**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский политехнический техникум»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода

информационных систем

Разработка информационной системы «Регистрация ДТП в ГИБДД»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Сухих М.А./

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc133366633)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133366634)

[1.1 Платформа .NET 4](#_Toc133366635)

[1.2 Язык программирования C# 7](#_Toc133366636)

[1.3 Windows Presentation Foundation (WPF) 9](#_Toc133366637)

[1.4 СУБД SQL server 11](#_Toc133366638)

[1.5 Microsoft SQL Server Management Studio 13](#_Toc133366639)

[1.6 Entity Framework 14](#_Toc133366640)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ. 16](#_Toc133366641)

[2.1. Анализ диаграммы ER (Entity-Relationship Diagram). 16](#_Toc133366642)

[2.2 Разработка базы данных 17](#_Toc133366643)

[ГЛАВА 3. РАЗРАБОТКА 20](#_Toc133366644)

[ИНФОРМАЦИОННОЙ СИСТЕМЫ 20](#_Toc133366645)

[3.1 Разработка интерфейса информационной системы 20](#_Toc133366646)

[3.2 Программирование информационной системы. 42](#_Toc133366647)

[ЗАКЛЮЧЕНИЕ 53](#_Toc133366648)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 54](#_Toc133366649)

# ВВЕДЕНИЕ

Актуальность темы курсового проекта обусловлена тем, что «Регистрация ДТП в ГИБДД» в настоящее время является обязательным процессом при ДТП. Поэтому появилось много баз данных, в которые заносится вся информация о водителях, ДТП. Одним из самых популярных видов ДТП является авто-ДТП. По статистике в год происходит около 150 тысяч ДТП.

Все ГИБДД регистрируют ДТП по определенному протоколу, в нем составляется вся информация от двух или нескольких лиц, участвовавших в ДТП, в произошедшем ДТП.

Современный мир информационных технологий трудно представить себе без возможности обработки больших объёмов информации. Такие объёмы информации удобно обрабатывать с помощью баз данных. Практически все системы в той или иной степени связаны с долговременным хранением и обработкой информации. Фактически, информация становится фактором, определяющим эффективность любой сферы деятельности. Увеличились информационные потоки и повысились требования к скорости обработки данных. Большинство операций не может быть выполнено вручную. Любые административные решения требуют более чёткой и точной оценки текущей ситуации и возможных перспектив её изменения.

Хранение в бумажной форме колоссальных объемов информации ведет к различным ошибкам. Применение новейших технологий позволяет свести к минимуму риск их совершения, полностью систематизировать и автоматизировать работу, сделать ее максимально удобной и эффективной.

**Объект:** информационная система " Регистрация ДТП в ГИБДД ".

**Предмет:** автоматизация бизнес-процесса регистрации ДТП.

**Цель**: разработать информационную систему " Регистрация ДТП в ГИБДД ".

**Задачи**:

* выбрать инструментарий;
* спроектировать базу данных;
* разработать информационную систему.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

## 1.1 Платформа .NET

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows. При разработке платформы .NET Framework учитывались следующие цели:

* Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
* Предоставление среды выполнения кода, в которой:
  + сведена к минимуму вероятность конфликтов в процессе развертывания программного обеспечения и управления его версиями;
  + гарантируется безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем;
  + исключаются проблемы с производительностью сред выполнения скриптов или интерпретируемого кода;
* обеспечиваются единые принципы разработки для разных типов приложений, таких как приложения Windows и веб-приложения;
* обеспечивается взаимодействие на основе промышленных стандартов, которое гарантирует интеграцию кода платформы .NET Framework с любым другим кодом.

Платформа .NET Framework состоит из общеязыковой среды выполнения (среды CLR) и библиотеки классов .NET Framework. Основой платформы .NET Framework является среда CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками и удаленное взаимодействие. При этом средой накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность.

Фактически основной задачей среды выполнения является управление кодом. Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом. Библиотека классов является комплексной объектно-ориентированной коллекцией повторно используемых типов, которые применяются для разработки приложений — начиная с обычных приложений, запускаемых из командной строки, и приложений с графическим интерфейсом (GUI) и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как веб-формы и веб-службы XML.

Платформа .NET Framework может размещаться неуправляемыми компонентами, которые загружают среду CLR в собственные процессы и запускают выполнение управляемого кода, создавая таким образом программную среду, позволяющую использовать средства как управляемого, так и неуправляемого выполнения. Платформа .NET Framework не только предоставляет несколько базовых сред выполнения, но также поддерживает разработку базовых сред выполнения независимыми производителями.

CLR – это некая «виртуальная машина», которая собственно и управляет нашими приложениями, написанными для .net.  
В ней есть такая занятная штука, как сборщик мусора (Garbage collector). Он подчищает всё ненужное, оставленное программой в оперативной памяти во время выполнения самой программы. То есть, если мы использовали, например, переменную всего один раз в программе, то после обращения к этой переменной, если она больше нигде не задействована – автоматический сборщик мусора её удаляет из оперативной памяти. Это абсолютно безопасно, а главное – это даёт огромный прирост в производительности масштабных и ресурсоёмких приложений. Это очень удобно, ведь в других языках, например в С++ чтобы достичь максимальной скорости работы приложения необходимо вручную удалять обьекты, а в этом случае нужно просчитать когда они не будут востребованы, чтобы их можно было безопасно удалить, чтобы не вызвать ошибку или крах программы.

Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчика за счёт предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах.

Вторым принципом стала ориентация на системы, работающие под управлением семейства операционных систем Microsoft Windows.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL) (ранее назывался Microsoft Intermediate Language, MSIL). В терминах .NET получается *сборка*, англ. *assembly*. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части.

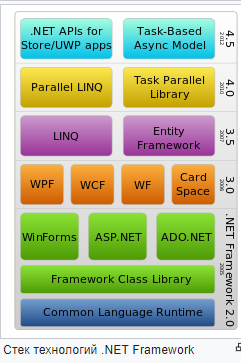


Рисунок 1.1 Стек технологий .NET Framework.

Архитектура .NET Framework описана и опубликована в спецификации Common Language Infrastructure (CLI), разработанной Microsoft и утверждённой ISO и ECMA. В CLI описаны типы данных .NET, формат метаданных о структуре программы, система исполнения байт-кода и многое другое.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке [Framework Class Library (FCL)](https://ru.wikipedia.org/wiki/Framework_Class_Library). В FCL входят классы [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), [ADO.NET](https://ru.wikipedia.org/wiki/ADO.NET), [ASP.NET](https://ru.wikipedia.org/wiki/ASP.NET), [Language Integrated Query](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [Windows Presentation Foundation](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Communication Foundation](https://ru.wikipedia.org/wiki/Windows_Communication_Foundation) и другие. Ядро FCL называется [Base Class Library (BCL)](https://ru.wikipedia.org/wiki/Base_Class_Library).

Среды разработки, поддерживающие .NET:

* Microsoft Visual Studio (C#, Visual Basic .NET, Managed C++, F#)
* SharpDevelop
* MonoDevelop
* Embarcadero RAD Studio (Delphi for .NET); ранее Borland Developer Studio (Delphi for .NET, C#)
* A#
* Zonnon
* PascalABC.NET
* JetBrains Rider

Приложения .NET также можно разрабатывать в текстовом редакторе, просто вызывая компилятор из командной строки.

## 1.2 **Язык программирования C#**

C# — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java.

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов.

С#‎ разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов С#‎, которая отражает BCL.

Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR.

Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET).

CLR предоставляет С#‎, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#‎, а производится CLR для программ, написанных на C#, точно так же, как это делается для программ на VB.NET, J# и др.

C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# — объектно-ориентированный язык. Вы определяете типы и их поведение. Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения. Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда-выражения поддерживают приемы функционального программирования. Синтаксис LINQ создает общий шаблон для работы с данными из любого источника. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем. В C# имеется Единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

Система типов в C# унифицирована таким образом, что значение любого типа можно рассматривать как object (объект). Каждый тип в C# является прямо или косвенно производным от типа класса object, и этот тип object является исходным базовым классом для всех типов. Чтобы значения ссылочного типа обрабатывались как объекты, им просто присваивается тип object. Чтобы значения типов значений обрабатывались как объекты, выполняются операции упаковки-преобразования и распаковки-преобразования. В следующем примере значение int преобразуется в object, а затем обратно в int.

