МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра анализа данных и искусственного интеллекта**

**КУРСОВАЯ РАБОТА**

**АВТОМАТИЗИРОВАННАЯ СИСТЕМА УЧЕТА ДЛЯ ДОРОЖНОЙ ИНСПЕКЦИИ**

Работу выполнил(а) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.Макаров

Направление подготовки 09.04.02 Информационные системы и технологии

Направленность (профиль) Искусственный интеллект и машинное обучение

Научный руководитель

д-р. пед. наук, канд. физ.-мат. наук, профессор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Юнов

Нормоконтролер

канд. физ.-мат. наук, доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Калайдина

Краснодар

2023

**РЕФЕРАТ**

Курсовая работа 32 с., 26 рис., 11 таблиц, 3 источника.

БАЗЫ ДАННЫХ, СТРУКТУРИРОВАННЫЙ ЯЗЫК ЗАПРОСОВ, СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ, АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ, УЧЁТ АВТОМОБИЛЕЙ

Объектами исследования являются системы управления базами данных, а также автоматизированные системы для бизнес-проектов.

Целью курсового проекта является закрепление основ и углубление знаний в области проектирования автоматизированных систем с внедренной системой баз данных, а также разработка модели базы данных с последующей реализация приложения, совместимого с данной базой.

К полученным результатам относятся модель базы данных в среде системы государственной инспекции безопасности дорожного движения, а также приложение, полностью совместимого с имеющейся базой.

К решённым задачам относятся описание предметной области, концептуальное проектирование структур базы данных, реализация модели базы данных при помощи средств СУБД, а также реализация приложения автоматизированной системы.

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc135913405)

[1 Описание предметной области 5](#_Toc135913406)

[1.1 Общее описание предметной области 5](#_Toc135913407)

[1.2 Назначение и цели создания ПО 6](#_Toc135913408)

[1.3 Выбор СУБД 7](#_Toc135913409)

[1.4 Выбор Среды разработки приложения 8](#_Toc135913410)

[1.5 Архитектура ПО 10](#_Toc135913411)

[2 Проектирование структуры базы данных 11](#_Toc135913412)

[2.1 Описание сущностей 11](#_Toc135913413)

[2.2 Построение концептуальной модели базы данных 12](#_Toc135913414)

[2.3 Проектирование таблиц и построение ER-диаграммы 13](#_Toc135913415)

[2.4 Реализация базы данных 16](#_Toc135913416)

[3 Программная реализация 18](#_Toc135913417)

[3.1 Пользовательский интерфейс 18](#_Toc135913418)

[3.2 Тестирование приложения 26](#_Toc135913419)

[Заключение 31](#_Toc135913420)

[Список использованных источников 32](#_Toc135913421)

# ВВЕДЕНИЕ

Проектирование автоматизированной системы в сфере бизнеса – довольно сложное дело, поскольку состоит из разработки и проектирования сразу нескольких компонентов, таких как: модель базы данных и модель программного обеспечения.

В свою очередь, каждый компонент такой системы требуется детально расписать, чтобы в итоге получить гибкое, устойчивое к множеству ошибок (особенно при операциях с таблицами базы данных) программное приложение. [1]

Для проектирования модели базы данных используется инструментарий в виде системы управления базами данных (СУБД), со встроенным в неё структурированным языком запросов (наиболее часто используемым таким языком является SQL). Для модели ПО, перед самим написанием программного кода, прибегают к созданию диаграмм, способных максимально подробно описать функциональность и структуру приложения. [2]

В качестве инструментария для такого описания применяют унифицированный язык моделирования, или сокращенно – UML. Фактически, язык представлен не кодом, а диаграммами, описывающими разные составные аспекты программного продукта. [3]

Целью курсового проекта является подкрепление и расширение имеющихся знаний в области автоматизированных систем и баз данных, посредством проектирования программного продукта для информационной системы безопасности дорожного движения.

