

**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО МГТУ «СТАНКИН»)**

**Институт                                                                                   Кафедра**

информационных                                                                      информационных систем

систем и технологий

**Отчет по самостоятельной работе**

по дисциплине **«Управление данными»**

на тему: «Проектирование БД ГИБДД»

**Студент**

группы ИДБ-16-06 **Агапов Д.О.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Подпись

**Руководитель Быстрикова В.А.**

старший преподаватель

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Подпись

Москва, 2018 г.

**Оглавление**

Введение

1. Анализ предметной области………………………………………………3
2. Концептуальное проектирование…………………………………………9
3. Логическое проектирование……………………………………………...12
4. Физическое проектирование…………………………………….……….22

Заключение…………………………………………………………………...30

Список использованной литературы………………………………………..31

Приложение А. SQL-операторы создания таблиц..………………………...32

Приложение Б. Пример заполнения таблиц………………………………...34

**Введение**

Объектом исследования данного проекта является ГИБДД. Актуальность проекта обуславливается тем, что у множества людей появляется возможность пользоваться личными транспортными средствами. В связи с этим, задачей данного курсового проекта является автоматизация работы ГИБДД.

Исходя из указанной цели можно выделить задачи, которые должны быть рассмотрены в данной работе:

1. Проектирование структуры БД
   1. Анализ предметной области
   2. Концептуальное проектирование
   3. Логическое проектирование
   4. Физическое проектирование
2. Создание пользовательского интерфейса

**1.1 Анализ предметной области**

Проектирование базы данных начинается с анализа предметной области. На данном этапе осуществляется анализ запросов пользователей, происходит выбор информационных объектов и их характеристик, от которых зависит содержание проектируемой БД. Анализ предметной области позволяет определить ее сущности и первоначальные требования к функциональности проекта. Предметной областью в данной работе является организация и работа ГИБДД.

Работа ГИБДД связана со множеством функций. Например, ведение данных о машинах, водителях, штрафах, истории ТС. Хотелось бы отметить другое. У ГИБДД есть функции, которые неявно выражены, однако это не отменяет того, что они есть: оформление ДТП, учет угнанных ТС, принятие квалификационных экзаменов на получение прав, регистрация и учет ТС.

В наши дни многие люди используют личный автотранспорт, с которым связанно множество проблем и забот. Только при покупке и продаже машины возникает необходимость проверить ряд документов, чтобы удостовериться, что с документами все в порядке. При этом, с развитием технологий, появилась возможность избегать долгий процесс проверки документов, проходя длинные очереди в ГИБДД. Достаточно открыть нужный сайт и узнать все необходимую информацию, при наличии соответствующих данных.

И в том и в другом случае, человеку все равно придется иметь дело с БД. Только в случае с личным посещением отделения ГИБДД, БД будет у сотрудника ГИБДД, он по вашим данным предоставит всю необходимую вам информацию. А в случае с посещением сайта, достаточно просто ввести необходимые данные и получить ответы на свои вопросы.

Одним из наиболее популярных сайтов является «Автокод» (https://avtokod.mos.ru). Свою задачу данный сайт выполняет полностью и популярен из-за простого и интуитивного интерфейса. На рис.1.1 представлена главная страница с различными вкладками. В зависимости от потребности пользователь может перейти на необходимую для него вкладку. Рассмотрим вкладку «Проверка автомобиля» (рис. 1.2). В данной вкладке пользователь вводит номер машины и СТС и получает информацию о том:

* Наименование машины;
* Год выпуска;
* Объем двигателя (Мощность);
* ДТП;
* Нахождение в розыске;
* Количество владельцев.



Рис. 1.1 Главная страница сайта Автокод

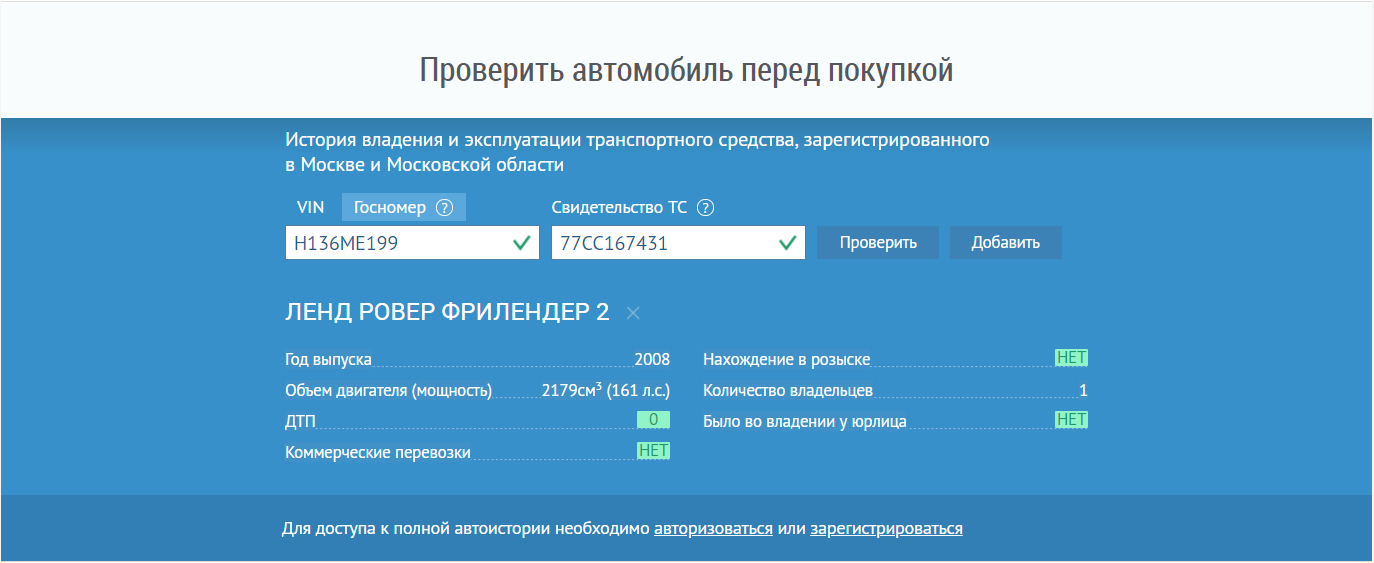


Рис. 1.2 Проверка автомобиля

На вкладке «Штрафы и нарушения» пользователь может проверить и оплатить выписанные ему штрафы (рис. 1.3) и ознакомиться с таблицей штрафов.

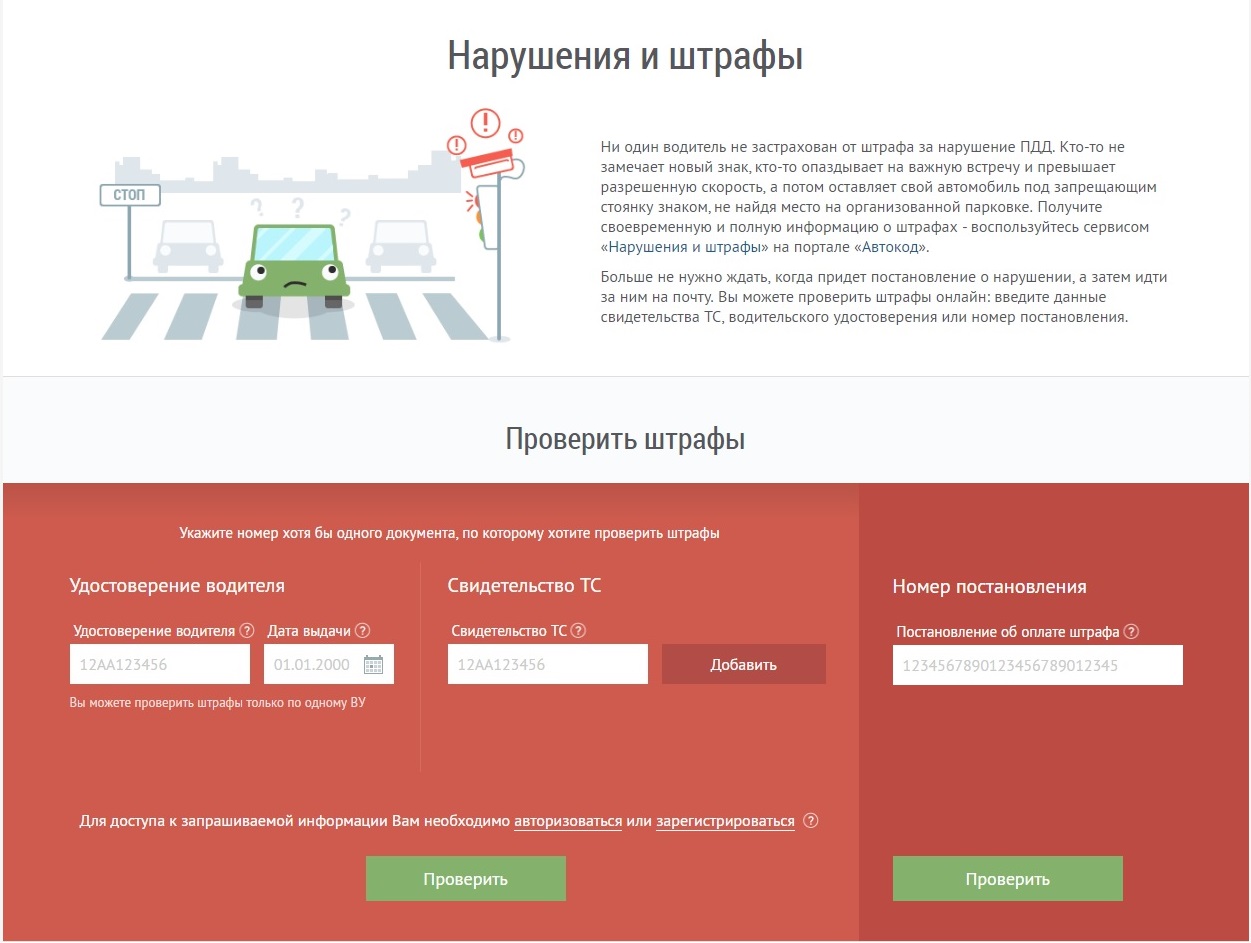


Рис. 1.3 Проверка штрафов

Также на вкладке «Запись в ГИБДД» пользователю доступно записаться не только в регистрационные подразделения ГИБДД, но и в экзаменационные отделение ГИБДД.

Есть еще вкладка «Проверка документов» (рис.1.4), где пользователь может проверить документы как автомобиля, так и водителя.

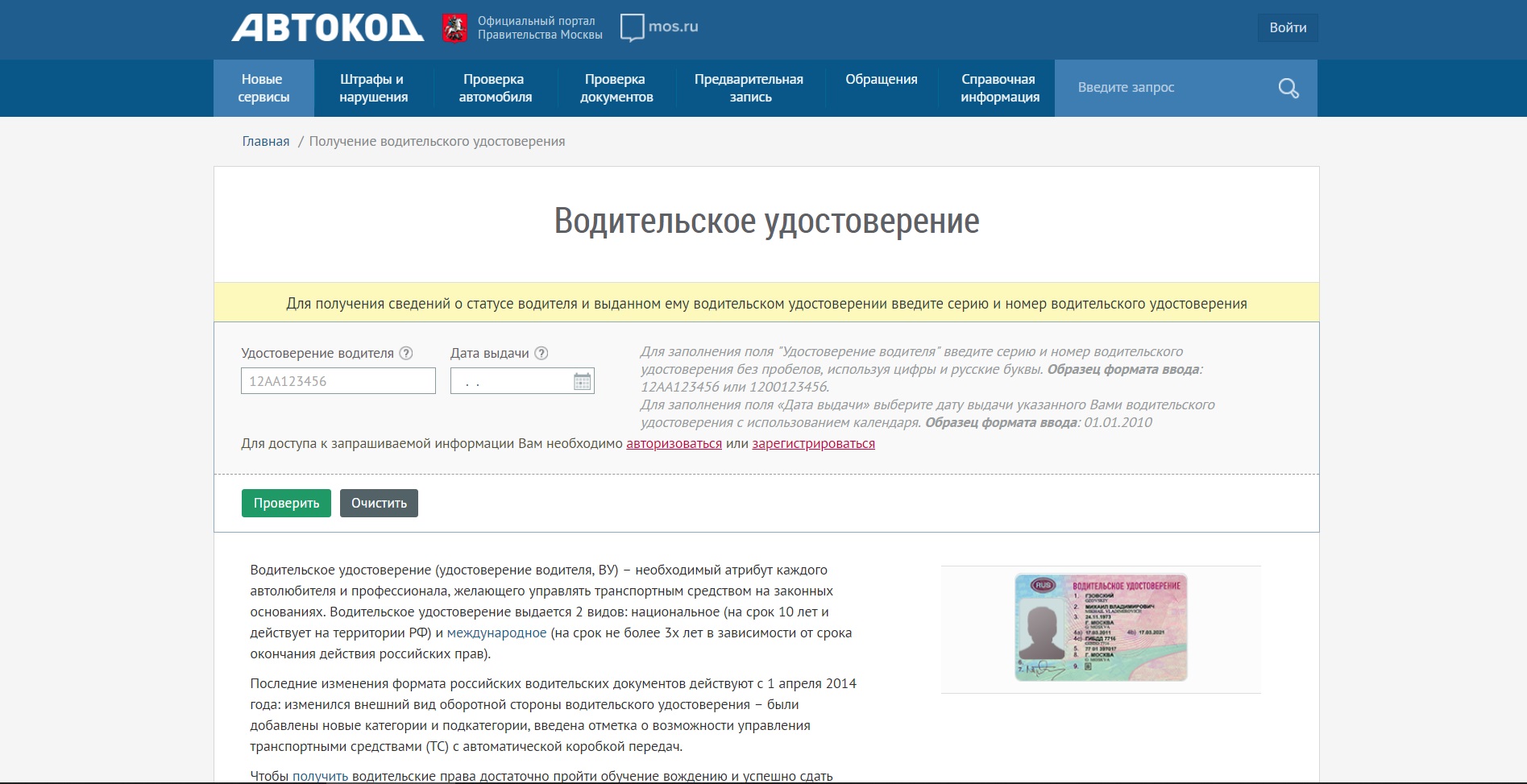


Рис. 1.4 Проверка документов

Аналогичный сайт есть и у ГИБДД (https://гибдд.рф). Информации на этом сайте в разы больше, но не вся она необходима пользователю. Именно поэтому данный сайт менее популярен чем «Автокод». На главной странице (рис. 1.5) также можно найти вкладки, такие как:

* Проверка штрафов;
* Проверка водителя;
* Проверка автомобиля;
* Нарушения и штрафы;
* Новости.

Перейдя по ним, пользователю необходимо будет ввести нужную информацию. В качестве примера рассмотрим вкладку «Проверка штрафов» (рис. 1.6). После предоставления данных пользователю предоставят штрафы, если они имеются, или сообщат, что штрафов нет.



Рис. 1.5 Главная страница сайта ГИБДД

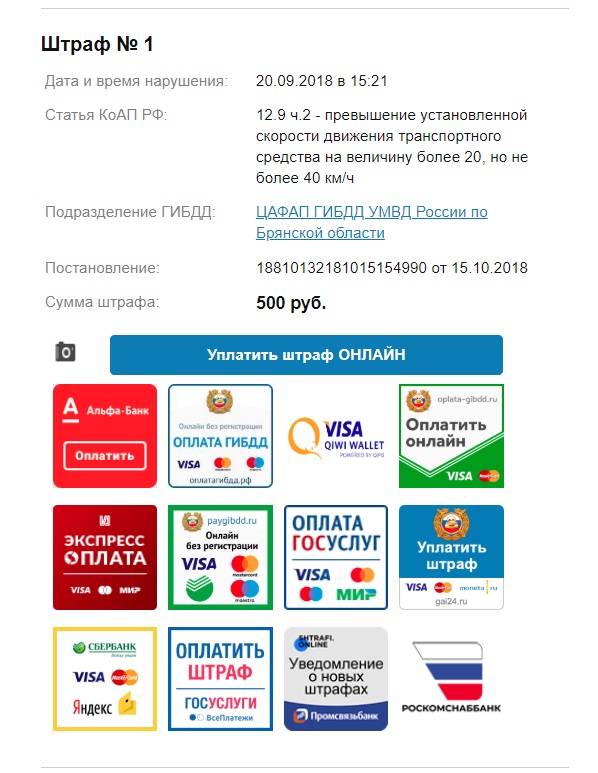


Рис. 1.6 Проверка штрафов

Проанализировав сайт «Автокод» и «ГИБДД», можно сказать, что оба сайта имеют схожие функции: проверка автомобиля, проверка штрафов, проверка документов водителя. При этом, если сравнивать различающиеся функции, то можно составить большой список. Сайтом «Автокод» пользоваться гораздо удобнее и приятнее, так как он более ориентирован на простых пользователей. Сайт «ГИБДД» имеет более обширную и специфическую информацию, которые не нужны рядовым пользователем, из-за чего пользоваться им немного сложнее.

Проанализировав предметную область, можно выделить цель проектирования БД ГИБДД: разработка такой БД ГИБДД, которой могли бы пользоваться как простые пользователи, которые хотят быстро и просто найти сведения о полученных штрафах, проверить историю и документы покупаемой машины, проверить ВУ, так и сотрудники ГИБДД, которым нужно быстро добавлять, искать и удалять необходимую информацию.

**1.2 Концептуальное проектирование**

Это первая стадия проектирования, которая заключается в построении информационной модели наиболее высокого уровня абстракций. Результатом концептуального проектирования является выявление объектов, информация о которых и будет храниться в БД. Чтобы реализовать концептуальное проектирование используется диаграмма вариантов использования.

