (ComboBoxDepArea.SelectedItem == null)

throw new Exception("Выберите район выезда");

if (ComboBoxStreet.SelectedItem == null)

throw new Exception("Выберите улицу");

// список объектов горения

List<BurnType> burnTypes = ComboBoxBurns.Items.Cast<BurnType>().Where(p => p.BurnSelected).ToList();

SaveUpdatedInfo();

// прибывшие подразделения 1 ранга пожара

List<Subdivision> subdivisionsRank = ListViewRank.Items.Cast<Subdivision>().Where(p => p.DepartureChecked && p.ArrivalChecked).ToList();

// прибывшие подразделения 1 БИС ранга пожара

List<Subdivision> subdivisionsRank1 = ListViewRank1.Items.Cast<Subdivision>().Where(p => p.DepartureChecked && p.ArrivalChecked).ToList();

// прибывшие подразделения 2 ранга пожара

List<Subdivision> subdivisionsRank2 = ListViewRank2.Items.Cast<Subdivision>().Where(p => p.DepartureChecked && p.ArrivalChecked).ToList();

// прибывшие подразделения 3 ранга пожара

List<Subdivision> subdivisionsRank3 = ListViewRank3.Items.Cast<Subdivision>().Where(p => p.DepartureChecked && p.ArrivalChecked).ToList();

// прибывшие подразделения 4 ранга пожара

List<Subdivision> subdivisionsRank4 = ListViewRank4.Items.Cast<Subdivision>().Where(p => p.DepartureChecked && p.ArrivalChecked).ToList();

// прибывшие подразделения 5 ранга пожара

List<Subdivision> subdivisionsRank5 = ListViewRank5.Items.Cast<Subdivision>().Where(p => p.DepartureChecked && p.ArrivalChecked).ToList();

// прибывшие подразделения общее

List<Subdivision> generalSubdivisions = new List<Subdivision>();

generalSubdivisions.AddRange(subdivisionsRank);

generalSubdivisions.AddRange(subdivisionsRank1);

generalSubdivisions.AddRange(subdivisionsRank2);

generalSubdivisions.AddRange(subdivisionsRank3);

generalSubdivisions.AddRange(subdivisionsRank4);

generalSubdivisions.AddRange(subdivisionsRank5);

// прибывшие иные службы

List<ServiceType> serviceTypes = ListViewServiceTypes.Items.Cast<ServiceType>().Where(p => p.DepartureChecked && p.ArrivalChecked).ToList();

// прибывшие должностные лица

List<PositionType> positionTypes = ListViewPositionTypes.Items.Cast<PositionType>().Where(p => p.ArrivalChecked).ToList();

if (\_currentDeparture.DepartureId == 0)

FireDepartEntities.GetContext().Departures.Add(\_currentDeparture);

FireDepartEntities.GetContext().SaveChanges();

// обновление, добавление данных в производных таблицах

FireDepartEntities.GetContext().DepartureBurns.RemoveRange(\_currentDeparture.DepartureBurns);

FireDepartEntities.GetContext().DepartureSubdivisions.RemoveRange(\_currentDeparture.DepartureSubdivisions);

FireDepartEntities.GetContext().DepartureServices.RemoveRange(\_currentDeparture.DepartureServices);

FireDepartEntities.GetContext().DeparturePositions.RemoveRange(\_currentDeparture.DeparturePositions);

foreach (var i in burnTypes)

FireDepartEntities.GetContext().DepartureBurns.Add(new DepartureBurn { BurnTypeId = i.BurnTypeId, DepartureId = \_currentDeparture.DepartureId });

foreach (var i in generalSubdivisions)

FireDepartEntities.GetContext().DepartureSubdivisions.Add(new DepartureSubdivision

{

SubdivisionId = i.SubdivisionId,

DepartureId = \_currentDeparture.DepartureId,

DateTimeDeparture = i.TimeDeparture,

DateTimeArrival = i.TimeArrival,

CountPeople = i.PeopleCount

});

foreach (var i in serviceTypes)