# Описание предметной области

# Общее описание предметной области

В современном быстро меняющемся мире управление транспортными средствами владельцами может быть сложной задачей. Часто важная информация теряется или упускается из виду, что приводит к целому ряду проблем. Чтобы помочь смягчить эти проблемы, это программное обеспечение было разработано, чтобы сделать учет транспортных средств более эффективным и результативным, с конечной целью повышения безопасности, сокращения нарушений и повышения удовлетворенности клиентов. Информационная система осуществляет множество операций, связанных с автомобилями и безопасностью дорожного движения. Система позволяет регистрировать в базе людей, получивших водительское удостоверение, а также регистрировать само водительское удостоверение.

Человек, имеющий водительское удостоверение, может зарегистрировать автомобиль на себя. Каждый новый добавленный автомобиль должен иметь регистрацию на какого-либо конкретного человека, причём только одного. Возможно переоформлять автомобиль, если между людьми произошел процесс купли-продажи автомобиля.

Каждый автомобиль должен обладать страховым полисом, прежде чем зарегистрируется в системе, согласно правилам дорожного движения.

В процессе регистрации автомобилю присваивается уникальный государственный номер, а также отмечается дата регистрации.

Также, на каждый автомобиль накладывается определённый налог в зависимости от объёма двигателя.

Система ведёт учёт дорожно-транспортных происшествий. Каждый автомобиль, попавший в ДТП, заносится в базу данных.

Также система позволяет вести учёт регулярного технического осмотра для конкретного автомобиля.

Сотрудники ГИБДД могут выписывать штрафы на конкретное водительское удостоверение. Нарушители могут оплачивать штрафы

# Назначение и цели создания ПО

Разрабатываемое приложение предназначено для ведения учета автомобилей, а также нарушений, ДТП и владельцев.

Рядовые пользователи могут найти информацию об автомобиле, прежде чем покупать его.

Сотрудники ГИБДД регистрируют автомобиль и выписывают штрафы.

Владельцы автомобиля могут просматривать и оплачивать штрафы.

Приведем 3 основные цели создания ПО:

1. отслеживание происшествий. Программное обеспечение предназначено для управления автомобилями путем предоставления важной информации заинтересованным сторонам. Пользователи могут регистрировать и просматривать сведения об аварии, такие как имя водителя, место и время аварии, а также создавать необходимые отчеты.
2. Повышенная безопасность. ПО предназначено для повышения стандартов безопасности за счет выявления и устранения потенциальных опасностей и нарушений при покупке автомобиля, путем отслеживания истории технического обслуживания транспортного средства.
3. Повышение удовлетворенности клиентов. Удобный интерфейс программного обеспечения и автоматизация задач направлены на повышение удовлетворенности клиентов за счет сокращения бумажной работы, сокращения времени отклика и предоставления клиентам информации в режиме реального времени.

# Выбор СУБД

В качестве системы управления базами данных бала взята самая распространённая в мире на сегодня СУБД – Microsoft SQL Server. Программа, при помощи которой происходит контакт администратора БД и СУБД – Microsoft SQL Server Management Studio.

В качестве аргументов для выбора данной СУБД я рассматривал 5 пунктов:

1. Масштабируемость: Microsoft SQL Server обладает высокой масштабируемостью и может обрабатывать большие объемы данных, что делает его идеальным выбором для предприятий с растущими потребностями в данных. Его также можно легко интегрировать с другими продуктами Microsoft, такими как Excel, SharePoint и Power BI.
2. Безопасность: Microsoft SQL Server имеет несколько функций безопасности, включая шифрование конфиденциальных данных, управление доступом на основе ролей, а также инструменты аудита и мониторинга для отслеживания активности базы данных. Это делает его очень безопасным вариантом для предприятий, имеющих дело с конфиденциальной информацией.
3. Производительность. Microsoft SQL Server оптимизирован для обеспечения высокой производительности и может быстро и эффективно обрабатывать сложные запросы и транзакции. Он также предлагает OLTP в памяти (онлайн-обработку транзакций), который может значительно повысить производительность для определенных рабочих нагрузок.
4. Поддержка: Microsoft предлагает обширную поддержку и документацию для SQL Server, что позволяет пользователям легко устранять неполадки и находить ответы на свои вопросы.
5. Экономичность. По сравнению с другими СУБД Microsoft SQL Server относительно рентабелен и предлагает ряд вариантов лицензирования для удовлетворения различных потребностей бизнеса. Он также предлагает ряд готовых инструментов и функций, что снижает потребность в сторонних надстройках.

