

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Информационная система автоматизации
работы службы 112**

Студент	_____	<u>Кривокоров Георгий Владимирович</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Прошкина Е.Н.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	<u>Такташкин Д.В.</u>
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	<u>Макарычев П.П.</u>
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза 2021

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ П.П. Макарычев
«__» _____ 2021 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент: Кривокоров Георгий Владимирович гр. 17ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы: Информационная система автоматизации работы службы 112

Тема утверждена приказом ПГУ № _____ от "___" _____ 20____

3. Руководитель работы _____ к.т.н., доцент кафедры «МОиПЭВМ» Прошкина Е.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение: информационная система автоматизации работы службы 112
предназначенf для приема, обработки и анализа заявок о
происшествиях.

Средства выполнения: Операционная система Windows (версия 10)

Средства разработки: Visual Studio 2019

Язык программирования: C#

Технологии разработки: ООП, UML

5. Перечень подлежащих разработке вопросов

1. Анализ предметной области и постановка задачи

2. Анализ требований на разработку

3. Разработка программных средства

4. Тестирование программных средств

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	14.04.2021 – 21.04.2021	
2	Проектирование	25%	22.04.2021 – 01.05.2021	
3	Реализация	30%	02.05.2021 – 18.05.2021	
4	Контроль качества программного обеспечения	15%	19.05.2021 – 25.05.2021	
5	План разработки бюджета	5%	26.05.2021 – 30.05.2021	
6	Оформление пояснительной записки	15%	01.06.2021 – 08.06.2021	

Дата выдачи задания "28" декабря 2020

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 85 листов, 10 таблиц, 31 рисунков, 17 использованных источников и 2 приложения.

С#, VISUAL STUDIO 2019, «СИСТЕМА 112», UML, ИНФОРМАЦИОННАЯ СИСТЕМА, GMAP, ИНТЕРФЕЙС.

Объект исследования: информационная система для автоматизации работы службы 112.

Цель работы: реализовать приложение, обеспечивающее прием и обработку заявок о происшествии.

Технологии разработки: ООП, С#, Visual Studio, WPF, Windows Forms, GMap, .Net Framework.

Задание: спроектировать и реализовать информационную систему, позволяющую обрабатывать заявки о происшествиях средствами приложения оператора, прием и обработка заявок с сайта, а также анализ места происшествия средствами ГИС.

Результаты работы: спроектирована и разработана информационная система автоматизации работы службы 112.

					ПГУ 09.03.02–8BP171.09 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	«Информационная система автоматизации работы службы 112» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Кривокоров Г.В.						
Провер.		Прошкина Е.Н.					4	85
Н. Контр.		Такташкин Д.В.				Группа 17ВИ1		
Утверд.		Макарычев П.П.						

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи	8
1.1 Задачи «Системы 112».....	8
1.2 Подсистемы «Системы 112».....	9
1.3 Ключевые объекты автоматизации «Системы 112».....	11
1.4 Основные возможности ЦОВ «Системы 112».....	13
1.5 Сравнительный анализ «Систем 112» в объектах РФ.....	14
1.6 Постановка задачи	16
2 Проектирование информационной системы.....	17
2.1 Выбор технологий проектирования	17
2.2 Проектирование логической модели базы данных	19
2.3 Анализ бизнес-процессов организации	21
2.4 Проектирование программных средств.....	24
2.4.1 Диаграмма прецедентов	24
2.4.2 Диаграмма классов.....	27
2.4.3 Диаграмма состояний	29
2.4.4 Диаграмма деятельности	30
2.4.5 Диаграмма последовательности	31
2.5 Проектирование пользовательского интерфейса	32
2.6 Планирование разработки и оценка бюджета.....	35
3 Разработка программных средств «Системы 112».....	38
3.1 Физическая модель базы данных	38
3.2 Реализация программных средств.....	39
3.2.1 Диаграмма компонентов	39
3.2.2 Диаграмма развертывания	41
4 Тестирование системы.....	48
4.1 Структурное тестирование.....	48
4.2 Метрики кода.....	51

4.2.1 Метрика кода средствами среды разработки Visual Studio	51
4.2.2 Метрики Холстеда	52
Заключение	55
Приложение А. Глоссарий	58
Приложение Б. Листинг программы	60

Введение

В современном мире одним из важных элементов обеспечения безопасности является повышение защиты населения, территорий и потенциально опасных объектов.

В настоящее время функционируют такие службы экстренного реагирования, как служба пожарной охраны, служба полиции, служба скорой медицинской помощи, аварийная служба газовой сети, служба реагирования в чрезвычайных ситуациях и служба «Антитеррор». Для удобства была создана служба 112, чтобы граждане имели возможность в чрезвычайной ситуации позвонить на единый номер и вызвать сразу несколько экстренных служб. Каждая служба 112 имеет своё программное и техническое обеспечение, образующее «Систему-112».

Автоматизация и усовершенствование процессов службы, позволяет упростить и улучшить качество работы, тем самым повышая безопасность жизни населения.

1 Анализ предметной области и постановка задачи

1.1 Задачи «Системы 112»

«Система 112» предназначена для информационного обеспечения единых дежурно-диспетчерских служб муниципальных образований и для решения следующих основных задач:

- прием по номеру «112» вызовов (сообщений о происшествиях);
- получение от оператора связи сведений о местонахождении лица, обратившегося по номеру «112», и (или) абонентского устройства, с которого был осуществлен вызов (сообщение о происшествии), а также иных данных, необходимых для обеспечения реагирования по вызову (сообщению о происшествии);
- анализ поступающей информации о происшествии;
- направление информации о происшествиях, в том числе вызовов (сообщений о происшествиях), в дежурно-диспетчерские службы экстренных оперативных служб в соответствии с их компетенцией для организации экстренного реагирования;
- обеспечение дистанционной психологической поддержки лицу, обратившемуся по номеру «112»;
- автоматическое восстановление соединения с пользовательским (оконечным) оборудованием лица, обратившегося по номеру «112», в случае внезапного прерывания соединения;
- регистрация всех входящих и исходящих вызовов (сообщений о происшествиях) по номеру «112»;
- ведение базы данных об основных характеристиках происшествий, о начале, завершении и об основных результатах экстренного реагирования на полученные вызовы (сообщения о происшествиях);
- возможность приема вызовов (сообщений о происшествиях) на иностранных языках [1].

«Система 112» является достаточно крупной и ресурсоемкой, поэтому разрабатываемая система будет решать только часть этих задач.

1.2 Подсистемы «Системы 112»

«Система-112» состоит из следующих основных подсистем:

- телекоммуникационная подсистема;
- информационно-коммуникационная подсистема;
- подсистема консультативного обслуживания;
- геоинформационная подсистема;
- подсистема мониторинга;
- подсистема обеспечения информационной безопасности.

Телекоммуникационная подсистема предназначена для обеспечения прохождения вызовов (сообщений о происшествиях), включая телефонные вызовы, короткие текстовые сообщения (SMS), от пользователей (абонентов) сетей фиксированной или подвижной связи в «Систему 112».

Информационно-коммуникационная подсистема предназначена для обеспечения хранения и актуализации баз данных, сбора, обработки и хранения информации о полученных вызовах (сообщениях о происшествиях) и обеспечения возможности получения информации о происшествии из архива в оперативном режиме, а также поддержки принятия решений по экстренному реагированию на происшествия и планированию мер реагирования.

Подсистема консультативного обслуживания предназначена для оказания информационно-справочной помощи лицам, обратившимся по номеру «112» по вопросам обеспечения безопасности жизнеобеспечения, в том числе через сеть Интернет.

Геоинформационная подсистема предназначена для отображения на основе электронных карт природно-географических, социально-демографических, экономических и других характеристик территорий, местонахождение лица, обратившегося по номеру «112», и (или)

1.3 Ключевые объекты автоматизации «Системы 112»

Сердцем «Системы 112» является центр обработки вызовов. Именно там происходит прием, анализ и обработка вызова с последующей передачей информации службам быстрого реагирования.

ЦОВ разделяются на:

- ЦОВ;
- РЦОВ;
- ЦОВ-ЕДДС.

ЦОВ (основной центр обработки вызовов) предназначен для обеспечения приема и обработки вызовов от населения с территории административного центра субъекта РФ, их обработки и передачи на реагирование в соответствующие дежурно-диспетчерские службы (ДДС) муниципальных образований, взаимодействия с персоналом ЦУКС, ЦОВ-ЕДДС, ДДС, навигационно-информационным центром системы экстренного реагирования при авариях ЭРА-ГЛОНАСС, а в случае необходимости, приема и обработки вызовов со всей территории субъекта РФ.

РЦОВ (резервный центр обработки вызовов) предназначен для резервирования функций основного ЦОВ, а также для организации обучения персонала «Системы 112».

ЦОВ-ЕДДС (муниципальные центры обработки вызовов) создаются на базе (являются подразделением) существующих единых дежурно-диспетчерских служб (ЕДДС) муниципальных районов, предназначены для приема и обработки вызовов от населения, находящегося в зоне обслуживания ЕДДС, а также для взаимодействия с ЦОВ, РЦОВ, ДДС (в рамках «Системы 112») и ЦОВ-ЕДДС соседних (смежных) муниципальных образований.

Взаимодействие ЦОВ представлено на рисунке 2.

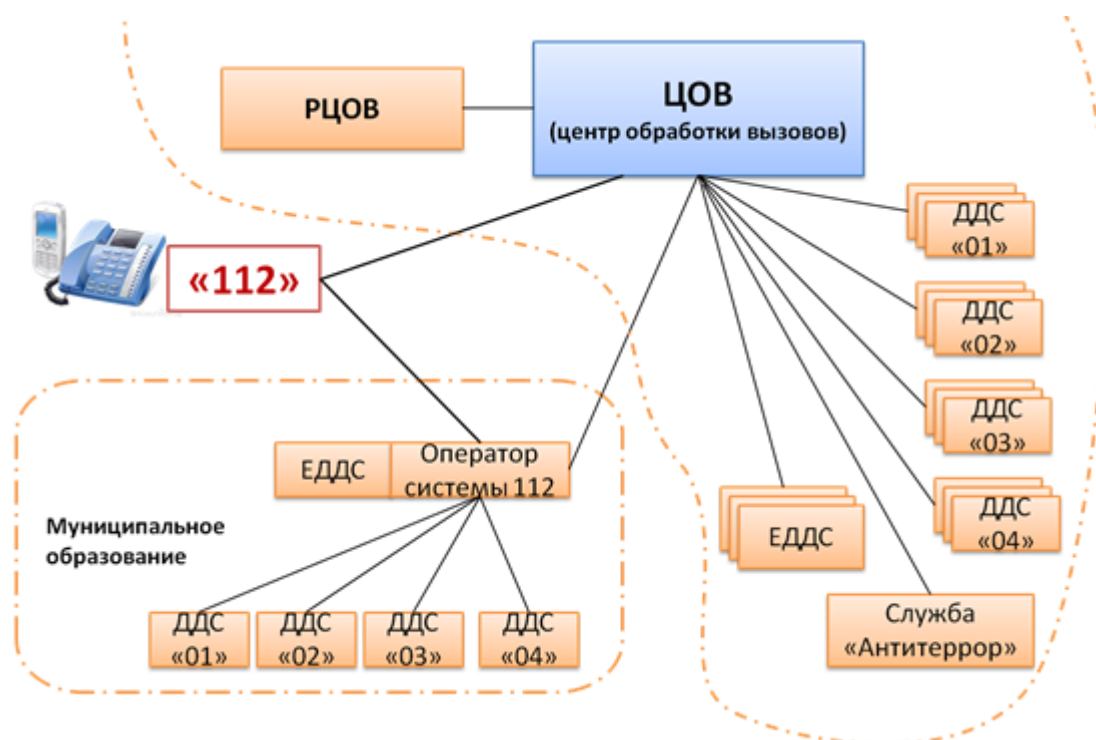


Рисунок 2 – Взаимодействие ЦОВ

На объектах автоматизации «Системы 112» создаются следующие автоматизированные рабочие места (АРМ):

- АРМ администратора «Системы 112»;
- АРМ администратора Информационной безопасности (ИБ);
- АРМ старшего Оператора;
- АРМ оператора;
- АРМ диспетчера;
- АРМ психолога;
- АРМ переводчика.

Автоматизированное рабочее место (АРМ) «Системы 112» – совокупность информационно-программно-технических ресурсов, обеспечивающих конечному оператору обработку данных и автоматизацию управленческих функций. При этом предполагается, что операции по накоплению, хранению и переработке информации возлагаются на вычислительную технику и программное обеспечение серверной части, а

оператор выполняет прием обращений, заполнение первичной информации о происшествиях и привлечение соответствующих служб.

АРМ оператора «Системы 112» предназначено для автоматизации деятельности дежурно-диспетчерских центров при приеме и регистрации вызова (обращения) о происшествии, обработке карточки информационного обмена и контроле выполнения заявки. АРМ «Системы 112» позволяет организовывать и следить за процессом реагирования на происшествия, принимать заявки, перенаправлять их адресно на городские службы, отправлять карточки обращений. «Система 112» позволяет вести учет сил и средств, направленных на разрешение инцидента. При поступлении вызова на АРМ в системе заводится уникальная карточка обращения.

1.4 Основные возможности ЦОВ «Системы 112»

Осуществлять, поддерживать и автоматизировать процессы ЦОВ помогают множество компонентов.

1. Телефонный вызов. Во время разговора с заявителем оператор 112 получает информацию о происшествии, которую необходимо занести в карточку происшествия.

2. Обработка SMS-сообщений. Обработка SMS-обращений аналогична обработке вызовов. Отличие состоит в том, что общение с заявителем происходит с помощью текстовых сообщений в отдельной вкладке SMS.

3. Взаимодействие с системой «ЭРА-ГЛОНАСС». При аварии автомобильное устройство в соответствии с заложенным в него алгоритмом определяет степень тяжести аварии, определяет местоположение пострадавшего транспортного средства через спутники системы ГЛОНАСС и/или GPS, устанавливает связь с инфраструктурой «ЭРА-ГЛОНАСС» и в соответствии с протоколом передаёт необходимые данные об аварии.

4. Работа с картами. На карте местности отображается геоинформационная-система (ГИС), представляющая собой карту местности

с различными данными. На карте отображаются (помимо названий улиц, населенных пунктов и номеров домов).

5. Взаимодействие с телефонными операторами. Взаимодействие с телефонными операторами позволило определять геолокацию человека, но точность геолокации крайне низкая.

6. Мобильное приложение. В московской области появилось мобильное приложение «112 МО». Приложение содержит ленту с информацией о последних чрезвычайных происшествиях, справочники и другие полезные функции. При вызове, сотрудник службы 112 получит данные личного профиля пользователя, номер телефона и геопозицию. Геопозиция со смартфона, является более точной, чем предоставленная оператором сотовой связи [3].

Данные пункты – это лишь малая часть внедренных технологий, позволяющая автоматизировать и улучшить работу «Системы 112». С развитием технологий совершенствуется и сама система.

1.5 Сравнительный анализ «Систем 112» в объектах РФ

Каждая область и регион имеют свою «Систему 112», соответственно возможности у систем тоже разные. Сравним возможности и средства «Систем 112» в ЯНАО, Московской области, Ульяновской области, а также разрабатываемой системы (таблица 1).

Таблица 1 – Сравнение «Систем 112» в регионах и областях РФ

	Московская область	ЯНАО	Ульяновская область	Разрабатываемая система
1	2	3	4	5
Обработка телефонных звонков	Да	Да	Да	Да

Продолжение таблицы 1

1	2	3	4	5
АРМ Оператора	Да	Да	Да	Да
АРМ для работы с глухонемыми	Да	В процессе разработки	Нет	Нет
Средства по работе с глухонемыми	SMS, видеосвязь, сайт	SMS	SMS	Сайт
Взаимодействи е с «ЭРА- ГЛОНАСС»	Да	Да	Нет	Нет
Получение геолокации через сотовые вышки	Да	Да	Да	Нет
Обработка вызова через спутниковый телефон	Да	Да	Да	Нет
Анализ ГИС	Да	Да	Да	Да
Ведение БД и оперативный анализ БД	Да	Да	Да	Да
Наличие мобильного приложение	Да	В процессе разработки	Нет	Нет
Получение данных о происшествии с сайта	Да	Нет	Нет	Да

В таблице выделены основные и дополнительные возможности «Системы 112». По результату сравнения можно сделать вывод о том, что все выбранные «Системы 112» выполняют свои основные функции.

Также можно выделить тенденцию по внедрению службами дополнительных функций и возможностей в систему, что позволяет расширить основной функционал системы, а также улучшить качество работы самой службы.

На основании проведенных исследований была изучена «Система 112», её функции, задачи и компоненты. Был проведен сравнительный анализ «Систем 112» в различных областях и регионах РФ, в результате чего были выделены ряд возможностей и средств этих систем, позволяющих автоматизировать работу службы 112.

Вследствие, будет реализована система, позволяющая автоматизировать основные задачи службы 112.

1.6 Постановка задачи

Целью данной выпускной квалификационной работы является разработка информационной системы автоматизации работы службы 112.

Для достижения указанной цели были поставлены следующие задачи:

- анализ предметной области и постановка задачи на разработку системы;
- анализ существующих аналогов информационной системы;
- определение вариантов использования информационной системы;
- проектирование системы;
- реализация приложения;
- тестирование приложения.

Создаваемая система должна упростить и ускорить прием и обработку заявок о происшествии службой.

2 Проектирование информационной системы

2.1 Выбор технологий проектирования

Для проектирования информационной системы автоматизации службы 112 были выбраны диаграммы нотации ArgoUML, а также CASE средства ERwin Data Modeler и Ramus Educational.