Данная диаграмма предназначена для того, чтобы представить проектируемую систему в форме вариантов использования, с которой взаимодействует внешние сущности или действующие лица. Действующим лицом называется любая сущность, взаимодействующая извне. Вариант использования — это описание сервисов, которые предоставлены действующему лицу системой, то есть определяет набор действий, совершаемой системой при диалоге с действующим лицом.

База данных ГИБДД должна содержать в себе информацию о водителях, ТС, штрафах, владельцах, ДТП, экзаменах, угнанных ТС. Действующими лицами будут:

* сотрудник ГИБДД;
* владелец ТС.

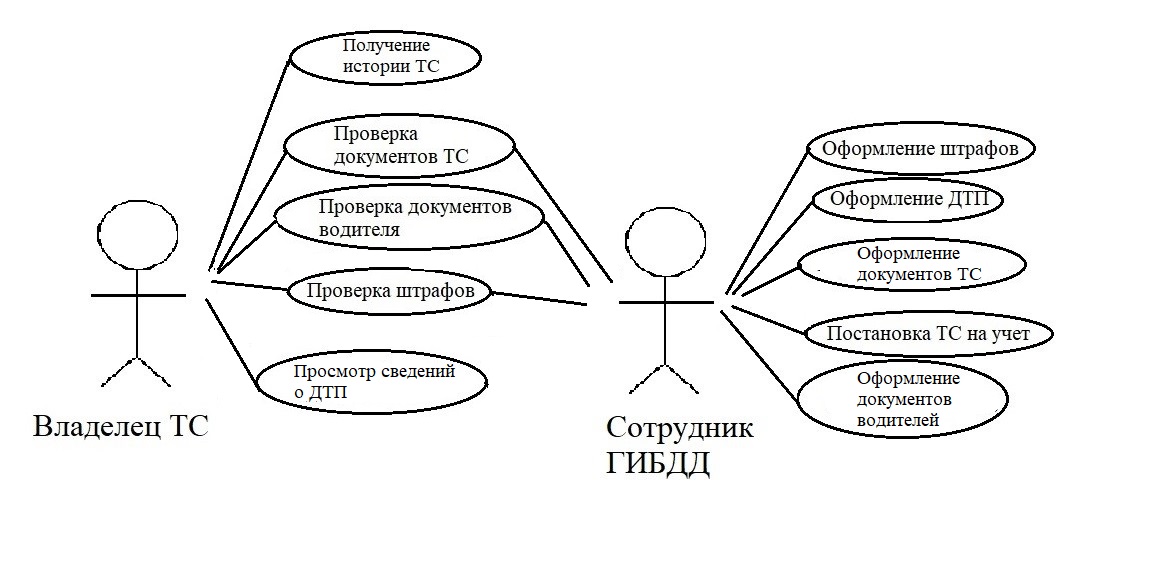
У владельца ТС есть следующие возможности:

1. Получение истории ТС
2. Проверка документов ТС
3. Проверка документов водителя
4. Проверка штрафов
5. Просмотр сведений о ДТП

Сотрудник ГИБДД может совершать следующие действия:

1. Проверка документов ТС
2. Проверка документов водителя
3. Проверка штрафов
4. Оформление штрафов
5. Оформление ДТП
6. Оформление документов ТС
7. Постановка на учет ТС
8. Оформление документов водителей

Исходя из возможных вариантов использования диаграмма вариантов использования будет иметь следующий вид (рис.2.1)



­­­­­­­Рис. 2.1 Диаграмма вариантов использования

Исходя из выделенных задач необходимо хранение и предоставление данных о следующих объектах:

1. Водитель
2. Владелец
3. Транспортное средство
4. Документы ТС
5. Штрафы
6. Сотрудник ГИБДД
7. История ТС
8. Водительское удостоверение
9. Таблица штрафов
10. ДТП

**1.3 Логическое проектирование**

Целью логического проектирования является создание логической модели, основываясь на отдельных моделях пользователей. Логическая модель является независимой от особенностей, реально используемых СУБД. Для реляционной модели данных логическая модель является набором схем отношений, с указанием первичных ключей и связей между этими отношениями, представляющих собой внешние ключи.

Диаграмма сущность-связь (ER-диаграмма) служит инструментом для построения семантической модели. Данные диаграммы используют изображение сущностей предметной области, их взаимодействие и соответствующих атрибутов.

В качестве сущности выступает объект (человек, событие, состояние, предмет и т.д.) данные о котором должны быть сохранены и доступны. Атрибут – это неотъемлемое и необходимое свойство сущности. Каждая сущность определяется определённым набором атрибутов. Связью будет выступать логическая связь между сущностями. Связь может принимать несколько состояний: 1:1, 1:М, М:1 и М:М.

На этапе концептуального проектирования были выделены объекты, информацию о которых необходимо хранить в БД. Эти объекты станут сущностями при построении ER-диаграммы. Рассмотрим ER-диаграмму «ТС имеет штраф»

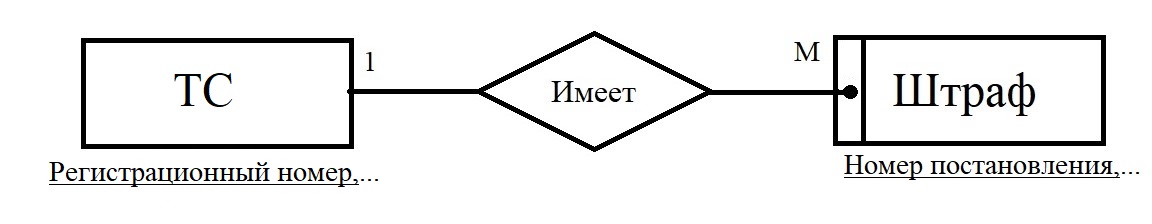


Рис 3.1 ТС имеет штраф

Связь ИМЕЕТ имеет тип 1:М, так как каждое ТС может получить один или множество штрафов, а штраф с определенным номером постановления может быть выписан только на одно ТС. Сущность ТС имеет необязательный класс принадлежности так как, на ТС необязательно можно получить штраф. Сущность ШТРАФ имеет обязательный класс принадлежности так как, ШТРАФ не может быть выписан, без указания на какое ТС его выписывают.

Рассмотрим ER-диаграмму «Владелец имеет ТС»

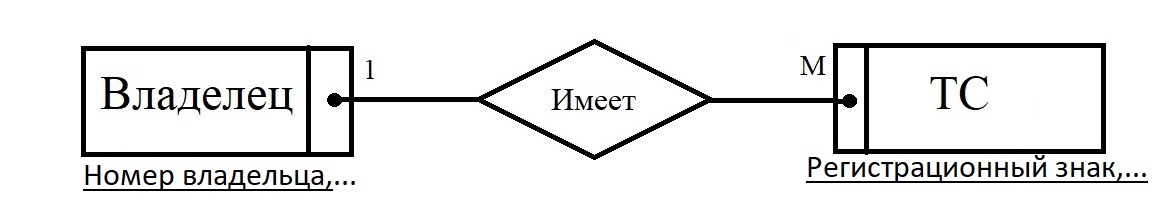


Рис. 3.2 Владелец имеет ТС

Связь ИМЕЕТ имеет тип 1:М, так как у одного человека может быть несколько машин, но у каждой машины может быть только один владелец. Сущность ВЛАДЕЛЕЦ имеет обязательный класс принадлежности, так как у него обязательно должна быть машина. Сущность ТС имеет обязательный класс принадлежности, так как мы предполагаем, что если машина есть в БД ГИБДД, то у нее есть ВЛАДЕЛЕЦ.

Рассмотрим ER-диаграмму «Владелец получает Документ ТС»

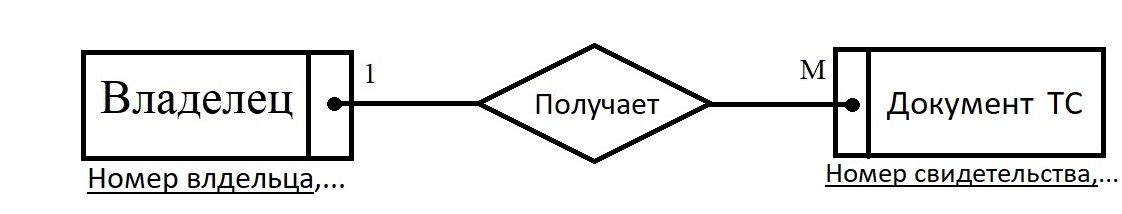


Рис. 3.3 Владелец получает документы ТС

Связь ПОЛУЧАЕТ имеет связь 1:М, так как каждый владелец может получить несколько свидетельств ТС. Сущность ВЛАДЕЛЕЦ имеет обязательный класс принадлежности, так как при постановке на учет ТС он должен получить ДОКУМЕНТ ТС. Сущность ДОКУМЕНТ ТС имеет обязательный класс принадлежности, так как мы предполагаем, что человек получает документы ТС обязательно владелец ТС, иначе он удаляется из базы.

Рассмотрим ER-диаграмму «ТС имеет Автоисторию»

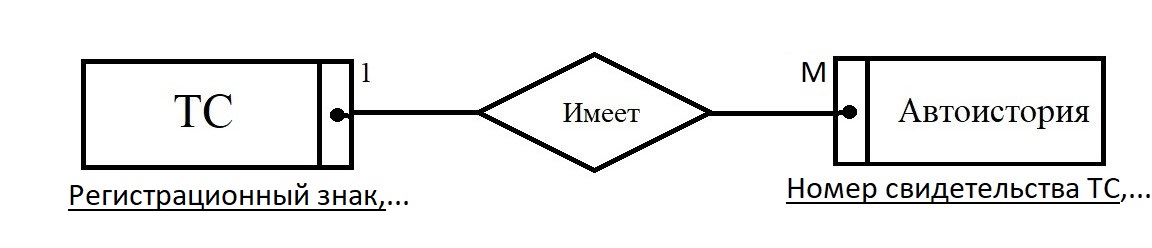


Рис. 3.4 ТС имеет Автоистоию

Связь ИМЕЕТ имеет связь 1:М, так как одна машина может иметь разные автоистории. Сущность ТС имеет обязательный класс принадлежности, так как у каждой машины есть автоистория. Сущность АВТОИСТОРИЯ имеет обязательный класс принадлежности, так как она не может существовать без ТС.

Рассмотрим ER-диаграмму «Сотрудник ГИБДД заполняет штраф»



Рис. 3.5 Сотрудник ГИБДД заполняет штраф

Связь ЗАПОЛНЯЕТ имеет связь 1:М, так как сотрудник ГИБДД может выписать один или несколько штрафов, но один штраф не может быть выписан несколькими сотрудниками ГИБДД. Сущность СОТРУДНИК ГИБДД имеет необязательный класс принадлежности, так как сотрудник ГИБДД может не выписать штраф. Сущность ШТРАФЫ имеет обязательный класс принадлежности так как, штраф может выписать только сотрудник ГИБДД.

Рассмотрим ER-диаграмму «Сотрудник ГИБДД выдает документ ТС»

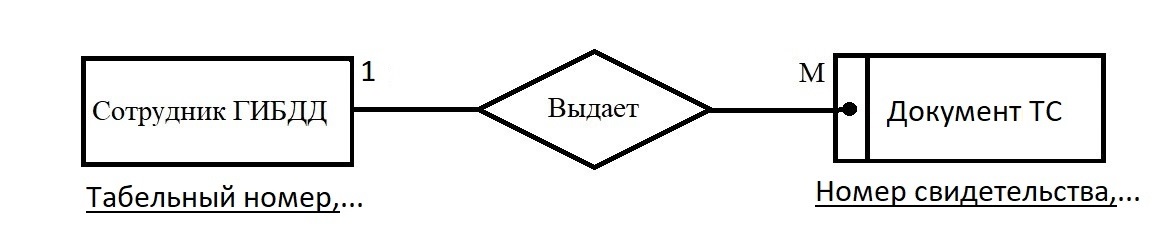


Рис. 3.6 Сотрудник ГИБДД выдаёт документ ТС

Связь ВЫДАЕТ имеет связь 1:М, так как СОТРУДНИК ГИБДД может выдать одни или несколько документов ТС но , одни ДОКУМЕНТ ТС могут быть выданы одним сотрудниками. Сущность СОТРУДНИК ГИБДД имеет необязательный класс принадлежности, так как не каждый сотрудник выдает ДОКУМЕНТ ТС. Сущность ДОКУМЕНТ ТС имеет обязательную сущность, так как оно обязательно будет выдано сотрудником ГИБДД.

Рассмотрим ER-диаграмму «ТС имеет документ ТС»

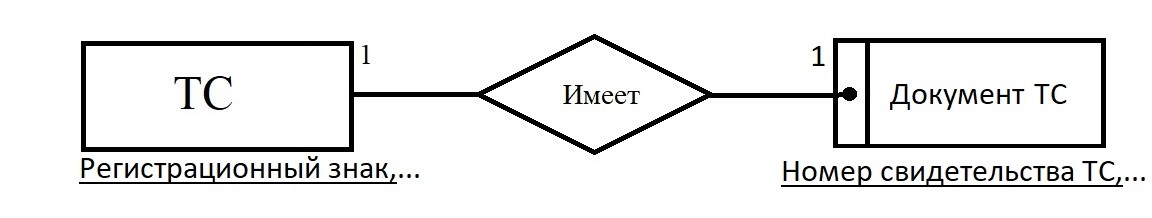


Рис. 3.7 ТС имеет документ ТС

Связь ИМЕЕТ имеет тип 1:1, так как у одной машины может быть только одни документы и наоборот. Сущность ТС имеет необязательный класс принадлежности, так как не все машины зарегистрированы. Сущность ДОКУМЕНТ ТС имеет обязательный класс принадлежности, так как документы ТС имеют принадлежность к той машине, для которой их сделали.

Рассмотрим ER-диаграмму «Водитель имеет ВУ»

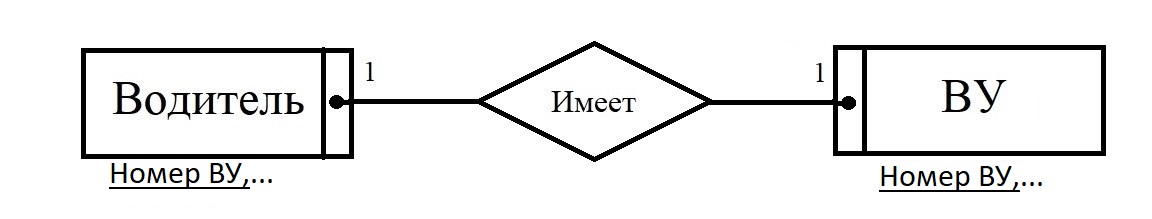


Рис. 3.8 Водитель имеет ВУ

Связь ИМЕЕТ имеет тип 1:1, так как одному водителю может принадлежать только одно ВУ и наоборот. Сущность ВОДИТЕЛЬ имеет обязательный класс принадлежности, так как у ВОДИТЕЛЯ обязательно должно быть ВУ. Сущность ВУ имеет обязательный класс принадлежности, так как оно не появиться без водителя.

Рассмотрим ER-диаграмму «Штраф выписан по статье»

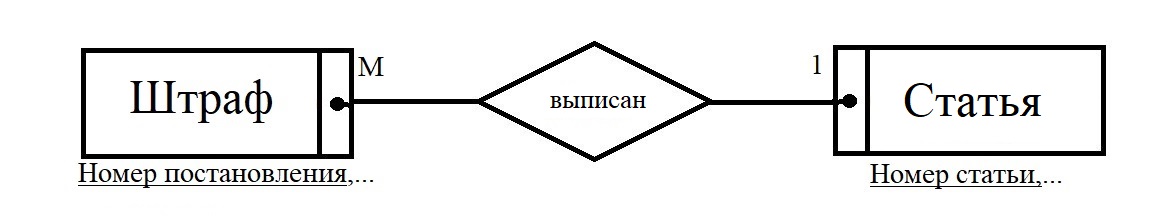


Рис. 3.9 Штраф выписан по статье

Связь ВЫПИСАН имеет тип М:1, так как множество различных штрафов описываются статьями. Сущность ШТРАФ имеет обязательный класс принадлежности, так как он обязательно должен быть выписан по статье. Сущность СТАТЬЯ имеет обязательный класс принадлежности, так как в статье обязательно содержится штраф.

Рассмотрим ER-диаграмму «Сотрудник ГИБДД оформляет ДТП»

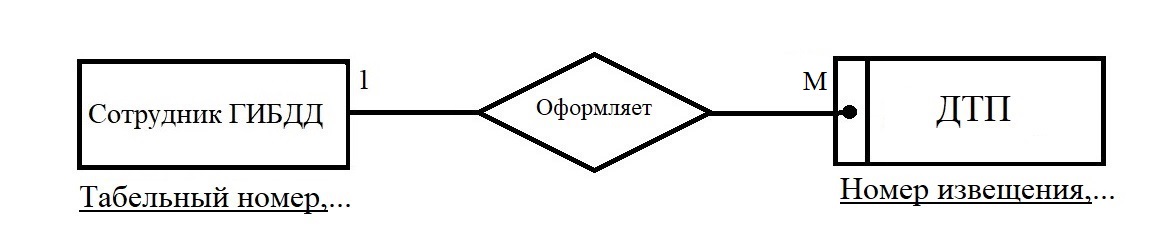


Рис. 3.10 Сотрудник ГИБДД оформляет ДТП

Связь ОФОРМЛЯЕТ имеет тип 1:М, так как один сотрудник ГИБДД оформляет множество различных ДТП. Сущность СОТРУДНИК ГИБДД имеет необязательный класс принадлежности, так как необязательно сотрудник ГИБДД будет оформлять ДТП. Сущность ДТП имеет обязательный класс принадлежности, так как ДТП обязательно оформляет сотрудник ГИБДД.