В целом, Microsoft SQL Server предлагает ряд преимуществ, которые делают его отличным выбором для предприятий любого размера. В то время как другие СУБД могут предлагать некоторые из этих функций, сочетание масштабируемости, безопасности, производительности, поддержки и экономичности Microsoft SQL Server делает его одним из главных претендентов на рынке.

Сущности, указанные в базе, должны чётко разграничивать свою функциональность, не иметь слишком много атрибутов, загромождающих память и не имеющих особого логического смысла.

Также, для первичных ключей рекомендуется ограничение по длине – 8 символов.

Тип данных для атрибутов, описывающих стоимость, ввиду практического отсутствия в денежном обороте копеек, рекомендуется установить, как целочисленный.

# Выбор Среды разработки приложения

В качестве среды программирования была выбрана среда разработки JetBrains Rider - мощная кроссплатформенная .NET IDE, предоставляющая разработчикам широкий спектр функций и преимуществ. Приведу 5 ключевых преимуществ использования:

1. Кроссплатформенная поддержка: JetBrains Rider совместим с операционными системами Windows, macOS и Linux, что помогает разработчикам беспрепятственно работать на разных платформах.
2. Всесторонняя поддержка .NET. JetBrains Rider обеспечивает полную поддержку .NET Framework, .NET Core, Xamarin и Unity. Это означает, что разработчики могут писать код для различных платформ и приложений в одной среде IDE.
3. Интуитивно понятный пользовательский интерфейс. Интерфейс JetBrains Rider создан для повышения производительности и эффективности. IDE включает полный набор функций и инструментов, доступ к которым можно получить через интуитивно понятный пользовательский интерфейс.
4. Широкие возможности настройки. JetBrains Rider предлагает широкие возможности настройки, которые позволяют разработчикам адаптировать IDE к своим конкретным потребностям. Это включает в себя настраиваемые сочетания клавиш, параметры форматирования кода и многое другое.
5. Большое сообщество: JetBrains Rider имеет большое и активное сообщество разработчиков, которые предлагают поддержку, делятся советами и рекомендациями и вносят свой вклад в разработку IDE.

Таким образом, JetBrains Rider — это мощная и универсальная IDE, которая предлагает разработчикам широкий спектр функций и преимуществ. Его межплатформенная поддержка, расширенные инструменты анализа кода и отладки, интуитивно понятный пользовательский интерфейс и широкие возможности настройки делают его отличным выбором для разработчиков .NET.

Программа написана на языке программирования С#, в частности была использована библиотека System.LINQ вместо стандартного для MS SQL языка SQL. LINQ расшифровывается как «Language-Integrated Query», то бишь – встроенные в язык запросы.

Также, преимущество LINQ ещё и в том, что можно использовать как полную форму записи, приближенную к стандартным SQL-запросам, так и укороченную, используя лямбда-функции.

Данные в модели Entity Framework представлены в виде сущностей, которые легко конвертировать в списки после применения выборки.

Само приложение написано на платформе «ASP.NET Core», и может запускаться на любой операционной системе. [4]

# Архитектура ПО

Архитектура нашего ПО имеет слоённую структуру. Самый низкий слой – домен, собственно сущности, с которыми мы работаем в БД.

Слой выше – репозитории. Этот слой используется для доступа к данным из базы данных.