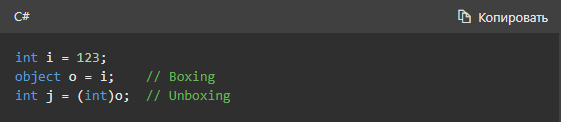
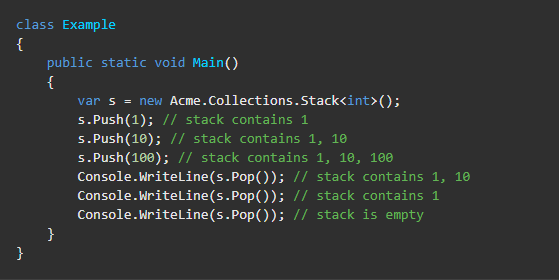


Рисунок 1.2 Пример кода.

Основными понятиями Организации в C# являются программы, пространства имен, типы, члены и сборки. В программе объявляются типы, которые содержат члены. Эти типы можно организовать в пространства имен. Примерами типов являются классы, структуры и интерфейсы. К членам относятся поля, методы, свойства и события. При компиляции программы на C# упаковываются в сборки. Сборки обычно имеют расширение .exe файла или .dll, в зависимости от того, реализуют ли они .exe или библиотек и соответственно.

Сборки содержат исполняемый код в виде инструкций промежуточного языка (IL) и символьную информацию в виде метаданных. Перед выполнением JIT-компилятор среды CLR .NET преобразует код IL в сборке в код, зависящий от процессора. Сборка полностью описывает сама себя и содержит весь код и метаданные, поэтому в C# не используются директивы #include и файлы заголовков. Чтобы использовать в программе C# открытые типы и члены, содержащиеся в определенной сборке, вам достаточно указать ссылку на эту сборку при компиляции программы. Например, эта программа использует класс Acme.Collections.Stack из сборки acme.dll:



1.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) представляет собой обширный API-интерфейс для создания настольных графических программ имеющих насыщенный дизайн и интерактивность. В отличие от устаревшей технологии Windows Forms, WPF включает новую модель построения пользовательских приложений (в основе WPF лежит мощная инфраструктура, основанная на DirectX).

Это означает возможность применения развитых графических эффектов, не платя за это производительностью, как это было в Windows Forms. Фактически даже становятся доступными такие расширенные средства, как поддержка видеофайлов и трехмерное содержимое. Используя эти средства (при наличии хорошего инструмента графического дизайна), можно создавать бросающиеся в глаза пользовательские интерфейсы и визуальные эффекты, которые были просто невозможны в Windows Forms.

**Графика и анимация WPF**

Описание построения базовых линий и фигур, использование трансформаций элементов, создание нестандартных текстур в виде пользовательских кистей, члены классов Geometry и Drawing. Так же приводятся основы эффектов и анимации.

**Шаблоны и пользовательские элементы управления WPF**

WPF существует в виде подмножества типов .NET, которые по большей части находятся в пространстве имен System.Windows. Если ранее вы создавали приложения в .NET с помощью таких платформ, как ASP.NET и Windows.Forms, то должны быть знакомы с фундаментальными принципами программирования WPF. Вы выполняете следующие действия. Создание экземпляров классов, настройка свойств, вызов методов, обработка событий.

WPF включает в себя дополнительные конструкции программирования, которые расширяют возможности свойств и событий.

WPF позволяет разрабатывать приложения, используя как разметку, так и код программной части, что привычно для разработчиков на ASP.NET. Разметка XAML обычно используется для определения внешнего вида приложения, а управляемые языки программирования (код программной части) — для реализации его поведения. Такое разделение внешнего вида и поведения имеет ряд преимуществ.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

XAML представляет собой язык декларативного описания интерфейса, основанный на XML. Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на элементы управления второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента, например, Button (кнопка).

Для работы с WPF требуется любой .NET-совместимый язык. В этот список входит множество языков: C#, F#, VB.NET, C++, Ruby, Python, Delphi (Prism), Lua и многие другие. Для полноценной работы может быть использована как Visual Studio, так и Expression Blend.

**Графика**

WPF представляет обширный, масштабируемый и гибкий набор графических возможностей:

* Графика, не зависящая от разрешения и устройства. Основной единицей измерения в графической системе WPF является аппаратно-независимый пиксель, который составляет 1/96 часть дюйма независимо от фактического разрешения экрана.
* Дополнительная поддержка графики и анимации. WPF упрощает программирование графики за счет автоматического управления анимацией. Разработчик не должен заниматься обработкой сцен анимации, циклами отрисовки и билинейной интерполяцией
* Аппаратное ускорение. Графическая система WPF использует преимущества графического оборудования, чтобы уменьшить использование ЦП.

**Двухмерная графика**

WPF предоставляет библиотеку общих двухмерных фигур, нарисованных с помощью векторов, таких, как прямоугольники и эллипсы, а также графические пути. И в своей функциональности фигуры реализуют многие возможности, которые доступны обычным элементам управления.

Двухмерная графика в WPF включает визуальные эффекты, такие как градиенты, точечные рисунки, чертежи, рисунки с видео, поворот, масштабирование и наклон.

**Трехмерная графика**

WPF также включает возможности трехмерной отрисовки, интегрированные с двухмерной графикой, что позволяет создавать более яркий и интересный пользовательский интерфейс.

## 1.4 СУБД SQL server

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

SQL является общепринятым интерфейсом к базам данных. «Все промышленные базы — Oracle, Microsoft SQL Server, PostgreSQL, MySQL — работают на SQL.SQL используют в Facebook, Google, Amazon, Uber, Netflix, Airbnb.

Вот основные типы операторов SQL:

**● DDL (Data Definition Language)** — операторы определения данных, которые работают с целыми таблицами. Например: CREATE — чтобы создать таблицу (TABLE) или базу данных (DATABASE), DROP (TABLE/DATABASE) — чтобы удалить всю таблицу или базу данных, USE — чтобы выбрать нужную базу данных.

**● DML (Data Manipulation Language)** — операторы манипуляции данными, которые работают с содержимым таблиц. Например, UPDATE — чтобы обновить данные, DELETE — чтобы удалить, INSERT — чтобы добавить новые, SELECT (FROM/WHERE) — чтобы выбрать нужные данные по заданному параметру.

**● DCL (Data Control Language)** — оператор определения доступа к данным для разных пользователей. Например, GRANT — чтобы открыть доступ или DENY — чтобы запретить.

**● SQL клиент-сервер** — для управления доступом к данных для разных пользователей.

**● SQL трёхуровневой архитектуры** — для защиты данных от несанкционированного копирования и распространения.

Язык SQL нужен разработчикам, тестировщикам, аналитикам данных, администраторам, маркетологам — всем тем, кому по работе нужно выгружать и обрабатывать большие объёмы данных. Правильно организованные запросы помогают извлекать полезную информацию о клиентах и пользователях, сортируют её по определённым категориям, анализируют работу сайта или бизнеса.

**Виды SQL-server:**

Для реализации функций СУБД на сегодняшний день чаще всего используются следующие SQL-серверы:

**MS SQL server** - многопользовательский программный продукт, разработанный компанией Microsoft, обладающий высокой производительностью и отказоустойчивостью, тесно интегрированный с ОС Windows. Этот сервер поддерживает удаленные подключения, работает с многими популярными типами данных, дает возможность создавать триггеры и хранимые данные, имеет практичные и удобные утилиты для настройки;

**Oracle Database server** - СУБД, предназначенная для создания, консолидации и управления базами данных в облачной среде. Используя этот сервер, можно как автоматизировать обычные бизнес-операции, так и выполнять динамический многомерный анализ данных (OLAP), проводить операции с документами xml-формата и управлять разделенной и локальной информацией.

## 1.5 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) — утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера[1].

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

Также есть SQL Server Management Studio Express для Express версии сервера, которая является бесплатной. Однако в ней нет поддержки ряда компонентов (Analysis Services, Integration Services, Notification Services, Reporting Services) и SQL Server 2005 Mobile Edition.

Среда SQL Server Management Studio предназначена для доступа к службам Службы Analysis Services, Integration Services и Reporting Services, а также для их настройки, администрирования и управления ими. Хотя все три технологии бизнес-аналитики полагаются на среду SQL Server Management Studio, административные задачи, связанные с каждой из этих технологий, несколько отличаются.