Пакет ArgoUML – среда визуального UML моделирования. ArgoUML является открытым программным обеспечением. Одним из основных достоинств ArgoUML является его среда моделирования. Программное обеспечение предлагает набор перекрывающихся диаграмм, позволяя получить полное представление о проекте.

Программное обеспечение «Ramus Educational» предназначено для использования в проектах, в которых необходимо описание бизнес-процессов предприятия и (или) создание систем классификации и кодирования. В Ramus реализованы следующие методологии:

- IDEF0 – методология функционального моделирования, позволяющая описать бизнес-процесс в виде иерархической системы взаимосвязанных функций;
- DFD (Data Flow Diagrams) – диаграммы потоков данных – методология структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Для проектирования базы данных веб-приложения было использовано CASE средство ERwin Data Modeler.

«Ramus Educational» позволяет создавать графические модели бизнеспроцессов согласно правилам синтаксиса IDEF0. Кроме того, «Ramus Educational» позволяет создавать систему классификации и кодирования всех объектов, которые фигурируют в бизнес-процессах предприятия и увязывать эту систему с графическими моделями бизнес-процессов. «Ramus Educational» обладает гибкими возможностями построения отчетности по

графическим моделям бизнес-процессов и системе классификации и кодирования. Данные возможности позволяют создавать отчетность в форме документов, которые регламентируют деятельность предприятия.

ERwin является одним из CASE средств, позволяющих моделировать бизнес процессы. Он относится к категории I - CASE. ERwin обеспечивает интеграцию моделей верхнего уровня с моделями нижнего уровня. Модели верхнего уровня разрабатываются на начальных стадиях проектирования информационных систем. Модели нижнего уровня разрабатываются на этапе создания программного кода и тестирования.

Основное предназначение ERwin data modeler это моделирование данных, поэтому с точки зрения моделирования бизнес процессов его применение ограничено. Он позволяет создавать модели данных высокого уровня, выполнять автоматическое преобразование этих моделей в модели низкого уровня, генерировать схемы базы данных и описание данных на уровне программного кода.

Для разработки приложения была выбрана среда разработки Microsoft Visual Studio. Он используется для разработки компьютерных программ, а также веб-сайтов, веб-приложений, веб-сервисов и мобильных приложений. Visual Studio использует платформы разработки программного обеспечения Microsoft, такие как Windows API, Windows Forms, Windows Presentation Foundation, Windows Store и Microsoft. Сильверлайт. Он может создавать как собственный код, так и управляемый код.

Visual Studio включает в себя поддерживающий IntelliSense (компонент завершения кода), а также рефакторинг кода. Интегрированный отладчик работает как отладчик на уровне исходного кода, так и как отладчик на уровне компьютера. Другие встроенные инструменты включают профилировщик кода, конструктор для создания приложений с графическим интерфейсом, веб-дизайнер, конструктор классов и схема базы данных конструктор. Он принимает плагины, которые расширяют функциональность почти на всех уровнях, включая добавление поддержки для систем

управления версиями (например, Subversion и Git) и добавление новых наборов инструментов, таких как редакторы и визуальные дизайнеры для предметных языков или наборы инструментов для других аспектов жизненного цикла разработки программного обеспечения (например, клиент Azure DevOps : Team Explorer).

Visual Studio поддерживает 36 различных языков программирования и позволяет редактору кода и отладчику поддерживать (в разной степени) практически любой язык программирования при условии, что существует служба для конкретного языка. Встроенные языки включают C, C ++, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XAML, XSLT, HTML и CSS.

2.2 Проектирование логической модели базы данных

CA ERwin Data Modeler (далее ERwin) – CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных.

ERwin позволяет создавать логическую, физическую модели и модель, совмещающую логический и физический уровни.

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире (например «Постоянный клиент», «Отдел» или «Заказ»).

Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физический уровень зависит от конкретной СУБД. В физической модели содержится информация о всех объектах БД. Физическая модель зависит от конкретной реализации СУБД. Одной и той же логической модели могут соответствовать несколько разных физических моделей.

Спроектированная логическая модель базы данных представлена на рисунке 3.

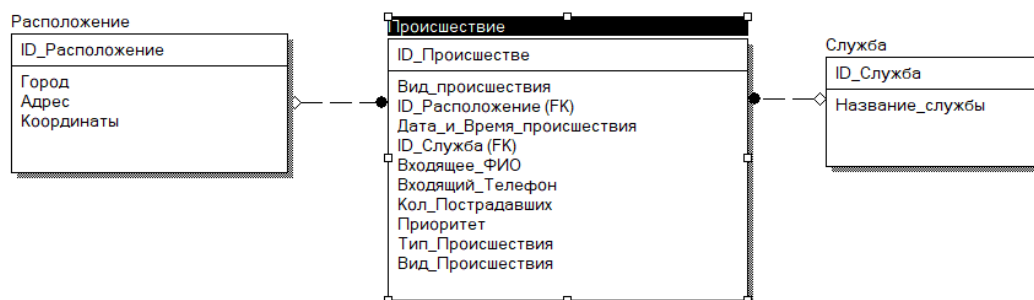


Рисунок 3 – Логическая модель базы данных

Описание сущностей и их атрибутов логической модели представлены в таблице 2.

Таблица 2 – Описание сущностей и их атрибутов логической модели

Сущность	Атрибуты	Роль атрибута	Описание атрибута
1	2	3	4
Происшествие	ID_Происшествие	РК	Уникальный код происшествия
	Вид_происшествия		Вид происшествия, о котором сообщил заявитель
	ID_Расположение	FK	Уникальный код расположения происшествия
	Дата_и_время_происшествия		Дата и время, когда произошло происшествие
	ID_Служба	FK	Уникальный номер служб
	Входящий_Телефон		Номер телефона обратившегося в службу
	Входящий_ФИО		ФИО обратившегося в службу
	Кол_Пострадавших		Количество пострадавших в происшествии
	Приоритет		Приоритет происшествия
	Тип_происшествия		Тип происшествия
	Вид_происшествия		Вид происшествия
Расположение	ID_Расположение	РК	Уникальный код расположения происшествия
	Город		Город, в котором произошло происшествие
	Адрес		Адрес, в котором произошло происшествие
	Координаты		Координаты, где произошло происшествие
Служба	ID_Служба	РК	Уникальный код службы
	Название_службы		Наименование службы

На основании логической модели базы данных будет создана физическая модель.

2.3 Анализ бизнес-процессов организации

В современных условиях непостоянной внешней среды главной задачей предприятия становится быстрое реагирование на изменения и соответствующее внедрение адекватных мероприятий в организации и осуществлении собственной предпринимательской деятельности.

Бизнес-процессы являются отправной точкой существования любой организации и зависят от заданий деятельности этой организации как субъекта хозяйствования. Они гарантируют выполнение различных типов работ на предприятии, в ходе которых изготавливаются товары или услуги, которые данный субъект хозяйствования или делает, или продаёт и поставляет, или делает все это в совокупности. Каждому виду работ, которые входят в общий процесс хозяйственной деятельности, присущи свои временные характеристики, устанавливающие очерёдность работ, продолжительность их проведения и пр. характеристики [8].

Контекстная диаграмма бизнес-процессов информационной системы автоматизации службы 112 в методологии IDEF0 представлена на рисунке 4.



Рисунок 4 – Контекстная диаграмма

В данной контекстной диаграмме входными данными является заявка.

Выходным данным являются:

- отчет о происшествиях;
- вызов экстренных служб.

Механизмы и управления – это карты и нормативные документы.

Ресурсами являются: оператор и МТО.

На основе созданной контекстной диаграммы в результате детализации была разработана детализирующая диаграмма, представленная на рисунке 5.



Рисунок 5 – Детализирующая диаграмма процесса «Обеспечение вызова экстренных служб» в нотации IDEF0

В процессе декомпозиции бизнес-процесса системы выделены 3 процесса: «Прием заявки», «Анализ заявки», «Передача сигнала экстренных служб».

Процесс «Анализ заявки» предназначен для обработки заявки и заполнения карточки происшествия, а также для анализа базы данных заявок.

Процесс «Передача сигнала экстренным службам» предназначен для уведомления экстренных служб о происшествии.

Декомпозиция процесса «Прием заявки» в нотации IDEF0 представлена на рисунке 6.

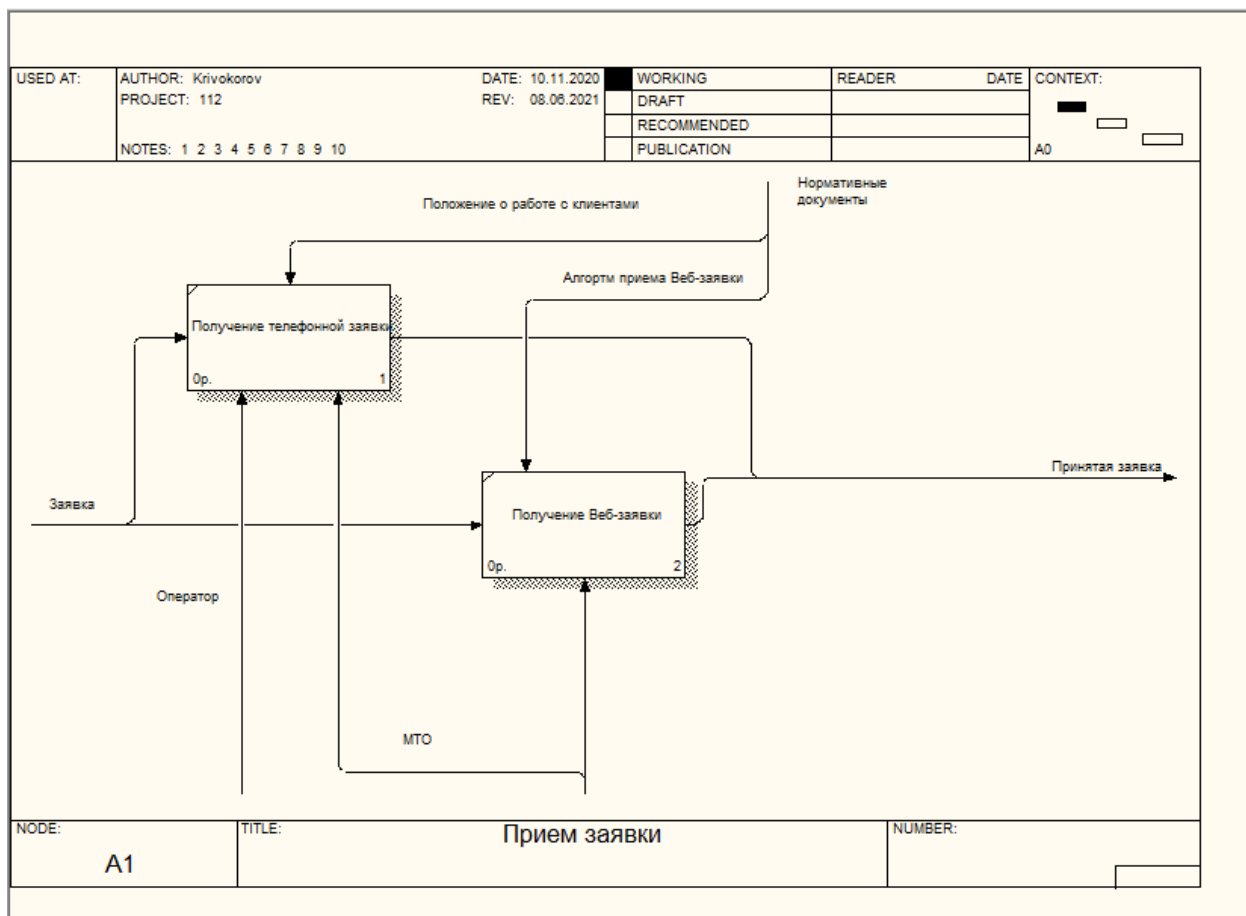


Рисунок 6 – Детализирующая диаграмма процесса «Прием заявки»

Существует два типа заявок:

- заявка по телефону;
- веб-заявка.

Заявка через телефонный вызов принимается оператором средствами МТО, на основании положения о работе с клиентами.

Веб-заявка осуществляется уже без оператора средствами МТО, на основании алгоритма приема веб-заявки.

2.4 Проектирование программных средств

2.4.1 Диаграмма прецедентов

Общая функциональность системы представлена с помощью диаграммы прецедентов. Она позволяет показать пользователей системы и ее функции. Основными элементами диаграммы являются актеры и варианты использования (прецеденты), с которыми они взаимодействуют. Между прецедентами также могут существовать отношения [9]. Диаграмма прецедентов представлена на рисунке 7.

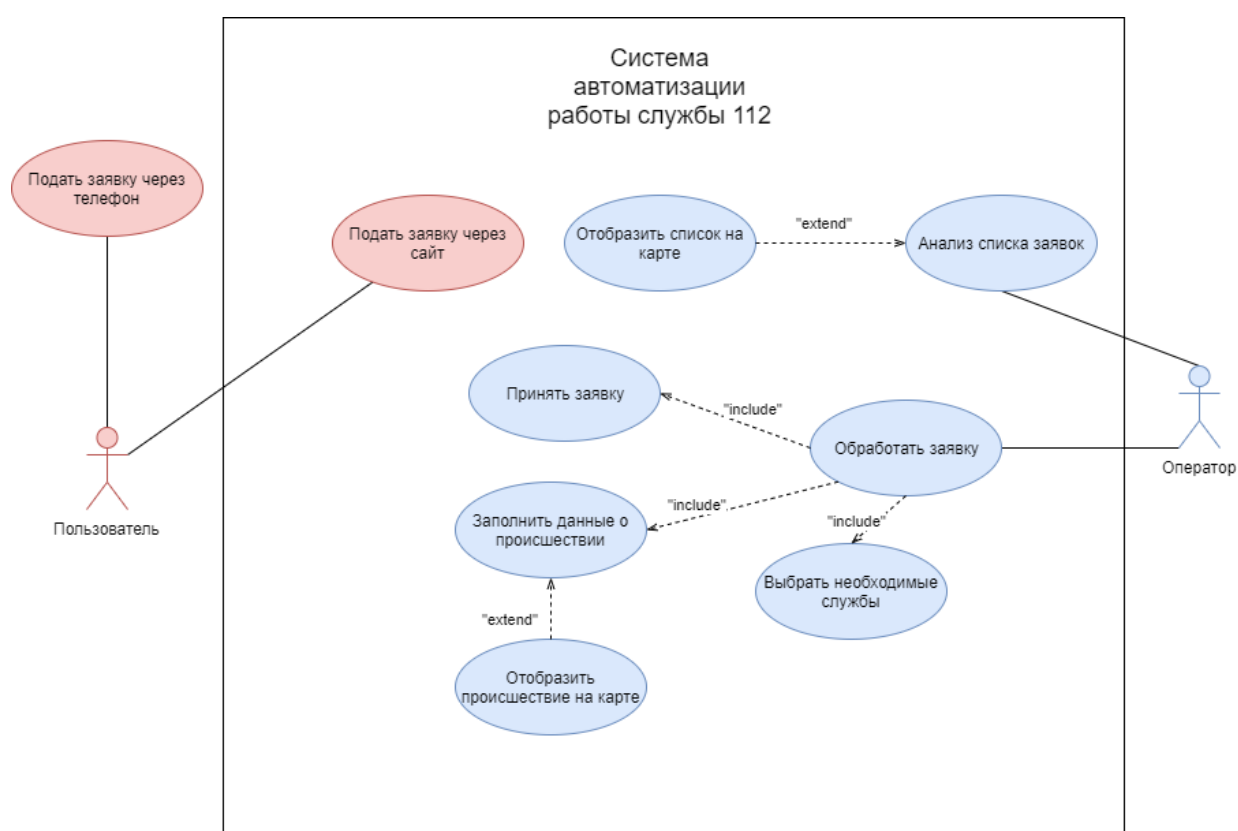


Рисунок 7 – Диаграмма прецедентов

Описание вариантов использования в таблице 3.

Таблица 3 – Описание вариантов использования

Варианты использования для актера «Пользователь»	
1	
Вариант использования	Подать заявку через телефон
Актер	Пользователь
Цель	Передать информацию о происшествии

Продолжение таблицы 3

Предусловие	нет
Основной поток:	1.Пользователь звонит службе 112 2.пользователь передает информацию о происшествии оператору
Постусловие	нет
Ссылки на другие варианты использования:	
Включения	нет
Расширение	нет
2	
Вариант использования	Подать заявку через сайт
Актер	Пользователь
Цель	Передать информацию о происшествии
Предусловие	Нет
Основной поток:	1. Пользователь через интернет заполняет заявку 2. Пользователь отправляет заявку
Постусловие	нет
Ссылки на другие варианты использования:	
Включения	нет
Расширение	нет
Варианты использования для актера «Оператор»	
1	
Вариант использования	Обработать заявку
Актер	Оператор
Цель	Обработка заявки с целью передачи информации о происшествии службам
Предусловие	нет
Основной поток:	Оператор с помощью поддержки системы заполняет необходимые данные о происшествии.
Постусловие	нет
Ссылки на другие варианты использования:	
Включения	Принять заявку; заполнить данные о происшествии; выбрать необходимые службы.
Расширение	нет
2	
Вариант использования	Выбрать необходимые службы
Актер	Оператор
Цель	Определить по происшествию необходимой службы.
Предусловие	
Основной поток:	1.Оператор вычленяет происшествие из заявки 2.Оператор сверяет происшествие с протоколом работы с происшествиями 3.Оператор вводит в систему необходимые службы 4.Система фиксирует службы.
Постусловие	нет
Ссылки на другие варианты использования:	

Продолжение таблицы 3

Включения	нет
Расширение	нет
3	
Вариант использования	Заполнить данные о происшествии
Актер	Оператор
Цель	Записать данные о происшествии, которые предоставил пользователь
Предусловие	
Основной поток:	1. Оператор заполняет данные в форму. 2. Данные с формы записываются в БД.
Постусловие:	
Ссылки на другие варианты использования:	
Включения	нет
Расширение	Отобразить происшествие на карте
4	
Вариант использования	Принять заявку
Актер	Оператор
Цель	Получить данные от пользователя о происшествии
Предусловие	нет
Основной поток:	Средствами системы оператор получает данные необходимые для составления карточки происшествия
Постусловие	нет
Ссылки на другие варианты использования:	
Включения	нет
Расширение	нет
5	
Вариант использования	Отобразить происшествие на карте
Актер	Оператор
Цель	Наглядно показать, где произошло происшествие
Предусловие	нет
Основной поток:	1. Оператор вводит координаты происшествия 2. Вызывается форма с картой, где меткой изображено место происшествия
Постусловие	нет
Ссылки на другие варианты использования:	
Включения	нет
Расширение	нет
6	
Вариант использования	Анализ списка заявок
Актер	Оператор
Цель	Проанализировать список заявок
Предусловие	нет
Основной поток:	1. Оператор получает список заявок 2. Заявки выводятся на форме
Ссылки на другие варианты использования:	
Включения	Отобразить список на карте

Окончание таблицы 3

Расширение	нет
7	
Вариант использования	Отобразить список на карте
Актер	Оператор
Цель	Наглядно показать, где произошли происшествия
Предусловие	нет
Основной поток:	1 Открывается форма с картой 2 Оператор выбирает вид происшествия 3 Происшествия с данным видом изображаются на карте
Постусловие	нет
Ссылки на другие варианты использования:	
Включения	нет
Расширение	нет

Благодаря диаграмме прецедентов были выделены основные функции системы.