Следующий слой – сервисы. Сервисы позволяют нам маппить объекты, полученные из базы данных, в понятный вид, и наоборот.

Самый внешний слой – контроллеры. Используются для обработки команд пользователя. Схематично эта архитектура представлена на Рисунке 1:

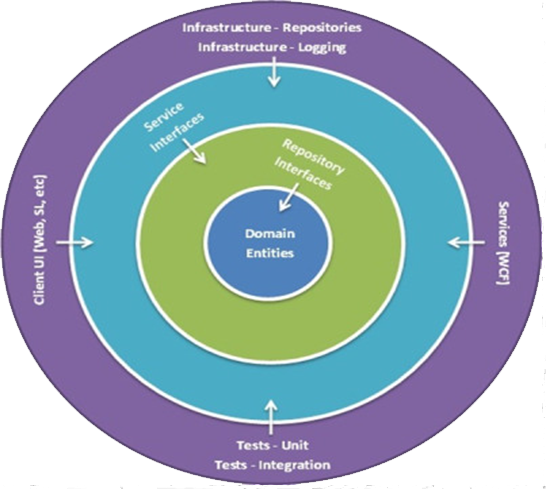


Рисунок 1 – Архитектура

Такая архитектура может называться «луковой архитектурой», или DDD. Такая конфигурация дает нам преимущества в качестве разработки и тестирования и безопасности. Для передачи объектов между слоями используется встроенная в .NET технология Dependency Injection.

# Проектирование структуры базы данных

# Описание сущностей

В базе данных представлено 9 сущностей.

Центральной сущностью является «Автомобиль». Сущность «Автомобиль» хранит данные о названии автомобиля, массе, мощности и объёме двигателя.

Сущность «Автомобиль» непосредственно связана с сущностями «Страховой полис», «Регистрация» и «Технический осмотр».

«Страховой полис» хранит информацию о дате начала страхования авто и дате окончания страхования авто.

«Регистрация» хранит данные о владельце автомобиля, номере, дате регистрации. Сущность «Регистрация» связана с сущностями «Водительское удостоверение», «Автомобиль».

«Технический осмотр» содержит информацию о последнем техническом осмотре авто, в частности на каком пробеге он был выполнен.

Сущность «Водительское удостоверение» содержит информацию о человеке, а также о категории удостоверения, и дату выдачи.

«Владелец» содержит информацию о человеке – его имени и дате рождения.

Также есть сущности «Налог», «Авария» и «Штраф».

«Налог» содержит несколько типов налогов в зависимости от объёма автомобиля, а также стоимость.

Сущность «Авария» содержит тип аварии и сведения об автомобиле, попавшем в ДТП.

«Штраф» содержит описание и стоимость.

# Построение концептуальной модели базы данных

Согласно пункту 2.1, имеется девять сущностей, которые необходимо связать в единую модель.

Сущность «Владелец» связана с «Водительским удостоверением» связью один к одному.

«Водительское удостоверение» связано со «Штрафами» - один ко многим, а также с «Регистрацией» - один ко многим.

«Регистрация» связана с «Автомобилем» связью один к одному, а также между этими сущностями есть таблица-посредник – «Владельцы автомобиля». Когда идет переоформление автомобиля, его старая регистрация удаляется, а таблица «Владельцы автомобиля» хранит в себе информацию о предыдущих владельцах авто.

«Страховой полис» связан с сущностью «Автомобиль» с помощью связи «один к одному».

«Автомобиль» связан с «ТО» - один ко многим, также эта сущность связана с «Налогом» через таблицу-посредник. Также «Автомобиль» имеет отношение к таблице «Аварии» через таблицу-посредник, так как в данном случае реализована связь «многие ко многим».

Конечная вариация концептуальной модели представлена на Рисунке 2:

Изображение выглядит как диаграмма, текст, линия, План

Автоматически созданное описание

Рисунок 2 – Концептуальная модель

# Проектирование таблиц и построение ER-диаграммы

Перед тем, как спроектировать таблицы основных сущностей, необходимо спроектировать сначала таблицы для сущностей побочных.