Рассмотрим ER-диаграмму «Водитель и ТС участвуют в ДТП»

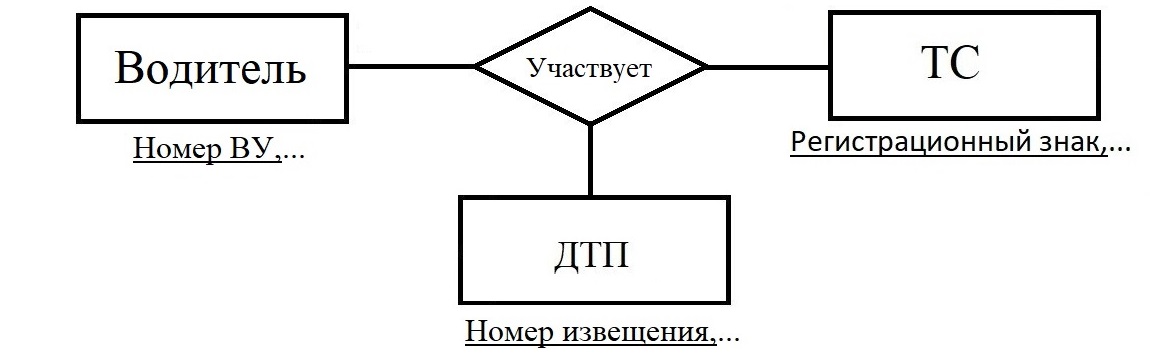


Рис. 3.11 Водитель и ТС участвуют в ДТП

Полученную сложную связь необходимо устранить, введя новую сущность УЧАСТНИК ДТП, и создать двусторонние связи 1:М между каждой из первоначальных сущностей и сущность участник ДТП.

Рассмотрим ER-диаграмму «Водитель является участником ДТП»



Рис. 3.12 Водитель является участником ДТП

Связь ЯВЛЯЕТСЯ имеет тип 1:М, так как один водитель может много раз участвовать в ДТП. Сущность ВОДИТЕЛЬ имеет необязательный класс принадлежности, так как необязательно водитель будет участником ДТП. Сущность УЧАСТНИК ДТП имеет обязательный класс принадлежности, так как в ДТП обязательно участвует водитель.

Рассмотрим ER-диаграмму «ТС является участником ДТП»



Рис. 3.13 ТС является участником ДТП

Связь ЯВЛЯЕТСЯ имеет тип 1:М, так как одно ТС может много раз участвовать в ДТП. Сущность ТС имеет необязательный класс принадлежности, так как необязательно ТС будет участвовать в ДТП. Сущность УЧАСТНИК ДТП имеет обязательный класс принадлежности, так как в ДТП обязательно участвует ТС.

Рассмотрим ER-диаграмму «В ДТП есть участники ДТП»

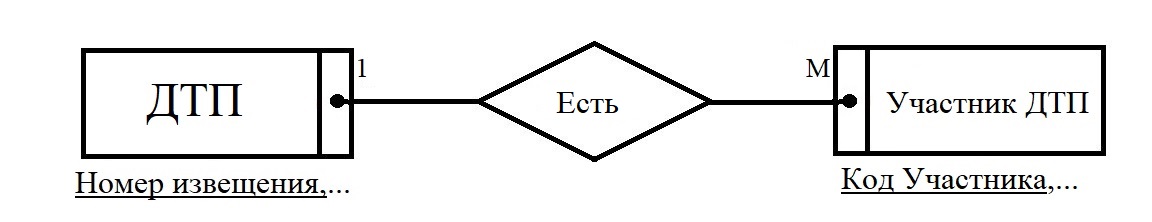


Рис. 3.14 В ДТП есть участники ДТП

Связь ЕСТЬ имеет тип 1:М, так как в одном ДТП может быть много участников ДТП. Сущность ДТП имеет обязательный класс принадлежности, так как обязательно в ДТП кто-то участвует. Сущность УЧАСТНИК ДТП имеет обязательный класс принадлежности, так как участники ДТП обязательно участвуют в ДТП.

Далее нужно сформировать набор предварительных отношений с указание первичного ключа для каждого из них, применив правила формирования отношений.

Связь ИМЕЕТ (рис. 3.1) удовлетворяет условиям правила 4, в соответствии с которыми получаем 2 отношения:

1. ТС (РегистрационныйЗнак,…)
2. Штраф (НомерПостановления, РегистрационныйЗнак,…)

Связь ИМЕЕТ (рис. 3.2) удовлетворяет условиям правила 4, в соответствии с которыми получаем 3 отношения:

1. Владелец (НомерВладельца,…)
2. ТС (РегистрационныйЗнак, НомерВладельца,…)

Связь ПОЛУЧАЕТ (рис. 3.3) удовлетворяет условиям правила 4, в соответствии с которыми получаем 2 отношения:

1. Владелец (НомерВладельца,…)
2. Документ ТС (НомерСвидетельстваТС, НомерВладельца,…)

Связь ИМЕЕТ (рис. 3.4) удовлетворяет условиям правила 4, в соответствии с которыми получаем 2 отношение:

1. ТС (РегистрационныйЗнак,…)
2. Автоистория (НомерСвидетельстваТС, РегистрационныйЗнак,…)

Связь ЗАПОЛНЯЕТ (рис. 3.5) удовлетворяет условиям правила 4, в соответствии с которыми получаем 2 отношения:

1. Сотрудник ГИБДД (ТабельныйНомер,…)
2. Штраф (НомерПостановления, ТабельныйНомер,…)

Связь ВЫДАЕТ (рис. 3.6) удовлетворяет условиям правила 4, в соответствии с которыми получаем 2 отношения:

1. Сотрудник ГИБДД (ТабельныйНомер,…)
2. Документ ТС (НомерСвидетельстваТС, ТабельныйНомер,…)

Связь ИМЕЕТ (рис. 3.7) удовлетворяет условиям правила 2, в соответствии с которыми получаем 2 отношение:

1. ТС (РегистрационныйЗнак,…)
2. Документ ТС (НомерСвидетельстваТС, РегистрационныйЗнак,…)

Связь ИМЕЕТ (рис. 3.8) удовлетворяет условиям правила 1, в соответствии с которыми получаем 1 отношение:

1. Водитель (НомерВУ,…)

Связь ВЫПИСАН (рис. 3.9) удовлетворяет условиям правила 4, в соответствии с которыми получаем 2 отношения:

1. Статья (НомерСтатьи,…)
2. Штраф (НомерПостановления, НомерСтатьи,…)

Связь ОФОРМЛЯЕТ (рис. 3.10) удовлетворяет условиям привала 4, в соответствии с которыми получаем 2 отношения

1. Сотрудник ГИБДД (ТабельныйНомер,…)
2. ДТП (НомерИзвещения, ТабельныйНомер,…)

Связь ЯВЛЯЕТСЯ (рис. 3.12) удовлетворяет условиям привала 4, в соответствии с которыми получаем 2 отношения

1. Водитель (НомерВУ,…)
2. Участник ДТП (КодУчастника, НомерВУ,…)

Связь ЯВЛЯЕТСЯ (рис. 3.13) удовлетворяет условиям привала 4, в соответствии с которыми получаем 2 отношения

1. ТС (РегистрационныйЗнак,…)
2. Участник ДТП (КодУчастника, РегистрационныйЗнак,…)

Связь ОФОРМЛЯЕТ (рис. 3.14) удовлетворяет условиям привала 4, в соответствии с которыми получаем 2 отношения

1. ДТП (НомерИзвещения,…)
2. Участник ДТП (КодУчастника, НомерИзвещения,…)

Добавим неключевые атрибуты в каждое из предварительных отношений:

1. Водитель (Номер ВУ, ФИО, Место прописки, Город, Телефон Дата рождения, Лишение прав, Открытые категории, Стаж, Дата выдачи,)
2. Штраф (Номер постановления, Регистрационный знак, Дата нарушения, Место нарушения, Номер статьи, Сумма штрафа, Лишение ВУ, Дата оплаты, Табельный номер)
3. Владелец (Номер владельца, ФИО, Место прописки, Дата рождения, Город, Телефон)
4. ТС (Регистрационный знак, Номер владельца, VIN, Марка, Модель, Год выпуска, Мощность, Объем двигателя, Цвет машины)
5. Документ ТС (Номер свидетельства ТС, Регистрационный знак, VIN, Номер владельца, Цвет Машины, Максимальная масса, Табельный номер)
6. Сотрудник ГИБДД (Табельный номер, ФИО, Звание, Стаж, Номер жетона, Код подразделения)
7. Автоистория (Номер свидетельства ТС, VIN, Регистрационный знак, Розыск, Количество владельцев, Ограничения)
8. Статья (Номер статьи, Описание штрафа, Штраф, Сумма штрафа)
9. ДТП (Номер Извещения, Дата ДТП, Место ДТП, Описание ДТП, Общий Ущерб, Табельный номер)
10. Участник ДТП (Код Участника, Номер ВУ, Регистрационный Знак, Номер Извещения, Статус)

**1.4 Физическое проектирование**

Физическое проектирование – этап проектирования схемы базы данных. Основным источником данных для данного этапа является этап логического проектирования. Исходя из всех особенностей СУБД происходит этап физического проектирования. Данные особенности могут выражаться в ограничениях на именования объектов или ограничения на различные типы данных. На этапе физического проектирования разработчиком могут быть приняты решения, которые могут повлиять или изменить логическую модель.

При создании базы данных была выбрана РСУБД Microsoft SQL Server. SQL сервер - СУБД, которая предназначена для хранения базы данных и обеспечения доступа к этим данным из других программ. Сложный доступ к данным используется для надежности их хранения. SQL сервер позволяет резервное копирование в любой момент рабочего дня без отключения пользователей. Если размер вашей базы данных стремиться к гигабайту и продолжает увеличиваться, то SQL сервер единственно возможный метод обеспечения ее функционирования.

Исходя из данных полученных на этапе логического проектирования, где были получены ER-диаграммы и сформированы отношения между сущностями, построим физическую базу данных. Для создания базы данных в SQL Server необходимо атрибутам установить типы данных, выбрать первичные и внешне ключи, создать ограничения для некоторых атрибутов, выбрать значение по умолчанию.

В приложении А (п.1) приведен код SQL-оператора для создания таблицы «Водитель», в приложении Б (рис. Б.1) приведен пример заполнения таблицы «Водитель».

Таблица 1

Структура таблицы «Водитель»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| НомерВУ | Integer | NOT NULL | Первичный |  |  |  |
| ФИО | Varchar(50) | NOT NULL |  |  |  |  |
| МестоПрописки | Varchar(100) | NOT NULL |  |  |  |  |
| Город | Varchar(50) | NOT NULL |  |  |  |  |
| Телефон | integer | NOT NULL |  |  |  |  |
| ДатаРождения | Date | NOT NULL |  |  |  |  |
| ЛишениеПрав | Varchar(3) |  |  | Нет | Да или Нет |  |
| ОткрытыеКатегории | Varchar(10) | NOT NULL |  | B | «А», «B», «C», «D», «M» |  |
| Стаж | Integer | NOT NULL |  | 0 | >=0 |  |
| ДатаВыдачи | date | NOT NULL |  |  |  |  |

В приложении А (п.2) приведен код SQL-оператора для создания таблицы «Владелец», в приложении Б (рис. Б.2) приведен пример заполнения таблицы «Владелец».

Таблица 2

Структура таблицы «Владелец»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| НомерВладельца | Integer | NOT NULL | Первичный |  |  |  |
| ФИО | Varchar(50) | NOT NULL |  |  |  |  |
| МестоПрописки | Varchar(50) | NOT NULL |  |  |  |  |
| ДатаРождения | Date | NOT NULL |  |  |  |  |
| Город | Varchar(50) | NOT NULL |  |  |  |  |
| Телефон | Integer |  |  |  |  |  |

В приложении А (п.3) приведен код SQL-оператора для создания таблицы «Штраф», в приложении Б (рис. Б.3) приведен пример заполнения таблицы «Штраф».

Таблица 3

Структура таблицы «Штраф»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| НомерПостановления | Integer | NOT NULL | Первичный |  |  |  |
| РегистрационныйЗнак | Nchar(9) | NOT NULL | Внешний |  |  | ТС(РегистрационныйЗнак) |
| ДатаНарушения | Date | NOT NULL |  |  |  |  |
| МестоНарушения | Varchar(100) | NOT NULL |  |  |  |  |
| НомерСтатьи | Varchar(10) | NOT NULL | Внешний |  |  | Статья(НомерСтатьи) |
| Сумма Штрафа | Smallint | NOT NULL |  |  |  |  |
| ЛишениеВУ | Varchar(10) | NOT NULL |  | Нет | Да или Нет |  |
| Оплата | Date |  |  |  | Не раньше даты нарушения |  |
| ТабельныйНомер | integer | NOT NULL | Внешний |  |  | Сотрудник ГИБДД(ТабельныйНомер) |

В приложении А (п.4) приведен код SQL-оператора для создания таблицы «ТС», в приложении Б (рис. Б.4) приведен пример заполнения таблицы «ТС».

Таблица 4

Структура таблицы «ТС»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| РегистрационыйЗнак | Nchar(9) | NOT NULL | Первичный |  |  |  |
| НомерВладельца | Integer | NOT NULL | Внешний |  |  | Владелец(НомерВладелца |
| VIN | Varchar(50) | NOT NULL |  |  |  |  |
| Марка | Varchar(50) | NOT NULL |  |  |  |  |
| Модель | Varchar(50) | NOT NULL |  |  |  |  |
| ГодВыпуска | Year | NOT NULL |  |  |  |  |
| Мощность | smallint | NOT NULL |  |  |  |  |
| ОбъемДвигателя | Char(3) | NOT NULL |  |  |  |  |
| ЦветМашины | Varchar(50) | NOT NULL |  |  |  |  |

В приложении А (п.5) приведен код SQL-оператора для создания таблицы «Статья», в приложении Б (рис. Б.5) приведен пример заполнения таблицы «Статья».

Таблица 5

Структура таблицы «Статья»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| НомерСтатьи | Varchar(10) | NOT NULL | Первичный |  |  |  |
| ОписаниеШтрафа | Varchar(50) | NOT NULL |  |  |  |  |
| Штраф | Varchar(50) | NOT NULL |  | Денежный |  |  |
| Сумма штрафа | Integer | NOT NULL |  |  | >=0 |  |

В приложении А (п.6) приведен код SQL-оператора для создания таблицы «Сотрудник ГИБДД», в приложении Б (рис. Б.6) приведен пример заполнения таблицы «Сотрудник ГИБДД».

Таблица 6

Структура таблицы «Сотрудник ГИБДД»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| Табельный номер | Integer | NOT NULL | Первичный |  |  |  |
| ФИО | Varchar(50) | NOT NULL |  |  |  |  |
| Звание | Varchar(50) | NOT NULL |  |  |  |  |
| Стаж | Integer | NOT NULL |  | 0 | >=0 |  |
| НомерЖетона | Integer | NOT NULL |  |  |  |  |
| КодПодразделения | Integer | NOT NULL |  |  |  |  |

В приложении А (п.7) приведен код SQL-оператора для создания таблицы «ДТП», в приложении Б (рис. Б.7) приведен пример заполнения таблицы «ДТП».

Таблица 7

Структура таблицы «ДТП»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| НомерИзвещения | integer | NOT NULL | Первичный |  |  |  |
| ДатаДТП | date | NOT NULL |  |  |  |  |
| МестоДТП | Varchar(50) | NOT NULL |  |  |  |  |
| ОписаниеДТП | Varchar(100) | NOT NULL |  |  |  |  |
| ОбщийУщерб | integer | NOT NULL |  |  |  |  |
| ТабельныйНомер | integer |  | внешний |  |  | СотрудникГибдд(ТабельныйНомер) |

В приложении А (п.8) приведен код SQL-оператора для создания таблицы «Документ ТС», в приложении Б (рис. Б.8) приведен пример заполнения таблицы «Документ ТС».

Таблица 8

Структура таблицы «Документ ТС»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| НомерСвидетельстваТС | Integer | NOT NULL | Первичный |  |  |  |
| VIN | Varchar(50) | NOT NULL |  |  |  |  |
| НомерВладельца | Integer | NOT NULL | Внешний |  |  | Владелец(НомерВладельца |
| ЦветМашины | Varchar(50) | NOT NULL |  |  |  |  |
| МаксимальнаяМасса | integer | NOT NULL |  |  |  |  |
| ТабельныйНомер | Integer | NOT NULL | Внешний |  |  | СотрудникГибдд(ТабельныйНомер) |
| РегистрационыйЗнак | Nchar(9) | NOT NULL | Внешний |  |  | ТС(РегистрационыйЗнак) |

В приложении А (п.9) приведен код SQL-оператора для создания таблицы «Автоистория», в приложении Б (рис. Б.9) приведен пример заполнения таблицы «Автоистория».

Таблица 9

Структура таблицы «Автоистория»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| РегистрационыйЗнак | Nchar(9) | NOT NULL | Внешний |  |  | ТС(РегистрационыйЗнак) |
| VIN | Varchar(50) | NOT NULL |  |  |  |  |
| ДТП | Integer | NOT NULL |  | 0 | >=0 |  |
| НомерСвидетельстваТС | Integer | NOT NULL | Первичный, Внешний |  |  | Документ ТС (Номер СвидетельстваТС) |
| Розыск | Varchar(3) | NOT NULL |  | Нет | Да или Нет |  |
| Количество Владельцев | Integer | NOT NULL |  | 1 | >0 |  |
| Ограничения | Varchar(50) | NOT NULL |  | Нет |  |  |

В приложении А (п.10) приведен код SQL-оператора для создания таблицы «Участник ДТП», в приложении Б (рис. Б.10) приведен пример заполнения таблицы «Участник ДТП».