2.4.2 Диаграмма классов

Класс представляет собой дескриптор набора объектов, имеющих одинаковые атрибуты, операции, методы, отношения и поведение. Иными словами класс – это дескриптор набора объектов, имеющих одинаковые свойства. Каждый объект – это экземпляр только одного класса. Класс надо рассматривать как шаблон объектов: класс определяет структуру (набор свойств) всех объектов этого класса. Все объекты одного класса должны иметь одинаковый набор операций, одинаковый набор атрибутов и одинаковый набор отношений, но значения атрибутов могут быть различными.

При создании экземпляра класса создается новый объект. Класс используется как шаблон. UML старается быть универсальным, поэтому создание экземпляров применяется не только к классам и объектам, но и к другим элементам модели. Понятие создания экземпляров, по сути, представляет общий процесс создания конкретного экземпляра чего-либо по шаблону.

В большинстве ОО языков есть специальные операции, называемые конструкторами, которые на самом деле принадлежат классу, а не объектам этого класса. Назначение операции конструктор – создание новых экземпляров класса. Конструктор выделяет память для нового объекта, присваивает ему уникальный идентификатор и задает исходные значения атрибутов. Он также настраивает все связи с другими объектами [10].

Диаграмма класса представлена на рисунке 8.

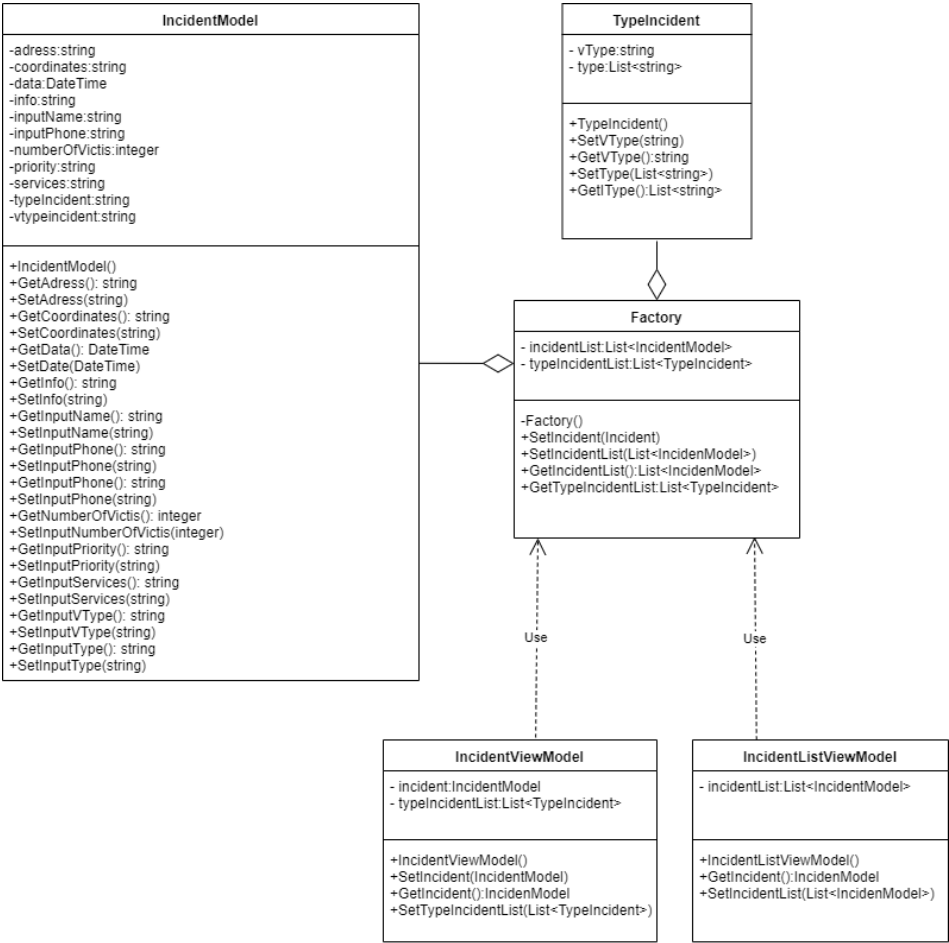


Рисунок 8 – Диаграмма классов

Класс «IncidentModel» описывает происшествие и позволяет хранить в себе информацию.

Класс «IncidentFactory» позволяет хранить в себе коллекцию объектов «IncidentModel» и «TypesIncident», а также содержит методы и свойства для доступа к этим коллекциям.

Класс «TypesIncident» предназначен для хранения информации о видах и типах происшествий.

Классы «IncidentViewModel» и «IncidentListViewModel» служат для подготовки информации для вывода на экранные формы, а также для получения данных с этих форм.

2.4.3 Диаграмма состояний

Диаграммы состояний моделируют аспекты динамического поведения системы в виде конечного автомата – автомата, который может существовать в конечном числе состояний. В ответ на события конечный автомат четко осуществляет переходы между этими состояниями.

Диаграмма состояний формы для принятия заявки о происшествии представлена на рисунке 9.

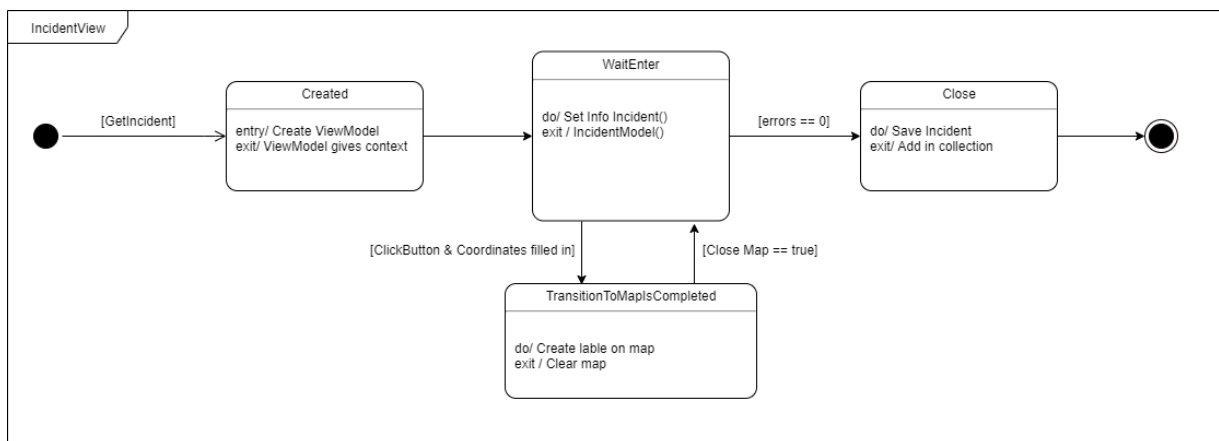


Рисунок 9 – Диаграмма состояний

Первое состояние – «CreateForm», состояние, которое показывает, что форма для принятия заявки создана, входным действием является создание «ViewModel» для данной формы, далее форма получает контекст от данной «ViewModel».

После система переходит в состояние «WaitEnter», в этом состоянии происходит заполнение данных о происшествии. При условии того, что заполнены данные о месте происшествия и нажата кнопка вызова карты, система переходит в состояние «TransitionToMapIsCompleted», в котором

форма переключается на карту. Когда карта закрывается, то система возвращается в состояние «WaitEnter», и выходом из него является создание объекта класса «IncidentModel».

Если ошибок не обнаружено, система переходит в состояние «Close», в котором происходит сохранение объекта класса и добавление в коллекцию.

2.4.4 Диаграмма деятельности

Диаграммы деятельности позволяют моделировать процесс как деятельность, которая состоит из коллекции соединенных ребрами узлов.

Деятельность может быть добавлена к любому элементу модели с целью моделирования его поведения. Элемент обеспечивает контекст для деятельности, и деятельность может использовать возможности своего контекста [11].

В данной системе диаграмма подробно отображает функцию системы «Прием заявки». Диаграмма деятельности в нотации UML представлена на рисунке 10.

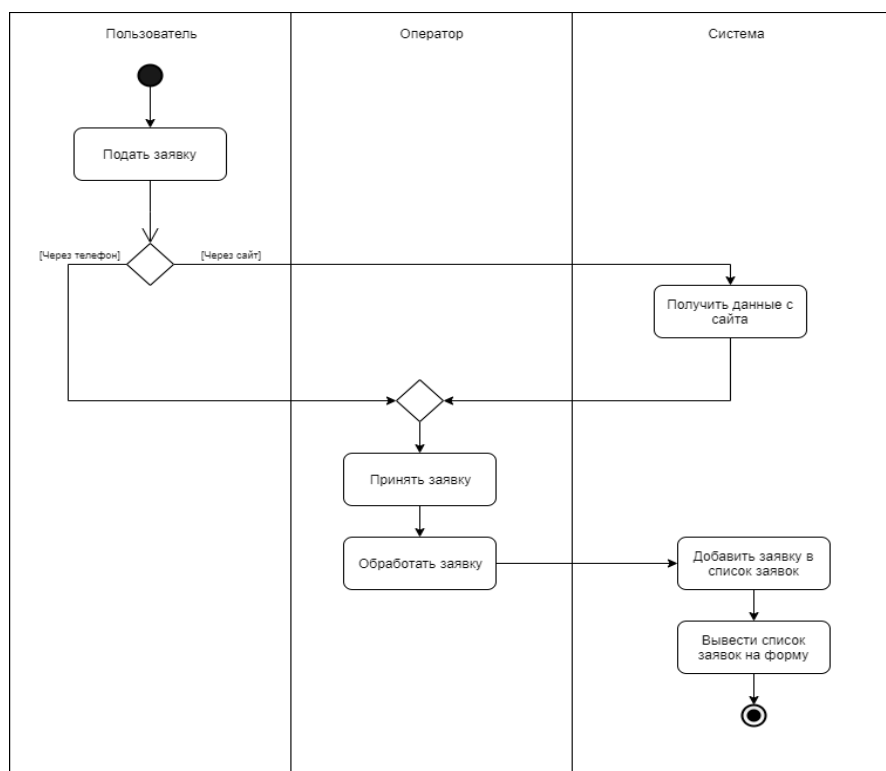


Рисунок 10 – Диаграмма деятельности

Пользователь подает заявку через телефон или через сайт. Если заявка подается через сайт, то Пользователь заполняет форму на сайте, после этого данные о происшествии передаются оператору. Если же Пользователь передает данные о происшествии через телефон, то Оператор заполняет соответствующую форму.

Все заявки обрабатываются и добавляются в список обработанных заявок. Данный список выводится на форму.

2.4.5 Диаграмма последовательности

Диаграммы последовательностей представляют взаимодействия между линиями жизни как упорядоченную последовательность событий.

Линия жизни (lifeline) представляет одного участника взаимодействия, т. е. она представляет, как экземпляр конкретного классификатора участвует во взаимодействии.

Диаграмма последовательности в нотации UML представлена на рисунке 11.

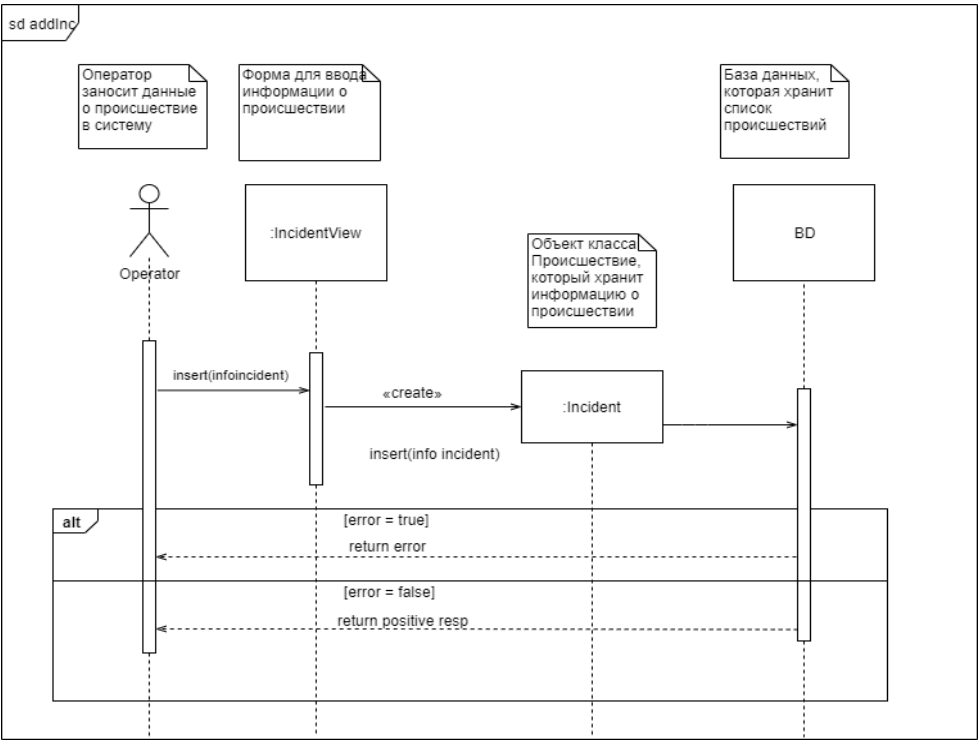


Рисунок 11 – Диаграмма последовательности

На диаграмме выделены 4 линии жизни: Operator, IncidentView, Incident, BD.

Operator (Оператор) заполняет данные на IncidentView (форма для ввода информации о происшествии). Создается объект класса Incident с внесенными в него данными. Далее объект класса записывается в BD (база данных). Если объект успешно добавился в коллекцию, то управление возвращается к оператору с положительным ответом. Если же произошла ошибка, то оператору возвращается соответствующий ответ.

Все объекты, представленные на диаграмме, существуют на протяжении всего сеанса общения.

2.5 Проектирование пользовательского интерфейса

Пользовательский интерфейс (ПИ) представляет собой совокупность программных и аппаратных средств, осуществляющих взаимодействие пользователя с информационной системой. ПИ включает в себя систему меню, диалоговые формы, сообщения об ошибках, справочную систему и т.п. Поскольку ПИ является частью информационной системы, то, с одной стороны, аналитику необходимо представлять базовые принципы проектирования ПИ, а с другой стороны, он должен быть вовлечен в этот процесс.

Для данной системы планируется разработать 5 интерфейсов для взаимодействия с пользователем:

- интерфейс осуществляющий прием заявки;
- форма для отображения списка принятых заявок;
- карта для отображения места происшествия;
- карта для отображения списка происшествий с фильтрацией по типу происшествия;
- веб форма, для принятия заявки с сайта.

Для интерфейса, осуществляющего прием заявки необходимы поля для ввода информации о происшествии:

- дата происшествия;
- вид и тип происшествия;
- ФИО и телефонный номер заявителя;
- место происшествия (город, улица, дом, координаты);
- список служб необходимых для обеспечения помощи.

Также планируется что на форме будет расположена кнопка, осуществляющая передачу данных о происшествии с формы в систему, а также кнопка для вызова карты.

Аналогичным интерфейсом является веб форма для принятия заявки.

Пример интерфейса представлен на рисунке 12.

The diagram shows a vertical web form with the following elements from top to bottom:

- Label: "Дата происшествия" (Date of incident)
- Input field: A rectangular box for the date.
- Label: "Вид и тип происшествия" (Type and kind of incident)
- Input field: A rectangular box for the incident type.
- Label: "Информация о заявителе" (Applicant information)
- Input field: A rectangular box for applicant details.
- Label: "Место происшествия" (Location of incident)
- Input field: A rectangular box for the location.
- Label: "Список служб" (List of services)
- Input field: A rectangular box for the list of services.
- Button: "Обработать" (Process)

Рисунок 12 – Пример интерфейса для приема заявки и веб приема заявки

Форма для отображения списка заявок будет состоять из таблицы. Столбцы таблицы именуются атрибутами происшествия. При каждом добавлении происшествия информация о нем будет отображаться в данной таблиц.

Пример формы для отображения списка заявок представлен на рисунке 13.