SSMS работает только в Windows (AMD или Intel). Если вам требуется средство, которое работает на платформах, отличных от Windows, рассмотрите Azure Data Studio. Azure Data Studio — это кроссплатформенное средство для macOS, Linux и Windows. Дополнительные сведения см. в разделе Azure Data Studio.

SSMS может устанавливать общие компоненты, если установлено, что они отсутствуют во время установки SSMS. SSMS не будет автоматически удалять эти компоненты при удалении SSMS.Общие компоненты:Azure Data StudioДрайвер Microsoft OLE DB для SQL ServerMicrosoft ODBC Driver for SQL Server версии 17Распространяемый компонент Microsoft Visual C++ 2013 (x86)Распространяемый компонент Microsoft Visual C++ 2017 (x86)Распространяемый компонент Microsoft Visual C++ 2017 (x64)набор средств Microsoft Visual Studio Tools для работы с приложениями 2019 г.Эти компоненты не удаляются, так как могут являться общими с другими продуктами. Их удаление может нарушить работу этих продуктов.

Среда SQL Server Management Studio позволяет включать компоненты служб Reporting Services, администрировать серверы и базы данных, управлять ролями и заданиями.

Она реализует функции управления общими расписаниями (в папке «Общие расписания») и базами данных сервера отчетов (ReportServer, ReportServerTempdb). Можно также создать роль RSExecRole в системной базе данных Master, когда база данных сервера отчетов перемещается в новое или другое ядро СУБД SQL Server.

## 1.6 Entity Framework

Entity Framework — это решение для работы с базами данных, которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Также код с использованием EF пишется гораздо быстрее.

Например, работая с базами данных напрямую, разработчик должен беспокоиться о подключении, подготовке SQL и параметров, отправке запросов и транзакций. На Entity Framework всё это делается автоматически — программист же работает непосредственно с сущностями и только говорит EF, что нужно сохранить изменения.

Entity Framework — это набор технологий в ADO.NET, которые поддерживают разработку программных приложений, ориентированных на данные. Архитекторам и разработчикам приложений, ориентированных на обработку данных, приходится учитывать необходимость достижения двух совершенно различных целей. Они должны моделировать сущности, связи и логику решаемых бизнес-задач, а также работать с ядрами СУБД, используемыми для сохранения и получения данных. Данные могут распределяться по нескольким системам хранения данных, в каждой из которых применяются свои протоколы, но даже в приложениях, работающих с одной системой хранения данных, необходимо поддерживать баланс между требованиями системы хранения данных и требованиями написания эффективного и удобного для обслуживания кода приложения.

Платформа Entity Framework позволяет работать с данными в форме специфических для домена объектов и свойств (например, с клиентами и их адресами) без необходимости учитывать формат базовых таблиц и столбцов базы данных, где хранятся эти данные. Entity Framework дает разработчикам возможность работать с данными на более высоком уровне абстракции, создавать и сопровождать приложения, ориентированные на работу с данными, одновременно с этим сокращая объем кода по сравнению с традиционными приложениями. поскольку Entity Framework является компонентом платформа .NET Framework, Entity Framework приложения могут работать на любом компьютере, на котором установлена платформа .NET Framework с пакетом обновления 1 (SP1) версии 3,5.

Entity Framework дает жизнь модели, позволяя разработчикам запрашивать сущности и связи в модели предметной области (называемой концептуальной моделью в Entity Framework), в то же время полагаться на Entity Framework преобразования этих операций в команды, относящиеся к источнику данных. Это позволяет отказаться от применения в приложениях жестко заданных зависимостей от конкретного источника данных.

Entity Framework включает поставщик данных EntityClient. Этот поставщик управляет соединениями, преобразует запросы сущностей в запросы, зависящие от источника данных, и возвращает модуль чтения данных, который Entity Framework использует для материализации данных сущности в объекты.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

## 2.1. **Анализ диаграммы ER (Entity-Relationship Diagram)**

**ER-модель** (от англ. *Entity-Relationship model*, *модель «сущность — связь»*) — модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование схемы, созданной на основе ER-модели, в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена *диаграмма «сущность-связь»* (англ. *Entity-Relationship diagram, ERD*, *ER-диаграмма*).

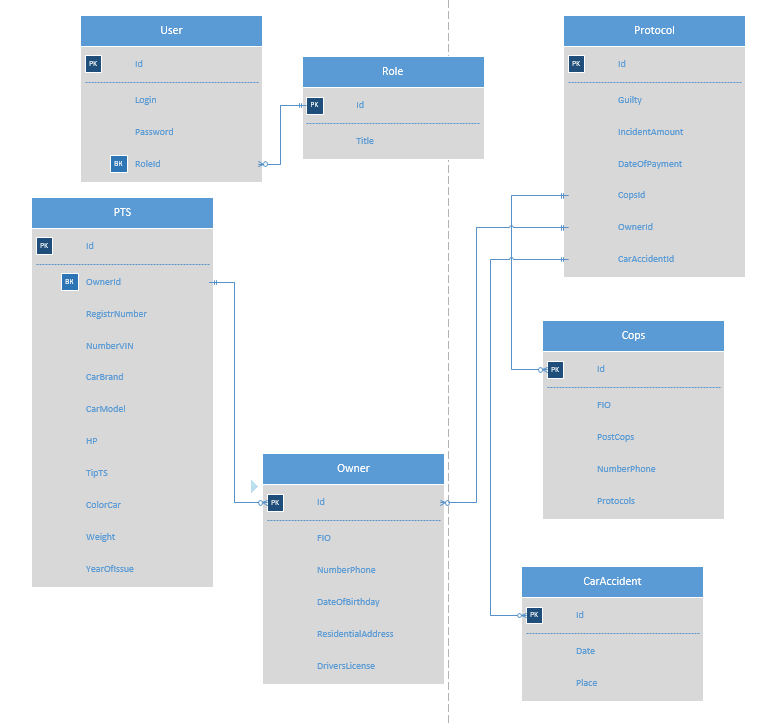


Рисунок 2.1 Диаграмма ER (Entity-Relationship Diagram)

Основными сущностями для решения указанной задачи являются: Пользователь, Роль, Водители, ПТС, Сотрудники ДПС, Протокол и Дорожное происшествие. Сразу возникает очевидная связь между сущностями – связь между клиентом и заявками, связь между работником банка и документами, связь между менеджером и заявками, связь между менеджером и документами.

## 2.2 Разработка базы данных

SQL Server Management Studio — это набор административных средств для управления компонентами, относящимися к SQL Server. Эта интегрированная среда позволяет пользователям выполнять разнообразные задачи, например резервное копирование данных, редактирование запросов и автоматизацию общих функций в одном интерфейсе.

Среда SQL Server Management Studio включает в себя следующие средства:

Редактор кода — богатый возможностями редактор скриптов для написания и редактирования скриптов. Среда SQL Server Management Studio предоставляет четыре версии редактора кода: редактор запросов ядра СУБД для скриптов Transact-SQL, редактор запросов многомерных выражений, редактор запросов расширения интеллектуального анализа данных и редактор запросов XML/A.

Обозреватель объектов для размещения, изменения, создания скрипта или выполнения объектов, принадлежащих экземплярам SQL Server.

Обозреватель шаблонов для размещения и написания сценариев шаблонов.

Обозреватель решений для организации и хранения связанных скриптов как части проекта.

Окно свойств для отображения текущих свойств выбранных объектов.

Среда SQL Server Management Studio обеспечивает эффективность рабочих процессов, предоставляя:

Отключенный доступ. Можно писать и изменять скрипты, не соединяясь с экземпляром SQL Server.

Создание сценариев из любого диалогового окна. Можно создать скрипт из любого диалогового окна, а также читать, изменять, сохранять и многократно использовать скрипты после создания.

Немодальные диалоговые окна. При обращении к диалоговому окну интерфейса можно просмотреть другие ресурсы в среде SQL Server Management Studio, не закрывая диалоговое окно.

Разработка базы данных начинается с ранее разработанной ERD диаграммой. Откроем Microsoft SQL Server Management Studio и подключаемся к нашему серверу.