Таблица 10

Структура таблицы «Участник ДТП»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя поля | Тип Данных | Обязательное поле | Ключ | Значение по умолчанию | Ограничения | Ссылка |
| НомерИзвещения | integer | NOT NULL | Внешний |  |  | ДТП(НомерИзвещения) |
| КодУчастника | integer | NOT NULL | Первичный |  |  |  |
| РегистрационныйЗнак | Nchar(9) | NOT NULL | Внешний |  |  | ТС(РегистрационыйЗнак) |
| НомерВУ | integer | NOT NULL | Внешний |  |  | Водитель(НомерВУ) |
| Статус | Varchar(9) | NOT NULL |  | Невиновен | Виновен или Невиновен |  |

После создания структуры таблиц необходимо установить связи между таблицами. Для построения схемы данных (рис. 4.1) была использована SQL Server Manegment Studio.

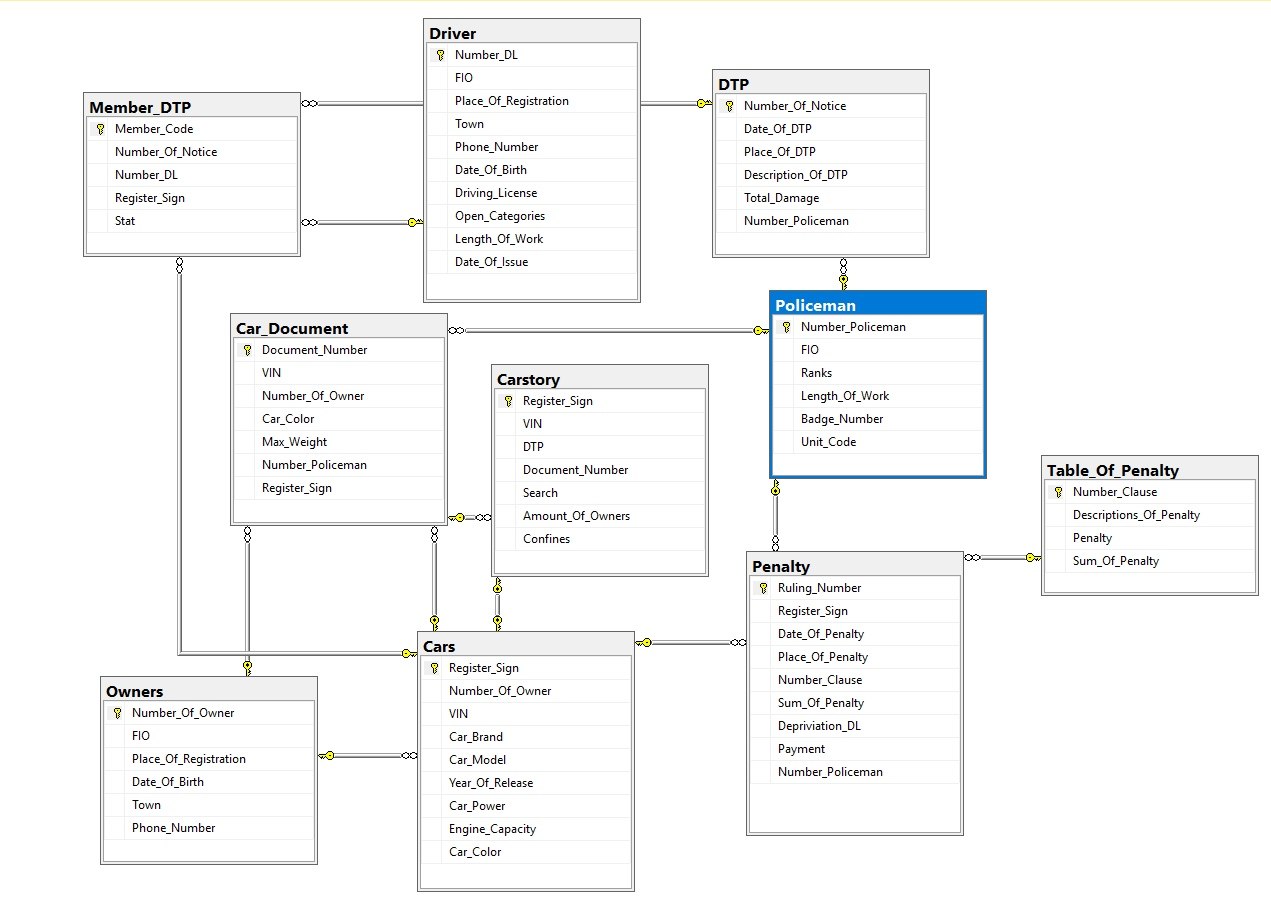


Рис. 4.1 Схема данных

**ГЛАВА 2. ОПИСАНИЕ ФУНКЦИОНИРОВАНИЯ БАЗЫ ДАННЫХ**

* 1. **Назначение и перечень функций базы данных**

Данная база данных предназначена для хранения, добавления, изменения и удаления данных необходимых для работы сотрудников ГИБДД. Базой данных могут пользоваться сотрудники ГИБДД и владельцы ТС.

Для сотрудников ГИБДД были реализованы следующие функции

* Просмотр, добавление и изменение документов ТС
* Просмотр, добавление и изменение документов водителей
* Просмотр и оформление штрафов
* Оформление и просмотр данных о ДТП
* Постановка ТС на учет

Для владельцев ТС были реализованы следующие функции

* Проверка истории ТС
* Проверка документов ТС
* Проверка ВУ
* Проверка штрафов
* Получение сведений о ДТП

Для разработки пользовательского интерфейса использовалась среда Visual Studio 2017, с языком программирования C#. В приложении В представлен программный код всех форм, с помощью которых осуществляется работа с базой данных.

**2.2 Описание работы с базой данных**

Главная форма программы предоставляет возможность начать работу в качестве сотрудник или владельца ТС (рис. 2.1).

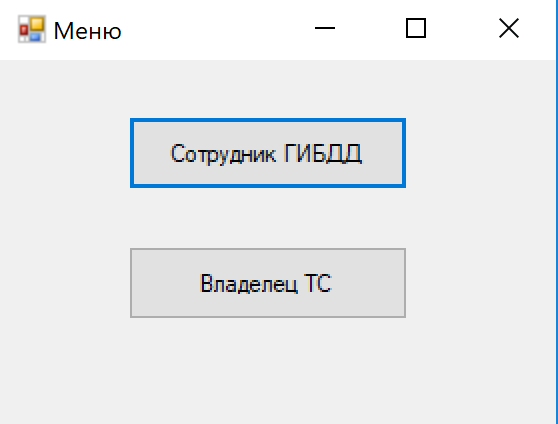


Рис. 2.1 Главная форма

После входа в систему в качестве сотрудника ГИБДД открывается окно «Меню сотрудника ГИБДД», в котором предоставляется выбор тех действий, которые необходимо совершить сотруднику ГИБДД(рис. 2.2)

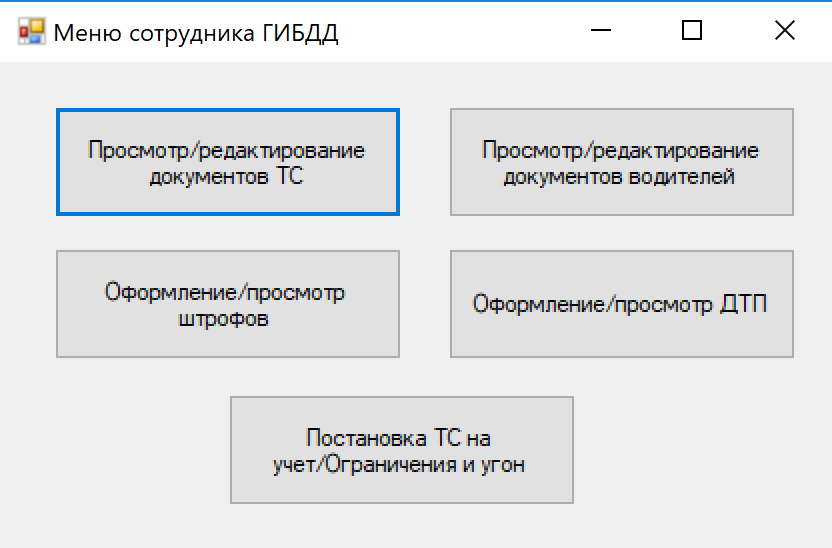


Рис. 2.2 Окно «Меню сотрудника ГИБДД»

Нажав на кнопку «Просмотр/редактирование документов ТС» откроется окно «Работа с документами ТС» , в котором можно добавлять, изменять, удалять документы на ТС (рис. 2.3).



Рис. 2.3 Окно «Работа с документами ТС»

При нажатии кнопки «Просмотр» откроется окно «Просмотр документов ТС», в котором можно совершать поиск необходимых документов введя соответствующую информацию (рис. 2.4).

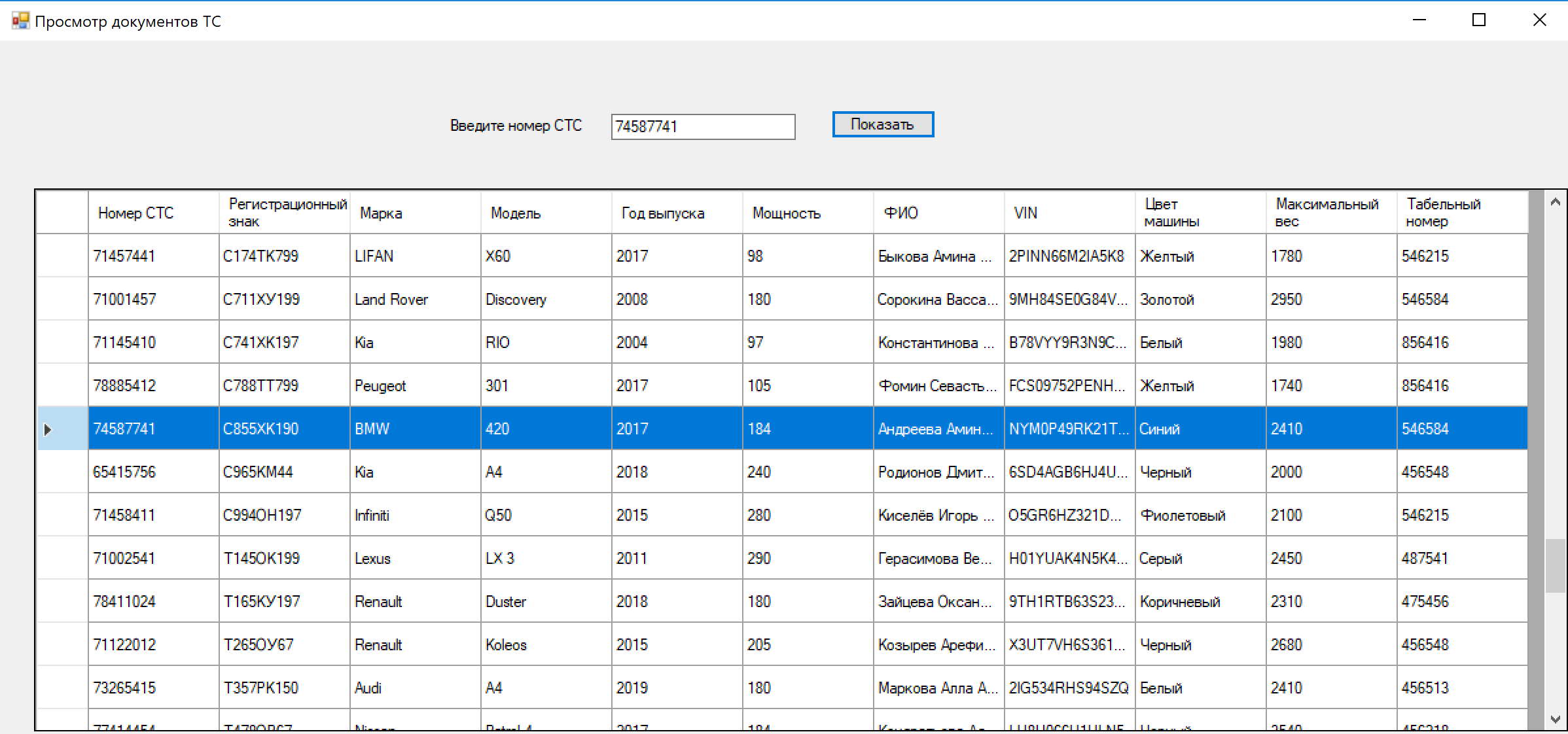


Рис. 2.4 Окно «Просмотр документов ТС»

В меню сотрудника ГИБДД (рис. 2.2) нажав на кнопку «Просмотр/редактирование документов водителей» откроется окно «Работа с документами водителей» , в котором можно добавлять, изменять, удалять документы водителей (рис. 2.5).

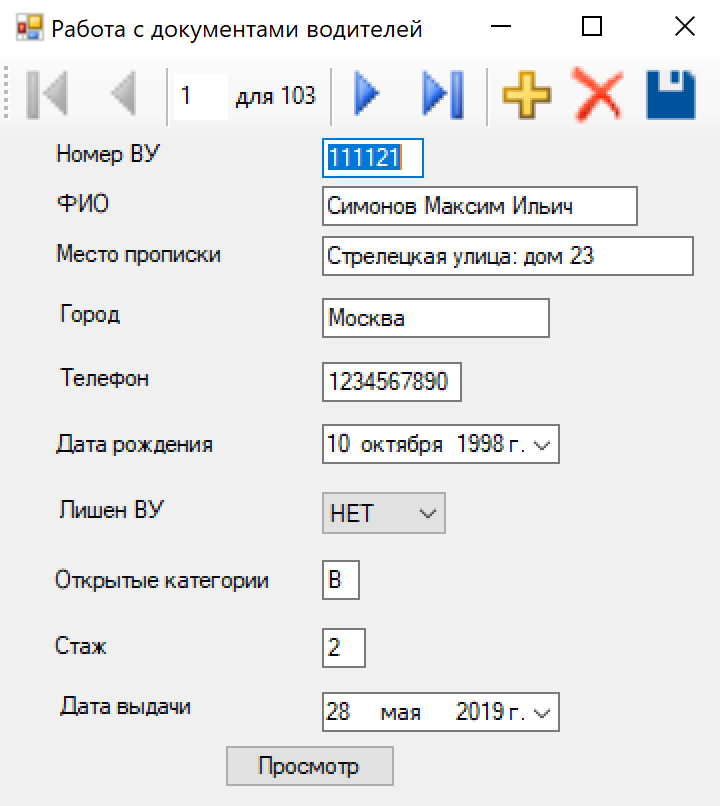


Рис. 2.5 Окно «Работа с документами водителей»

При нажатии кнопки «Просмотр» откроется окно «Просмотр документов водителей», в котором можно совершать поиск необходимых документов введя соответствующую информацию (рис. 2.6).

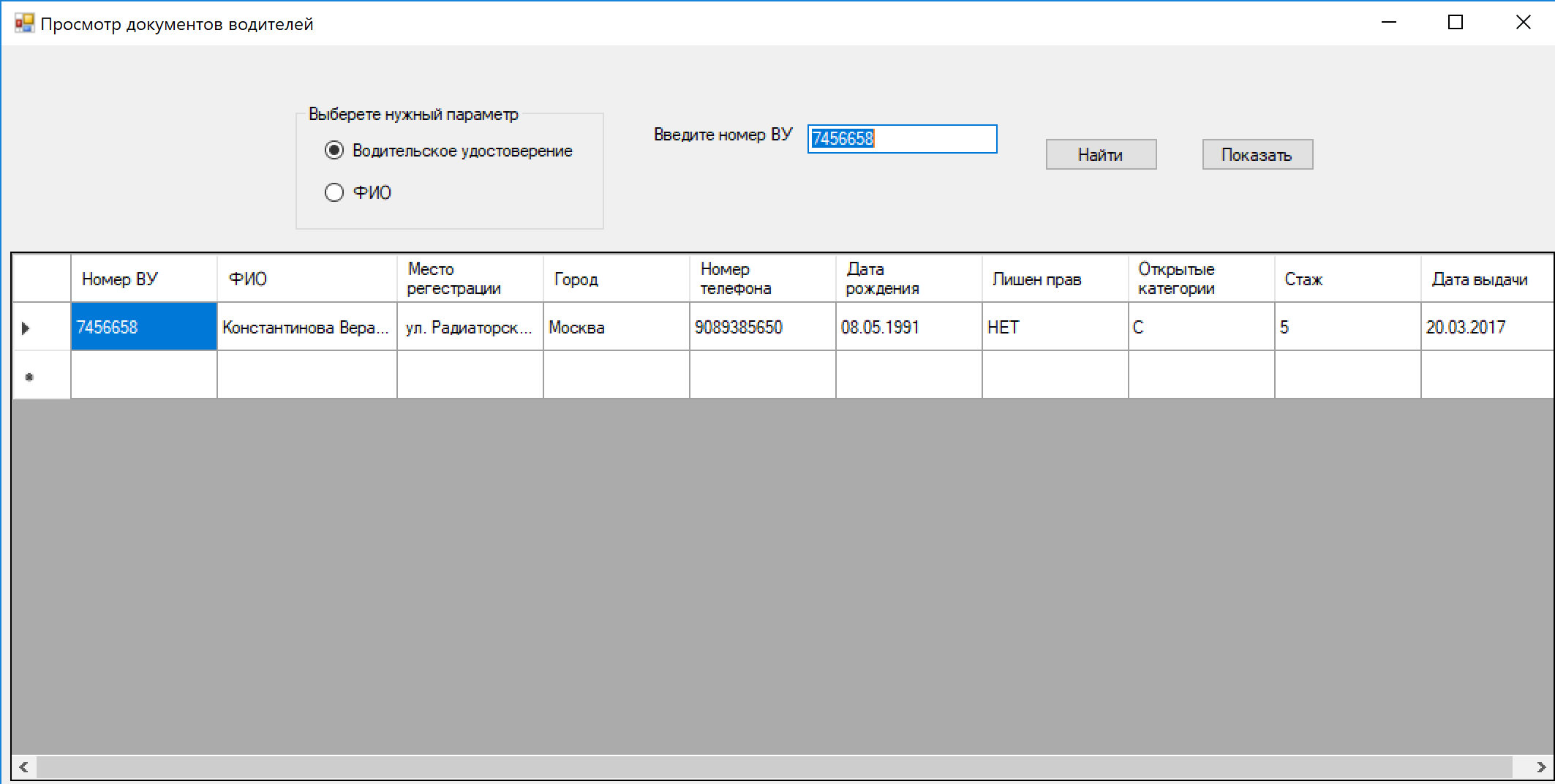


Рис. 2.6 Окно «Просмотр документов водителей»

В меню сотрудника ГИБДД (рис. 2.2) нажав на кнопку «Оформление/просмотр штрафов» откроется окно «Работа со штрафами» , в котором можно добавлять, изменять, удалять штрафы (рис. 2.7).