Дата происшествия	Вид и тип происшествия	Информация о заявителе	Место происшествия	Список служб
11.12.2021	Транспортное происшествие на дороге	Кривокоров Г.В. 89615552358	Ул. Красная 64а	Скорая Пожарная
12.10.2021	Пожар в квартире	Кривокоров Г.В. 89615552358	Ул. Красная 64а	Скорая Пожарная
<div>Показать происшествия на карте</div>				

Рисунок 13 – Пример интерфейса для отображения происшествий

На карте для отображения происшествия при загрузке стоит метка происшествия. Имеется возможность увеличивать и уменьшать масштаб карты, а также разворачивать карту, подключаться к картам разных производителей (Яндекс, Google и т.п.). Планируется возможность отображать выбранный список служб, и каждой службе присвоена своя метка. Также планируется, что на карте находятся дополнительные компоненты для работы с картой.

Карта для отображения списка происшествий вызывается с помощью кнопки, расположенной на форме для отображения списка принятых заявок, и имеет схожий функционал с картой для отображения происшествий. Вместо выбора списка служб пользователь выбирает вид происшествия. В зависимости от выбора, на карте отображаются метки, соответствующие виду происшествия.

Пример карты представлен на рисунке 14.

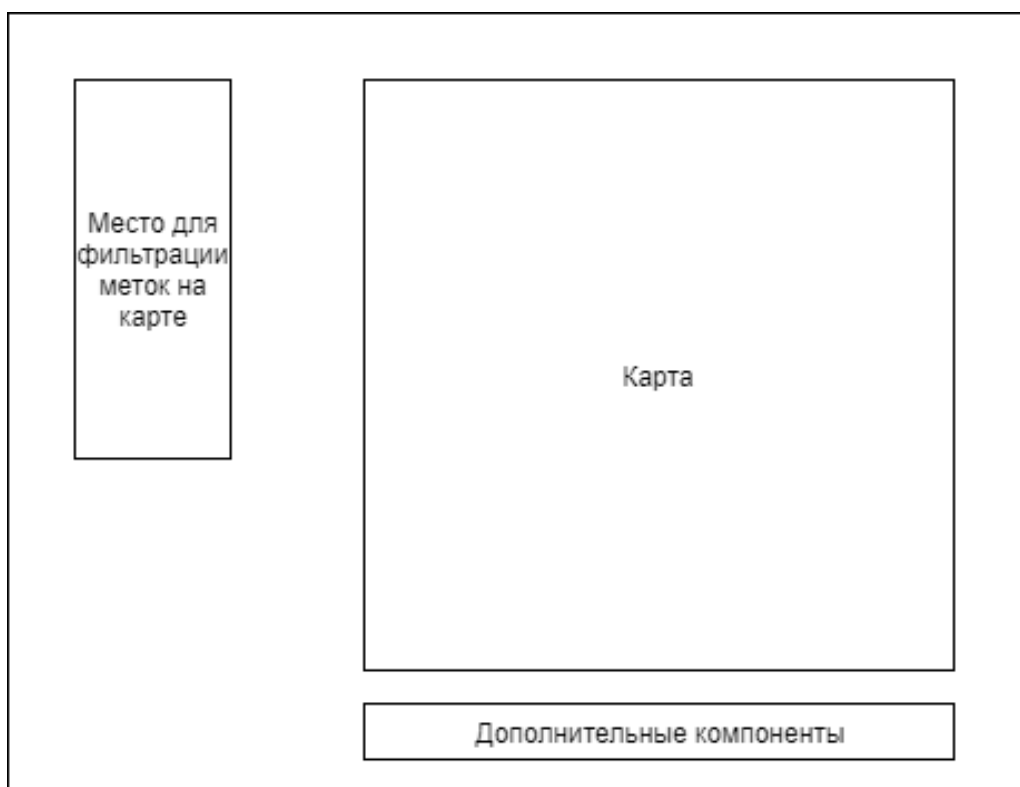


Рисунок 14 – Пример интерфейса для отображения происшествий

Таким образом, были спроектированы интерфейсы для работы с пользователем, которые в дальнейшем будут реализованы средствами среды разработки.

2.6 Планирование разработки и оценка бюджета

Использование бюджетирования в компании позволяет планировать и контролировать эффективность использования ресурсов компании, согласовывать деятельность различных подразделений внутри компании, контролировать их вклад в достижение целей и выполнение задач компании. Если компания использует проектный подход к деятельности, то для получения полной информации необходимо организовать как отдельный сбор фактических показателей, так и осуществлять планирование по каждому проекту отдельно для корректного расчета финансовых показателей каждого проекта и компании в целом.

Данные для формирования диаграммы Ганта представлены в таблице 4.

Таблица 4 – Данные для формирования диаграммы Ганта

Название задачи	Длительность	Начало	Окончание	Предшественники	Название ресурсов
Разработка приложения	33,5 дней	Чт 15.04.21	Вт 01.06.21		
Анализ предметной области	2 дня	Чт 15.04.21	Пт 16.04.21		Аналитик
Постановка задачи	4 дня	Пн 19.04.21	Чт 22.04.21	2	Аналитик
Проектирование	2,5 дня	Пт 23.04.21	Вт 27.04.21	3	Аналитик, Программист
Реализация	15 дней	Вт 27.04.21	Вт 18.05.21	4	Программист
Тестирование	7 дней	Вт 18.05.21	Чт 27.05.21	5	Тестирование
Внедрение	3 дня	Чт 27.05.21	Вт 01.06.21	6	Программист

Диаграмма Ганта представлена на рисунке 15.

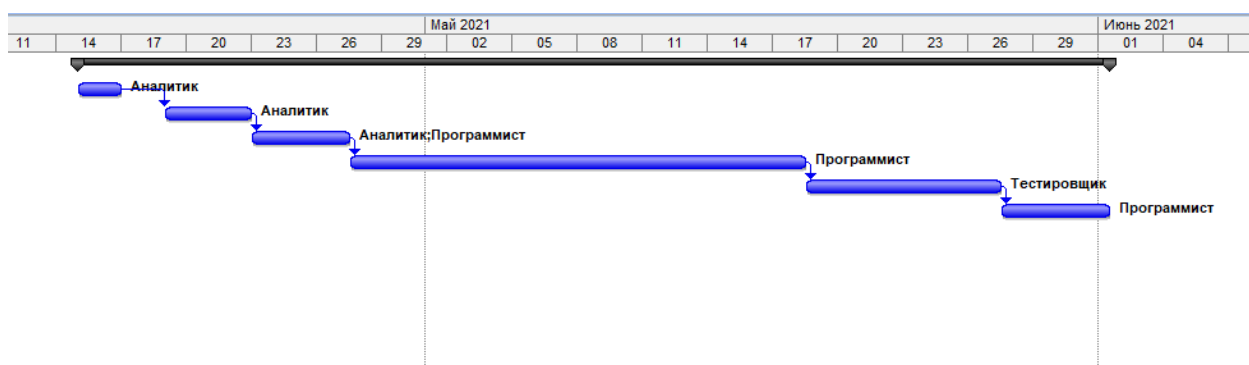


Рисунок 15 – Диаграмма Ганта

Результаты вычислений представлены в таблице 5.

Таблица 5 – Вычисления бюджета

Название задачи	Затраты
Анализ предметной области	1 600,00 р
Постановка задачи	3 200,00 р
Проектирование	4 000,00 р
Реализация	12 000,00 р
Тестирование	5 600,00 р
Внедрение	2 400,00 р
Итого	28 800,00 р

По результатам отчета видно, что фактические затраты реализации приложения составили 28 800,00 р., длительность проекта, с учетом праздников и выходных дней, составила 33,5 дня.

3 Разработка программных средств «Системы 112»

3.1 Физическая модель базы данных

Физическая модель базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т.п.

Физическая модель данных для разрабатываемой системы представлена на рисунке 16.

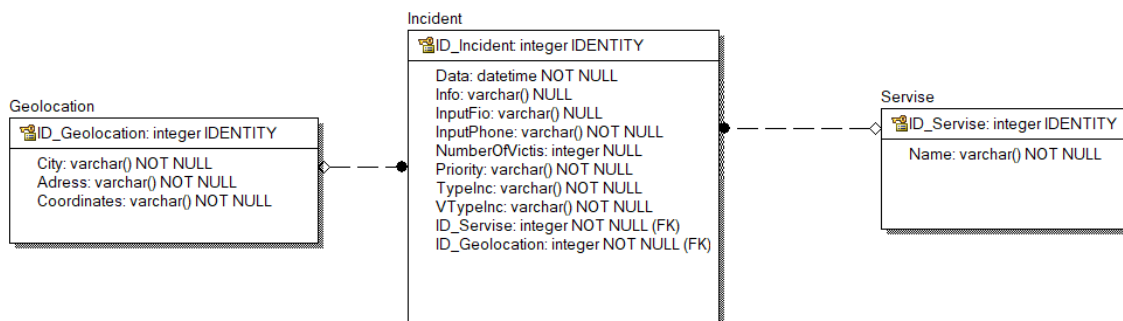


Рисунок 16 – Физическая модель БД

Описание сущностей и их атрибутов логической модели представлены в таблице 6.

Таблица 6 – Описание сущностей и их атрибутов физической модели

Сущность	Атрибуты	Роль атрибута	Описание атрибута
1	2	3	4
Incident	ID_Incident: integer Identity	PK	Уникальный код происшествия
	VTypeInc: varchar() Not Null		Вид происшествия, о котором сообщил заявитель
	ID_Geolocation: integer Not Null	FK	Уникальный код расположения происшествия
	Data: datetime Not Null		Дата и время, когда произошло происшествие
	ID_Service: integer Not Null	FK	Уникальный номер служб
	InputPhone: varchar() Not Null		Номер телефона обратившегося в службу
	InputFio: varchar() Null		ФИО обратившегося в службу
	NumberOfVictis integer Not Null		Количество пострадавших в происшествии

Продолжение таблицы 6

1	2	3	4
	Priority: varchar() Not Null		Приоритет происшествия
	TypeInc: varchar() Not Null		Тип происшествия
	VTypeInc: varchar() Not Null		Вид происшествия
	Info: varchar() Null		Дополнительная информация о происшествии
Geolocation	ID_Geolocation: integer Identity	PK	Уникальный код расположения происшествия
	Sity: varchar() Not Null		Город, в котором произошло происшествие
	Adress: varchar() Not Null		Адрес, в котором произошло происшествие
	Coordinates: varchar() Not Null		Координаты, где произошло происшествие
Service	ID_Service: integer Identity	PK	Уникальный код службы
	Name: varchar() Not Null		Наименование службы

На основании физической модели базы данных средствами ERwin реализован код SQL (приложение Б, листинг А.1).

3.2 Реализация программных средств

3.2.1 Диаграмма компонентов

Компонент представляет модульную часть системы, которая инкапсулирует ее содержимое, и реализация компонента замещается в рамках его окружения. Компонент как черный ящик, внешнее поведение которого полностью определяется его предоставляемыми и требуемыми интерфейсами. Поэтому один компонент может быть заменен другим, поддерживающим тот же протокол.

Компоненты могут иметь атрибуты и операции и участвовать в отношениях ассоциации и обобщения. Компоненты – это структурированные классификаторы. У них может быть внутренняя структура, включающая части и соединители.

Компонент может быть представлен одним или более артефактами. Артефакт представляет некоторую физическую сущность, например исходный файл.

Диаграмма компонентов представлена на рисунке 17.

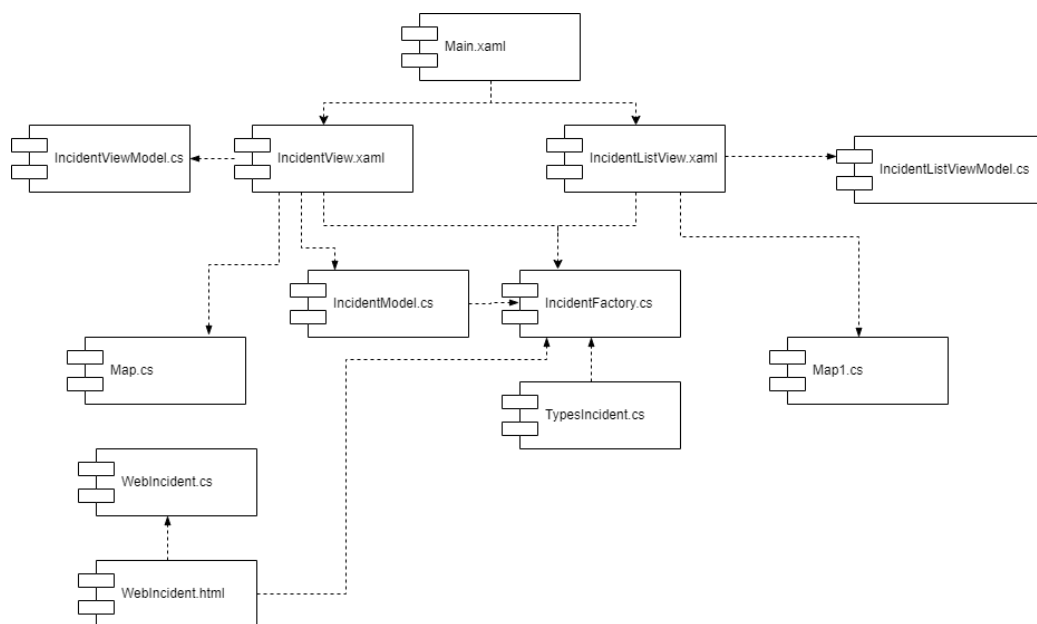


Рисунок 17 – Диаграмма компонентов

Краткое описание каждого компонента приведено далее:

- Main.xaml – главная форма приложения;
 - IncidentView.xaml – форма для заполнения заявок;
 - IncidentListView – форма для вывода списка заявок;
 - IncidentViewModal.cs – класс для подготовки данных для ввода или вывода на форму IncidentView;
 - IncidentListModel.cs – класс для подготовки данных для ввода или вывода на форму IncidentListView;
 - IncidentModel.cs – класс описывающий заявку(происшествие);
 - IncidentFactory – класс для хранения информации о видах и типах происшествия, а также содержит методы для доступа к ним;
 - Map.cs – карта для отображения происшествия;
 - Map1.cs – карта для отображения списка происшествий;
 - TypesIncident предназначен для хранения информации о видах и типах происшествия;
 - WebIncident.cs/html предназначены для получения заявки с сайта.
- Таким образом были выделены основные компоненты

информационной системы.

3.2.2 Диаграмма развертывания

В UML развертывание – это процесс распределения артефактов по узлам или экземплярам артефактов по экземплярам узлов.

Диаграмма развертывания определяет физическое оборудование, на котором будет выполняться программная система, а также описывает, как программное обеспечение развертывается на это оборудование.

Диаграмма развертывания проецирует программную архитектуру, созданную при проектировании, на исполняющую ее физическую архитектуру системы. В распределенных системах она моделирует распределение программного обеспечения по физическим узлам. Диаграмма развертывания представлена на рисунке 18.

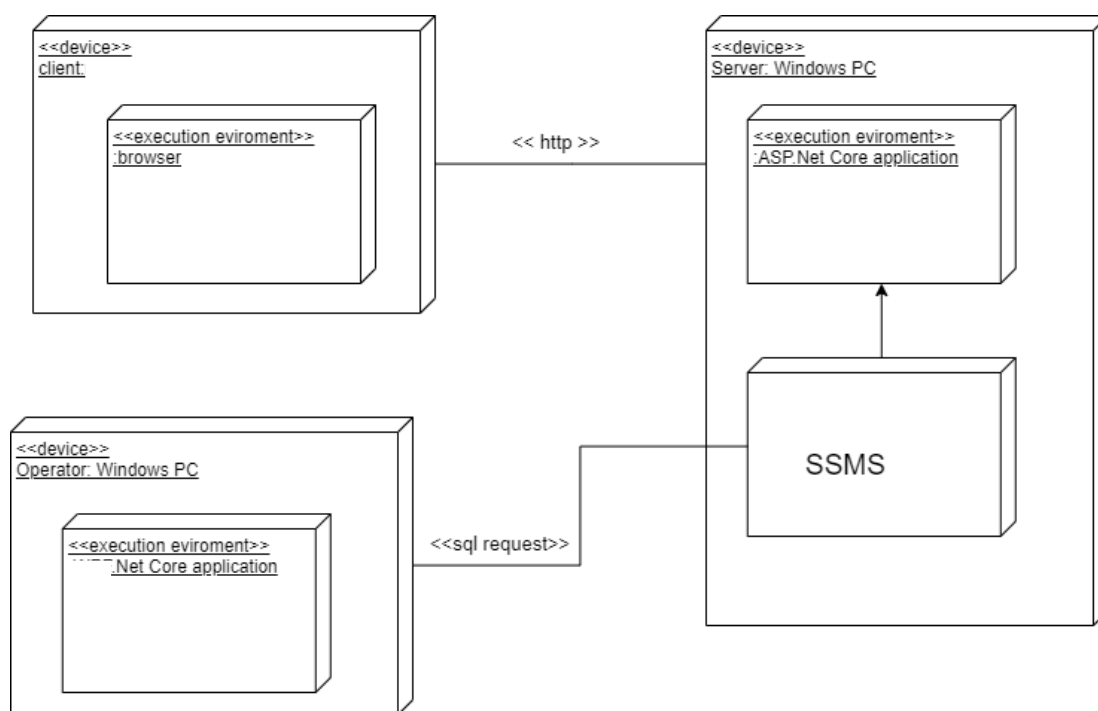


Рисунок 18 – Диаграмма развертывания

Пользователь подает заявку через браузер. Через протокол HTTP данные попадают на сервер, на сервере происходит добавление заявки в БД. Оператор принимает телефонную заявку пользователя и добавляет её в БД.