Таблица 1 – Табличное представление сущности «Человек»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
|  | Дата рождения | date |
|  | Имя | nvarchar |

Таблица – Табличное представление сущности «Водительское удостоверение»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
|  | Дата выдачи | date |
|  | Категория | nvarchar |
|  | Договор до | Date |

Таблица 3 – Табличное представление сущности «Штраф»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
| (вторичный) | Права | Int |
|  | Описание | Nvarchar |
|  | Стоимость | decimal |

Таблица – Табличное представление сущности «Регистрация»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | Int |
| (вторичный) | Права | int |
| (вторичный) | Автомобиль | int |
|  | Гос номер | nvarchar |
|  | Дата регистрации | date |

Таблица 5 – Табличное представление для сущности «Страховка»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | Int |
|  | Дата начала страхования | Date |
|  | Дата окончания страхования | date |

Таблица 6 – Табличное представление для сущности «Автомобиль»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
|  | Модель | Nvarchar(50) |
|  | Объём двигателя | int |
|  | Мощность | int |
|  | Масса | int |

Таблица 7 – Табличное представление сущности «ДТП»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
|  | Тип | Nvarchar(50) |

Таблица – Табличное представление сущности «ДТПиАвто»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
| (вторичный) | Авто | int |
| (вторичный) | Авария | Int |
|  | Дата | date |

Таблица – Табличное представление сущности «Налог»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
|  | Тип | Nvarchar(50) |
|  | Стоимость | decimal |

Таблица – Табличное представление сущности «НалогИАвто»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |

Таблица 11 – Табличное представление сущности «ТО»

|  |  |  |
| --- | --- | --- |
| **Ключевой атрибут** | **Атрибут** | **Тип данных** |
| (первичный) | Ид | int |
| (вторичный) | Авто | int |
|  | Пробег | int |

В итоге диаграмма сущностей выглядит следующим образом (См. Рисунок 3):

Изображение выглядит как текст, диаграмма, План, Параллельный

Автоматически созданное описание

Рисунок – Диаграмма сущностей

# Реализация базы данных

Исходя из ранее указанной диаграммы, составим в соответствующем приложении (MS SQL) сущности и свяжем их между собой прописанными в ER-диаграмме связями.

Проектирование таблиц может быть реализовано либо вручную или путём запроса SQL из разряда Create Table (См. Рисунок 4):

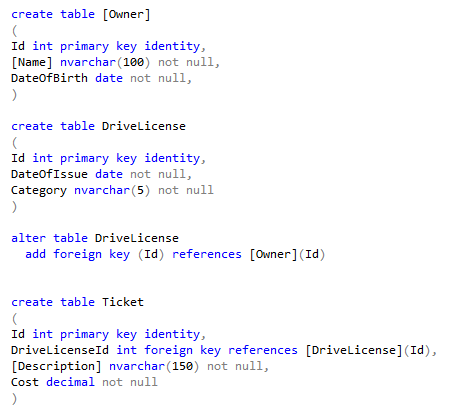


Рисунок – Процесс создания таблицы в MS SQL

Создав таким образом все таблицы, загружаем их в диаграмму, где и проставляются ранее спроектированные связи. Основное требование для создания записи – совпадение типов данных первичного и вторичного ключа в таблицах сущностей.

# Программная реализация

# Пользовательский интерфейс

Представим ситуацию, когда человек, ранее не зарегистрированный в базе, получает водительское удостоверение и покупает новый автомобиль с автосалона. Человека и автомобиль необходимо зарегистрировать. Общая схема регистрации выглядит таким образом:

1. Регистрируется человек, ему выдается водительское удостоверение.
2. Новый автомобиль заносится в базу данных существующих авто.
3. Автомобиль регистрируется на конкретного человека. У автомобиля может быть только один владелец. Автомобилю автоматически делается страхование сроком на 1 год. На этом этапе владельцу выдается документ – свидетельство о регистрации транспортного средства (СТС) с уникальным номером.
4. Автомобиль должен пройти технический осмотр и получить соответствующий штамп.