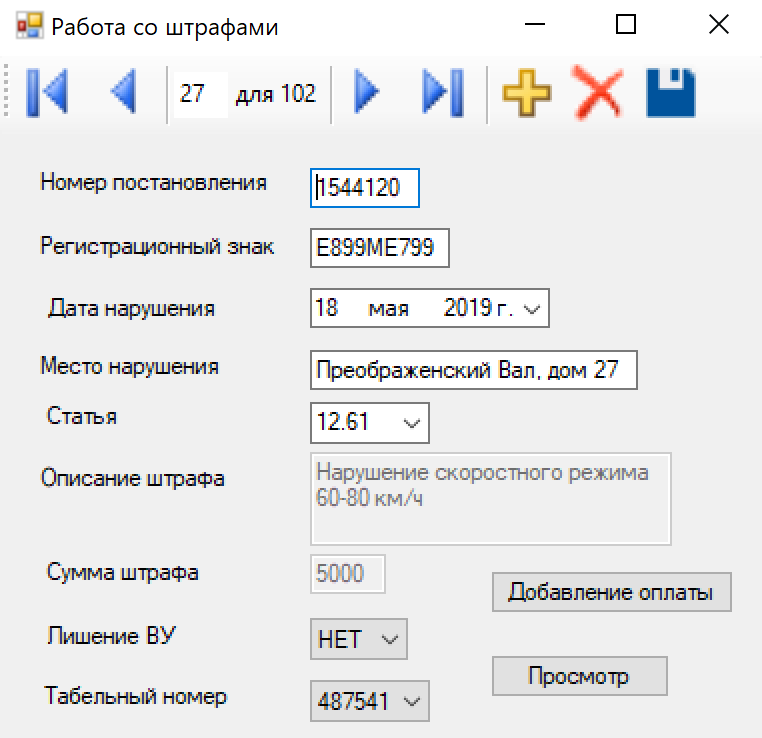


Рис. 2.7 Окно «Работа со штрафами»

При нажатии кнопки «Просмотр» откроется окно «Просмотр штрафов», в котором можно совершать поиск необходимых штрафов введя соответствующую информацию (рис. 2.8).

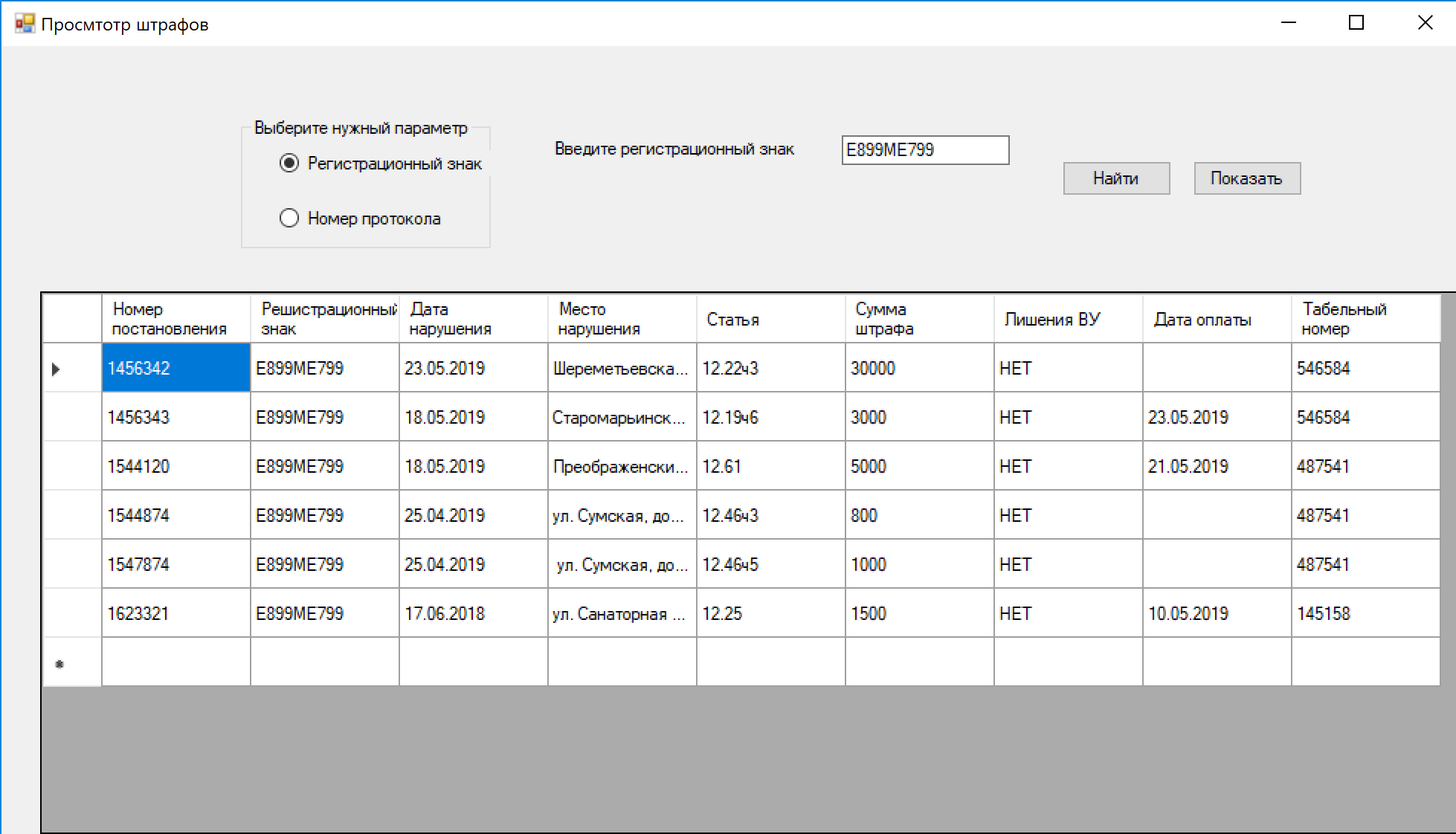


Рис. 2.8 Окно «Просмотр штрафов»

При нажатии кнопки «Добавление оплаты» откроется окно «Добавление оплаты штрафов», в котором можно совершить добавление даты оплаты штрафа по номеру протокола (рис. 2.9).

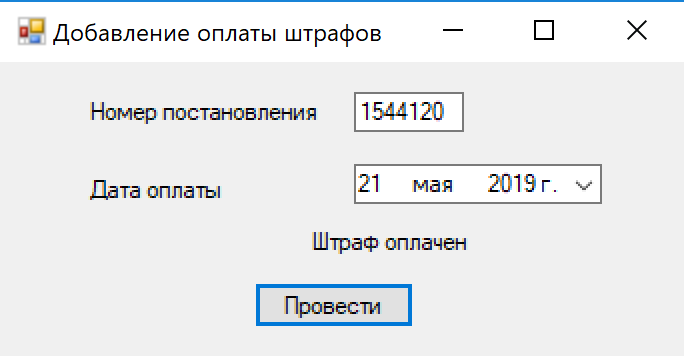


Рис. 2.9 Окно «Добавление оплаты штрафов»

В меню сотрудника ГИБДД (рис. 2.2) нажав на кнопку «Оформление/просмотр ДТП» откроется окно «ДТП» , в котором можно добавлять, изменять, удалять информацию о ДТП (рис. 2.10).

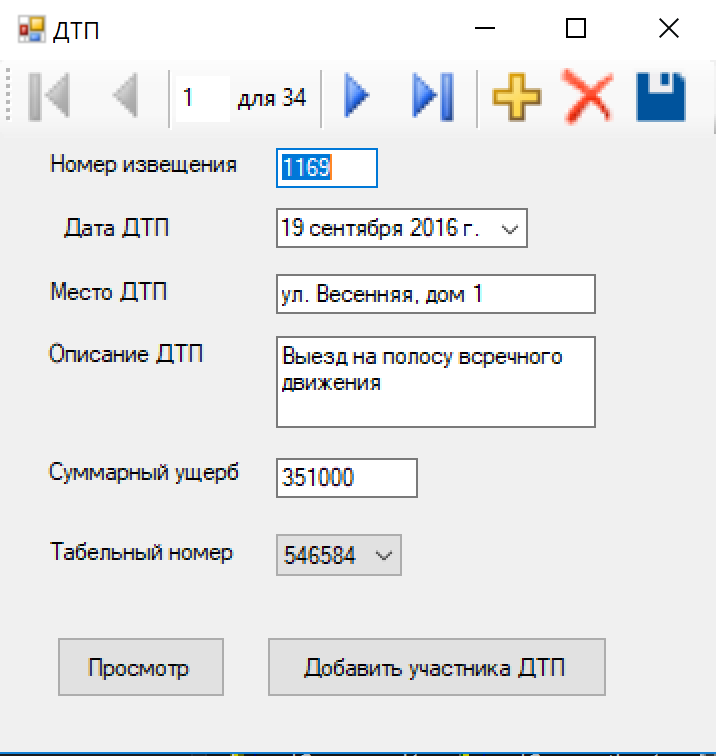


Рис. 2.10 Окно «ДТП»

При нажатии кнопки «Просмотр» откроется окно «Просмотр ДТП», в котором можно совершать поиск информации о ДТП введя соответствующую информацию (рис. 2.11).

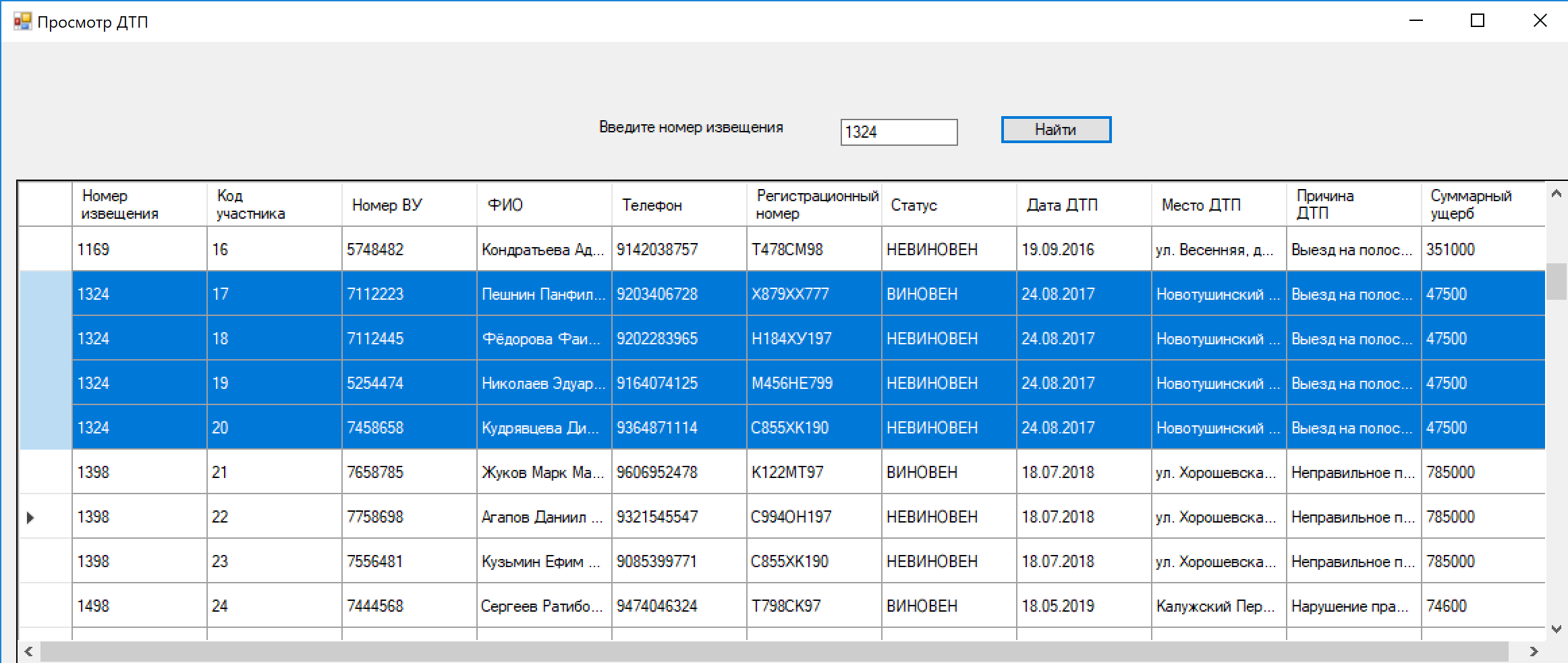


Рис. 2.11 Окно «Просмотр ДТП»

При нажатии кнопки «Добавить участника ДТП» откроется окно «Участник ДТП», в котором можно добавлять участников ДТП с соответствующими данными(рис. 2.12).

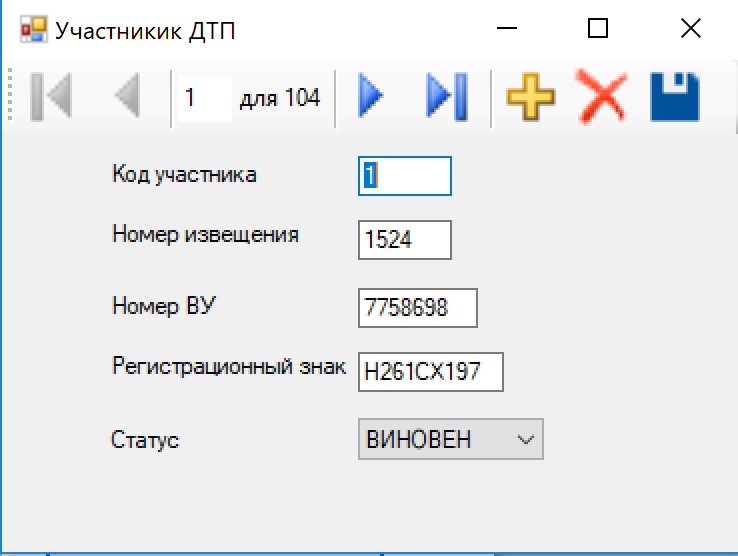


Рис. 2.12 Окно «Участники ДТП»

В меню сотрудника ГИБДД (рис. 2.2) нажав на кнопку «Постановка ТС на учет/Ограничение и угон» откроется окно «Учет ТС» , в котором можно добавлять, изменять, удалять информацию о ТС (рис. 2.13).

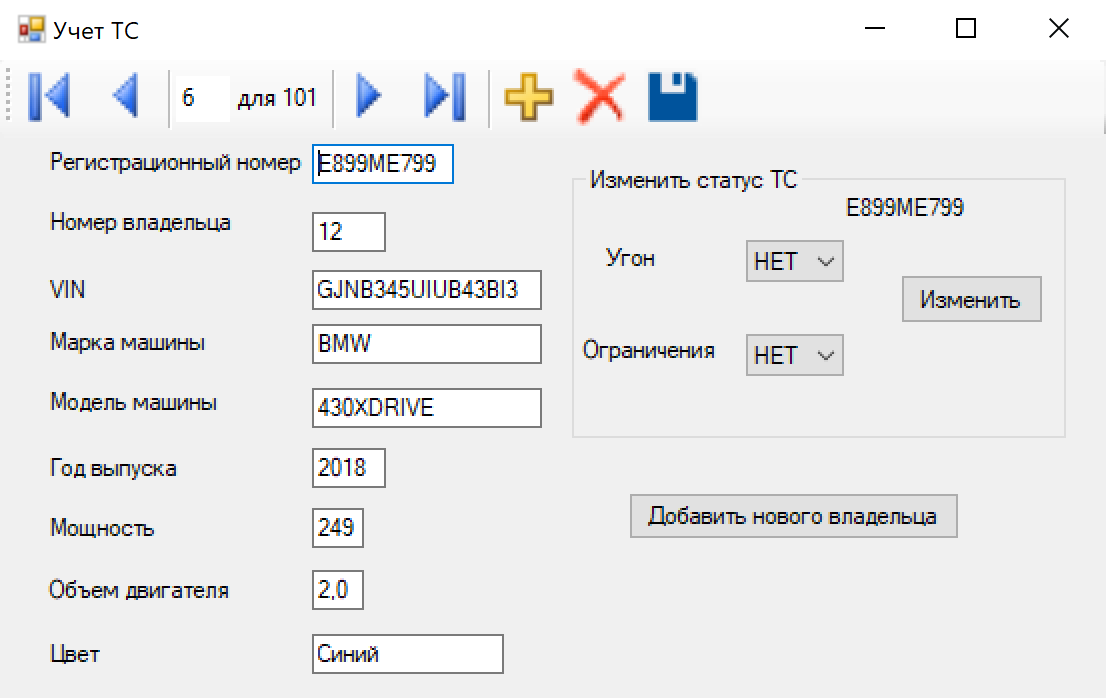


Рис. 2.13 Окно «Учет ТС»

При нажатии кнопки «Добавить нового владельца» откроется окно «Добавить владельца», в котором можно добавлять, изменять, удалять данные о новых владельцах(рис. 2.14).