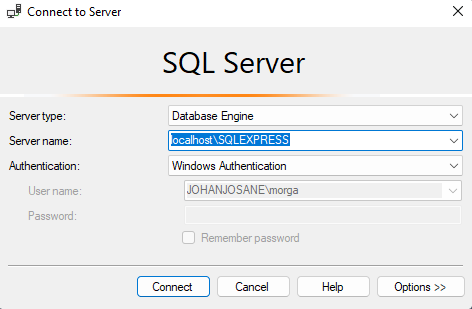


Рисунок 2.2 «Connect to Server»»

Далее нужно создать базу данных, наживаем правой кнопкой мыши на DataBase и нажимаем New DataBase.

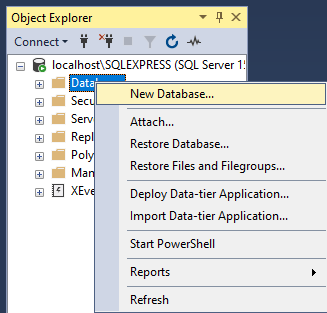


Рисунок 2.3 «New Database»

Далее в окне даём название для базы данных и нажимаем на кнопку OK. В Object Explorer появиться созданная база данных, развернём её, нажмём ПКМ по папке Database Diagrams и нажмём на New Database Diagram.

После чего нам выведется поле для создания диаграммы базы данных, на котором можно создавать таблицы с названием столбцов и их типами данных.

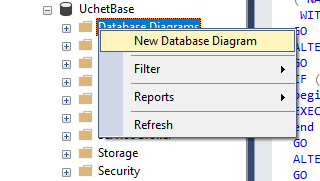


Рисунок 2.4 Создание диаграммы

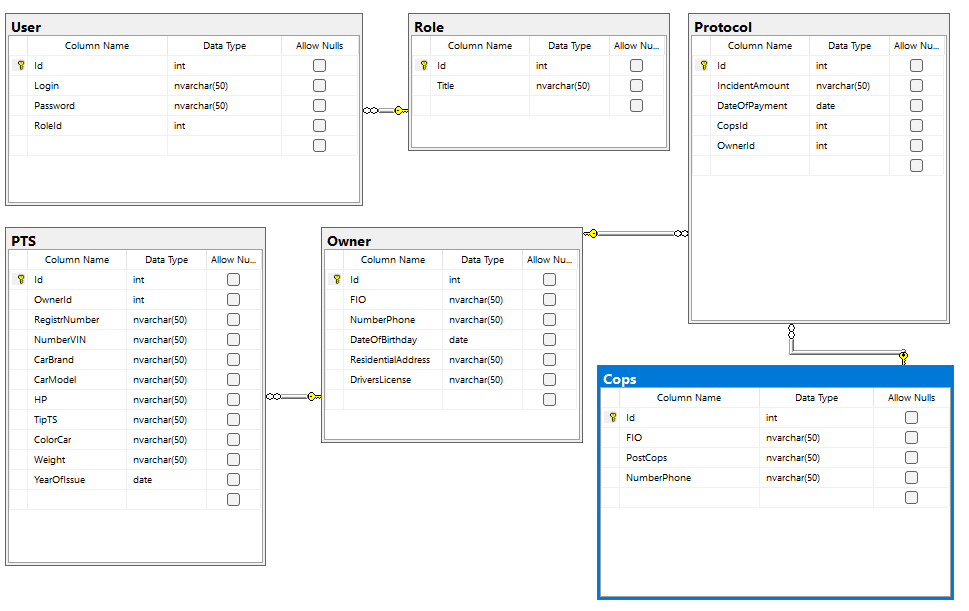


Рисунок 2.5 «База данных «Регистрация ДТП в ГИБДД»»

В данной базе данных присутствует 6 сущностей User, Role, Owner, PTS, Protocol и Cops. User и Role используется для формы авторизации.

Owner, PTS, Protocol и Cops используется в корне программы. Owner – таблица с водителями, PTS – паспорт транспортного средства водителей, Cops – сотрудники ГИБДД, Protocol – содержит данные о ДТП.

Далее создаём связи как в ERD сущностях и после сохраняем таблицы нажав сочетание клавиш «Ctrl + S» и в высветившимся окне пишем название нашей диаграммы.

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 3.1 **Разработка интерфейса информационной системы**

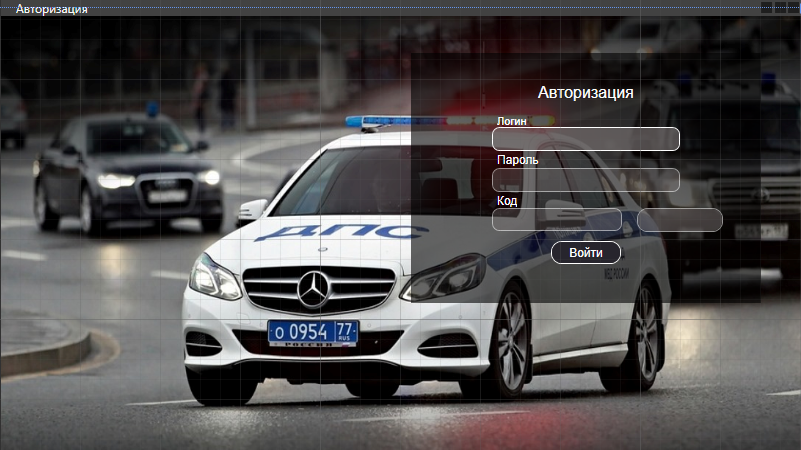


Рисунок 3.1 «Окно авторизации»

На рисунке 3.1 показана форма авторизации, в котором имеется TextBox, PasswordBox, TextBlock, кнопка(Войти). При правильном вводе логина, откроется окно ввода пароля. При правильном вводе пароля появится код подтверждения, который надо подтвердить.

<Grid>

<Grid.Background>

<ImageBrush ImageSource="/merz.jpg"/>

</Grid.Background>

<Grid Margin="410,37,40,0" Width="350" Height="250" VerticalAlignment="Top" Background="#99000000">

<Grid.RowDefinitions>

<RowDefinition Height="252\*" />

</Grid.RowDefinitions>

<TextBlock Height="11" HorizontalAlignment="Left" Margin="86,62,0,0" x:Name="textBlockLogin"

VerticalAlignment="Top" Width="31" Foreground="White" Text="Логин" FontFamily="Arial" FontSize="11"></TextBlock>

<TextBlock Height="14" HorizontalAlignment="Left" Margin="86,100,0,0" x:Name="textBlockPassword"

VerticalAlignment="Top" Width="43" Foreground="White" Text="Пароль" FontFamily="Arial" RenderTransformOrigin="0.322,0.574"/>

<TextBox Height="24" HorizontalAlignment="Center" Margin="0,74,0,0" Name="textBoxLogin"

VerticalAlignment="Top" KeyUp="LogBlock\_Up" Width="188" Background="#997D7676" BorderBrush="White" IsEnabled="True" Foreground="White">

<TextBox.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="7"></Setter>

</Style>

</TextBox.Resources>

</TextBox>

<PasswordBox Style="{DynamicResource PasswordBoxStyle1}" Height="24" HorizontalAlignment="Center" Name="textBoxPassword"

VerticalAlignment="Top" KeyUp="PassBlock\_Up" Width="188" Background="#997D7676" BorderBrush="White" IsEnabled="False" Foreground="White" Margin="0,115,0,0">

<PasswordBox.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="7"></Setter>

</Style>

</PasswordBox.Resources>

</PasswordBox>

<Button Style="{DynamicResource ButtonStyle1}" Content="Войти" Height="23" HorizontalAlignment="Center" Margin="0,188,0,0" Name="Vhod"

VerticalAlignment="Top" Width="70" Click="Vhod\_Click" Background="#FF27252D" Foreground="White" BorderBrush="White" FontFamily="Arial" />

<TextBox Style="{DynamicResource TextBoxStyle1}" Height="23" HorizontalAlignment="Left" Margin="81,155,0,0" x:Name="textBoxCod"

VerticalAlignment="Top" Width="130" Background="#997D7676" BorderBrush="White" IsEnabled="False" Foreground="White">

<TextBox.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="7"></Setter>

</Style>

</TextBox.Resources>

</TextBox>

<TextBlock Height="13" HorizontalAlignment="Left" Margin="86,141,0,0" x:Name="textBlockCod"

VerticalAlignment="Top" Width="20" Foreground="White" Text="Код" FontFamily="Arial"/>

<Label Content="Авторизация" HorizontalAlignment="Center" Margin="0,25,0,0" VerticalAlignment="Top" FontSize="16" FontFamily="Arial" Foreground="White"/>