Также оператор имеет возможность получить список происшествий для дальнейшего анализа.

3.3. Разработка пользовательского интерфейса

Интерфейс для приема происшествий разработан с помощью WPF на C# в Visual Studio 19. Интерфейс приема и обработки происшествия представлен на рисунке 19.

Прием происшествия

Горячая перезагрузка

Дата и время происшествия

Входящий номер

ФИО заявителя

Вид происшествия

Количество пострадавших

Приоритет

Тип происшествия

Службы

Место происшествия

Координаты

Город

Улица

Дом

Квартира

Этаж

Доп. Инф

Показать на карте

Записать

Рисунок 19 – Интерфейс для приема и обработки происшествия

Для заполнения информации о входящем номере, ФИО заявителя,

количестве пострадавших, месте происшествия, координатах происшествия используется компонент `textbox`.

Вид, тип происшествия и приоритет заполняются с помощью компонента `ComboBox`.

Выбор необходимых служб происходит средствами компонента `CheckBox`.

Кнопка (компонент `Button`) «Показать на карте» открывает форму `Map.cs` и передает туда координаты происшествия.

Кнопка (компонент `Button`) «Записать» вызывает функцию, которая считывает все заполненные данные о происшествии и создает объект класса `Incident`.

Пример работы формы представлен на рисунке 20.

Прием происшествия

Дата и время происшествия: 19/06/2021 12:00:00

Входящий номер: 89615552358

ФИО заявителя: Кривокоров Г.В.

Вид происшествия: Попадание инородного тела

Количество пострадавших: 1

Приоритет: Низкий

Тип происшествия: Через кожу (край или обло.)

Службы:

- ☒ Скорая
- ☐ Полиция
- ☐ Пожарная
- ☐ Антитеррор
- ☐ Газовая
- ☐ Реагирования в чс

Место происшествия:

Город: Пенза

Улица: Красная

Дом: 64а

Квартира: 29

Этаж: 1

Доп. Инф:

Координаты: 53,187888 45,006404

Показать на карте

Записать

Рисунок 20 – Интерфейс для приема и обработки происшествия

Интерфейс для работы с картой создан с помощью Windows Forms и GMap. Интерфейс представлен на рисунке 21.

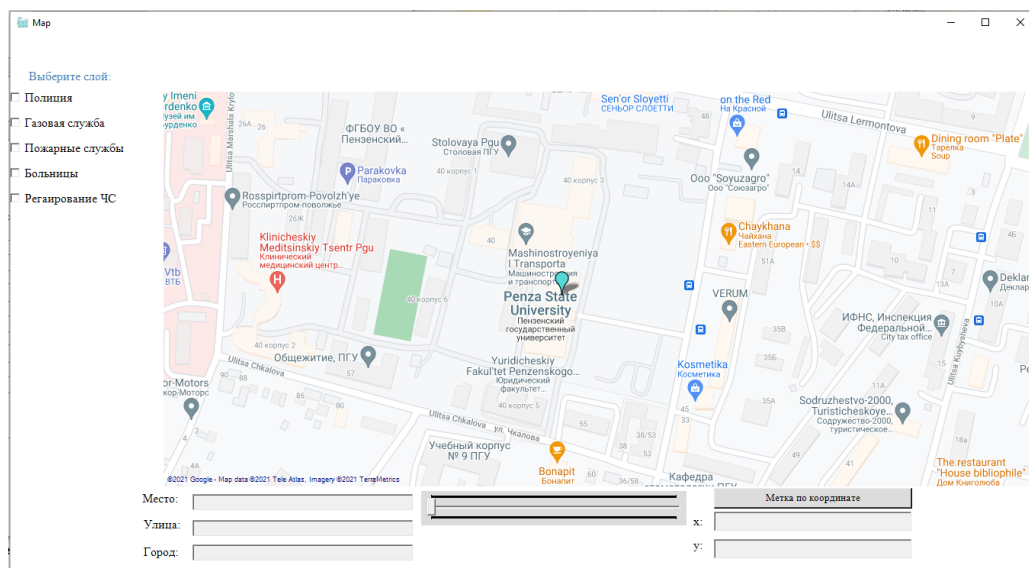


Рисунок 21 – Интерфейс для работы с картой

С помощью компонентов CheckBox пользователь может выбрать службы, которые необходимо отобразить на карте.

TrackBar необходим для вращения карты.

Кнопка «Метка по координатам» позволяет считать координаты из TextBox ниже и поставить метку.

Пример использования интерфейса представлен на рисунках 22, 23.

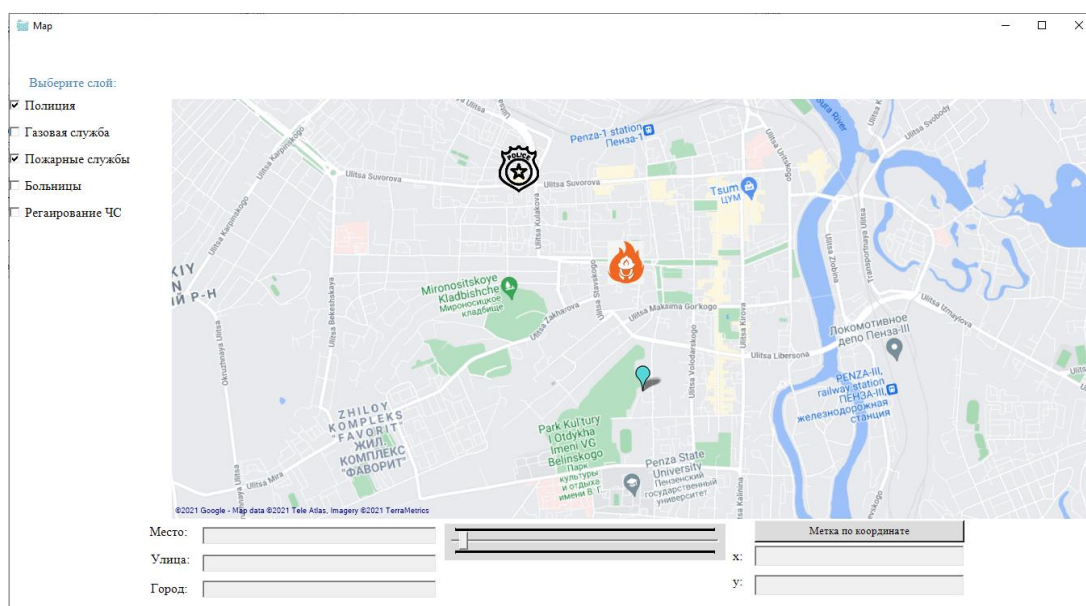


Рисунок 22 – Пример использования интерфейса для работы с картой

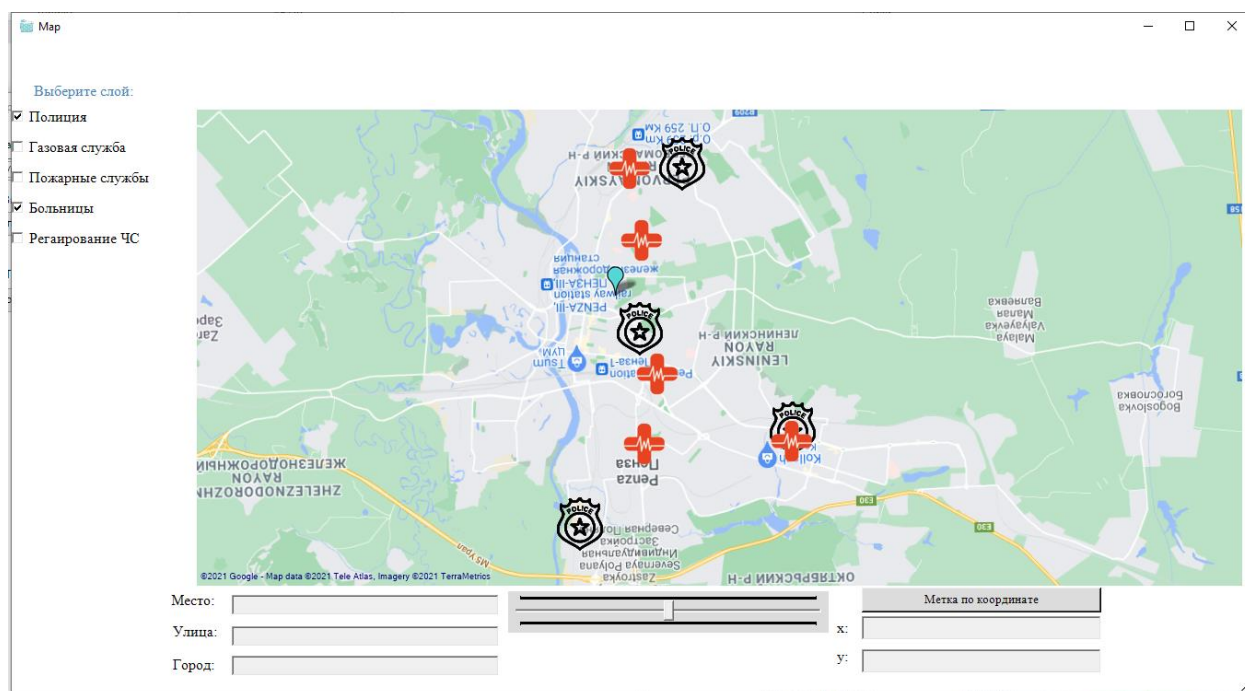


Рисунок 23 – Пример использования интерфейса для работы с картой

Аналогичным интерфейсом является интерфейс для анализа происшествий на карте (рисунок 24)

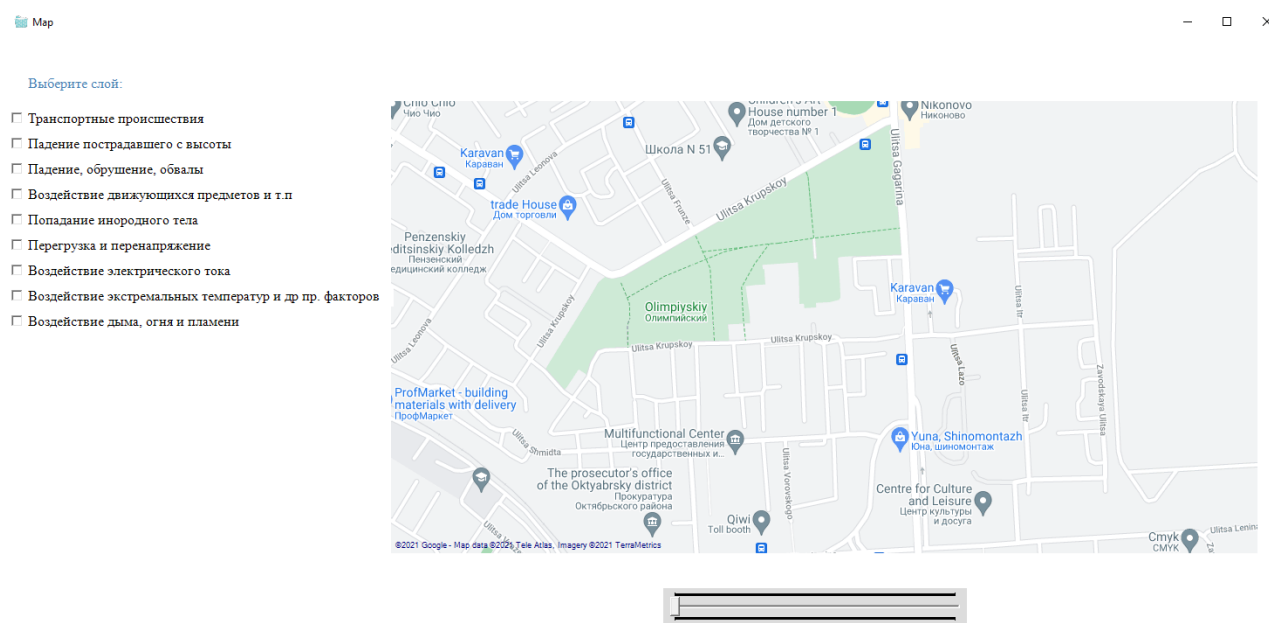


Рисунок 24 – Использование интерфейса для анализа происшествий

Пример работы интерфейса представлен на рисунке 25.

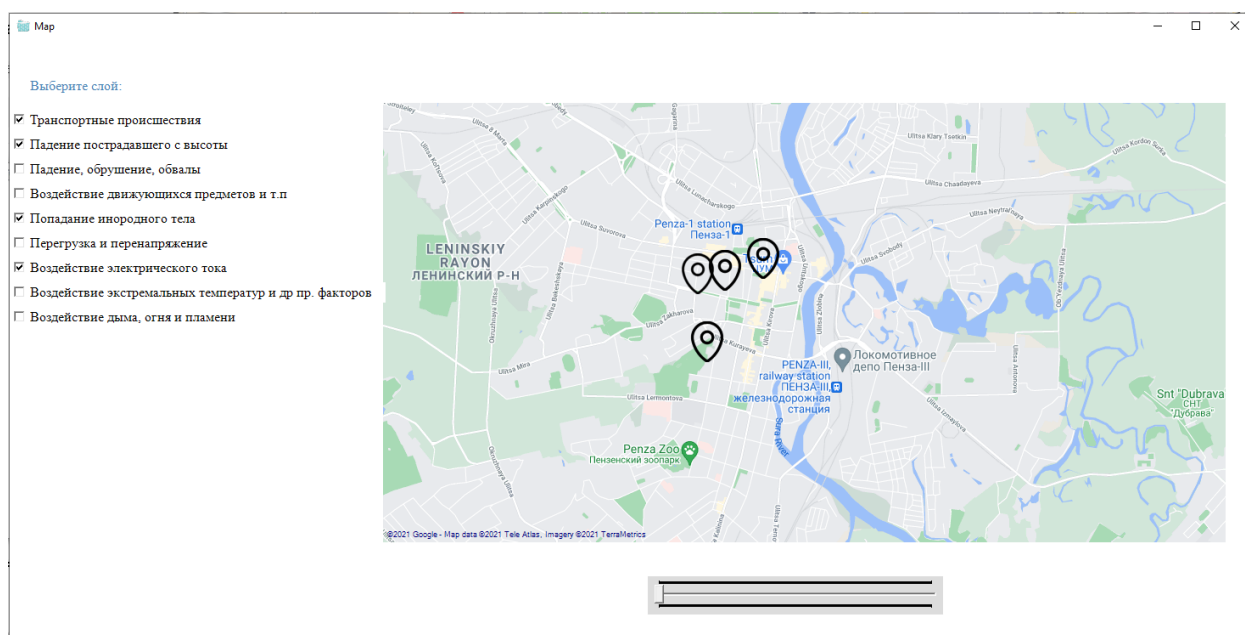


Рисунок 25 – Пример использования интерфейса для работы с картой

Интерфейс для анализа списка заявок написан с помощью WPF, на языке C#. Интерфейс представлен на рисунке 26.

Время	Вид	Тип	Приоритет	Адрес	Службы
6/19/2021 12:00:00 AM	Попадание инородного тела	Через кожу (край или обломок другого предмета, заноза и т.п.)	Низкий	ПензаКрасная64а291	Скорая;
6/19/2021 12:00:00 AM	Транспортные происшествия	на железнодорожном транспорте	Средний	ПензаПушкина30	Пожарная; Полиция; Скорая;
6/19/2021 12:00:00 AM	Падение пострадавшего с высоты	Падение при разности уровней высот	Средний	ПензаПушкина7	Пожарная; Полиция; Скорая;
6/19/2021 12:00:00 AM	Воздействие электрического тока	Искусственного электричества	Низкий	ПензаПушкина7	Пожарная; Скорая;
6/19/2021 12:00:00 AM	Физические перегрузки и перенапряжения	Чрезмерные физические усилия при толкании или демонтаже предметов и деталей	Низкий	ПензаКрасная7	Пожарная; Скорая;

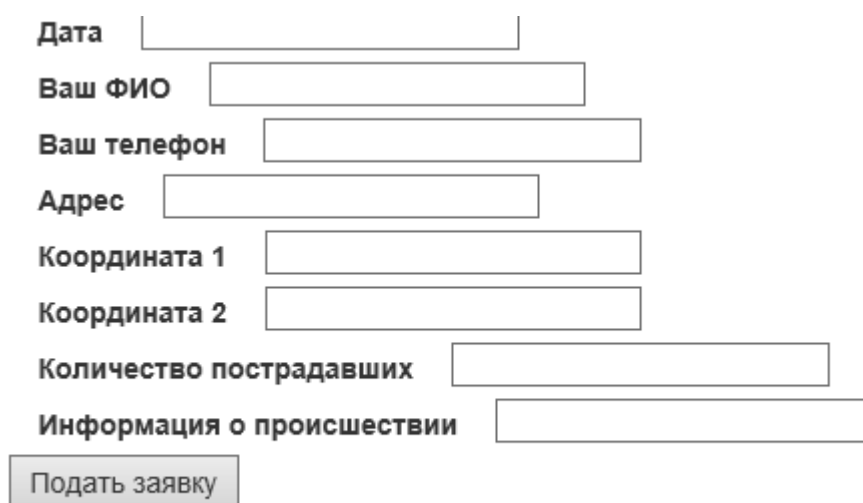
Показать происшествия на карте

Рисунок 26 – Интерфейс для анализа списка заявок

Компонент DataGrid служит для отображения списка происшествий на форме.

Кнопка «Показать происшествия на карте» открывает интерфейс для анализа происшествий.

Веб форма написана на HTML и изображена на рисунке 27.



The image shows a web form with the following fields and labels:

- Дата
- Ваш ФИО
- Ваш телефон
- Адрес
- Координата 1
- Координата 2
- Количество пострадавших
- Информация о происшествии
-

Рисунок 27 – Интерфейс для подачи заявки с сайта

Данные о происшествии вводятся в компоненты TextBox. Компонент Submit служит для обработки заявки.