FireDepartEntities.GetContext().DepartureServices.Add(new DepartureService

{

ServiceTypeId = i.ServiceTypeId,

DepartureId = \_currentDeparture.DepartureId,

DateTimeDeparture = i.TimeDeparture,

DateTimeArrival = i.TimeArrival,

CountTechnics = i.TechnicCount,

CountPeople = i.PeopleCount

});

foreach (var i in positionTypes)

FireDepartEntities.GetContext().DeparturePositions.Add(new DeparturePosition

{

PositionTypeId = i.PositionTypeId,

DepartureId = \_currentDeparture.DepartureId,

DateTimeArrival = i.TimeArrival

});

FireDepartEntities.GetContext().SaveChanges();

//SaveFileDialog saveFileDialog = new SaveFileDialog() { Filter = "Файлы данных|\*.xml" };

//if (saveFileDialog.ShowDialog() == true)

//{

// XmlSerializer xmlSerializer = new XmlSerializer(typeof(Departure));

// using (FileStream fs = new FileStream(saveFileDialog.FileName, FileMode.Create))

// {

// Departure departure = new Departure

// {

// DepartureId = \_currentDeparture.DepartureId,

// DateTimeDepart = \_currentDeparture.DateTimeDepart,

// NumHouse = \_currentDeparture.NumHouse

// };

// Departure departure2 = \_currentDeparture;

// xmlSerializer.Serialize(fs, departure2);

// }

//}

MessageBox.Show("Запись изменена/добавлена");

}

catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); }

}

private void LoadInfo()

{

// загрузка инфорации

//ComboBoxBurns.Items.Clear();

selectedBurns = FireDepartEntities.GetContext().BurnTypes.ToList();

foreach (var i in selectedBurns)

{

i.BurnSelected = false;

if (\_currentDeparture.DepartureId != 0)

{

DepartureBurn depBurn = \_currentDeparture.DepartureBurns.FirstOrDefault(p => p.BurnTypeId == i.BurnTypeId);

if (depBurn != null)

{

i.BurnSelected = true;

}

}

}

ComboBoxBurns.ItemsSource = selectedBurns;

selectedServices = FireDepartEntities.GetContext().ServiceTypes.ToList();

foreach (var i in selectedServices)

{

i.DepartureChecked = false;

i.ArrivalChecked = false;

i.PeopleCount = 0;

i.TechnicCount = 0;

if (\_currentDeparture.DepartureId != 0)

{

DepartureService depSerivce = \_currentDeparture.DepartureServices.FirstOrDefault(p => p.ServiceTypeId == i.ServiceTypeId);

if (depSerivce != null)

{

i.DepartureChecked = true;

i.TimeDeparture = depSerivce.DateTimeDeparture;

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depSerivce.DateTimeArrival;

i.PeopleCount = (short)depSerivce.CountPeople;

i.TechnicCount = (short)depSerivce.CountTechnics;

}

}

}

ListViewServiceTypes.ItemsSource = selectedServices;

selectedPositions = FireDepartEntities.GetContext().PositionTypes.ToList();

foreach (var i in selectedPositions)

{

i.ArrivalChecked = false;

if (\_currentDeparture.DepartureId != 0)

{

DeparturePosition depPosition = \_currentDeparture.DeparturePositions.FirstOrDefault(p => p.PositionTypeId == i.PositionTypeId);

if (depPosition != null)

{

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depPosition.DateTimeArrival;

}

}

}

ListViewPositionTypes.ItemsSource = selectedPositions;

}

private void LoadSubdivisionInfo()

{

if (ComboBoxDepArea.SelectedItem is DepartureArea departureArea)

{

// 1 ранг пожара

List<Subdivision> subdivisionRank = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 1 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

foreach (var i in subdivisionRank)

{

i.DepartureChecked = false;

i.ArrivalChecked = false;

i.PeopleCount = 0;

if (\_currentDeparture.DepartureId != 0)

{

DepartureSubdivision depSubdivision = \_currentDeparture.DepartureSubdivisions.FirstOrDefault(p => p.SubdivisionId == i.SubdivisionId);

if (depSubdivision != null)

{

i.DepartureChecked = true;

i.TimeDeparture = depSubdivision.DateTimeDeparture;

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depSubdivision.DateTimeArrival;

i.PeopleCount = (short)depSubdivision.CountPeople;

}

}

}

ListViewRank.ItemsSource = subdivisionRank;