<TextBox Style="{DynamicResource TextBoxStyle3}" Height="24" HorizontalAlignment="Left" Margin="226,155,0,0" x:Name="textBoxCodSpawn" FontSize="14" FontFamily="Arial"

VerticalAlignment="Top" Width="86" Background="#DC7D7676" BorderBrush="White" IsEnabled="False" Foreground="White"/>

<Button Style="{DynamicResource ButtonStyle2}" Content="Обновить" Height="23" HorizontalAlignment="Left" Margin="226,188,0,0" x:Name="Cod"

VerticalAlignment="Top" Width="86" Click="Cod\_Click" Background="#FF8976E6" Foreground="Black" BorderBrush="White" IsEnabled="False" Visibility="Hidden"/>

</Grid>

</Grid>

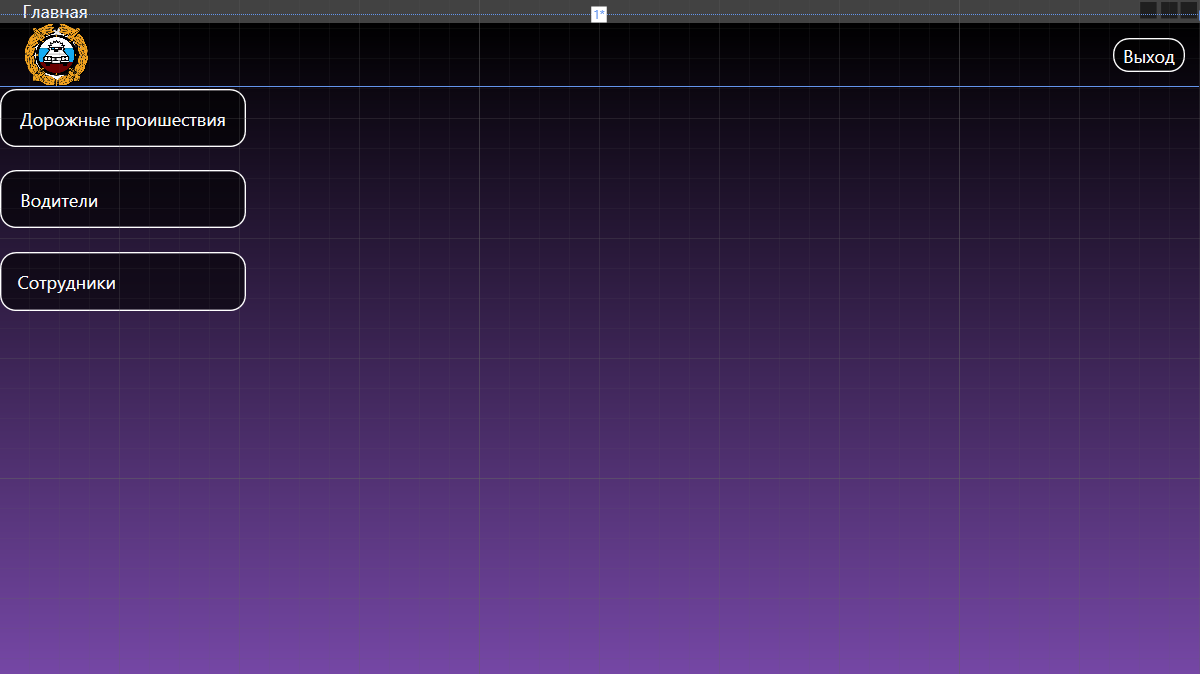


Рисунок 3.2 «Главное окно отчетов»

На рисунке 3.2 показано главное окно, в котором есть четыре кнопки: Дорожные происшествия, Водители, Сотрудники, Выход. Так же имеется иконка приложения.

//Основной грид с в котором отображаются все страницы

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Black"/>

<GradientStop Color="#FF7547A5" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.RowDefinitions>

<RowDefinition Height="43\*"/>

<RowDefinition Height="391\*"/>

</Grid.RowDefinitions>

<Button Style="{DynamicResource ButtonStyle3}" Click="Owner\_Click" Padding="0,0,85,0" Background="#99000000" Content="Водители" Foreground="White" BorderBrush="White"

Margin="0,55,0,0" Width="164" HorizontalAlignment="Left" Height="39" VerticalAlignment="Top" Grid.Row="1"/>

<Button Style="{DynamicResource ButtonStyle1}" Click="CarAccident\_Click" Background="#99000000" Content="Дорожные проишествия" Foreground="White" BorderBrush="White"

Margin="0,1,0,0" Height="39" VerticalAlignment="Top" HorizontalAlignment="Left" Width="164" Grid.Row="1"/>

<Button Style="{DynamicResource ButtonStyle2}" Click="Cops\_Click" Padding="0,0,75,0" Background="#99000000" Content="Сотрудники" Foreground="White" BorderBrush="White"

Margin="0,110,0,0" Height="39" VerticalAlignment="Top" HorizontalAlignment="Left" Width="164" Grid.Row="1"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="Exit\_Click" Background="#99000000" Content="Выход" Foreground="White" BorderBrush="White"

Margin="0,0,10,0" Height="23" VerticalAlignment="Center" HorizontalAlignment="Right" Width="48"/>

<Frame NavigationUIVisibility="Hidden" x:Name="Frame1" Margin="169,1,0,0" Grid.Row="1"/>

<Image HorizontalAlignment="Left" Height="43" VerticalAlignment="Center" Width="76" Source="/Views/icon.png"/>

</Grid>

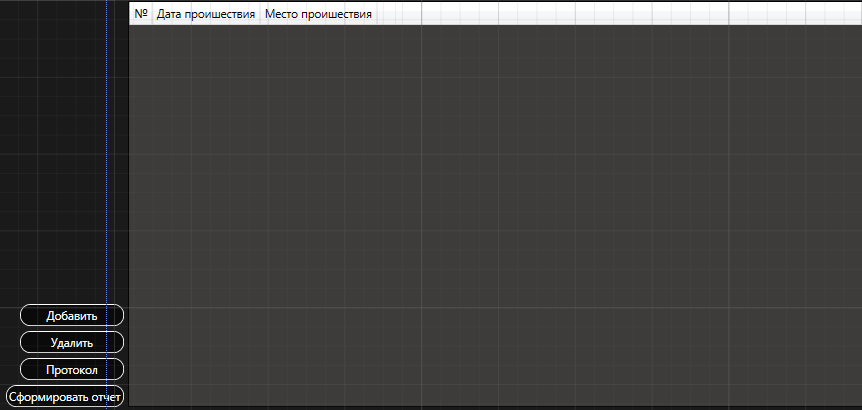


Рисунок 3.3 «Страница дорожных происшествий»

Все следующие страницы имеют идентичный код.

//Кнопка закрытия программы

<Grid>

<DataGrid Margin="14,0,21,27" x:Name="CarRP" IsReadOnly="True" AutoGenerateColumns="False" BorderBrush="Black">

<DataGrid.Background>

<SolidColorBrush Color="#997D7676" Opacity="0.6"/>

</DataGrid.Background>

<DataGrid.Columns>

//DataGrid создание таблицы с данными, которые импортируются из БД

<DataGridTextColumn Binding="{Binding Id}" Header="№" Width="auto" ></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Date, StringFormat='{}{0:dd.MM.yyyy}'}" Header="Дата проишествия" Width="auto"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Place}" Header="Место проишествия" Width="auto"></DataGridTextColumn>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Редактировать" Name="BtnEdit" HorizontalAlignment="Left" Click="BtnEdit\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Button Style="{DynamicResource ButtonStyle1}" Click="Add\_Click" Background="#99000000" Content="Добавить" Foreground="White" BorderBrush="White"

Margin="-99,0,0,111" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="Delete\_Click" Background="#99000000" Content="Удалить" Foreground="White" BorderBrush="White"

Margin="-99,0,0,83" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="Protocol\_Click" Background="#99000000" Content="Протокол" Foreground="White" BorderBrush="White"

Margin="-99,0,0,55" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="otchet\_Click" Background="#99000000" Content="Сформировать отчет" Foreground="White" BorderBrush="White"

Margin="-114,0,0,27" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="123"/>