4 Тестирование системы

4.1 Структурное тестирование

Структурное тестирование называют также тестированием по «маршрутам», так как в этом случае тестовые наборы формируют путем анализа маршрутов, предусмотренных алгоритмом. Под маршрутами при этом понимают последовательности операторов программы, которые выполняются при конкретном варианте исходных данных.

В основе структурного тестирования лежит концепция максимально полного тестирования всех маршрутов программы. Так, если алгоритм программы включает ветвление, то при одном наборе исходных данных может быть выполнена последовательность операторов, реализующая действия, которые предусматривает одна ветвь, а при втором - другая. Соответственно для программы будут существовать маршруты, различающиеся выбранным при ветвлении вариантом.

Считают, что программа проверена полностью, если с помощью тестов удастся осуществить выполнение программы по всем возможным маршрутам передач управления. Однако нетрудно заметить, что даже в программе среднего уровня сложности число неповторяющихся маршрутов может быть очень велико, и, следовательно, полное или исчерпывающее тестирование маршрутов, как правило, невозможно.

Алгоритм работы системы представлен на рисунке 28.



Рисунок 28 – Алгоритм прохождения тестирования

По алгоритму работу программы можно построить граф переходов системы, который представлен на рисунке 29.

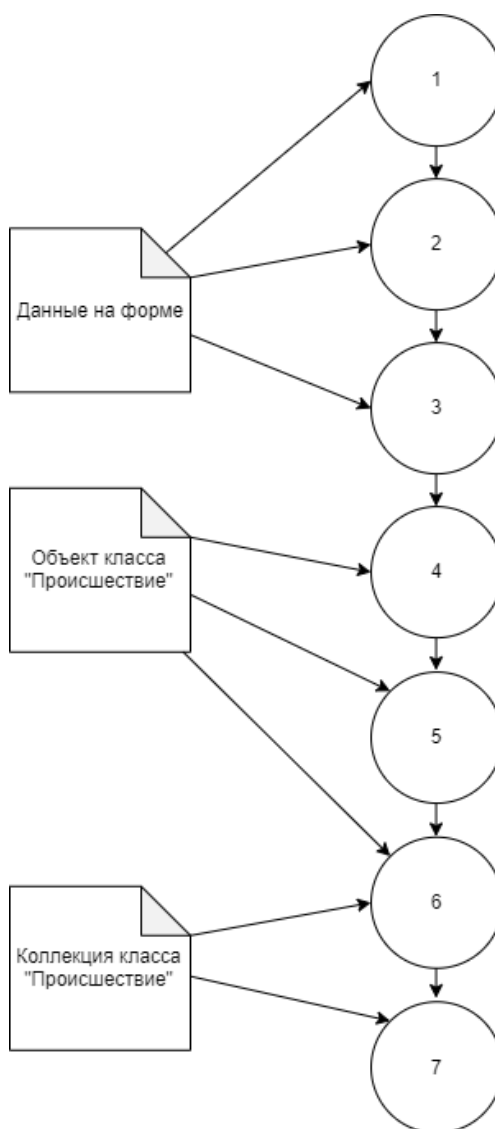


Рисунок 29 – Граф переходов системы

Покрытие операторов представлено в таблице 7.

Таблица 7 – Покрытие операторов

№ оператора	Входные данные	Предполагаемый результат	Фактический результат
1, 2, 3, 4, 5, 6, 7	Дата происшествия "09.06.2021"; ФИО заявителя "Кривокоров Г.В."; Входящий номер "89615552358"; Вид происшествия "Пожар"; Тип происшествия "В доме"; Количество пострадавших "3"; Приоритет "Средний"; Служба "Пожарная"; Адрес "Красная 64 а"; Координаты "53.187888, 45.006404";	Создан объект класса, добавлен в коллекцию и выведен на форму	Создан объект класса, добавлен в коллекцию и выведен на форму

Покрытие маршрутов представлено в таблице 8

Таблица 8 – Покрытие маршрутов

№ оператора	Входные данные	Предполагаемый результат	Фактический результат
1, 2, 3, 4, 5,6,7	Дата происшествия “09.06.2021”; ФИО заявителя “Кривокоров Г.В.”; Входящий номер ”89615552358”; Вид происшествия “Пожар”; Тип происшествия “В доме”; Количество пострадавших “3”; Приоритет “Средний”; Служба “Пожарная”; Адрес “Красная 64 а”; Координаты “53.187888, 45.006404”;	Создан объект класса, добавлен в коллекцию и выведен на форму	Создан объект класса, добавлен в коллекцию и выведен на форму

Таким образом, проанализировав результаты структурного тестирования можно сделать вывод о том, что разработанная информационная система работает корректно и соответствует разработанным требованиям.

4.2 Метрики кода

4.2.1 Метрика кода средствами среды разработки Visual Studio

Метрика программного обеспечения – это некая мера определенного свойства программного обеспечения или же его спецификаций. Как известно, мера – это средство измерения. Важно понять, что мера – это числовое значение. Таким образом, метрика программного обеспечения будет показывать некое числовое значение определенного свойства ПО.

Метрики кода, посчитанные средствами Visual Studio 2019 представлено на рисунке 30.

Иерархия	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов	Строки исходного кода	Строки исполняемого кода
112 (Debug)	74	561	9	226	5 201	2 135

Рисунок 30 – Метрики кода средствами Visual Studio

В информационной системе автоматизации работы службы 112 индекс удобства поддержки равен 74, сложность организации циклов равен 561, глубина наследования равна 9, взаимозависимость классов равна 226, строки исходного кода равна 5201, количество строк исполняемого кода равно 2135.

4.2.2 Метрики Холстеда

Метрики Холстеда включены в ряд современных коммерческих инструментов, которые подсчитывают количество строк программного кода. Подсчитывая и определяя, какие являются операторами, а какие – операндами, можно собрать следующие базовые меры:

- $n1$ – количество различных операторов;
- $n2$ – количество различных операндов;
- $N1$ – общее количество появлений операторов;
- $N2$ – общее количество вхождений операндов.

В дополнение к вышесказанному, Холстед определяет следующее:

- $n1^*$ – количество потенциальных операторов;
- $n2^*$ – количество потенциальных операндов.

В данной работе расчет метрик приведен по метрикам Холстеда для функции получения данных из полей формы, представленного на рисунке 31.

```

private void Button_Click(object sender, RoutedEventArgs e)
{
    DateTime date = Convert.ToDateTime(Picker.SelectedDate);
    var inputPhone = InputPhone.Text.ToString();
    var inputName = InputName.Text.ToString();
    var index = Vtype.SelectedIndex;
    String typeIncident = Type.SelectedItem.ToString();
    int numberOfVictims = Convert.ToInt32(NumberOfVictims.Text);
    var priority = Priority.Text.ToString();
    var Address = City.Text.ToString() + Street.Text.ToString() + House.Text.ToString() + Flat.Text.ToString() + Floor.Text.ToString();
    string coordinates1 = Coordinates1.Text.ToString();
    string coordinates2 = Coordinates2.Text.ToString();
    List<String> service = new List<String>();
    String info = Info.Text.ToString();
    if (c01.IsChecked == true)
    {
        service.Add("Пожарная");
    }
    if (c02.IsChecked == true)
    {
        service.Add("Полиция");
    }
    if (c03.IsChecked == true)
    {
        service.Add("Скорая");
    }
    if (c04.IsChecked == true)
    {
        service.Add("Газовая");
    }
    if (c05.IsChecked == true)
    {
        service.Add("Антитеррор");
    }
    if (c06.IsChecked == true)
    {
        service.Add("Реагирования в час");
    }
    vm.SetIncident(date, inputPhone, inputName, index, typeIncident,
        numberOfVictims, priority, Address, coordinates1, coordinates2,
        service, info);
}

```

Рисунок 31 – Функция получения данных с формы

Полученные результаты находятся в таблице 9.

Таблица 9 – Словарь операндов и операторов

Операторы	Число вхождений	Операнды	Число вхождений
=	12	date	2
==	6	inputPhone	2
If()	6	inputName	2
+	4	index	2
		typeIncident	2
		numberOfVictims	2
		priority	2
		Adress	2
		coordinates1	2
		coordinates2	2
		service	2
		info	2
n1 = 4	N1 =28	n2 = 12	N2=24

На основе полученных данных определяем показатели программы (таблица 10).

Таблица 10 – Показатели программы

Обозначение	Расшифровка	Формула	Полученное значение
n	Словарь программы	$n = n_1 + n_2$	16
N	Длинна программы	$N = N_1 + N_2$	52
V	Объем кода	$N * \log_2 n$	5,7
L'	Уровень качества программирования	$L' = (2 * n_2) / (n_1 * N_2)$	0,25
Ec	Сложность качества	$Ec = (V / L') * 2$	45,6

Таким образом, были рассчитаны показатели для написанной процедуры.

Заключение

В рамках данной выпускной квалификационной работы был выполнен анализ предметной области, в котором была рассмотрена актуальность создания информационной системы автоматизации работы службы 112, также был проведен обзор систем аналогов.

С помощью многофункционального языка UML и Microsoft Project система была спроектирована, а также был распланирован бюджет.

Реализация система проходила в среде разработки Visual Studio 2019. Использовались технологий WPF, Windows Forms. Система написана на языке C#.

В рамках тестирования использовались различные методологии: метрики кода, структурное тестирование, тестирование средствами среды разработки.

В результате была разработана информационная система автоматизации работы службы 112.

Список использованных источников

1. Постановление Правительства РФ от 21 ноября 2011 г. N 958 «О системе обеспечения вызова экстренных оперативных служб по единому номеру «112»».
2. Сайт МЧС России Постановления Правительства РФ [Электронный ресурс]. – Режим доступа: <https://www.mchs.gov.ru/dokumenty/postanovleniya-pravitelstva-rf/623> (Дата обращения 25.04.2021).
3. Центр обработки вызовов системы «112» [Электронный ресурс]. – Режим доступа: <http://www.cov112-48.ru/system112/history> (Дата обращения 24.04.2021).
4. Постановление Правительства ЯНАО от 24 декабря 2019 года N 1377-П «О системе обеспечения вызова экстренных оперативных служб по единому номеру «112» на территории Ямало-Ненецкого автономного округа».
5. Постановление Правительства Московской Области от 18 июля 2017 года N 598/24 «Об утверждении Регламента информационного взаимодействия при обеспечении вызова экстренных оперативных служб по единому номеру «112» Московской области».
6. Постановление Правительства Московской области от 25.02.2016 № 143/5 «Об утверждении Положения о системе обеспечения вызова экстренных оперативных служб по единому номеру «112» на территории Московской области».
7. Постановление Правительства Ульяновской области от 28 октября 2013 г. N 502-П «Положение о создании и функционировании системы обеспечения вызова экстренных оперативных служб по единому номеру «112» на территории Ульяновской области».

8. Арлоу, Д. UML 2 и унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу, И. Нейштадт – СПб: СимволПлюс, 2007. – 624 с.
9. И.Б. Балашова, Е.Н. Прошкина Язык моделирования информационных процессов UML 2.0. Учебное пособие. – Пенза:ПГУ, 2018. – 121с.
10. Прошкина, Е. Н. Методы и средства проектирования информационных систем и технологий. Методические указания к выполнению лабораторных работ / Е. Н. Прошкина. – Пенза: ПГУ, 2014. – 92с.
11. Буч Г., Рамбо Дж., Якобсон И. Введение в UML от создателей языка. – Изд-во «ДМК Пресс», 2015. – 496с.
12. Ларман К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку, 3-е издание. – Изд-во: «Вильямс», 2013. – 736с.
13. Леоненков А.В. Самоучитель UML 2. – Изд-во: «БХВ-Петербург», 2007. – 576с.
14. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка, 2-е издание. – Изд-во: «Питер», 2007. 544с.
15. Скотт К. Фаулер М. UML. Основы. – СПб.: Символ, 2006. – 184с.
16. Windows Forms // Wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Windows_Forms (дата обращения: 03.06.2021).
17. Общие сведения о WPF [Электронный ресурс]. – URL: <https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8> (Дата обращения 02.05.2021).

Приложение А
ГЛОССАРИЙ
(справочное)

Информационная система (ИС) – взаимосвязанный комплекс средств, методов и трудовых ресурсов, который направлен на хранение, модификацию и выдачу информации для дальнейшего использования.

БД – база данных.

Выпускная квалификационная работа (ВКР) – это итоговая аттестационная работа студента, выполненная им на последнем курсе обучения. ВКР оформляется в письменном виде в соответствии с методическими указаниями и с соблюдением всех правил написания таких работ, на завершающем этапе обучения подлежит защите перед государственной аттестационной комиссией.

Программное обеспечение (ПО) – это совокупность программ, позволяющих взаимодействовать с компьютером, решая с его помощью все необходимые задачи.

МТО – Материально-техническое обеспечение.

SMS – сервис коротких сообщений (Short Messaging Service), осуществляющий передачу небольшого объема данных в сети сотовой связи.

ЦОВ – основной центр обработки вызовов.

РЦОВ – резервный центр обработки вызовов.

Приложение Б
ЛИСТИНГ ПРОГРАММЫ
(рекомендуемое)

Листинг А.1 – Код базы данных

```

CREATE TABLE Geolocation
(
    ID_Geolocation    integer IDENTITY ( 1,1 ) ,
    City              varchar() NOT NULL ,
    Adress             varchar() NOT NULL ,
    Coordinates        varchar() NOT NULL
)
go
ALTER TABLE Geolocation
    ADD CONSTRAINT XPKGeolocation PRIMARY KEY CLUSTERED
(ID_Geolocation ASC)
go
CREATE TABLE Incident
(
    ID_Incident        integer IDENTITY ( 1,1 ) ,
    Data                datetime NOT NULL ,
    InputFio            varchar() NULL ,
    Info                varchar() NULL ,
    InputPhone          varchar() NOT NULL ,
    NumberOfVictis      integer NULL ,
    Priority             varchar() NOT NULL ,
    TypeInc             varchar() NOT NULL ,
    VTypeInc            varchar() NOT NULL ,
    ID_Servise           integer NOT NULL ,
    ID_Geolocation       integer NOT NULL
)
go
ALTER TABLE Incident
    ADD CONSTRAINT XPKIncident PRIMARY KEY CLUSTERED (ID_Incident ASC)
go
CREATE TABLE Servise
(
    ID_Servise          integer IDENTITY ( 1,1 ) ,
    Name                varchar() NOT NULL
)
go
ALTER TABLE Servise
    ADD CONSTRAINT XPKServise PRIMARY KEY CLUSTERED (ID_Servise ASC)
Go

```

Листинг А.2 – Класс IncidentModel

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _112.Model

```

```

{
public class IncidentModel : INotifyPropertyChanged
{

    public IncidentModel ()
    {
        data = new DateTime();
        data = DateTime.Now;
        inputPhone = "890000000000";
        inputName = "";
        vtype = "";
        typeIncident = "";
        numberOfVictims = 0;
        priority = "";
        Adress = "";
        coordinates1 = "";
        coordinates2 = "";
        services = new List<string>();
        info = "";
    }
    public IncidentModel(DateTime data, String inputPhone, String inputName, String vtype,
String typeInciden, int numberOfVictims, String priority, String address, String coordinates1,
String coordinates2, List<String> services, String info)
    {
        this.Data = data;
        this.InputPhone = inputPhone;
        this.InputName = inputName;
        this.VType = vtype ;
        this.TypeIncident = typeInciden;
        this.NumberOfVictim = numberOfVictims ;
        this.Priority = priority;
        this.Adress = address;
        this.Coordinates1 = coordinates1;
        this.Coordinates2 = coordinates2;
        this.Services = services;
        this.Info = info;
    }
    //Дата и время происшествия
    private DateTime data;

    public DateTime Data
    {
        get { return data; }
        set { data = value; OnPropertyChanged("Data"); }
    }

    //Входящий телефон
    private String inputPhone;

    public String InputPhone
    {
        get { return inputPhone; }
    }
}

```

```

    set { inputPhone = value; OnPropertyChanged("InputPhone"); }
}
//Имя заявителя
private String inputName;

public String InputName
{
    get { return inputName; }
    set { inputName = value; OnPropertyChanged("InputName"); }
}
//Вид происшествия
private String vtype;

public String VType
{
    get { return vtype; }
    set { vtype = value; OnPropertyChanged("VType"); }
}
//Тип происшествия
private String typeIncident;

public String TypeIncident
{
    get { return typeIncident; }
    set { typeIncident = value; OnPropertyChanged("TypeIncident"); }
}
//Количество пострадавших
private int numberOfVictims;

public int NumberOfVictim
{
    get { return numberOfVictims; }
    set { numberOfVictims = value; OnPropertyChanged("NumberOfVictim"); }
}

//Приоритет
private String priority;

public String Priority
{
    get { return priority; }
    set { priority = value; OnPropertyChanged("Priority"); }
}

//Адресс
private String adress;

public String Adress
{
    get { return adress; }
    set { adress = value; OnPropertyChanged("Adress"); }
}

```

```

//Координаты
private String coordinates1;

public String Coordinates1
{
    get { return coordinates1; }
    set { coordinates1= value; OnPropertyChanged("Coordinates1"); }
}
private String coordinates2;
public String Coordinates2
{
    get { return coordinates2; }
    set { coordinates2 = value; OnPropertyChanged("Coordinates2"); }
}
private List<String> services;
public String ServicesString
{
    get
    {
        String buf="";
        for (int i = 0; i < services.Count; i++)
        {
            buf = buf + services[i]+"\\n";
        }
        return buf;
    }
}

public List<String> Services
{
    get
    {
        return services;
    }
    set
    {
        services = value;
        OnPropertyChanged("Services");
    }
}

private String info;

public String Info
{
    get { return info; }
    set { info = value; }
}

//Интерфейс OnPropertyChanged

```

```

#region Property
public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName = "")
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
}
#endregion
}
}

```

Листинг А.3 – Класс TypesIncident