// 1 Бис ранг пожара

List<Subdivision> subdivisionRank1 = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 2 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

foreach (var i in subdivisionRank1)

{

i.DepartureChecked = false;

i.ArrivalChecked = false;

i.PeopleCount = 0;

if (\_currentDeparture.DepartureId != 0)

{

DepartureSubdivision depSubdivision = \_currentDeparture.DepartureSubdivisions.FirstOrDefault(p => p.SubdivisionId == i.SubdivisionId);

if (depSubdivision != null)

{

i.DepartureChecked = true;

i.TimeDeparture = depSubdivision.DateTimeDeparture;

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depSubdivision.DateTimeArrival;

i.PeopleCount = (short)depSubdivision.CountPeople;

}

}

}

ListViewRank1.ItemsSource = subdivisionRank1;

// 2 ранг пожара

List<Subdivision> subdivisionRank2 = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 3 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

foreach (var i in subdivisionRank2)

{

i.DepartureChecked = false;

i.ArrivalChecked = false;

i.PeopleCount = 0;

if (\_currentDeparture.DepartureId != 0)

{

DepartureSubdivision depSubdivision = \_currentDeparture.DepartureSubdivisions.FirstOrDefault(p => p.SubdivisionId == i.SubdivisionId);

if (depSubdivision != null)

{

i.DepartureChecked = true;

i.TimeDeparture = depSubdivision.DateTimeDeparture;

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depSubdivision.DateTimeArrival;

i.PeopleCount = (short)depSubdivision.CountPeople;

}

}

}

ListViewRank2.ItemsSource = subdivisionRank2;

// 3 ранг пожара

List<Subdivision> subdivisionRank3 = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 4 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

foreach (var i in subdivisionRank3)

{

i.DepartureChecked = false;

i.ArrivalChecked = false;

i.PeopleCount = 0;

if (\_currentDeparture.DepartureId != 0)

{

DepartureSubdivision depSubdivision = \_currentDeparture.DepartureSubdivisions.FirstOrDefault(p => p.SubdivisionId == i.SubdivisionId);

if (depSubdivision != null)

{

i.DepartureChecked = true;

i.TimeDeparture = depSubdivision.DateTimeDeparture;

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depSubdivision.DateTimeArrival;

i.PeopleCount = (short)depSubdivision.CountPeople;

}

}

}

ListViewRank3.ItemsSource = subdivisionRank3;

// 4 ранг пожара

List<Subdivision> subdivisionRank4 = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 5 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

foreach (var i in subdivisionRank4)

{

i.DepartureChecked = false;

i.ArrivalChecked = false;

i.PeopleCount = 0;

if (\_currentDeparture.DepartureId != 0)

{

DepartureSubdivision depSubdivision = \_currentDeparture.DepartureSubdivisions.FirstOrDefault(p => p.SubdivisionId == i.SubdivisionId);

if (depSubdivision != null)

{

i.DepartureChecked = true;

i.TimeDeparture = depSubdivision.DateTimeDeparture;

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depSubdivision.DateTimeArrival;

i.PeopleCount = (short)depSubdivision.CountPeople;

}

}

}

ListViewRank4.ItemsSource = subdivisionRank4;

// 5 ранг пожара

List<Subdivision> subdivisionRank5 = FireDepartEntities.GetContext().Subdivisions.Where(p => p.FireRank == 6 && p.DepartureAreaId == departureArea.DepartureAreaId).ToList();

foreach (var i in subdivisionRank5)

{

i.DepartureChecked = false;

i.ArrivalChecked = false;

i.PeopleCount = 0;

if (\_currentDeparture.DepartureId != 0)

{

DepartureSubdivision depSubdivision = \_currentDeparture.DepartureSubdivisions.FirstOrDefault(p => p.SubdivisionId == i.SubdivisionId);

if (depSubdivision != null)

{

i.DepartureChecked = true;

i.TimeDeparture = depSubdivision.DateTimeDeparture;

i.ArrivalChecked = true;

i.TimeArrival = (DateTime)depSubdivision.DateTimeArrival;

i.PeopleCount = (short)depSubdivision.CountPeople;

}

}

}

ListViewRank5.ItemsSource = subdivisionRank5;