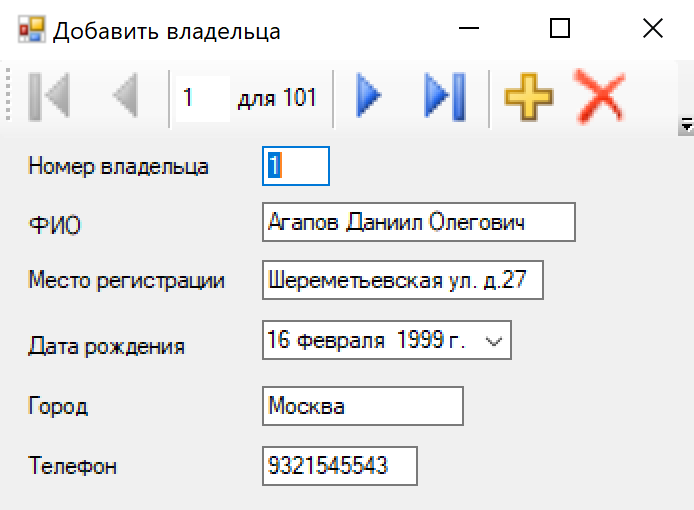


Рис. 2.14 Окно «Добавить владельца»

При входе в систему в качестве владельца ТС открывается окно «Меню владельца ТС», в котором предоставляется выбор тех действий, которые необходимо совершить владельцу ТС (рис. 2.15)

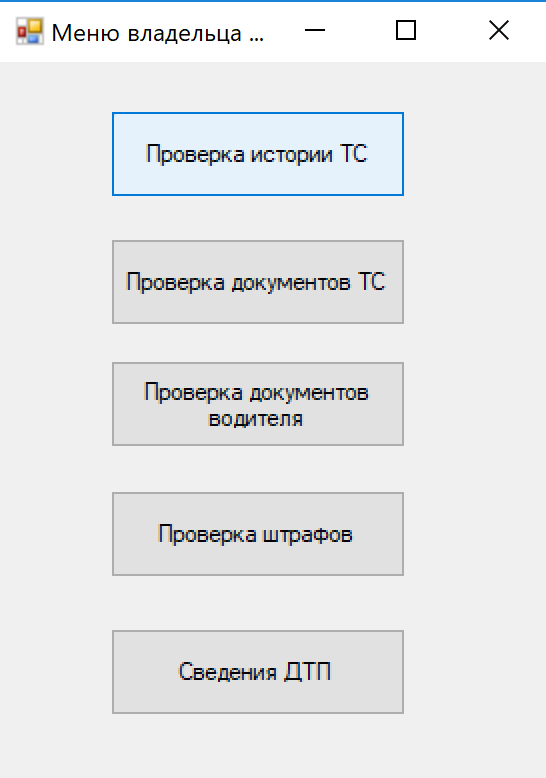


Рис. 2.15 Окно «Меню владельца ТС»

В меню владельца ТС (рис. 2.15) нажав на кнопку «Проверка истории ТС» откроется окно «Автоистория» , в котором введя соответствующие данные и нажав кнопку «Проверить» будет выведена история ТС(рис. 2.16).

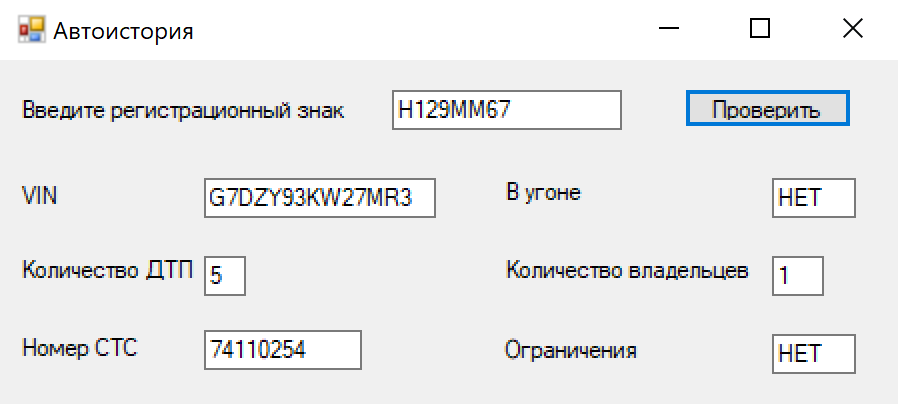


Рис. 2.16 Окно «Автоистория»

В меню владельца ТС (рис. 2.15) нажав на кнопку «Проверка документов ТС» откроется окно «Проверка документов ТС» , в котором введя соответствующие данные и нажав кнопку «Проверить» будет выведена информация, которая позволить удостоверится в подлинности данных в документе(рис. 2.17).

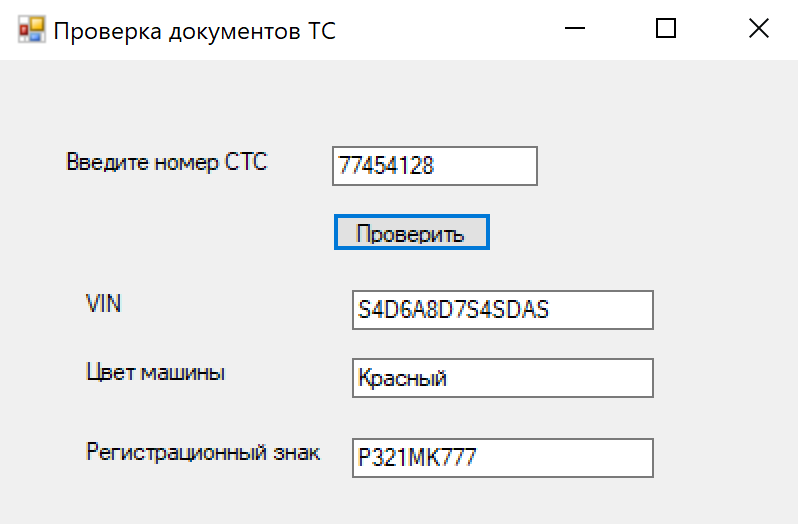


Рис. 2.17 Окно «Проверка документов ТС»

В меню владельца ТС (рис. 2.15) нажав на кнопку «Проверка документов водителя» откроется окно «Проверка водителя» , в котором введя соответствующие данные и нажав кнопку «Проверить» будет выведена информация, которая позволить удостоверится в подлинности данных в ВУ (рис. 2.18).

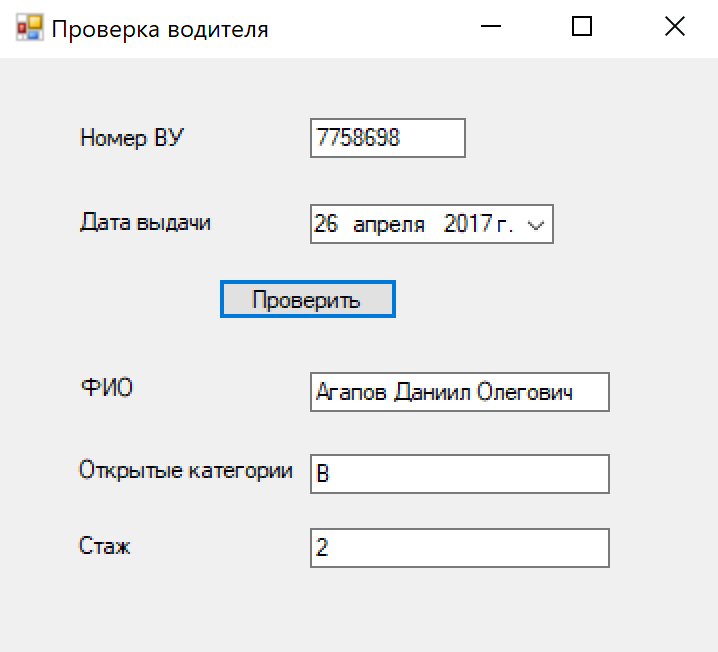


Рис. 2.18 Окно «Проверка водителя»

В меню владельца ТС (рис. 2.15) нажав на кнопку «Проверка штрафов» откроется окно «Поиск штрафов» , в котором введя соответствующие данные и нажав кнопку «Проверить» будет выведена информация, отображающая информацию о штрафах на данное ТС (рис. 2.19).

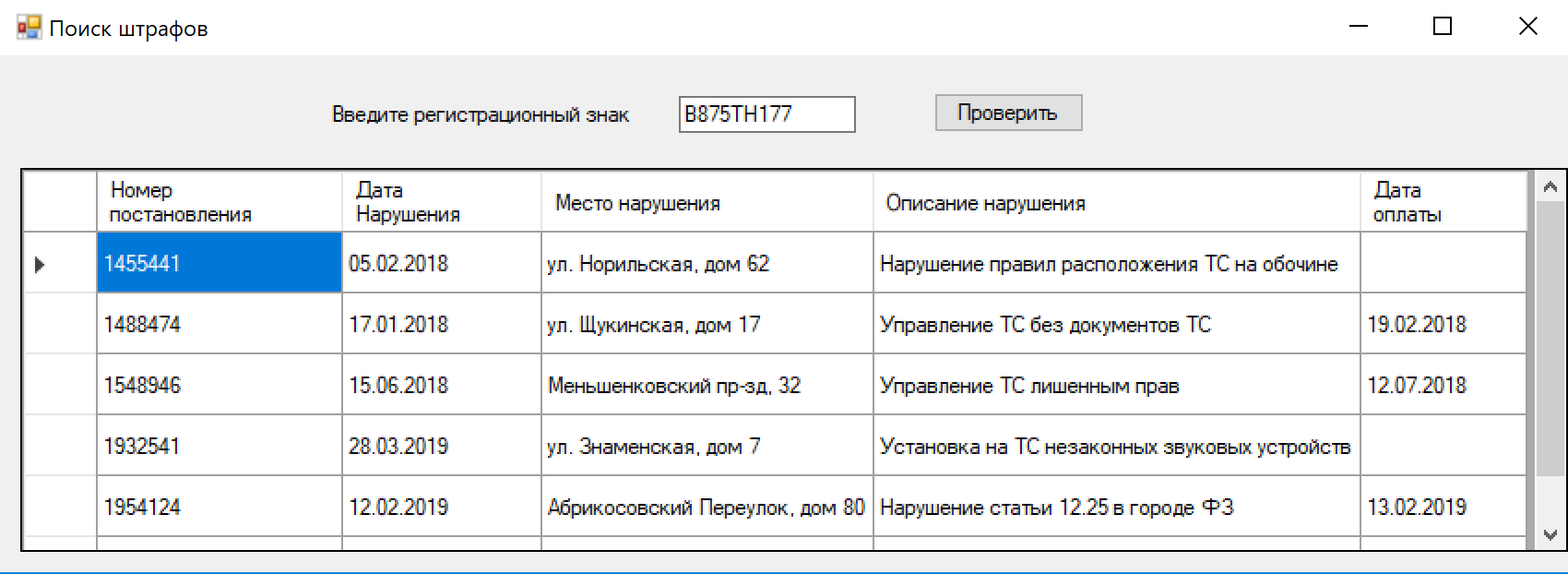


Рис. 2.19 Окно «Поиск штрафов»

В меню владельца ТС (рис. 2.15) нажав на кнопку «Сведения о ДТП» откроется окно «Информация о ДТП» , в котором введя соответствующие данные и нажав кнопку «Проверить» будет выведена информация, отображающая информацию ДТП в которых учувствовало ТС (рис. 2.20).

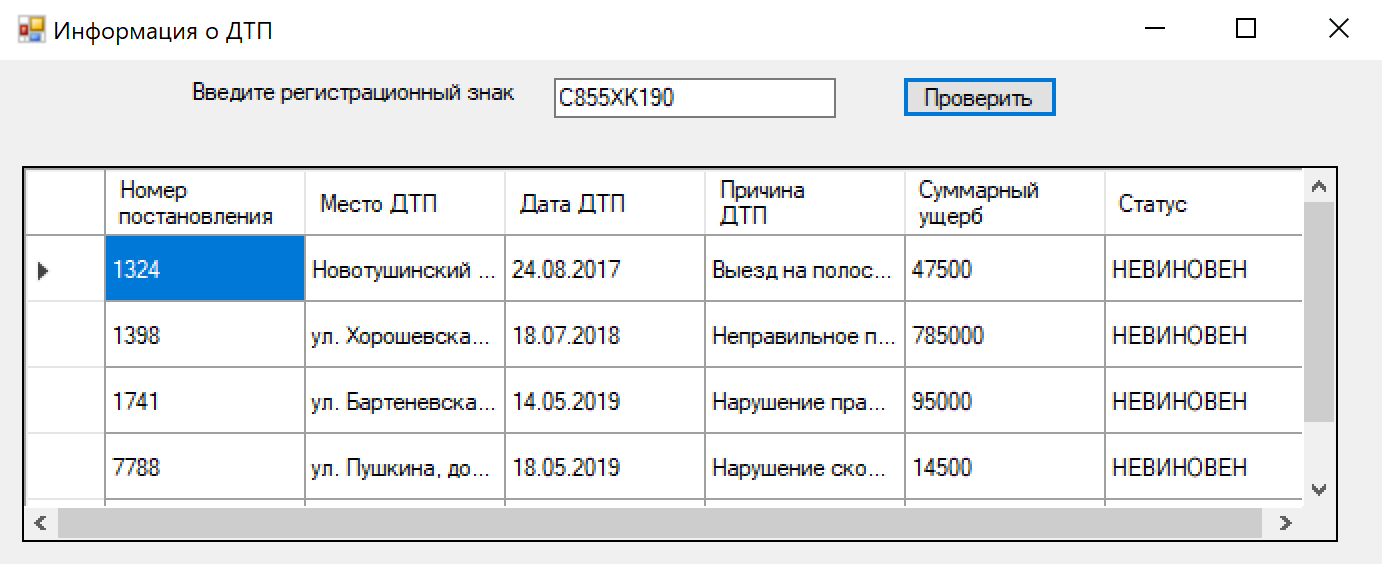


Рис. 2.20 Окно «Информация о ДТП»

**Заключение**

Данная работа посвящена проектированию базы данных ГИБДД.

Данная тема является актуальной как для всей структуры ГИБДД. Сотрудникам ГИБДД база данных упростит поиск и обработку данных, что повлияет на качество работы.

В ходе выполнения работы были получены следующие результаты:

1. Изучена предметная область, проанализированы имеющиеся варианты приложений или сайтов («Автокод» и «ГИБДД»), выделены основные функции будущей системы.
2. На этапе концептуального проектирования были выделены основные действующие лица БД: сотрудник ГИБДД и владелец ТС. Определены их функции, и исходя из этого, составлена диаграмма вариантов использования. В конце этапа концептуального проектирования выделены основные сущности, информация о которых и будет храниться в БД.
3. На этапе логического проектирования на основе информации полученной на этапе концептуального проектирования построены ER-диаграммы. Для всех сущностей определены типы связей и классы принадлежности. На основе полученных диаграмм сформированы отношения, такие как: водитель, штраф, владелец, ТС и другие. Указаны первичные и внешние ключи, а также неключевые атрибуты.
4. На этапе физического проектирования были определены требования к структуре таблиц базы данных. В среде SQL Server Management Studio с помощью SQL-операторов были созданы таблицы и связи между ними. Также были созданы хранимые процедуры для решения различных задач по выборке и изменению данных.
5. На последнем этапе был создан пользовательский интерфейс, позволяющий работать с базой данных. Данная система позволяет облегчить работу сотрудников ГИБДД с базой данных и позволяет просматривать необходимую информацию отдельным пользователям. Данная система выполняет все поставленные требования в курсовом проекте.

**Список использованной литературы**

1. Сайт «Автокод» [Электронный ресурс] – Режим доступа: https://avtokod.mos.ru/, свободный. Дата обращения: 9.11.2018 г.
2. Сайт «ГОСАВТОИНСПЕКЦИЯ» [Электронный ресурс] – Режим доступа: https://xn--90adear.xn--p1ai/, свободный. Дата обращения: 9.11.2018 г.
3. Сайт «Habr» [Электронный ресурс] – Режим доступа: https://habr.com/post/255361/, свободный. Дата обращения: 18.11.2018 г.
4. Сайт «Microsoft» [Электронный ресурс] – Режим доступа: https://msdn.microsoft.com/ru-ru/library/azure/ms174123(v=sql.100).aspx, свободный. Дата обращения: 18.11.2018 г.