Итак, запустим наше приложение и приступим к процессу регистрации. Перейдём во вкладку «Владелец», укажем данные человека и нажмём кнопку «Добавить». Приложение уведомляет нас об успешности действия и информирует нас о выданном водительском удостоверении и его номере (См. Рисунок 5):

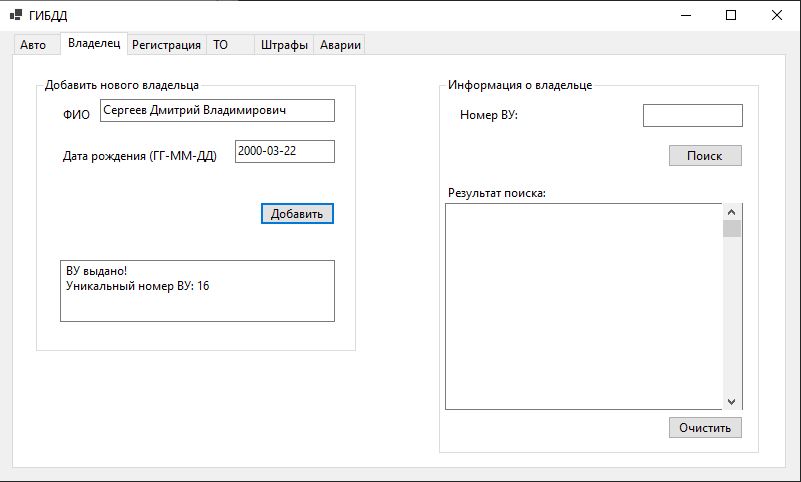


Рисунок 5 – Выдача ВУ

С помощью Microsoft SQL Management Studio проверим, добавился ли в нашем случае Сергеев Дмитрий Владимирович в базу данных (См. Рисунок 6).

C:\Users\ashur\Desktop\Projects\Study\Gibdd_KubSTU\Скрины\2 Владелец в базе.JPG

Рисунок – Владелец в базе данных

Далее добавляем автомобиль. Заходим во вкладку «Авто», вводим данные об автомобиле. Следует обратить внимание, что, если объём двигателя у автомобиля в диапазоне до 2.0 литров, система автоматически добавит соответствующий налог. Жмём кнопку «Добавить» и получаем отклик системы (См. Рисунок 7):

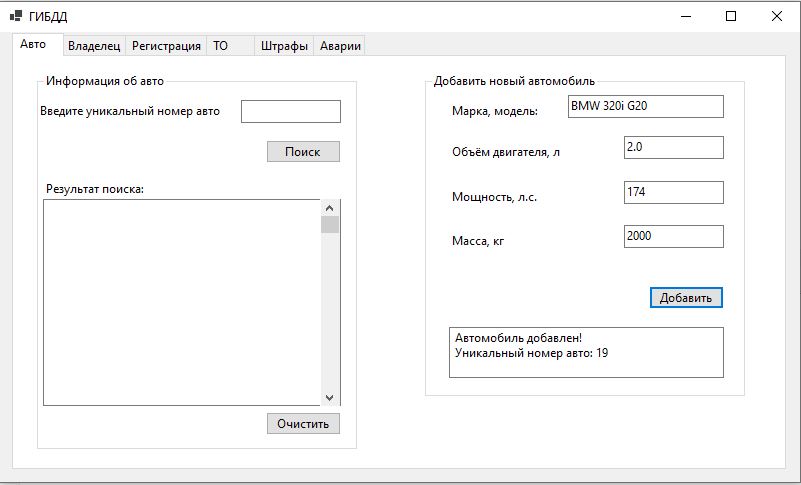


Рисунок 7 – Добавление автомобиля

Проверяем в базе данных (См. Рисунок 8):