}

}

private void LoadUpdatedInfo()

{

// уточненная информация

if (\_currentDeparture.ExternalSignsInfo != null)

{

if (\_currentDeparture.ExternalSignsInfo.Contains("Слабое выделение дыма"))

RButtonSign1.IsChecked = true;

else if (\_currentDeparture.ExternalSignsInfo.Contains("Сильное выделение дыма"))

RButtonSign2.IsChecked = true;

else if (\_currentDeparture.ExternalSignsInfo.Contains("Вырывающиеся языки пламени"))

RButtonSign3.IsChecked = true;

}

if (\_currentDeparture.ThreatPeopleInfo != null)

{

if (\_currentDeparture.ThreatPeopleInfo.Contains("Имеется угроза"))

ThreatPeople.IsChecked = true;

else

ThreatPeople.IsChecked = false;

}

if (\_currentDeparture.AffectedInfo != null)

{

if (\_currentDeparture.AffectedInfo.Contains("Да, в количестве"))

YesAffected.IsChecked = true;

else if (\_currentDeparture.AffectedInfo.Contains("Пострадавших нет"))

NoAffected.IsChecked = true;

else if (\_currentDeparture.AffectedInfo.Contains("Не знаю"))

UnknownAffected.IsChecked = true;

}

if (\_currentDeparture.StatePeopleInfo != null)

{

if (\_currentDeparture.StatePeopleInfo.Contains("Без сознания"))

StatePeople1.IsChecked = true;

if (\_currentDeparture.StatePeopleInfo.Contains("ожоги"))

StatePeople2.IsChecked = true;

if (\_currentDeparture.StatePeopleInfo.Contains("Не знаю"))

StatePeople3.IsChecked = true;

}

if (\_currentDeparture.PeopleLocationInfo != null)

{

if (\_currentDeparture.PeopleLocationInfo.Contains("Самостоятельно эвакуируются"))

CheckBoxLocation1.IsChecked = true;

if (\_currentDeparture.PeopleLocationInfo.Contains("В зоне воздействия"))

CheckBoxLocation2.IsChecked = true;

}

if (\_currentDeparture.ThreatFireInfo != null)

{

if (\_currentDeparture.ThreatFireInfo.Contains("Имеется"))

ThreatFire1.IsChecked = true;

else if (\_currentDeparture.ThreatFireInfo.Contains("Имеется на этажи"))

ThreatFire2.IsChecked = true;

else if (\_currentDeparture.ThreatFireInfo.Contains("Имеется на соседние объекты"))

ThreatFire3.IsChecked = true;

else if (\_currentDeparture.ThreatFireInfo.Contains("Не знаю"))

ThreatFire4.IsChecked = true;

}

if (\_currentDeparture.NearbyObjectInfo != null)

{

if (\_currentDeparture.NearbyObjectInfo.Contains("Социально - значимые"))

CheckBoxObject1.IsChecked = true;

if (\_currentDeparture.NearbyObjectInfo.Contains("С массовым пребыванием"))

CheckBoxObject2.IsChecked = true;

if (\_currentDeparture.NearbyObjectInfo.Contains("Производственные"))

CheckBoxObject3.IsChecked = true;

if (\_currentDeparture.NearbyObjectInfo.Contains("Социально - значимых объектов рядом нет"))

CheckBoxObject4.IsChecked = true;

if (\_currentDeparture.NearbyObjectInfo.Contains("Рядом объектов нет"))

CheckBoxObject5.IsChecked = true;

}

if (\_currentDeparture.PlaceInfo != null)

{

if (\_currentDeparture.PlaceInfo.Contains("На объекте"))

RButtonPlace1.IsChecked = true;

else if (\_currentDeparture.PlaceInfo.Contains("Рядом с объектом"))

RButtonPlace2.IsChecked = true;

else if (\_currentDeparture.PlaceInfo.Contains("Внутри здания"))

RButtonPlace3.IsChecked = true;

}

if (\_currentDeparture.BarrierInfo != null)

{

if (\_currentDeparture.BarrierInfo.Contains("Есть"))

RButtonBarrier1.IsChecked = true;

else if (\_currentDeparture.BarrierInfo.Contains("Припаркованные ТС"))