**Приложение А**

SQL-операторы создания таблиц

1. SQL-операторы создания таблицы «Водитель»

Create table Driver (Number\_DL integer not null Primary key,

FIO Varchar(50) not null,

Place\_Of\_Registration Varchar(100) not null,

Town Varchar(50) not null,

Phone\_Number integer,

Date\_Of\_Birth Date not null,

Driving\_License varchar(3) check(Driving\_License = 'ДА' or Driving\_License = 'НЕТ') Default 'НЕТ',

Open\_Categories Varchar(10) not null check( Open\_Categories = 'A' or Open\_Categories = 'B' or Open\_Categories = 'C' or Open\_Categories = 'D' or Open\_Categories = 'M') Default 'B',

Length\_Of\_Work integer not null check (Length\_Of\_Work >=0) default 0,

Date\_Of\_Issue date not null)

1. SQL-операторы создания таблицы «Владелец»

Create table Owners (Number\_Of\_Owner integer not null primary key,

FIO varchar(50) not null,

Place\_Of\_Registration varchar(50) not null,

Date\_Of\_Birth date not null,

Town varchar(50) not null,

Phone\_Number integer)

1. SQL-операторы создания таблицы «Штрафы»

Create table Penalty (Ruling\_Number integer not null Primary key,

Register\_Sign nchar(9) not null default('-'),

Date\_Of\_Penalty date not null,

Place\_Of\_Penalty varchar(100) not null,

Number\_Clause varchar(10) not null,

Sum\_Of\_Penalty integer not null,

Depriviation\_DL varchar(3) not null check(Depriviation\_DL = 'ДА' or Depriviation\_DL = 'НЕТ') Default 'НЕТ',

Payment date,

Number\_Policeman integer not null)

Alter table Penalty Add Constraint SKR check (Date\_Of\_Penalty < Payment)

1. SQL-операторы создания таблицы «ТС»

Create table Cars (Register\_Sign nchar(9) not null Primary key,

Number\_Of\_Owner integer not null,

VIN varchar(50) not null,

Car\_Brand varchar(50) not null,

Car\_Model varchar(50) not null,

Year\_Of\_Release char(4) not null,

Car\_Power integer not null,

Engine\_Capacity char(3) not null,

Car\_Color varchar(50) not null)

1. SQL-операторы создания таблицы «Статья»

Create table Table\_Of\_Penalty(Number\_Clause varchar(10) not null Primary key,

Descriptions\_Of\_Penalty Varchar(50) not null,

Penalty varchar(50) not null default('Денежный'),

Sum\_Of\_Penalty integer not null check(Sum\_Of\_Penalty >=0))

1. SQL-операторы создания таблицы «Сотрудник ГИБДД»

Create table Policeman (Number\_Policeman integer not null Primary key,

FIO varchar(50) not null,

Ranks varchar(50) not null,

Length\_Of\_Work integer not null check (Length\_Of\_Work >=0) default 0,

Badge\_Number integer not null,

Unit\_Code integer not null)

1. SQL-операторы создания таблицы «ДТП»

Create table DTP (Number\_Of\_Notice integer not null Primary key,

Date\_Of\_DTP date not null,

Place\_Of\_DTP varchar(50) not null,

Description\_Of\_DTP varchar(100) not null,

Total\_Damage integer not null,

Number\_Policeman integer not null)

1. SQL-операторы создания таблицы «Документы ТС»

Create table Car\_Document (Document\_Number integer not null Primary key,

VIN varchar(50) not null,

Number\_Of\_Owner integer not null,

Car\_Color varchar(50) not null,

Max\_Weight integer not null,

Number\_Policeman integer not null,

Register\_Sign nchar(9) not null)

1. SQL-операторы создания таблицы «Автоистория»

Create table Carstory (Register\_Sign nchar(9) not null Primary key,

VIN varchar(50) not null,

DTP integer not null check(DTP >=0) default 0,

Document\_Number integer not null,

Search varchar (3) not null check (Search = 'ДА' or Search = 'НЕТ') default 'НЕТ',

Amount\_Of\_Owners integer not null check (Amount\_Of\_Owners > 0) default 1,

Confines varchar(50) not null default 'НЕТ')

1. SQL-операторы создания таблицы «Участник ДТП»

Create table Member\_DTP (Member\_Code integer not null Primary key,

Number\_Of\_Notice integer not null references DTP(Number\_Of\_Notice),

Number\_DL integer not null references Driver (Number\_DL),

Register\_Sign nchar(9) not null references Cars (Register\_Sign),

Stat varchar(9) not null check(Stat = 'НЕВИНОВЕН' or Stat= 'ВИНОВЕН') Default ('ВИНОВЕН'))

**Приложение Б**

**Пример заполненных таблиц**



Рис Б.1 Пример заполнения таблицы «Водитель»

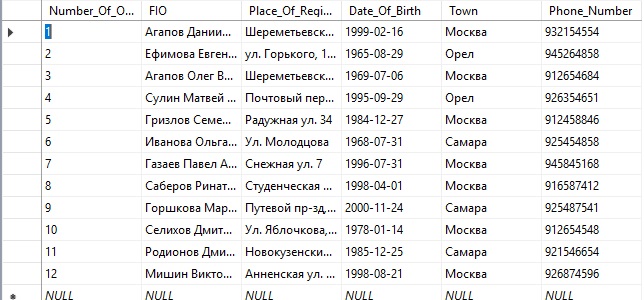


Рис Б.2 Пример заполнения таблицы «Владелец ТС»



Рис Б.3 Пример заполнения таблицы «Штраф»



Рис Б.4 Пример заполнения таблицы «ТС»

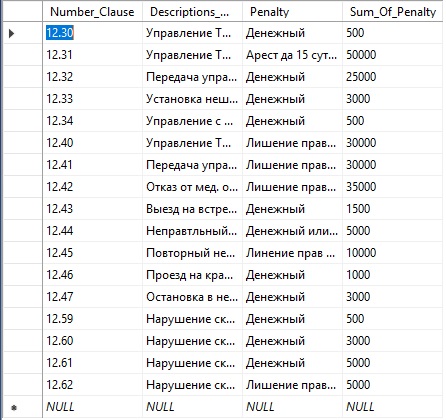


Рис Б.5 Пример заполнения таблицы «Статья»



Рис Б.6 Пример заполнения таблицы «Сотрудник ГИБДД»

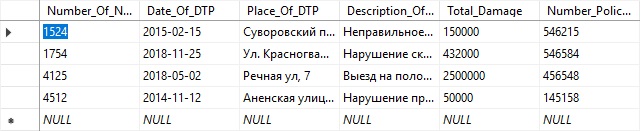


Рис Б.7 Пример заполнения таблицы «ДТП»

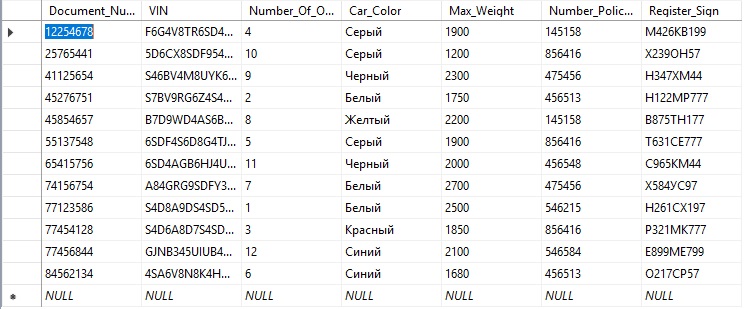


Рис Б.8 Пример заполнения таблицы «Документ ТС»

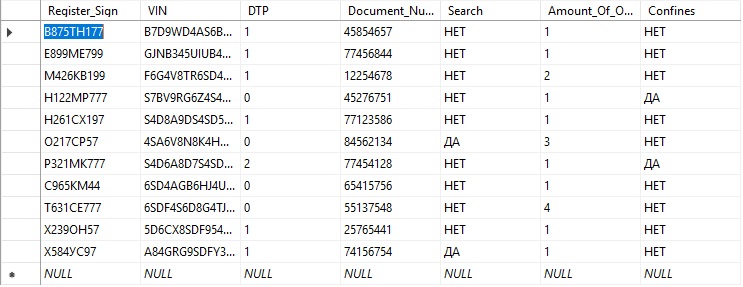


Рис Б.9 Пример заполнения таблицы «Автоистория»

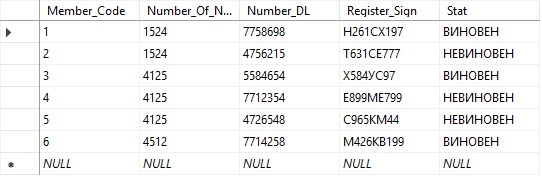


Рис Б.10 Пример заполнения таблицы «Участник ДТП»

**Приложение В**

**Код пользовательского интерфейса**

1. Код окна «Меню сотрудника ГИБДД»

public partial class Policeman : Form

{

public Policeman()

{

InitializeComponent();

}

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

{

Form form10 = new CarDocP();

form10.Show();

}

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

{

Form form12 = new DriverP();

form12.Show();

}

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

{

Form form13 = new PenaltyP();

form13.Show();

}

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

{

Form form14 = new DTPP();

form14.Show();

}

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

{

Form form15 = new CarP();

form15.Show();

}

private void Policeman\_Load(object sender, EventArgs e)

{

}

}

1. Код окна «Работа с документами ТС»

public partial class CarDocP : Form

{

public CarDocP()

{

InitializeComponent();

}

private void car\_DocumentBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

string CTC = document\_NumberTextBox.Text;

var validator = new Regex("([0-9]{6,9})$");

var validateResult = validator.Match(CTC);

if (validateResult.Success)

{

string VIN = vINTextBox.Text;

var validator2 = new Regex("([0-9A-Z]{14})$");

var validateResult2 = validator2.Match(VIN);

if (validateResult2.Success)

{

string weight = max\_WeightTextBox.Text;

var validator4 = new Regex("([0-9]{3,5})$");

var validateResult4 = validator4.Match(weight);

if (validateResult4.Success)

{

var validator3 = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult3 = validator3.Match(register\_SignTextBox.Text);

if (validateResult3.Success)

{

this.Validate();

this.car\_DocumentBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

MessageBox.Show("Save completed");

}

else

{

MessageBox.Show("Проверьте корректность госномера");

}

}

else

{

MessageBox.Show("Проверьте правильность введённой массы");

}

}

else

{

MessageBox.Show("Проверьте корректность ввода VIN");

}

}

else

{

MessageBox.Show("Проверьте корректность ввода СТС");

}

}

private void CarDocP\_Load(object sender, EventArgs e)

{

this.policemanTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Policeman);

this.car\_DocumentTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Car\_Document);

}

private void button8\_Click(object sender, EventArgs e)

{

Form form1 = new CarDocPView();

form1.Show();

}

private void number\_Of\_OwnerTextBox\_TextChanged(object sender, EventArgs e)

{

if (number\_Of\_OwnerTextBox.Text != "")

{

sqlCommand1.Parameters["@idOwner"].Value = Convert.ToInt32(number\_Of\_OwnerTextBox.Text);

sqlConnection1.Open();

sqlCommand1.ExecuteNonQuery();

sqlConnection1.Close();

textBox2.Text = Convert.ToString(sqlCommand1.Parameters["@fio"].Value);

}

textBox2.Visible = true;

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

textBox2.Visible = false;

}

1. Код окна «Просмотр документов ТС»

public partial class CarDocPView : Form

{

public CarDocPView()

{

InitializeComponent();

}

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

{

string gosznak = textBox1.Text;

var validator = new Regex("([0-9]{8})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

for (int i = 0; i < dataGridView1.RowCount; i++)

{

dataGridView1.Rows[i].Selected = false;

for (int j = 0; j < dataGridView1.ColumnCount; j++)

if (dataGridView1.Rows[i].Cells[j].Value != null)

if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(textBox1.Text))

{

dataGridView1.Rows[i].Selected = true;

break;

}

}

MessageBox.Show("Нужные данные выделены");

}

else

{

MessageBox.Show("Проверьте корректность ввода СТС");

}

}

private void CarDocPView\_Load\_1(object sender, EventArgs e)

{

this.car\_DocumentTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Car\_Document);

sqlCommand1.Parameters["@res"].Value = 0;

sqlConnection1.Open();

var temp = new DataTable();

temp.Load(sqlCommand1.ExecuteReader());

dataGridView1.DataSource = temp;

sqlConnection1.Close();

}

1. Код окна «Работа с документами водителей»

public partial class DriverP : Form

{

public DriverP()

{

InitializeComponent();

}

private void driverBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

int age = DateTime.Now.Year - date\_Of\_BirthDateTimePicker.Value.Year - 18;

if (age > Convert.ToInt32(length\_Of\_WorkTextBox.Text))

{

string BY = number\_DLTextBox.Text;

var validator = new Regex("([0-9]{6,9})$");

var validateResult = validator.Match(BY);

if (validateResult.Success)

{

string phone = phone\_NumberTextBox.Text;

var validator2 = new Regex("([0-9]{10})$");

var validateResult2 = validator2.Match(phone);

if (validateResult2.Success)

{

string cat = open\_CategoriesTextBox.Text;

var validator3 = new Regex("([A,B,C,D,M]{1})$");

var validateResult3 = validator3.Match(cat);

if (validateResult3.Success)

{

this.Validate();

this.driverBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

MessageBox.Show("Save completed");

}

else

{

MessageBox.Show("Некорректно введены категории");

}

}

else

{

MessageBox.Show("Некорректно введен телефон");

}

}

else

{

MessageBox.Show("Некорректный ввод ВУ");

}

}

else

{

MessageBox.Show("Некорректный ввод стажа");

}

}

private void DriverP\_Load(object sender, EventArgs e)

{

this.driverTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Driver);

DateTime dt = DateTime.Now;

dt = dt.AddYears(-10);

string curDate = dt.ToShortDateString();

date\_Of\_IssueDateTimePicker.MinDate = Convert.ToDateTime(curDate);

date\_Of\_IssueDateTimePicker.MaxDate = DateTime.Now;

DateTime dt2 = DateTime.Now;

dt2 = dt2.AddYears(-18);

string curDate2 = dt2.ToShortDateString();

date\_Of\_BirthDateTimePicker.MaxDate = Convert.ToDateTime(curDate2);

}

private void button8\_Click(object sender, EventArgs e)

{

Form form1 = new DriverPView();

form1.Show();

}

private void bindingNavigatorAddNewItem\_Click(object sender, EventArgs e)

{

length\_Of\_WorkTextBox.Text = "0";

}

}

1. Код окна «Просмотр документов водителей»

public partial class DriverPView : Form

{

public DriverPView()

{

InitializeComponent();

}

private void DriverPView\_Load(object sender, EventArgs e)

{

this.driverTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Driver);

}

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

{

if (radioButton1.Checked == true)

{

string BY = textBox1.Text;

var validator = new Regex("([0-9]{6,9})$");

var validateResult = validator.Match(BY);

if (validateResult.Success)

{

driverBindingSource.Filter = "Number\_DL =" + Convert.ToInt32(textBox1.Text);

}

else

{

MessageBox.Show("Некорректно введен номер ВУ");

}

}

else if (radioButton2.Checked == true)

{

driverBindingSource.Filter = "FIO Like'" + textBox2.Text + "%'";

}

if (driverDataGridView.Rows.Count < 2)

{

MessageBox.Show("Такого водителя нет!");

}

}

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

{

driverBindingSource.Filter = "";

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = false;

textBox2.Visible = false;

label1.Visible = true;

textBox1.Visible = true;

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

textBox2.Visible = true;

label1.Visible = false;

textBox1.Visible = false;

}

}

1. Код окна «Работа со штрафами»

public partial class PenaltyP : Form

{

public PenaltyP()

{

InitializeComponent();

}

private void penaltyBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

depriviation\_DLComboBox.Text = "НЕТ";

string number = ruling\_NumberTextBox.Text;

var validator = new Regex("([0-9]{7,8})$");

var validateResult = validator.Match(number);

if (validateResult.Success)

{

var validator3 = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult3 = validator3.Match(register\_SignTextBox.Text);

if (validateResult3.Success)

{

this.Validate();

this.penaltyBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

MessageBox.Show("Save completed");

}

else

{

MessageBox.Show("Проверьте ГосНомер");

}

}

else

{

MessageBox.Show("Проверьте протокол");

}

}

private void PenaltyP\_Load(object sender, EventArgs e)

{

this.table\_Of\_PenaltyTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Table\_Of\_Penalty);

this.policemanTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Policeman);

this.penaltyTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Penalty);

date\_Of\_PenaltyDateTimePicker.MaxDate = DateTime.Now;

}

private void button8\_Click(object sender, EventArgs e)

{

Form form12 = new PenaltyPView();

form12.Show();

}

private void button9\_Click(object sender, EventArgs e)

{

Form form12 = new PaymentPenalty();

form12.Show();

}

private void number\_ClauseComboBox\_TextChanged(object sender, EventArgs e)

{

if (number\_ClauseComboBox.Text != "")

{

sqlCommand1.Parameters["@number"].Value = Convert.ToString(number\_ClauseComboBox.Text);

sqlConnection1.Open();

sqlCommand1.ExecuteNonQuery();

sqlConnection1.Close();

sum\_Of\_PenaltyTextBox.Text = Convert.ToString(sqlCommand1.Parameters["@sum\_penalty"].Value);

textBox1.Text = Convert.ToString(sqlCommand1.Parameters["@description"].Value);

}

}

}

1. Код окна «Просмотр штрафов»

public partial class PenaltyPView : Form

{

public PenaltyPView()

{

InitializeComponent();

}

private void penaltyBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

this.Validate();

this.penaltyBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

}

private void PenaltyPView\_Load(object sender, EventArgs e)

{

this.penaltyTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Penalty);

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = false;

textBox2.Visible = false;

label1.Visible = true;

textBox1.Visible = true;

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

label2.Visible = true;

textBox2.Visible = true;

label1.Visible = false;

textBox1.Visible = false;

}

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

{

if (radioButton1.Checked == true)

{

string gosznak = textBox1.Text;

var validator = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

penaltyBindingSource.Filter = "Register\_Sign Like'" + textBox1.Text + "%'";

}

else

{

MessageBox.Show("Некорректно введен госзнак");

}

}

else if (radioButton2.Checked == true)

{

string number = textBox2.Text;

var validator2 = new Regex("([0-9]{7,8})$");

var validateResult2 = validator2.Match(number);

if (validateResult2.Success)

{

penaltyBindingSource.Filter = "Ruling\_Number =" + Convert.ToInt32(textBox2.Text);

}

else

{

MessageBox.Show("Некорректно введен номер постановления");

}

}

}

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

{

penaltyBindingSource.Filter = "";

}

}

1. Код окна «Добавление оплаты штрафов»

public partial class PaymentPenalty : Form

{

public PaymentPenalty()

{

InitializeComponent();

}

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

{

label3.Text = "";

string number = textBox1.Text;

var validator = new Regex("([0-9]{7,8})$");

var validateResult = validator.Match(number);

if (validateResult.Success)