</Grid>

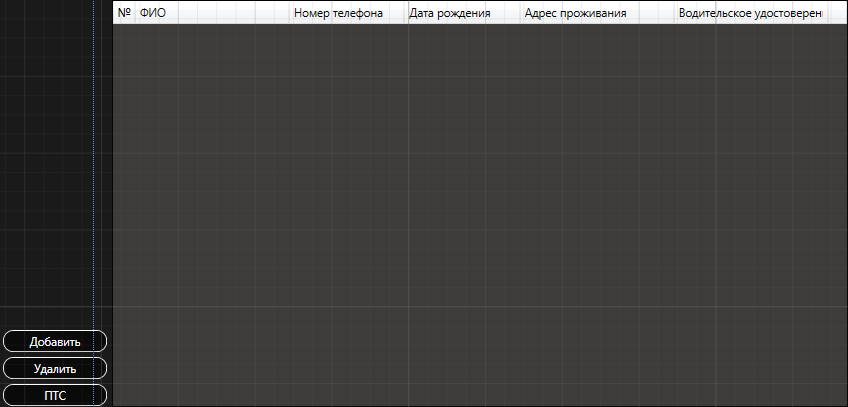


Рисунок 3.5 «Страница данных о водителях»

<Grid>

<DataGrid Margin="10,0,23,26" x:Name="OwnerGP" IsReadOnly="True" AutoGenerateColumns="False" BorderBrush="Black">

<DataGrid.Background>

<SolidColorBrush Color="#997D7676" Opacity="0.6"/>

</DataGrid.Background>

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Id}" Header="№" Width="auto" ></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding FIO}" Header="ФИО" Width="200\*"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding NumberPhone}" Header="Номер телефона" Width="150\*"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding DateOfBirthday, StringFormat='{}{0:dd.MM.yyyy}'}" Header="Дата рождения" Width="150\*"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding ResidentialAddress}" Header="Адрес проживания" Width="200\*"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding DriversLicense}" Header="Водительское удостоверение" Width="200\*"></DataGridTextColumn>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Редактировать" Name="BtnEdit" HorizontalAlignment="Left" Click="BtnEdit\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Button Style="{DynamicResource ButtonStyle1}" Click="EditDrivers\_Click" Background="#99000000" Content="Добавить" Foreground="White" BorderBrush="White"

Margin="-103,0,0,83" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="DeleteDrivers\_Click" Background="#99000000" Content="Удалить" Foreground="White" BorderBrush="White"

Margin="-103,0,0,55" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="PTS\_Click" Background="#99000000" Content="ПТС" Foreground="White" BorderBrush="White"

Margin="-103,0,0,27" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

</Grid>

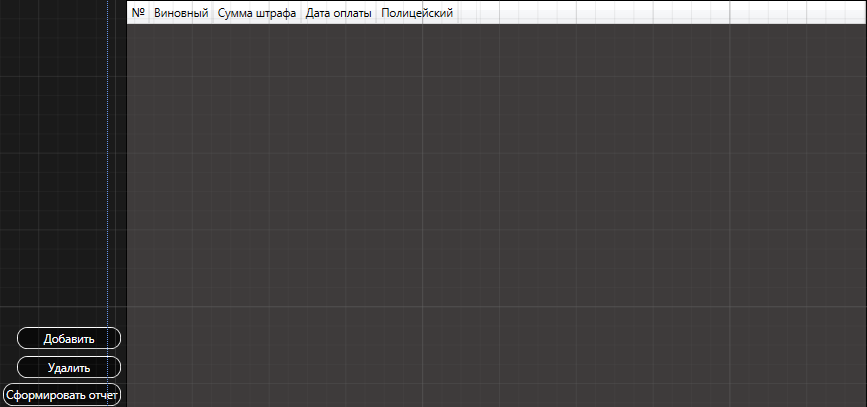


Рисунок 3.6 «Страница протоколов»

<Grid>

<DataGrid Margin="10,0,18,25" x:Name="ProtocolPP" IsReadOnly="True" AutoGenerateColumns="False" BorderBrush="Black">

<DataGrid.Background>

<SolidColorBrush Color="#997D7676" Opacity="0.6"/>

</DataGrid.Background>

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Id}" Header="№" Width="auto" ></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Owner.FIO}" Header="Виновный" Width="auto"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding IncidentAmount}" Header="Сумма штрафа" Width="auto"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding DateOfPayment, StringFormat='{}{0:dd.MM.yyyy}'}" Header="Дата оплаты" Width="auto"></DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Cops.FIO}" Header="Полицейский" Width="auto"></DataGridTextColumn>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Редактировать" Name="BtnEdit" HorizontalAlignment="Left" Click="BtnEdit\_Click">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Button Style="{DynamicResource ButtonStyle1}" Click="EditProtocol\_Click" Background="#99000000" Content="Добавить" Foreground="White" BorderBrush="White"

Margin="-103,0,0,86" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="DeleteProtocol\_Click" Background="#99000000" Content="Удалить" Foreground="White" BorderBrush="White"

Margin="-103,0,0,56" Height="23" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="108"/>

<Button Style="{DynamicResource ButtonStyle3}" Click="otchet\_Click" Background="#99000000" Content="Сформировать отчет" Foreground="White" BorderBrush="White"

Margin="-118,0,0,27" Height="24" VerticalAlignment="Bottom" HorizontalAlignment="Left" Width="123"/>

</Grid>

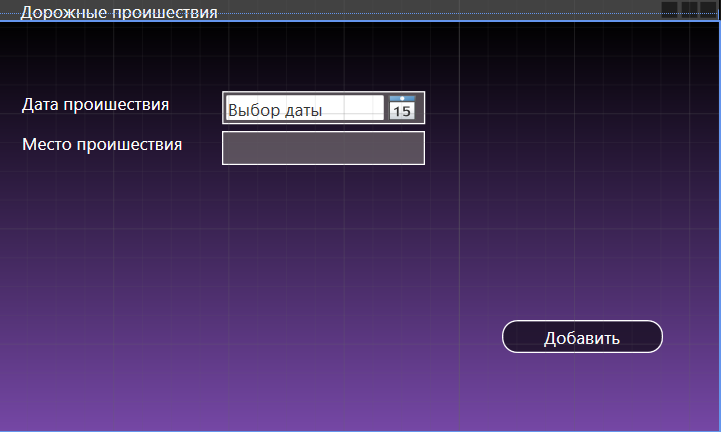


Рисунок 3.8 «Окно редактирования ДТП»

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Black"/>

<GradientStop Color="#FF7547A5" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Button Style="{DynamicResource ButtonStyle1}" Click="Edit\_Click" Background="#99000000" Content="Добавить" Foreground="White" BorderBrush="White"

Margin="349,207,0,0" Width="112" HorizontalAlignment="Left" Height="23" VerticalAlignment="Top"/>

<TextBox Text="{Binding Place}" Margin="155,76,0,0" Name="textBoxPlace" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="141" Height="23" VerticalAlignment="Top"/>

<DatePicker SelectedDate="{Binding Date}" Margin="155,48,0,0" Name="textBoxDate" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="141" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,48,0,0" x:Name="textBlockAccident" Foreground="White" Text="Дата проишествия" HorizontalAlignment="Left" Width="108" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,76,0,0" x:Name="textBlockPlace" Foreground="White" Text="Место проишествия" HorizontalAlignment="Left" Width="114" Height="23" VerticalAlignment="Top"/>

</Grid>

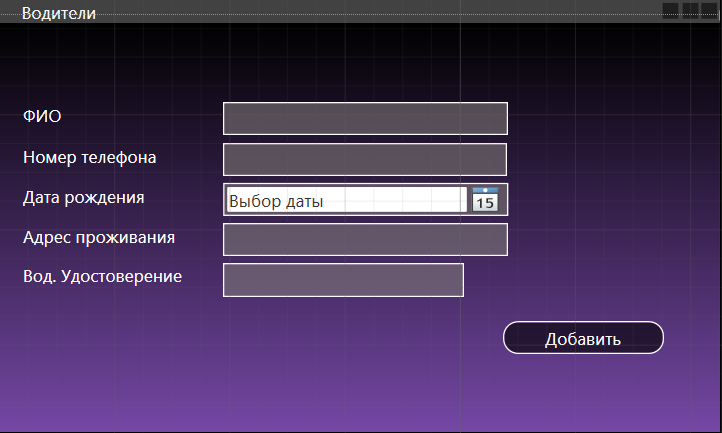


Рисунок 3.10 «Окно редактирования информации о водителях»