RButtonBarrier2.IsChecked = true;

else if (\_currentDeparture.BarrierInfo.Contains("Котлованы"))

RButtonBarrier3.IsChecked = true;

else if (\_currentDeparture.BarrierInfo.Contains("Шлагбаум"))

RButtonBarrier4.IsChecked = true;

else if (\_currentDeparture.BarrierInfo.Contains("Искусственные заграждения"))

RButtonBarrier5.IsChecked = true;

}

if (\_currentDeparture.MeetMCSInfo != null)

{

if (\_currentDeparture.MeetMCSInfo.Contains("Имеется"))

RButtonMCS1.IsChecked = true;

else if (\_currentDeparture.MeetMCSInfo.Contains("Нет"))

RButtonMCS2.IsChecked = true;

}

if (\_currentDeparture.WhyApplicantInfo != null)

{

if (\_currentDeparture.WhyApplicantInfo.Contains("Собственник"))

RButtonWhy1.IsChecked = true;

else if (\_currentDeparture.WhyApplicantInfo.Contains("Сосед"))

RButtonWhy2.IsChecked = true;

else if (\_currentDeparture.WhyApplicantInfo.Contains("Арендатор"))

RButtonWhy3.IsChecked = true;

else if (\_currentDeparture.WhyApplicantInfo.Contains("Очевидец"))

RButtonWhy4.IsChecked = true;

}

// привлеченная техника

foreach (var i in \_currentDeparture.DepartureSubdivisions)

{

if (i.Subdivision.DepartureAreaId == selectedDepartureArea.DepartureAreaId)

i.Subdivision.Technic.TechnicCountGeneral += 1;

}

// кол-во привлеченной техники и л.с

foreach (var i in \_currentDeparture.DepartureSubdivisions)

if (i.Subdivision.DepartureAreaId == selectedDepartureArea.DepartureAreaId)

Subdivision.PeopleCountGeneral += (int)i.CountPeople;

if (\_currentDeparture.DepartureSubdivisions.FirstOrDefault(p => p.Subdivision.DepartureAreaId == selectedDepartureArea.DepartureAreaId) != null)

Technic.AllTechnicCount = \_currentDeparture.DepartureSubdivisions.Count;

PositionType.PeopleCountGeneral = \_currentDeparture.DeparturePositions.Count;

foreach (var i in \_currentDeparture.DepartureServices)

{

ServiceType.TechnicCountGeneral += (int)i.CountTechnics;

ServiceType.PeopleCountGeneral += (int)i.CountPeople;

}

TbMCSCountTechnic.Text = Technic.AllTechnicCount.ToString();

TbMCSCountPeople.Text = (Subdivision.PeopleCountGeneral + PositionType.PeopleCountGeneral).ToString();

TbGeneralCountPeople.Text = (Subdivision.PeopleCountGeneral + ServiceType.PeopleCountGeneral + PositionType.PeopleCountGeneral).ToString();

TbGeneralCountTechnic.Text = (Technic.AllTechnicCount + ServiceType.TechnicCountGeneral).ToString();

// загрузка в раздел "информация с места пожара"

DataGridFireInfo.ItemsSource = \_currentDeparture.FireInfoes.ToList();

}

private void UpdateBarrelAndSquareCount()

{

var listFireInfo = DataGridFireInfo.Items.Cast<FireInfo>().ToList();

short barrelCount = 0;

foreach (var i in listFireInfo)

{

if (i.BarrelCount != null)

barrelCount += Convert.ToInt16(i.BarrelCount);

}

var lastFireArea = listFireInfo.Where(p => p.FireAreaCount != null).OrderByDescending(p => p.DateTimeEntry).ToList();

if (lastFireArea.Count > 0)

{

\_currentDeparture.FireAreaCount = lastFireArea[0].FireAreaCount;

TbSquareFire.Text = lastFireArea[0].FireAreaCount.ToString();

}

else

{

\_currentDeparture.FireAreaCount = 0;

TbSquareFire.Text = "0";

}

\_currentDeparture.BarrelCount = barrelCount;

TbCountSteam.Text = barrelCount.ToString();