{

sqlCommand1.Parameters["@ruling\_number"].Value = Convert.ToString(textBox1.Text);

sqlCommand1.Parameters["@payment\_date"].Value = dateTimePicker1.Value;

sqlConnection1.Open();

sqlCommand1.ExecuteNonQuery();

sqlConnection1.Close();

var result = sqlCommand1.Parameters["@res"].Value;

label3.Text = Convert.ToString(result);

if (label3.Text != "")

{

label3.Text = "Штраф оплачен";

}

}

else

{

MessageBox.Show("Проверьте номер протокола");

}

if (label3.Text == "")

{

MessageBox.Show("Такой протокол не найден");

}

}

private void PaymentPenalty\_Load(object sender, EventArgs e)

{

dateTimePicker1.MaxDate = DateTime.Now;

DateTime dt2 = DateTime.Now;

dt2 = dt2.AddYears(-1);

string curDate2 = dt2.ToShortDateString();

dateTimePicker1.MinDate = Convert.ToDateTime(curDate2);

}

1. Код окна «ДТП»

public DTPP()

{

InitializeComponent();

}

private void dTPBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

string izveshenie = number\_Of\_NoticeTextBox.Text;

var validator = new Regex("([0-9]{4,8})$");

var validateResult = validator.Match(izveshenie);

if (validateResult.Success)

{

string total = total\_DamageTextBox.Text;

var validator2 = new Regex("([0-9]{3,10})$");

var validateResult2 = validator2.Match(total);

if (validateResult2.Success)

{

this.Validate();

this.dTPBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

MessageBox.Show("Save completed");

}

else

{

MessageBox.Show("Проверьте корректность суммарного ущерба");

}

}

else

{

MessageBox.Show("Проверьте корректность номера извещения");

}

}

private void DTPP\_Load(object sender, EventArgs e)

{

this.policemanTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Policeman);

this.dTPTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.DTP);

date\_Of\_DTPDateTimePicker.MaxDate = DateTime.Now;

}

private void button8\_Click(object sender, EventArgs e)

{

Form form13 = new DTPPView();

form13.Show();

}

private void button9\_Click(object sender, EventArgs e)

{

Form form14 = new MembersDTPP();

form14.Show();

}

1. Код окна «Просмотр ДТП»

public DTPPView()

{

InitializeComponent();

}

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

{

string izveshenie = textBox1.Text;

var validator = new Regex("([0-9]{4,8})$");

var validateResult = validator.Match(izveshenie);

if (validateResult.Success)

{

for (int i = 0; i < dataGridView1.RowCount; i++)

{

dataGridView1.Rows[i].Selected = false;

for (int j = 0; j < dataGridView1.ColumnCount; j++)

if (dataGridView1.Rows[i].Cells[j].Value != null)

if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(textBox1.Text))

{

dataGridView1.Rows[i].Selected = true;

break;

}

}

MessageBox.Show("Нужные данные выделены");

}

else

{

MessageBox.Show("Некорректно введено номер извещения");

}

}

private void DTPPView\_Load(object sender, EventArgs e)

{

this.dTPTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.DTP);

sqlCommand1.Parameters["@res"].Value = 0;

sqlConnection1.Open();

var temp = new DataTable();

temp.Load(sqlCommand1.ExecuteReader());

dataGridView1.DataSource = temp;

sqlConnection1.Close();

}

1. Код окна «Участники ДТП»

private void member\_DTPBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

string member = number\_Of\_NoticeTextBox.Text;

var validator = new Regex("([0-9]{1,8})$");

var validateResult = validator.Match(member);

if (validateResult.Success)

{

string izveshenie = number\_Of\_NoticeTextBox.Text;

var validator2 = new Regex("([0-9]{4,8})$");

var validateResult2 = validator2.Match(izveshenie);

if (validateResult2.Success)

{

string BY = number\_DLTextBox.Text;

var validator3 = new Regex("([0-9]{6,9})$");

var validateResult3 = validator3.Match(BY);

if (validateResult3.Success)

{

var validator4 = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult4 = validator4.Match(register\_SignTextBox.Text);

if (validateResult4.Success)

{

this.Validate();

this.member\_DTPBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

MessageBox.Show("Save compeleted");

}

else

{

MessageBox.Show("Некорректный ГосНомер");

}

}

else

{

MessageBox.Show("Некорректное ВУ");

}

}

else

{

MessageBox.Show("Некорректное извещение");

}

}

else

{

MessageBox.Show("Некорректный код участника");

}

}

private void MembersDTPP\_Load(object sender, EventArgs e)

{

this.member\_DTPTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Member\_DTP);

}

1. Код окна «Учет ТС»

public partial class CarP : Form

{

public CarP()

{

InitializeComponent();

}

private void carsBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

string gosznak = register\_SignTextBox.Text;

var validator = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

string VIN = vINTextBox.Text;

var validator2 = new Regex("([0-9A-Z]{14})$");

var validateResult2 = validator2.Match(VIN);

if (validateResult2.Success)

{

string owner = number\_Of\_OwnerTextBox.Text;

var validator3 = new Regex("([0-9]{1,8})$");

var validateResult3 = validator3.Match(owner);

if (validateResult3.Success)

{

string power = car\_PowerTextBox.Text;

var validator4 = new Regex("([0-9]{2,4})$");

var validateResult4 = validator4.Match(power);

if (validateResult4.Success)

{

int year = Convert.ToInt32(year\_Of\_ReleaseTextBox.Text);

int nowyear = Convert.ToInt32(DateTime.Now.Year + 1);

if (year > 1900)

{

if (year < nowyear)

{

this.Validate();

this.carsBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

sqlCommand2.Parameters["@register"].Value = register\_SignTextBox.Text;

sqlCommand2.Parameters["@vin"].Value = vINTextBox.Text;

sqlCommand2.Parameters["@dtp"].Value = 0;

sqlCommand2.Parameters["@search"].Value = comboBox1.Text;

sqlCommand2.Parameters["@owners"].Value = 1;

sqlCommand2.Parameters["@con"].Value = comboBox2.Text;

sqlConnection1.Open();

sqlCommand2.ExecuteNonQuery();

sqlConnection1.Close();

MessageBox.Show("Save completed");

}

else

{

MessageBox.Show("Год больще чем текущий");

}

}

else

{

MessageBox.Show("Слишком старая машины");

}

}

else

{

MessageBox.Show("Некорректная мощность ТС");

}

}

else

{

MessageBox.Show("Некорректный номер владельца");

}

}

else

{

MessageBox.Show("Некорректный VIN");

}

}

else

{

MessageBox.Show("Некорректный госномер");

}

}

private void CarP\_Load(object sender, EventArgs e)

{

this.carsTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Cars);

label3.Text = register\_SignTextBox.Text;

}

private void button8\_Click(object sender, EventArgs e)

{

Form form15 = new OwnerP();

form15.Show();

}

private void register\_SignTextBox\_TextChanged(object sender, EventArgs e)

{

label3.Text = register\_SignTextBox.Text;

sqlCommand1.Parameters["@register"].Value = register\_SignTextBox.Text;

sqlCommand1.Parameters["@search"].Value = comboBox1.Text;

sqlCommand1.Parameters["@con"].Value = comboBox2.Text;

sqlConnection1.Open();

sqlCommand1.ExecuteNonQuery();

sqlConnection1.Close();

comboBox1.Text = Convert.ToString(sqlCommand1.Parameters["@search2"].Value);

comboBox2.Text = Convert.ToString(sqlCommand1.Parameters["@con2"].Value);

}

private void button9\_Click(object sender, EventArgs e)

{

sqlCommand1.Parameters["@register"].Value = register\_SignTextBox.Text;

sqlCommand1.Parameters["@search"].Value = comboBox1.Text;

sqlCommand1.Parameters["@con"].Value = comboBox2.Text;

sqlConnection1.Open();

sqlCommand1.ExecuteNonQuery();

sqlConnection1.Close();

MessageBox.Show("Данные изменены");

}

}

1. Код окна «Добавить владельца»

public partial class OwnerP : Form

{

public OwnerP()

{

InitializeComponent();

}

private void ownersBindingNavigatorSaveItem\_Click(object sender, EventArgs e)

{

{

string owner = number\_Of\_OwnerTextBox.Text;

var validator3 = new Regex("([0-9]{1,8})$");

var validateResult3 = validator3.Match(owner);

if (validateResult3.Success)

{

string phone = phone\_NumberTextBox.Text;

var validator2 = new Regex("([0-9]{10})$");

var validateResult2 = validator2.Match(phone);

if (validateResult2.Success)

{

this.Validate();

this.ownersBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.\_Курсовая\_2\_0DataSet);

MessageBox.Show("Save completed");

}

else

{

MessageBox.Show("Некорректно введен телефон");

}

}

else

{

MessageBox.Show("Некорректный номер владельца");

}

}

}

private void OwnerP\_Load(object sender, EventArgs e)

{

this.ownersTableAdapter.Fill(this.\_Курсовая\_2\_0DataSet.Owners);

DateTime dt = DateTime.Now;

dt = dt.AddDays(-7);

string curDate = dt.ToShortDateString();

date\_Of\_BirthDateTimePicker.MaxDate = Convert.ToDateTime(curDate);

DateTime dt2 = DateTime.Now;

dt2 = dt2.AddYears(-119);

string curDate2 = dt2.ToShortDateString();

date\_Of\_BirthDateTimePicker.MinDate = Convert.ToDateTime(curDate2);

}

1. Код окна «Меню владельца ТС»

public partial class Owners : Form

{

public Owners()

{

InitializeComponent();

}

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

{

Form form1 = new CarstoryO();

form1.Show();

}

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

{

Form form2 = new Car\_DocumentO();

form2.Show();

}

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

{

Form form3 = new DriverO();

form3.Show();

}

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

{

Form form4 = new PenaltyO();

form4.Show();

}

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

{

Form form5 = new DTPO();

form5.Show();

}

}

1. Код окна «Автоистория»

public partial class CarstoryO : Form

{

public CarstoryO()

{

InitializeComponent();

}

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

{

if (textBox1.Text != "")

{

string gosznak = textBox1.Text;

var validator = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

sqlCommand2.Parameters["@register"].Value = textBox1.Text;

sqlCommand3.Parameters["@registr\_Sing"].Value = textBox1.Text;

sqlConnection1.Open();

sqlCommand2.ExecuteNonQuery();

sqlCommand3.ExecuteNonQuery();

sqlConnection1.Close();

var result = sqlCommand3.Parameters["@vin"].Value;

var result2 = sqlCommand3.Parameters["@dtp"].Value;

var result3 = sqlCommand3.Parameters["@ctc"].Value;

var result4 = sqlCommand3.Parameters["@search"].Value;

var result5 = sqlCommand3.Parameters["@owners"].Value;

var result6 = sqlCommand3.Parameters["@con"].Value;

textBox2.Text = Convert.ToString(result);

textBox3.Text = Convert.ToString(result2);

textBox4.Text = Convert.ToString(result3);

textBox5.Text = Convert.ToString(result4);

textBox6.Text = Convert.ToString(result5);

textBox7.Text = Convert.ToString(result6);

if (textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "" || textBox5.Text == "" || textBox6.Text == "" || textBox7.Text == "")

{

MessageBox.Show("Такого номера нет");

}

}

else

{

MessageBox.Show("Проверьте корректность ввода номера");

}

}

else

{

MessageBox.Show("Вы ничего не ввели");

}

}

1. Код окна «Проверка документов ТС»

public partial class Car\_DocumentO : Form

{

public Car\_DocumentO()

{

InitializeComponent();

}

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

{

if (textBox1.Text != "")

{

string gosznak = textBox1.Text;

var validator = new Regex("([0-9]{6,9})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

myCommand.Parameters["@doc\_Number"].Value = textBox1.Text;

sqlConnection1.Open();

myCommand.ExecuteNonQuery();

sqlConnection1.Close();

var result = myCommand.Parameters["@vin"].Value;

var result2 = myCommand.Parameters["@car\_color"].Value;

var result3 = myCommand.Parameters["@register\_Sing"].Value;

textBox2.Text = Convert.ToString(result);

textBox3.Text = Convert.ToString(result2);

textBox4.Text = Convert.ToString(result3);

if (textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "")

{

MessageBox.Show("Такой машины не найдено");

}

}

else

{

MessageBox.Show("Провеьте корректность данных");

}

}

else

{

MessageBox.Show("Вы ничего не ввели");

}

}

}

1. Код окна «Проверка водителя»

public partial class DriverO : Form

{

public DriverO()

{

InitializeComponent();

}

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

{

if (textBox1.Text != "")

{

string gosznak = textBox1.Text;

var validator = new Regex("([0-9]{6,9})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

sqlCommand1.Parameters["@number\_DL"].Value = Convert.ToInt32(textBox1.Text);

sqlCommand1.Parameters["@date\_of\_birth"].Value = dateTimePicker1.Value;

sqlConnection1.Open();

sqlCommand1.ExecuteNonQuery();

sqlConnection1.Close();

var result = sqlCommand1.Parameters["@fio"].Value;

var result2 = sqlCommand1.Parameters["@Open\_c"].Value;

var result3 = sqlCommand1.Parameters["@Legth"].Value;

textBox2.Text = Convert.ToString(result);

textBox3.Text = Convert.ToString(result2);

textBox4.Text = Convert.ToString(result3);

if (textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "")

{

MessageBox.Show("Такого водителя не найдено");

}

}

else

{

MessageBox.Show("Провеьте корректность данных");

}

}

else

{

MessageBox.Show("Вы ничего не ввели");

}

}

private void DriverO\_Load(object sender, EventArgs e)

{

DateTime dt = DateTime.Now;

dt = dt.AddYears(-10);

string curDate = dt.ToShortDateString();

dateTimePicker1.MinDate = Convert.ToDateTime(curDate);

}

}

1. Код окна «Поиск штрафов»

public partial class PenaltyO : Form

{

public PenaltyO()

{

InitializeComponent();

}

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

{

if (textBox1.Text != "")

{

string gosznak = textBox1.Text;

var validator = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

sqlCommand1.Parameters["@registr\_Sing"].Value = textBox1.Text;

sqlConnection1.Open();

var temp = new DataTable();

temp.Load(sqlCommand1.ExecuteReader());

dataGridView1.DataSource = temp;

sqlConnection1.Close();

if (dataGridView1.Rows.Count < 2)

{

MessageBox.Show("Такого номера нет");

}

}

else

{

MessageBox.Show("Проверьте корректность ввода номера");

}

}

else

{

MessageBox.Show("Вы ничего не ввели");

}

}

}

1. Код окна «Информация о ДТП»

public partial class DTPO : Form

{

public DTPO()

{

InitializeComponent();

}

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

{

if (textBox1.Text != "")

{

string gosznak = textBox1.Text;

var validator = new Regex("([У,К,Е,Н,Х,В,А,Р,О,С,М,Т])+([0-9]{3})+([У,К,Е,Н,Х,В,А,Р,О,С,М,Т]{2})+([0-9]{2,3})$");

var validateResult = validator.Match(gosznak);

if (validateResult.Success)

{

sqlCommand1.Parameters["@registr\_Sing"].Value = textBox1.Text;

sqlConnection1.Open();

var temp = new DataTable();

temp.Load(sqlCommand1.ExecuteReader());

dataGridView1.DataSource = temp;

sqlConnection1.Close();

if (dataGridView1.Rows.Count < 2)

{

MessageBox.Show("Такого номера нет");

}

}

else

{

MessageBox.Show("Проверьте корректность ввода номера");

}

}

else

{

MessageBox.Show("Вы ничего не ввели");

}

}

}

1. Код хранимой процедуры «Добавление автоистории»

CREATE PROCEDURE [dbo].[AddCarstory] @register nchar(9), @vin nvarchar(50), @dtp int, @search nvarchar(3), @owners int, @con nvarchar(3)

AS

BEGIN

declare @ctc int

Select @ctc = Document\_Number from Car\_Document where Register\_Sign = @register

if exists (Select \* from Carstory where Register\_Sign = @register)

update Carstory Set VIN = @vin, DTP = @dtp, Document\_Number = @ctc, Search = @search, Amount\_Of\_Owners = @owners, Confines = @con where Register\_Sign = @register

else

Insert into Carstory Values(@register, @vin, @dtp, @ctc, @search, @owners, @con)

END

1. Код хранимой процедуры «Обновление Автоистории»

ALTER PROCEDURE [dbo].[Search] @register nchar(9), @search nvarchar(3), @con nvarchar(3), @search2 nvarchar(3) output, @con2 nvarchar(3) output

AS

BEGIN

Select @search2 = Search, @con2 = Confines from Carstory where Register\_Sign = @register

Update Carstory Set Search = @search, Confines = @con where Register\_Sign = @register

END

1. Код хранимой процедуры «Добавлние даты платежа»

ALTER PROCEDURE [dbo].[PaymentPenalty]

@ruling\_number int, @payment\_date date output, @res int output

AS

BEGIN

if exists (select \* from Penalty where Ruling\_Number = @ruling\_number)

set @res = 1

update Penalty set Payment = @payment\_date where Ruling\_Number = @ruling\_number

END

1. Код хранимой процедуры «Проверка документов ТС»

ALTER PROCEDURE [dbo].[Check\_CarDoc]

@doc\_Number int, @vin nvarchar(50) output, @car\_color nvarchar(50) output, @register\_Sing nchar(9) output

AS

BEGIN

select @vin = VIN, @car\_color = Car\_Color, @register\_Sing = Register\_Sign from Car\_Document where Document\_Number = @doc\_Number

END

1. Код хранимой процедуры «Проверка водителя»

ALTER PROCEDURE [dbo].[Check\_Driver]

@number\_DL int, @date\_of\_birth date, @fio nvarchar(50) Output, @Open\_c nvarchar(10) output, @Legth int output

AS

BEGIN

Select @fio = FIO, @Open\_c = Open\_Categories, @Legth = Length\_Of\_Work from Driver where Number\_DL = @number\_DL and Date\_Of\_Issue = @date\_of\_birth

END