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Black"/>

<GradientStop Color="#FF7547A5" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Button Style="{DynamicResource ButtonStyle1}" Click="Edit\_Click" Background="#99000000" Content="Добавить" Foreground="White" BorderBrush="White"

Margin="349,207,0,0" Width="112" HorizontalAlignment="Left" Height="23" VerticalAlignment="Top"/>

<TextBox Margin="155,55,0,0" Text="{Binding FIO}" Name="TextBoxFIO" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="198" Height="23" VerticalAlignment="Top"/>

<DatePicker SelectedDate="{Binding DateOfBirthday}" Margin="155,111,0,0" Name="textBoxDateBRZD" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="198" Height="23" VerticalAlignment="Top"/>

<TextBox Margin="155,83,0,0" Text="{Binding NumberPhone}" Name="textBoxNumberPhone" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="197" Height="23" VerticalAlignment="Top" RenderTransformOrigin="0.5,0.5"/>

<TextBlock Margin="16,56,0,0" x:Name="textBlockFIO" Foreground="White" Text="ФИО" Height="23" VerticalAlignment="Top" HorizontalAlignment="Left" Width="118"/>

<TextBlock Margin="16,84,0,0" x:Name="textBlockNumberPhone" Foreground="White" Text="Номер телефона" HorizontalAlignment="Left" Width="108" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,112,0,0" x:Name="textBlockDateOfBry" Foreground="White" Text="Дата рождения" HorizontalAlignment="Left" Width="114" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,140,0,0" x:Name="textBlockAdress" Foreground="White" Text="Адрес проживания" HorizontalAlignment="Left" Width="114" Height="23" VerticalAlignment="Top"/>

<TextBox Margin="155,139,0,0" Text="{Binding ResidentialAddress}" x:Name="textBoxPlaceHome" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="198" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,167,0,0" x:Name="textBlockUdostoverenie" Foreground="White" Text="Вод. Удостоверение" HorizontalAlignment="Left" Width="118" Height="23" VerticalAlignment="Top"/>

<TextBox Margin="155,167,0,0" Text="{Binding DriversLicense}" x:Name="textBoxAccident\_Copy1" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="167" Height="23" VerticalAlignment="Top"/>

</Grid>

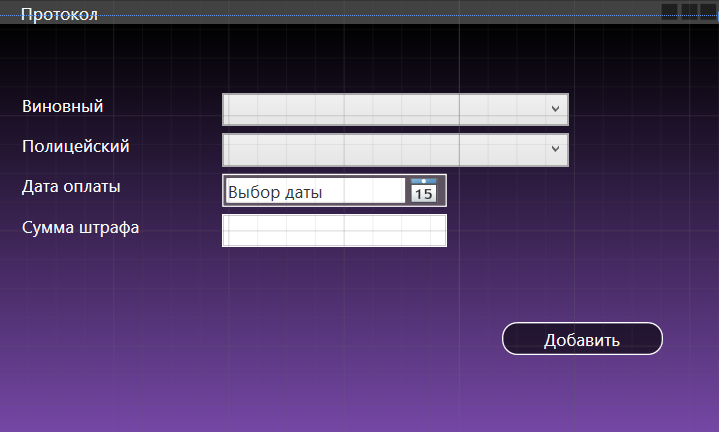


Рисунок 3.11 «Окно редактирования протокола»

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Black"/>

<GradientStop Color="#FF7547A5" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Button Style="{DynamicResource ButtonStyle1}" Click="Edit\_Click" Background="#99000000" Content="Добавить" Foreground="White" BorderBrush="White"

Margin="349,207,0,0" Width="112" HorizontalAlignment="Left" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,48,0,0" Foreground="White" Text="Виновный" HorizontalAlignment="Left" Width="114" Height="23" VerticalAlignment="Top"/>

<ComboBox Margin="155,48,0,0" Text="{Binding OwnerID}" x:Name="OwnerId" DisplayMemberPath="FIO" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="241" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,76,0,0" Foreground="White" HorizontalAlignment="Left" Width="114" Height="23" VerticalAlignment="Top"><Run Language="ru-ru" Text="Полицейский"/><LineBreak/><Run Language="ru-ru"/></TextBlock>

<TextBlock Margin="16,104,0,0" Foreground="White" Text="Дата оплаты" HorizontalAlignment="Left" Width="114" Height="23" VerticalAlignment="Top"/>

<ComboBox Margin="155,76,0,0" Text="{Binding CopsId}" x:Name="CopsId" DisplayMemberPath="FIO" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="241" Height="23" VerticalAlignment="Top"/>

<DatePicker SelectedDate="{Binding DateOfPayment}" Margin="155,104,0,0" Name="textBoxDateBRZD" Background="#997D7676" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="156" Height="23" VerticalAlignment="Top"/>

<TextBlock Margin="16,132,0,0" Foreground="White" HorizontalAlignment="Left" Width="114" Height="23" VerticalAlignment="Top"><Run Text="Сумма"/><Run Language="ru-ru" Text=" штрафа"/></TextBlock>

<TextBox Margin="155,132,0,0" Text="{Binding IncidentAmount}" BorderBrush="White" IsEnabled="True" HorizontalAlignment="Left" Width="156" Height="23" VerticalAlignment="Top"></TextBox>

</Grid>

## 3.2 Программирование информационной системы

**Код окна авторизации:**

namespace CopsSnitch

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

DispatcherTimer timer = new DispatcherTimer();

string code;

public MainWindow()

{

InitializeComponent();

}

public static class Globals

{

public static int userrole;

public static User userinfo { get; set; }

}

//Код для кода доступа в систему

private void gencode()

{

code = null;

Random random = new Random();

string[] massiveCharacters = new string[] { "2", "6", "7", "8", "a", "y", "e" };

for (int i = 0; i < 3; i++)

{

code += massiveCharacters[random.Next(0, massiveCharacters.Length)];

}

textBoxCodSpawn.Text = code;

timer.Interval = TimeSpan.FromSeconds(10);

timer.Tick += Timer\_Tick;

timer.Start();

textBoxCod.IsEnabled = true;

Vhod.IsEnabled = true;

Cod.IsEnabled = true;

Cod.Visibility = Visibility.Visible;

}

//Код времени ожидания ввода кода

void Timer\_Tick(object sender, EventArgs e)

{

code = null;

MessageBox.Show("Закончилось время ожидания. Повторите попытку");

timer.Stop();

}

//Код обновления кода для входа в систему

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

{

timer.Stop();

gencode();

Cod.Focus();

}

//Код проверки логина

private void LogBlock\_Up(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

using (var db = new CopsBaseEntities())

{

var login = db.User.AsNoTracking().FirstOrDefault(l => l.Login == textBoxLogin.Text.Trim());

if (login == null)

{

MessageBox.Show("Неверный логин");

}

else

{

textBoxPassword.IsEnabled = true;

textBoxLogin.IsEnabled = false;

textBoxPassword.Focus();

}

}

}

}

//Код проверки пароля

private void PassBlock\_Up(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

using (var db = new CopsBaseEntities())

{

var login = db.User.AsNoTracking().FirstOrDefault(l => l.Login == textBoxLogin.Text.Trim() & l.Password == textBoxPassword.Password.Trim());

if (login == null)

{

MessageBox.Show("Неверный пароль");

}

else

{

textBoxPassword.IsEnabled = false;

gencode();

Cod.Focus();

}

}

}

}

//Код для кнопки "Войти"

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

{

using (var db = new CopsBaseEntities())

{

var auth = db.User.AsNoTracking().FirstOrDefault(m => m.Login == textBoxLogin.Text && m.Password == textBoxPassword.Password);

if (auth != null & code == textBoxCod.Text)

{

timer.Stop();

Globals.userrole = auth.RoleId;

Globals.userinfo = auth;

Main Mwin = new Main();

Mwin.Show();

Close();

}

else

{

MessageBox.Show("Неверно написан код, повторите снова!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

timer.Stop();

}

}

}

}

}

**Код для главного окна отчетов:**

namespace CopsSnitch.Views

{

/// <summary>

/// Логика взаимодействия для Main.xaml

/// </summary>

public partial class Main : Window

{

public Main()

{

InitializeComponent();

Frame1.Navigate(new OwnerPage());

Manager.Frame1 = Frame1;

}

//Код для кнопки перехеда на страницу ДТП

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

{

Manager.Frame1.Navigate(new CarAccidentPage());