C:\Users\ashur\Desktop\Projects\Study\Gibdd_KubSTU\Скрины\0.5 Авто в базе.JPG

Рисунок – Запись автомобиля в базе данных

Далее идёт регистрация. Аналогично, переходим во вкладку «Регистрация», указываем номера ВУ и автомобиля, вводим государственный номер, нажимаем кнопку «Зарегистрировать» (См. Рисунок 9). Если такой государственный номер уже существует, приложение выдаст нам ошибку, так как база данных сконструирована таким образом, что это значение должно быть уникальным. В случае перепродажи автомобиля, предыдущего владельца необходимо снять с регистрации для данного автомобиля, и зарегистрировать нового владельца.

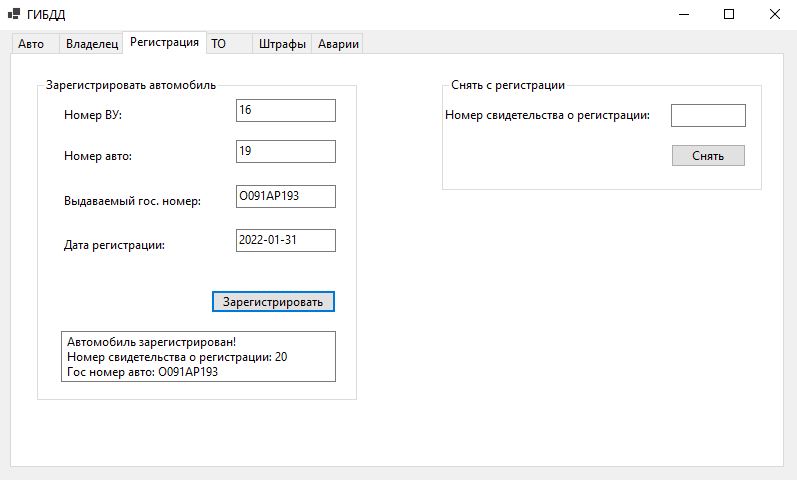


Рисунок – Регистрация

Далее необходимо пройти технический осмотр. Если автомобиль успешно прошёл технический осмотр, переходим во вкладку «ТО» и отмечаем (См. Рисунок 10):

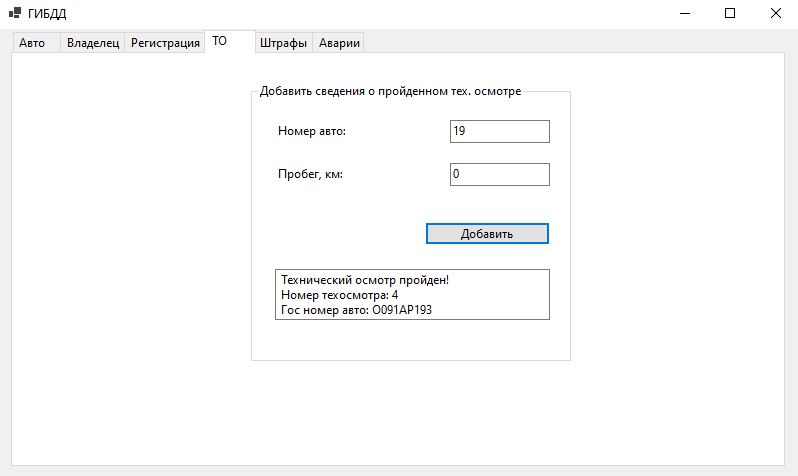


Рисунок 10 – Добавление сведений о техническом осмотре

Здесь нам необходимо указать уникальный номер автомобиля (VIN код) и пробег, на котором данный технический осмотр проводился. В нашем случае машина новая, поэтому пробег указываем 0. Сведение о ТО успешно заносятся в нашу базу данных (См. Рисунок 11):