}

/// <summary>

/// Добавление информации с места пожара

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

AddOrEditFireInfoWindow addOrEditFireInfoWindow = new AddOrEditFireInfoWindow(null, \_currentDeparture);

if (addOrEditFireInfoWindow.ShowDialog() == false)

{

FireDepartEntities.GetContext().ChangeTracker.Entries<FireInfo>().ToList().ForEach(p => p.Reload());

}

DataGridFireInfo.ItemsSource = \_currentDeparture.FireInfoes.ToList();

UpdateBarrelAndSquareCount();

}

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

{

if (DataGridFireInfo.SelectedItem is FireInfo fireInfo)

{

AddOrEditFireInfoWindow addOrEditFireInfoWindow = new AddOrEditFireInfoWindow(fireInfo, \_currentDeparture);

if (addOrEditFireInfoWindow.ShowDialog() == false)

{

FireDepartEntities.GetContext().ChangeTracker.Entries<FireInfo>().ToList().ForEach(p => p.Reload());

}

DataGridFireInfo.ItemsSource = \_currentDeparture.FireInfoes.ToList();

UpdateBarrelAndSquareCount();

}

}

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

{

MessageBoxResult result = MessageBox.Show("Вы хотите удалить данную запись?", "Удаление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes && DataGridFireInfo.SelectedItem is FireInfo fireInfo)

{

try

{

//if (positionType.Specialties.Count > 0)

// throw new Exception("Существуют записи в производных таблицах, удаление запрещено!");

if (fireInfo.FireInfoId != 0)

{

FireDepartEntities.GetContext().FireInfoSubdivisions.RemoveRange(fireInfo.FireInfoSubdivisions);

FireDepartEntities.GetContext().FireInfoes.Remove(fireInfo);

}

else

{

fireInfo.FireInfoSubdivisions.Clear();

\_currentDeparture.FireInfoes.Remove(fireInfo);

}

//FireDepartEntities.GetContext().SaveChanges();

MessageBox.Show("Запись удалена!");

DataGridFireInfo.ItemsSource = \_currentDeparture.FireInfoes.ToList();

UpdateBarrelAndSquareCount();

FireDepartEntities.GetContext().SaveChanges();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

/// <summary>

/// добавление уточненной информации

/// </summary>

private void SaveUpdatedInfo()

{

// форматы разделители: - - перечисление, , - доп информация, % - текстовое поле, | - след информация

//string generalInfo = $"Что вы видите по внешним признакам? - {\_currentDeparture.SignInfo} , {\_currentDeparture.SignAddInfo}|Угроза людям, просят ли о помощи? - " +

// $"{\_currentDeparture.ThreatPeople} % {\_currentDeparture.ThreatCountPeople} , {\_currentDeparture.AddThreatPeople}|Имеются ли пострадавшие? - {\_currentDeparture.Affected}" +

// $" % {\_currentDeparture.AffectedCount} , {\_currentDeparture.AddAffected}|Что с ними?";

//foreach (var i in \_currentDeparture.StatePeoples)

// generalInfo += $" - {i}";

//generalInfo += $" , {\_currentDeparture.AddStatePeople}";

//var list = generalInfo.Split(new char[] { '|' }, StringSplitOptions.RemoveEmptyEntries);

//foreach(var i in list)

//{

// var test = i.Split(new char[] { '-' }, StringSplitOptions.RemoveEmptyEntries);

//}

}

private void RButtonSign\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.ExternalSignsInfo = (((RadioButton)sender).Content as TextBlock)?.Text;

}

private void ThreatPeople\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.ThreatPeopleInfo = (((CheckBox)sender).Content as TextBlock)?.Text;

}

private void ThreatPeople\_Unchecked(object sender, RoutedEventArgs e)

{

\_currentDeparture.ThreatPeopleInfo = null;

}

private void YesAffected\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.AffectedInfo = (((RadioButton)sender).Content as TextBlock)?.Text;

}

private void StatePeople\_Checked(object sender, RoutedEventArgs e)

{

if (\_currentDeparture.StatePeopleInfo != null && !\_currentDeparture.StatePeopleInfo.Contains($"{(((CheckBox)sender).Content as TextBlock)?.Text},"))