}

//Код для кнопки перехеда на страницу Сотрудников

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

{

Manager.Frame1.Navigate(new CopsPage());

}

//Код для кнопки выходы из программы

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

{

MainWindow Auth = new MainWindow();

Auth.Show();

Close();

}

//Код для кнопки перехеда на страницу Водителей

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

{

Manager.Frame1.Navigate(new OwnerPage());

}

}

}

**Код страницы Протокол:**

namespace CopsSnitch.Views

{

/// <summary>

/// Логика взаимодействия для ProtocolPage.xaml

/// </summary>

public partial class ProtocolPage : Page

{

public ProtocolPage()

{

InitializeComponent();

}

//Кнопка добавления

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

{

WindowEditProtocol WEP = new WindowEditProtocol(null);

WEP.Show();

}

//Кнопка редактирования

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

{

WindowEditProtocol WEG = new WindowEditProtocol((sender as Button).DataContext as Protocol);

WEG.Show();

}

//Код для кнопки удаления

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

{

var ProtocolForRemoving = ProtocolPP.SelectedItems.Cast<Protocol>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить следующие {ProtocolForRemoving.Count()} элементов?", "Внимание",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

CopsBaseEntities.GetContext().Protocol.RemoveRange(ProtocolForRemoving);

CopsBaseEntities.GetContext().SaveChanges();

MessageBox.Show("Данные удалены!");

ProtocolPP.ItemsSource = CopsBaseEntities.GetContext().Protocol.ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

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

{

if (Visibility == Visibility.Visible)

{

CopsBaseEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

ProtocolPP.ItemsSource = CopsBaseEntities.GetContext().Protocol.ToList();

}

}

//Кнопка создания отчетов

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

{

var allRequest = CopsBaseEntities.GetContext().Protocol.ToList();

var application = new Word.Application();

Word.Document document = application.Documents.Add();

Word.Paragraph userParagraph = document.Paragraphs.Add();

Word.Range userRange = userParagraph.Range;

userRange.Text = "Отчет по протоколу";

userRange.InsertParagraphAfter();

Word.Paragraph tableParagraph = document.Paragraphs.Add();

Word.Range tableRange = tableParagraph.Range;

Word.Table paymentsTable = document.Tables.Add(tableRange, allRequest.Count() + 1, 5);

paymentsTable.Borders.InsideLineStyle = paymentsTable.Borders.OutsideLineStyle

= Word.WdLineStyle.wdLineStyleSingle;

paymentsTable.Range.Cells.VerticalAlignment = Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

Word.Range cellRange;

//Создание таблицы отчета в Word

cellRange = paymentsTable.Cell(1, 1).Range;

cellRange.Text = "Номер протокола";

cellRange = paymentsTable.Cell(1, 2).Range;

cellRange.Text = "Виновный";

cellRange = paymentsTable.Cell(1, 3).Range;

cellRange.Text = "Сумма штрафа";

cellRange = paymentsTable.Cell(1, 4).Range;

cellRange.Text = "Дата оплаты";

cellRange = paymentsTable.Cell(1, 5).Range;

cellRange.Text = "Полицейский";

paymentsTable.Rows[1].Range.Bold = 1;

paymentsTable.Rows[1].Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

for (int i = 0; i < allRequest.Count(); i++)

{

var currentCategory = allRequest[i];

cellRange = paymentsTable.Cell(i + 2, 1).Range;

cellRange.Text = Convert.ToString(currentCategory.Id);

cellRange.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

cellRange = paymentsTable.Cell(i + 2, 2).Range;

cellRange.Text = Convert.ToString(currentCategory.Owner.FIO);

cellRange = paymentsTable.Cell(i + 2, 3).Range;

cellRange.Text = Convert.ToString(currentCategory.IncidentAmount);

cellRange = paymentsTable.Cell(i + 2, 4).Range;

cellRange.Text = currentCategory.DateOfPayment.ToString("dd.MM.yyyy");

cellRange = paymentsTable.Cell(i + 2, 5).Range;

cellRange.Text = Convert.ToString(currentCategory.Cops.FIO);

}

application.Visible = true;

}

}

}

**Код для таблицы редактирования ПТС водителя:**

namespace CopsSnitch.Views

{

/// <summary>

/// Логика взаимодействия для WindowEditPTS.xaml

/// </summary>

public partial class WindowEditPTS : Window

{

//Объявление переменной для бд

private PTS \_currentPTS= new PTS();

public WindowEditPTS(PTS selectedPTS)

{

InitializeComponent();

if (selectedPTS != null)

{

\_currentPTS = selectedPTS;

}

DataContext = \_currentPTS;

CmbOwner.ItemsSource = CopsBaseEntities.GetContext().Owner.ToList();

}

//Код для проверки редактирования

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

{

//Создание предупреждение с пустым полем

StringBuilder Errors = new StringBuilder();

var CurrentOwner = CmbOwner.SelectedItem as Owner;

if (CurrentOwner == null)

Errors.AppendLine("Выберите статус");

if (String.IsNullOrEmpty(\_currentPTS.RegistrNumber))

Errors.AppendLine("Укажите номер автомобиля");

if (String.IsNullOrEmpty(\_currentPTS.NumberVIN))

Errors.AppendLine("Укажите VIN номер автомобиля");

if (String.IsNullOrEmpty(\_currentPTS.CarBrand))

Errors.AppendLine("Укажите марку автомобиля");

if (String.IsNullOrEmpty(\_currentPTS.CarModel))

Errors.AppendLine("Укажите модель автомобиля");

if (String.IsNullOrEmpty(\_currentPTS.HP))

Errors.AppendLine("Укажите кол-во лошадинных сил автомобиля");

if (String.IsNullOrEmpty(\_currentPTS.TipTS))

Errors.AppendLine("Укажите тип автомобиля");

if (String.IsNullOrEmpty(\_currentPTS.ColorCar))

Errors.AppendLine("Укажите цвет автомобиля");

if (textBoxDate.SelectedDate == null)

Errors.AppendLine("Укажите дату выпуска");

if (Errors.Length > 0)

{

MessageBox.Show(Errors.ToString());

return;

}

if (\_currentPTS.Id >= 0)

{

CopsBaseEntities.GetContext().PTS.AddOrUpdate(\_currentPTS);

}

//Сохранение таблицы и бд

try

{

CopsBaseEntities.GetContext().SaveChanges();

MessageBox.Show("Информация о сотруднике сохранена");

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

}

# **ЗАКЛЮЧЕНИЕ**

В данный момент автоматизированные информационные системы являются актуальной темой.

При выполнении данной работы, а именно конструирования автоматизированной системы «Регистрация ДТП в ГИБДД», была спроектирована диаграмма: Entity-Relationship Diagram, была спроектирована база данных в приложении Microsoft SQL Server Management Studio 18. Данная автоматизированная информационная система была разработана для сотрудников ГИБДД. С её помощью можно быстро производить оформление протоколов, отслеживать произошедшие ДТП и выполнять необходимые отчёты.

Экономический эффект от внедрения автоматизированной информационной системы «Регистрация ДТП в ГИБДД» ожидается за счет сокращения затрат времени на выполняемые менеджером и работником ГИБДД, исключения ошибок при формировании отчетов, увеличения времени на анализ. При ручном создании отчетов могут быть допущены ошибки, правильно составленный алгоритм разрабатываемой программы экономит время и исключает ошибки.

Исходя из общего положения и опираясь на совокупность всех ранее вышеперечисленных и упомянутых фактов можно сделать вывод, что поставленные цель и задачи при проектировании информационной системы были выполнены в полной мере.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

* + 1. Джон, Скит C#. Программирование для профессионалов / Скит Джон. - М.: Диалектика / Вильямс**, 2017.**
    2. Вагнер, Билл С# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, **2021.**
    3. Климов, А. C#. Советы программистам / А. Климов. - М.: БХВ-Петербург, **2018.**

Интернет-ресурсы

* + 1. Документация по C#. Начало работы, руководства, справочные материалы. // Microsoft Learn. – Режим доступа:  <https://learn.microsoft.com/ru-ru/dotnet/csharp/>
    2. Общие сведения о WPF. – Режим доступа: <https://learn.microsoft.com/ru-RU/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8>
    3. Windows Presentation Foundation. Материал из Википедии. Режим доступа: <https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation>
    4. Что такое Windows Presentation Foundation - WPF .NET // Microsoft Learn – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0>