```

using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _112.Model
{
    public class TypesIncident
    {
        private ObservableCollection<string> type_childrens = new
        ObservableCollection<string>();

        public string VtypeName { get; set; }
        public ObservableCollection<string> Childrens
        {
            get { return type_childrens; }
            set { type_childrens = value; }
        }
    }
}

```

Листинг А.4 – Класс IncidentFactory

```

using _112.Model;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace _112.Sup
{
    public class IncidentFactory : INotifyPropertyChanged
    {
        #region Singletin
        private static readonly Lazy<IncidentFactory> instance =
            new Lazy<IncidentFactory>(() => new IncidentFactory());
        public static IncidentFactory Instance { get { return instance.Value; } }
        #endregion
        private ObservableCollection<TypesIncident> TypesIncident = new
ObservableCollection<TypesIncident>();
        private ObservableCollection<IncidentModel> incidentModels;
        private ObservableCollection<IncidentModel> webIncidentModels;

        private IncidentFactory()
        {
            incidentModels = new ObservableCollection<IncidentModel>();
            webIncidentModels = new ObservableCollection<IncidentModel>();
            #region Типы происшествий
            TypesIncident typesIncidents;
            typesIncidents = new TypesIncident { VtypeName = "Транспортные происшествия" };
            TypesIncident.Add(typesIncidents);
            typesIncidents.Children.Add("на железнодорожном транспорте");
            typesIncidents.Children.Add("на водном транспорте");
            typesIncidents.Children.Add("на воздушном транспорте");
            typesIncidents.Children.Add("на наземном транспорте");

            typesIncidents = new TypesIncident { VtypeName = "Падение пострадавшего с
высоты" };
            TypesIncident.Add(typesIncidents);
            typesIncidents.Children.Add("Падение на ровной поверхности одного уровня");
            typesIncidents.Children.Add("Падение при разности уровней высот");

            typesIncidents = new TypesIncident { VtypeName = "Падение, обрушение, обвалы
предметов, материалов, земли и пр." };
            TypesIncident.Add(typesIncidents);
            typesIncidents.Children.Add("Обрушение и осыпь земляных масс, скал, камней,
снега и др.");
            typesIncidents.Children.Add("Обвалы зданий, стен, строительных лесов, лестниц,
складированных товаров и др.");
            typesIncidents.Children.Add("Удары падающими предметами и деталями (включая
их осколки и частицы) при работе (обращении) с ними");
            typesIncidents.Children.Add("Удары случайными падающими предметами");
            typesIncidents = new TypesIncident { VtypeName = "Воздействие движущихся,
разлетающихся, вращающихся предметов, деталей, машин и т.д." };
            TypesIncident.Add(typesIncidents);
            typesIncidents.Children.Add("Контактные удары (ушибы) при столкновении с
движущимися предметами, деталями и машинами (за исключением случаев падения
предметов и деталей), в том числе в результате взрыва ");
            typesIncidents.Children.Add("Контактные удары (ушибы) при столкновении с

```

```

неподвижными предметами, деталями и машинами, в том числе в результате взрыва ");
    typesIncidents.Childrens.Add("Защемление между неподвижными и движущимися
предметами, деталями и машинами (или между ними)");
    typesIncidents.Childrens.Add("Защемление между движущимися предметами,
детальями и машинами (за исключением летящих или падающих предметов, деталей и
машин)");
    typesIncidents = new TypesIncident { VtypeName = "Попадание инородного тела" };
    TypesIncident.Add(typesIncidents);
    typesIncidents.Childrens.Add("Через естественные отверстия в организме");
    typesIncidents.Childrens.Add("Через кожу (край или обломок другого предмета,
заноза и т.п.)");
    typesIncidents.Childrens.Add("Вдыхание и заглатывание пищи либо инородного
предмета, приводящее к закупорке дыхательных путей ");
    typesIncidents = new TypesIncident { VtypeName = "Физические перегрузки и
перенапряжения" };
    TypesIncident.Add(typesIncidents);
    typesIncidents.Childrens.Add("Чрезмерные физические усилия при подъеме
предметов и деталей");
    typesIncidents.Childrens.Add("Чрезмерные физические усилия при толкании или
демонтаже предметов и деталей");
    typesIncidents.Childrens.Add("Чрезмерные физические усилия при переноске или
бросании предметов");

    typesIncidents = new TypesIncident { VtypeName = "Воздействие электрического
тока" };
    TypesIncident.Add(typesIncidents);
    typesIncidents.Childrens.Add("Природного электричества (молнии)");
    typesIncidents.Childrens.Add("Искусственного электричества");
    typesIncidents = new TypesIncident { VtypeName = "Воздействие экстремальных
температур и других природных факторов" };
    TypesIncident.Add(typesIncidents);
    typesIncidents.Childrens.Add("Воздействие повышенной температуры воздуха
окружающей или рабочей среды");
    typesIncidents.Childrens.Add("Воздействие пониженной температуры воздуха
окружающей или рабочей среды");
    typesIncidents.Childrens.Add("Соприкосновение с горячими и раскаленными
частями оборудования, предметами или материалами, включая воздействие пара и
горячей воды");
    typesIncidents.Childrens.Add("Соприкосновение с чрезмерно холодными частями
оборудования, предметами и материалами");
    typesIncidents.Childrens.Add("Воздействие высокого или низкого атмосферного
давления ");

    typesIncidents = new TypesIncident { VtypeName = "Воздействие дыма, огня и
пламени" };
    TypesIncident.Add(typesIncidents);
    typesIncidents.Childrens.Add("Воздействие неконтролируемого огня (пожара) в
здании или сооружении");
    typesIncidents.Childrens.Add("Воздействие неконтролируемого огня (пожара) вне
здания или сооружения, в том числе пламени от костра");
    typesIncidents.Childrens.Add("Воздействие контролируемого огня в здании или
сооружении (огня в печи, камине и т.д.)");

```

```

        typesIncidents.Childrens.Add("Повреждения при возгорании
        легковоспламеняющихся веществ и одежды");
        typesIncidents.Childrens.Add("Воздействие высокого или низкого атмосферного
        давления");
        #endregion
    }
    public ObservableCollection<IncidentModel> Incidents
    {
        get { return incidentModels; }
        set
        {
            if (value != incidentModels)
            {
                incidentModels = value;
                OnPropertyChanged("Incidents");
            }
        }
    }

    public ObservableCollection<IncidentModel> WebIncidents
    {
        get { return webIncidentModels; }
        set
        {
            if (value != webIncidentModels)
            {
                webIncidentModels = value;
                OnPropertyChanged("WebIncidents");
            }
        }
    }

    public ObservableCollection<TypesIncident> GetTypesIncident()
    {
        return TypesIncident;
    }

    #region Property
    public event PropertyChangedEventHandler PropertyChanged;
    protected virtual void OnPropertyChanged(string propertyName = "")
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
    #endregion
}

```

Листинг А.5 – Класс IncidentView

```

using _112.Sup;
using _112.ViewModel;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
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.Shapes;
namespace _112.View
{
    /// <summary>
    /// Логика взаимодействия для Incident.xaml
    /// </summary>
    public partial class IncidentView : Window
    {
        readonly IncidentViewModel vm = new IncidentViewModel();
        public IncidentView()
        {
            InitializeComponent();
            DataContext = vm;
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            DateTime date = Convert.ToDateTime(Picker.SelectedDate);
            var inputPhone = InputPhone.Text.ToString();
            var inputName = InputName.Text.ToString();
            var index = Vtype.SelectedIndex;
            String typeIncident = Type.SelectedItem.ToString();
            int numberOfVictims = Convert.ToInt32(NumberOfVictims.Text);
            var priority = Priority.Text.ToString();
            var Adress = City.Text.ToString() + Street.Text.ToString() + House.Text.ToString() +
            Flat.Text.ToString() + Floor.Text.ToString();
            string coordinates1 = Coordinates1.Text.ToString();
            string coordinates2 = Coordinates2.Text.ToString();
            List<String> service = new List<String>();
            String info = Info.Text.ToString();
            if (c01.IsChecked == true)
            {
                service.Add("Пожарная");
            }
            if (c02.IsChecked == true)

```

```

        {
            service.Add("Полиция");
        }
        if (c03.IsChecked == true)
        {
            service.Add("Скорая");
        }
        if (c04.IsChecked == true)
        {
            service.Add("Газовая");
        }
        if (c05.IsChecked == true)
        {
            service.Add("Антитеррор");
        }
        if (c06.IsChecked == true)
        {
            service.Add("Реагирования в час");
        }

        vm.SetIncident(date, inputPhone, inputName, index, typeIncident,
            numberOfVictims, priority, Address, coordinates1, coordinates2,
            service, info);
    }
    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        Map map = new Map();
        if ((Coordinates1.Text.Length != 0 & Coordinates2.Text.Length != 0) )
        {
            map.GetPoint(Convert.ToDouble(Coordinates1.Text),
                Convert.ToDouble(Coordinates2.Text));
        }
        map.ShowDialog();
    }
}
}

```

Листинг А.5 – Класс Map1

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.IO;
using System.Xml;
using System.Device.Location;
using System.Net;
// Библиотеки для карты
using GMap.NET;

```

```

using GMap.NET.MapProviders;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.WindowsForms.ToolTips;
using _112.Sup;

namespace _112
{
    public partial class Map1 : Form
    {
        private readonly IncidentFactory incidentFactory = IncidentFactory.Instance;
        // Слой
        private GMapOverlay markersOverlay;

        public CPoint cPoint;

        // Класс точка - координаты
        public class CPoint
        {
            public double x { get; set; }
            public double y { get; set; }
            public string place { get; set; }
            public string building { get; set; }
            public string street { get; set; }
            public string town { get; set; }
            public string Image;

            public CPoint() { }
            public CPoint(double _x, double _y)
            {
                x = _x;
                y = _y;
            }

            public CPoint(double _x, double _y, string _place)
            {
                x = _x;
                y = _y;
                place = _place;
            }

            public CPoint(double _x, double _y, string _building, string _street, string _town)
            {
                x = _x;
                y = _y;
                building = _building;
                street = _street;
                town = _town;
            }

            public CPoint(double _x, double _y, string _building, string _street, string _town, string
_image)

```

```

    {
        x = _x;
        y = _y;
        building = _building;
        street = _street;
        town = _town;
        Image = _image;
    }
}

// Мой ключ для АПИ
string myAPI = "";

public Map1()
{
    InitializeComponent();

    StartPosition = FormStartPosition.CenterScreen;
}

// Список точек для учреждений
List<CPoint> ListWithPoins1 = new List<CPoint>();
List<CPoint> ListWithPoins2 = new List<CPoint>();
List<CPoint> ListWithPoins3 = new List<CPoint>();
List<CPoint> ListWithPoins4 = new List<CPoint>();
List<CPoint> ListWithPoins5 = new List<CPoint>();
List<CPoint> ListWithPoins6 = new List<CPoint>();
List<CPoint> ListWithPoins7 = new List<CPoint>();
List<CPoint> ListWithPoins8 = new List<CPoint>();
List<CPoint> ListWithPoins9 = new List<CPoint>();
/*
List<CPoint> ListWithPoinsOfUser = new List<CPoint>();
List<CPoint> ListWithPoinsFromXML = new List<CPoint>();
List<CPoint> ListWithPoinsPlaces = new List<CPoint>();
List<CPoint> ListWithPoinsPict = new List<CPoint>();*/

/* GMapOverlay PointsOfMyHouse = new GMapOverlay("МойДомик");
GMapOverlay MyPosition = new GMapOverlay("Позиция");
GMapOverlay PositionsForUser = new GMapOverlay("ПозицияпоЛКМ");
GMapOverlay PositionFromTextBoxes = new GMapOverlay("ПозицияДляТекстБокса");
GMapOverlay ListOfXML = new GMapOverlay("XML");
GMapOverlay ListOfTooltip = new GMapOverlay("Надписи");
GMapOverlay ListOfPict = new GMapOverlay("Картинки");*/
GMapOverlay OverlayCLICK = new GMapOverlay("Click");
List<CPoint> ListClick = new List<CPoint>();

private void gMapControl1_Load(object sender, EventArgs e)
{

```

```

// Создание элементов меню

ToolStripMenuItem YandexMenuItem = new ToolStripMenuItem("Установить Яндекс-
карту");
ToolStripMenuItem GoogleMenuItem = new ToolStripMenuItem("Установить Google-
карту");
ToolStripMenuItem OpenCycleMapMenuItem = new ToolStripMenuItem("Установить
OpenCycleMap-карту");

ToolStripMenuItem ClearMap = new ToolStripMenuItem("Очистить карту");

contextMenuStrip1.Items.AddRange(new[] { YandexMenuItem, GoogleMenuItem,
OpenCycleMapMenuItem, ClearMap });

gmap.ContextMenuStrip = contextMenuStrip1;

// saveMenuItem.Click += saveMenuItem_Click;
YandexMenuItem.Click += YandexMenuItem_Click;
GoogleMenuItem.Click += GoogleMenuItem_Click;
OpenCycleMapMenuItem.Click += OpenCycleMapMenuItem_Click;

ClearMap.Click += ClearMap_Click;

// Настройки для компонента GMap
gmap.Bearing = 0;
// Перетаскивание левой кнопки мыши
gmap.CanDragMap = true;
// Перетаскивание карты левой кнопкой мыши
gmap.DragButton = MouseButtons.Left;

gmap.GrayScaleMode = true;

// Все маркеры будут показаны
gmap.MarkersEnabled = true;
// Максимальное приближение
gmap.MaxZoom = 18;
// Минимальное приближение
gmap.MinZoom = 2;
// Курсор мыши в центр карты
gmap.MouseWheelZoomType = MouseWheelZoomType.MousePositionWithoutCenter;

// Отключение нигативного режима
gmap.NegativeMode = false;
// Разрешение полигонов
gmap.PolygonsEnabled = true;
// Разрешение маршрутов
gmap.RoutesEnabled = true;
// Скрытие внешней сетки карты
gmap.ShowTileGridLines = false;
// При загрузке 10-кратное увеличение
gmap.Zoom = 11;
// Убрать красный крестик по центру

```

```

gmap.ShowCenter = false;

// Чья карта используется
gmap.MapProvider = GMapProviders.GoogleMap;
GMaps.Instance.Mode = AccessMode.ServerOnly;
//gmap.Position = new PointLatLng(cPoint.x, cPoint.y);

/*
    GMapMarker marker1 = new GMarkerGoogle(
new PointLatLng(cPoint.x, cPoint.y),
GMarkerGoogleType.lightblue);
    marker1.Tag = "Происшествие";

    GMapOverlay markers = new GMapOverlay("markers");
    markers.Markers.Add(marker1);
    gmap.Overlays.Add(markers);*/

// Для запросов
GMapProvider.WebProxy = WebRequest.GetSystemWebProxy();
GMapProvider.WebProxy.Credentials = CredentialCache.DefaultCredentials;

}

// Подпись полигона
void ClearMap_Click(object sender, EventArgs e)
{
    gmap.Overlays.Clear();

    checkBox1.Checked = false;
    checkBox2.Checked = false;
    checkBox3.Checked = false;
    checkBox5.Checked = false;
    checkBox6.Checked = false;
    ListWithPoins1.Clear();
    ListWithPoins2.Clear();
    ListWithPoins3.Clear();
    ListWithPoins4.Clear();
    ListWithPoins5.Clear();
    ListWithPoins6.Clear();
    ListWithPoins7.Clear();
    ListWithPoins8.Clear();
    ListWithPoins9.Clear();
    ListOf1.Clear();
    ListOf2.Clear();
    ListOf3.Clear();
}

```

```

ListOf4.Clear();
ListOf5.Clear();
ListOf6.Clear();
ListOf7.Clear();
ListOf8.Clear();
ListOf9.Clear();

OverlayCLICK.Clear();
ListClick.Clear();

markersOverlayMy111.Clear();
Areas.Clear();

polyOverlay12.Clear();
}

// Смена поставщика карт
void YandexMenuItem_Click(object sender, EventArgs e)
{
    gmap.MapProvider = GMapProviders.YandexMap;
    gmap.Zoom = 11;
    gmap.Position = new PointLatLng(53.182600, 45.000546);
}
void GoogleMenuItem_Click(object sender, EventArgs e)
{
    gmap.MapProvider = GMapProviders.GoogleMap;
    gmap.Zoom = 11;
    gmap.Position = new PointLatLng(53.182600, 45.000546);
}

void OpenCycleMapMenuItem_Click(object sender, EventArgs e)
{
    gmap.MapProvider = GMapProviders.OpenCycleMap;
    gmap.Zoom = 11;
    gmap.Position = new PointLatLng(53.182600, 45.000546);
}
private void checkBox1_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf1);

    if (checkBox1.Checked)
    {
        for (int i = 0; i < incidentFactory.Incidents.Count; i++)
        {
            if (incidentFactory.Incidents[i].VType == "Транспортные происшествия")
            {
                ListWithPoins1.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));
            }
        }
    }
}