\_currentDeparture.StatePeopleInfo += $"{(((CheckBox)sender).Content as TextBlock)?.Text},";

else if (\_currentDeparture.StatePeopleInfo == null)

\_currentDeparture.StatePeopleInfo += $"{(((CheckBox)sender).Content as TextBlock)?.Text},";

//\_currentDeparture.StatePeoples.Add((((CheckBox)sender).Content as TextBlock)?.Text);

}

private void StatePeople\_Unchecked(object sender, RoutedEventArgs e)

{

\_currentDeparture.StatePeopleInfo = \_currentDeparture.StatePeopleInfo.Replace($"{(((CheckBox)sender).Content as TextBlock)?.Text},", "");

//\_currentDeparture.StatePeoples.Remove((((CheckBox)sender).Content as TextBlock)?.Text);

}

private void CheckBoxLocation\_Checked(object sender, RoutedEventArgs e)

{

if (\_currentDeparture.PeopleLocationInfo != null && !\_currentDeparture.PeopleLocationInfo.Contains($"{(((CheckBox)sender).Content as TextBlock)?.Text},"))

\_currentDeparture.PeopleLocationInfo += $"{(((CheckBox)sender).Content as TextBlock)?.Text},";

else if (\_currentDeparture.PeopleLocationInfo == null)

\_currentDeparture.PeopleLocationInfo += $"{(((CheckBox)sender).Content as TextBlock)?.Text},";

//\_currentDeparture.Locations.Add((((CheckBox)sender).Content as TextBlock)?.Text);

}

private void CheckBoxLocation\_Unchecked(object sender, RoutedEventArgs e)

{

\_currentDeparture.PeopleLocationInfo = \_currentDeparture.PeopleLocationInfo.Replace($"{(((CheckBox)sender).Content as TextBlock)?.Text},", "");

//\_currentDeparture.Locations.Remove((((CheckBox)sender).Content as TextBlock)?.Text);

}

private void ThreatFire\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.ThreatFireInfo = (((RadioButton)sender).Content as TextBlock)?.Text;

}

private void CheckBoxObject\_Checked(object sender, RoutedEventArgs e)

{

if (\_currentDeparture.NearbyObjectInfo != null && !\_currentDeparture.NearbyObjectInfo.Contains($"{(((CheckBox)sender).Content as TextBlock)?.Text},"))

\_currentDeparture.NearbyObjectInfo += $"{(((CheckBox)sender).Content as TextBlock)?.Text},";

else if (\_currentDeparture.NearbyObjectInfo == null)

\_currentDeparture.NearbyObjectInfo += $"{(((CheckBox)sender).Content as TextBlock)?.Text},";

}

private void CheckBoxObject\_Unchecked(object sender, RoutedEventArgs e)

{

\_currentDeparture.NearbyObjectInfo = \_currentDeparture.NearbyObjectInfo.Replace($"{(((CheckBox)sender).Content as TextBlock)?.Text},", "");

}

private void RButtonPlace\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.PlaceInfo = (((RadioButton)sender).Content as TextBlock)?.Text;

}

private void RButtonBarrier\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.BarrierInfo = (((RadioButton)sender).Content as TextBlock)?.Text;

}

private void RButtonMCS\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.MeetMCSInfo = (((RadioButton)sender).Content as TextBlock)?.Text;

}

private void RButtonWhy\_Checked(object sender, RoutedEventArgs e)

{

\_currentDeparture.WhyApplicantInfo = (((RadioButton)sender).Content as TextBlock)?.Text;

}

}

}

**Приложение Б**

Автоматизированная информационная система

«Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарстан»

Спецификация

460.ЗМК.00015-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
|  | |  | | --- | | Документация | |  |
| 460.ЗМК.00015-01 12 01 | Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарсатан» |  |
|  | Текст программы |  |
| 460.ЗМК.00015-01 13 01 | Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарсатан» |  |
|  | Описание программы |  |
| 460.ЗМК.00015-01 34 01 | Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарсатан» |  |
|  | Руководство пользователя |  |
| 460.ЗМК.00015-01 81 01 | Автоматизированная информационная система «Учет привлекаемых сил и средств местных пожарно-спасательных гарнизонов Республики Татарсатан» |  |
|  | Пояснительная записка |  |

**Приложение B**

**Диск с программным средством**