```

```

    }
    for (int i = 0; i < ListWithPoins1.Count; i++)
    {
        GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
            (ListWithPoins1[i].x, ListWithPoins1[i].y), new Bitmap(@"Icon/metka.png"));
        marker.ToolTip = new GMapRoundedToolTip(marker);
        ListOf1.Markers.Add(marker);
    }
}
else
{
    gmap.Overlays.Remove(ListOf1);
    ListOf1.Clear();
    ListWithPoins1.Clear();
}
}

private void checkBox2_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf2);

    if (checkBox2.Checked)
    {
        for (int i = 0; i < incidentFactory.Incidents.Count; i++)
        {
            if (incidentFactory.Incidents[i].VType == "Падение пострадавшего с высоты")
            {
                ListWithPoins2.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));
            }
        }
        for (int i = 0; i < ListWithPoins2.Count; i++)
        {
            GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
                (ListWithPoins2[i].x, ListWithPoins2[i].y), new Bitmap(@"Icon/metka.png"));
            marker.ToolTip = new GMapRoundedToolTip(marker);
            ListOf2.Markers.Add(marker);
        }
    }
    else
    {
        gmap.Overlays.Remove(ListOf2);
        ListOf2.Clear();
        ListWithPoins2.Clear();
    }
}

private void checkBox3_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf3);

```

```

        if (checkBox3.Checked)
        {
            for (int i = 0; i < incidentFactory.Incidents.Count; i++)
            {
                if (incidentFactory.Incidents[i].VType == "Падение, обрушение, обвалы
предметов, материалов, земли и пр.")
                {
                    ListWithPoins3.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));
                }
            }
            for (int i = 0; i < ListWithPoins3.Count; i++)
            {
                GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
(ListWithPoins3[i].x, ListWithPoins3[i].y), new Bitmap(@"Icon/metka.png"));
                marker.ToolTip = new GMapRoundedToolTip(marker);
                ListOf3.Markers.Add(marker);
            }
        }
        else
        {
            gmap.Overlays.Remove(ListOf3);
            ListOf3.Clear();
            ListWithPoins3.Clear();
        }
    }
    private void checkBox4_Click(object sender, EventArgs e)
    {
        gmap.Overlays.Add(ListOf4);

        if (checkBox4.Checked)
        {
            for (int i = 0; i < incidentFactory.Incidents.Count; i++)
            {
                if (incidentFactory.Incidents[i].VType == "Воздействие движущихся,
разлетающихся, вращающихся предметов, деталей, машин и т.д.")
                {
                    ListWithPoins4.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));
                }
            }
            for (int i = 0; i < ListWithPoins3.Count; i++)
            {
                GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
(ListWithPoins4[i].x, ListWithPoins4[i].y), new Bitmap(@"Icon/metka.png"));
            }
        }
    }

```

```

        marker.ToolTip = new GMapRoundedToolTip(marker);
        ListOf4.Markers.Add(marker);
    }
}
else
{
    gmap.Overlays.Remove(ListOf4);
    ListOf4.Clear();
    ListWithPoins4.Clear();
}
}

private void checkBox5_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf5);
    if (checkBox5.Checked)
    {
        for (int i = 0; i < incidentFactory.Incidents.Count; i++)
        {
            if (incidentFactory.Incidents[i].VType == "Попадание инородного тела")
            {
                ListWithPoins5.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));
            }
        }
        for (int i = 0; i < ListWithPoins5.Count; i++)
        {
            GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
                (ListWithPoins5[i].x, ListWithPoins5[i].y), new Bitmap(@"Icon/metka.png"));
            marker.ToolTip = new GMapRoundedToolTip(marker);
            ListOf5.Markers.Add(marker);
        }
    }
    else
    {
        gmap.Overlays.Remove(ListOf5);
        ListOf5.Clear();
        ListWithPoins5.Clear();
    }
}

private void checkBox6_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf6);

    if (checkBox6.Checked)
    {
        for (int i = 0; i < incidentFactory.Incidents.Count; i++)
        {
            if (incidentFactory.Incidents[i].VType == "Физические перегрузки и
перенапряжения")

```

```

        {
            ListWithPoins6.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));

        }
    }
    for (int i = 0; i < ListWithPoins6.Count; i++)
    {
        GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
            (ListWithPoins6[i].x, ListWithPoins6[i].y), new Bitmap(@"Icon/metka.png"));
        marker.ToolTip = new GMapRoundedToolTip(marker);
        ListOf6.Markers.Add(marker);
    }
}
else
{
    gmap.Overlays.Remove(ListOf6);
    ListOf6.Clear();
    ListWithPoins6.Clear();
}
}
private void checkBox7_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf7);

    if (checkBox7.Checked)
    {
        for (int i = 0; i < incidentFactory.Incidents.Count; i++)
        {
            if (incidentFactory.Incidents[i].VType == "Воздействие электрического тока")
            {
                ListWithPoins7.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));
            }
        }
        for (int i = 0; i < ListWithPoins7.Count; i++)
        {
            GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
                (ListWithPoins7[i].x, ListWithPoins7[i].y), new Bitmap(@"Icon/metka.png"));
            marker.ToolTip = new GMapRoundedToolTip(marker);
            ListOf7.Markers.Add(marker);
        }
    }
    else
    {
        gmap.Overlays.Remove(ListOf7);
        ListOf7.Clear();
        ListWithPoins7.Clear();
    }
}
}

```

```

private void checkBox8_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf8);

    if (checkBox8.Checked)
    {
        for (int i = 0; i < incidentFactory.Incidents.Count; i++)
        {
            if (incidentFactory.Incidents[i].VType == "Воздействие экстремальных
температур и других природных факторов")
            {
                ListWithPoins8.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));

            }

        }

        for (int i = 0; i < ListWithPoins8.Count; i++)
        {
            GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
(ListWithPoins8[i].x, ListWithPoins8[i].y), new Bitmap(@"Icon/metka.png"));
            marker.ToolTip = new GMapRoundedToolTip(marker);
            ListOf8.Markers.Add(marker);
        }
    }
    else
    {
        gmap.Overlays.Remove(ListOf8);
        ListOf8.Clear();
        ListWithPoins8.Clear();
    }
}

private void checkBox9_Click(object sender, EventArgs e)
{
    gmap.Overlays.Add(ListOf9);

    if (checkBox9.Checked)
    {
        for (int i = 0; i < incidentFactory.Incidents.Count; i++)
        {
            if (incidentFactory.Incidents[i].VType == "Воздействие дыма, огня и пламени")
            {
                ListWithPoins9.Add(new
CPoint(Convert.ToDouble(incidentFactory.Incidents[i].Coordinates1),
Convert.ToDouble(incidentFactory.Incidents[i].Coordinates2)));

            }

        }
    }
}

```

```

for (int i = 0; i < ListWithPoins9.Count; i++)
{
    GMarkerGoogle marker = new GMarkerGoogle(new PointLatLng
        (ListWithPoins9[i].x, ListWithPoins9[i].y), new Bitmap(@"Icon/metka.png"));
    marker.ToolTip = new GMapRoundedToolTip(marker);
    ListOf9.Markers.Add(marker);
}
}
else
{
    gmap.Overlays.Remove(ListOf9);
    ListOf9.Clear();
    ListWithPoins9.Clear();
}
}
// Загрузка формы
private void Map_Load(object sender, EventArgs e)
{
    trackBar1.Minimum = 0;
    trackBar1.Maximum = 360;
}
private void очиститьМеткиToolStripMenuItem_Click(object sender, EventArgs e) { }
// Очистить все метки с карты

// Иницирует получение данных расположения от текущего поставщика
расположения
if (watcher.TryStart(false, TimeSpan.FromMilliseconds(1000)))
{
    while (watcher.Status != GeoPositionStatus.Ready)
        watcher.TryStart(false, TimeSpan.FromMilliseconds(1000));
    coord = watcher.Position.Location;
    if (!coord.IsUnknown)
    {
        // Сохраняем в переменные наши координаты
        double Latitude = coord.Latitude;
        double Longitude = coord.Longitude;
        GMarkerGoogle marker11 = new GMarkerGoogle(new PointLatLng(Latitude,
Longitude), GMarkerGoogleType.arrow);
        marker11.ToolTip = new GMapRoundedToolTip(marker11);
        marker11.ToolTipText = "Mof";
        aaaaa.Markers.Add(marker11);
        gmap.Overlays.Add(aaaaa);
    }
    // Если координаты не получены
    else
        throw new Exception("Координаты - не числа");
}
// Если не может получить данные
else
    throw new Exception("Данные не получены");
}
public static double DegreesToRadians(double degrees)

```

```

    {
        const double degToRadFactor = Math.PI / 180;
        return degrees * degToRadFactor;
    }
    public static double RadiansToDegrees(double radians)
    {
        const double radToDegFactor = 180 / Math.PI;
        return radians * radToDegFactor;
    }
    public static double DistanceTwoPoint(double startLat, double startLong, double endLat,
double endLong)
    {
        var startPoint = new GeoCoordinate(startLat, startLong);
        var endPoint = new GeoCoordinate(endLat, endLong);
        return startPoint.GetDistanceTo(endPoint);
    }
    // -----
    // Нажатие на красный крестик формы
    private void Map_FormClosing(object sender, FormClosingEventArgs e)
    {
        //if (MessageBox.Show("Вы действительно хотите выйти?", "Выход",
MessageBoxButtons.YesNo) == DialogResult.Yes)
        //    e.Cancel = false;
        //else
        //    e.Cancel = true;
    }
    private void trackBar1_Scroll(object sender, EventArgs e)
    {
        gmap.Bearing = trackBar1.Value;
    }
    /* public void GetPoint(double x, double y)
    {
        cPoint = new CPoint(x, y);
    }
    */
}
}

```

Листинг А.6 – Класс WebIncidentListViewModel

```

using _112.Model;
using _112.Sup;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace _112.ViewModel
{

```

```

class WebIncidentListViewModel : INotifyPropertyChanged
{
    private readonly IncidentFactory incidentFactory = IncidentFactory.Instance;
    public WebIncidentListViewModel()
    {
    }
    public ObservableCollection<IncidentModel> WebIncidents
    {
        get { return incidentFactory.WebIncidents; }
        set
        {
            if (value != incidentFactory.WebIncidents)
            {
                incidentFactory.WebIncidents = value;
                OnPropertyChanged("WebIncidents");
            }
        }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    protected virtual void OnPropertyChanged(string propertyName = "")
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}

```

Листинг А.7 – Код IncidentView.xaml

```

<Window x:Class="_112.View.IncidentView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:_112.View"
    mc:Ignorable="d"
    Title="Прием происшествия" Height="765.282" Width="742.878">
    <Grid RenderTransformOrigin="0.46,0.485">
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Label Content="Дата и время происшествия" HorizontalAlignment="Left" Height="27"
            Margin="13,12,0,0" VerticalAlignment="Top" Width="167"/>
        <Label Content="Входящий номер" HorizontalAlignment="Left" Height="27"
            Margin="225,13,0,0" VerticalAlignment="Top" Width="167"/>
        <TextBox x:Name="InputPhone" HorizontalAlignment="Left" Height="21"
            Margin="228,38,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="151"
            RenderTransformOrigin="0.462,0.452"/>
        <Label Content="ФИО заявителя" HorizontalAlignment="Left" Height="27"
            Margin="405,14,0,0" VerticalAlignment="Top" Width="167"/>
        <TextBox x:Name="InputName" HorizontalAlignment="Left" Height="21"
            Margin="410,39,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="151"
            RenderTransformOrigin="0.462,0.452"/>
        <Label Content="Вид происшествия" HorizontalAlignment="Left" Height="27"

```

```

Margin="30,72,0,0" VerticalAlignment="Top" Width="154"/>
    <ComboBox Name="Vtype" HorizontalAlignment="Left" Height="26"
Margin="17,99,0,0" VerticalAlignment="Top" Width="184" ItemsSource="{Binding
Path=TypesIncidents}" SelectedIndex="0">
    <ComboBox.ItemTemplate>
        <DataTemplate x:Name="Vt">
            <TextBlock x:Name="Vtn" Text="{Binding VtypeName}" />
        </DataTemplate>
    </ComboBox.ItemTemplate>
</ComboBox>
<Label Content="Место происхождения" HorizontalAlignment="Left" Height="28"
Margin="122,294,0,0" VerticalAlignment="Top" Width="154"/>
<Label Content="Улица" HorizontalAlignment="Left" Height="26" Margin="40,378,0,0"
VerticalAlignment="Top" Width="68"/>
<Label Content="Дом" HorizontalAlignment="Left" Height="26" Margin="40,418,0,0"
VerticalAlignment="Top" Width="64"/>
<Label Content="Квартира" HorizontalAlignment="Left" Height="26"
Margin="40,458,0,0" VerticalAlignment="Top" Width="62"/>
<Label Content="Этаж" HorizontalAlignment="Left" Height="28" Margin="40,498,0,0"
VerticalAlignment="Top" Width="62"/>
<TextBox x:Name="Street" HorizontalAlignment="Left" Height="18"
Margin="130,378,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="150"
RenderTransformOrigin="0.5,0.5"/>
<TextBox x:Name="House" HorizontalAlignment="Left" Height="20"
Margin="130,418,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="152"
RenderTransformOrigin="0.462,0.452"/>
<TextBox x:Name="Flat" HorizontalAlignment="Left" Height="18" Margin="130,458,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="152"
RenderTransformOrigin="0.462,0.452"/>
<TextBox x:Name="Floor" HorizontalAlignment="Left" Height="20"
Margin="130,498,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="152"
RenderTransformOrigin="0.501,0.55"/>
<Label Content="Количество пострадавших" HorizontalAlignment="Left" Height="27"
Margin="226,75,0,0" VerticalAlignment="Top" Width="161"/>
<TextBox x:Name="NumberOfVictims" HorizontalAlignment="Left" Height="21"
Margin="229,104,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="151"
RenderTransformOrigin="0.462,0.452"/>
<Button Content="Записать" HorizontalAlignment="Left" Margin="319,648,0,0"
VerticalAlignment="Top" Width="106" Click="Button_Click" Height="32"/>
<Button Content="Показать на карте" HorizontalAlignment="Left" Height="40"
Margin="481,344,0,0" VerticalAlignment="Top" Width="142" Click="Button_Click_1"/>
<Label Content="Приоритет" HorizontalAlignment="Left" Height="27"
Margin="406,78,0,0" VerticalAlignment="Top" Width="161"/>
<ComboBox Name="Priority" HorizontalAlignment="Left" Height="26"
Margin="410,101,0,0" VerticalAlignment="Top" Width="184">
    <TextBlock>Низкий</TextBlock>
    <TextBlock>Средний</TextBlock>
    <TextBlock>Высокий</TextBlock>
</ComboBox>
<Label Content="Город" HorizontalAlignment="Left" Height="26" Margin="40,338,0,0"
VerticalAlignment="Top" Width="68"/>
<TextBox x:Name="City" HorizontalAlignment="Left" Height="20"

```

```

Margin="130,338,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="152"
RenderTransformOrigin="0.5,0.5"/>
    <Label Content="Координаты" HorizontalAlignment="Left" Height="26"
Margin="364,292,0,0" VerticalAlignment="Top" Width="84"/>
    <TextBox x:Name="Coordinates1" HorizontalAlignment="Left" Height="20"
Margin="454,296,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="88"
RenderTransformOrigin="0.5,0.5"/>
    <TextBox x:Name="Coordinates2" HorizontalAlignment="Left" Height="20"
Margin="560,296,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="86"
RenderTransformOrigin="0.5,0.5"/>
    <Label Content="Тип происшествия" HorizontalAlignment="Left" Height="27"
Margin="32,153,0,0" VerticalAlignment="Top" Width="154"/>
    <ComboBox Name="Type" HorizontalAlignment="Left" Height="26"
Margin="17,182,0,0" VerticalAlignment="Top" Width="184" ItemsSource="{Binding
ElementName=Vtype, Path=SelectedItem.Childrens}" SelectedIndex="0"/>
    <DatePicker Name="Picker" HorizontalAlignment="Left" Margin="19,41,0,0"
VerticalAlignment="Top" Width="172">
        <DatePicker.Resources>
            <Style TargetType="DatePickerTextBox">
                <Setter Property="Template">
                    <Setter.Value>
                        <ControlTemplate>
                            <TextBox Text="{Binding Path=SelectedDate,
RelativeSource={RelativeSource AncestorType=DatePicker}, StringFormat={}{0:dd/MM/yyyy
hh:mm:ss}}"/>
                        </ControlTemplate>
                    </Setter.Value>
                </Setter>
            </Style>
        </DatePicker.Resources>
    </DatePicker>
    <Label Content="Доп. Инф" HorizontalAlignment="Left" Height="28"
Margin="40,538,0,0" VerticalAlignment="Top" Width="104"/>
    <TextBox x:Name="Info" HorizontalAlignment="Left" Height="60"
Margin="130,538,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="152"
RenderTransformOrigin="0.501,0.55"/>
    <Label Content="Службы" HorizontalAlignment="Left" Height="27"
Margin="244,153,0,0" VerticalAlignment="Top" Width="154"/>
    <CheckBox x:Name="c03" Content="Скорая" HorizontalAlignment="Left" Height="23"
Margin="250,186,0,0" VerticalAlignment="Top" Width="110"/>
    <CheckBox x:Name="c02" Content="Полиция" HorizontalAlignment="Left" Height="24"
Margin="250,214,0,0" VerticalAlignment="Top" Width="110"/>
    <CheckBox x:Name="c01" Content="Пожарная" HorizontalAlignment="Left"
Height="24" Margin="250,240,0,0" VerticalAlignment="Top" Width="110"/>
    <CheckBox x:Name="c05" Content="Антитеппор" HorizontalAlignment="Left"
Height="24" Margin="370,186,0,0" VerticalAlignment="Top" Width="110"/>
    <CheckBox x:Name="c04" Content="Газовая" HorizontalAlignment="Left" Height="24"
Margin="370,214,0,0" VerticalAlignment="Top" Width="110"/>
    <CheckBox x:Name="c06" Content="Реагирования в час" HorizontalAlignment="Left"
Height="24" Margin="370,240,0,0" VerticalAlignment="Top" Width="126"/>
</Grid>
</Window>

```