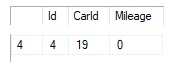


Рисунок 11 – Сведения о ТО в БД

Сотрудник ГИБДД оставляет за собой право выписать штраф за нарушение ПДД. В данном случае мы переходим во вкладку «Штрафы», обозначаем сумму в рублях, номер ВУ оштрафованного человека и описание штрафа (См. Рисунок 12):

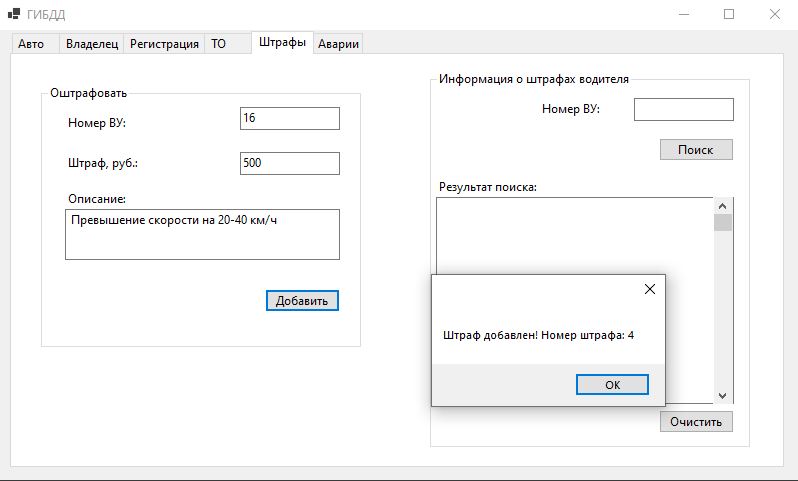


Рисунок – Выписать штраф

В результате нам высвечивается диалоговое окно о результате операции. Сотрудник ГИБДД также может просмотреть наличие штрафов у того или иного водителя. Для этого ему надо указать номер ВУ человека и нажать на кнопку «Поиск» (См. Рисунок 13). В целях проверки проверим нашего тестируемого – Сергеева Дмитрия Владимировича, которому мы только что выписали штраф.

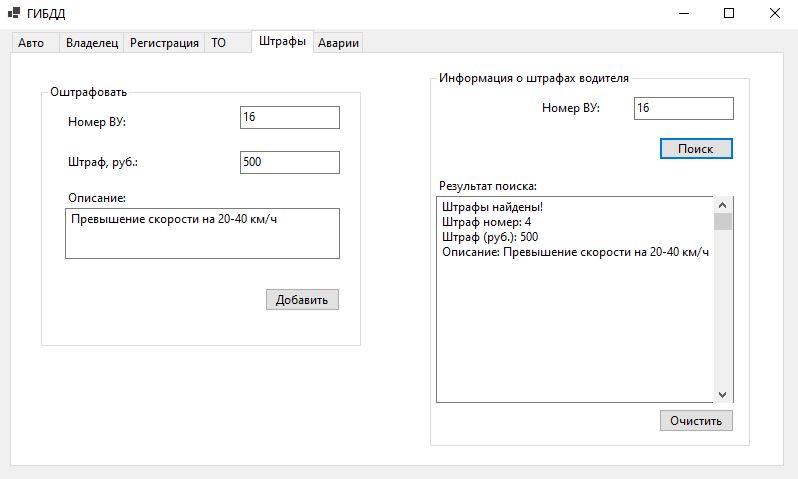


Рисунок – Проверка штрафов

Также сотрудник ГИБДД обязан выписывать протокол о ДТП, если автомобиль попадает в аварию. Для этого в соответствующей вкладке указываем уникальный номер автомобиля и тип ДТП, и дату (См. Рисунок 14). В данном случае тип ДТП – 1, что означает, что ДТП произошло без изменение геометрических характеристик автомобиля. Бывают также другие типы ДТП, например, с нарушением геометрических и эксплуатационных свойств автомобиля, ДТП с участием пешехода и так далее.

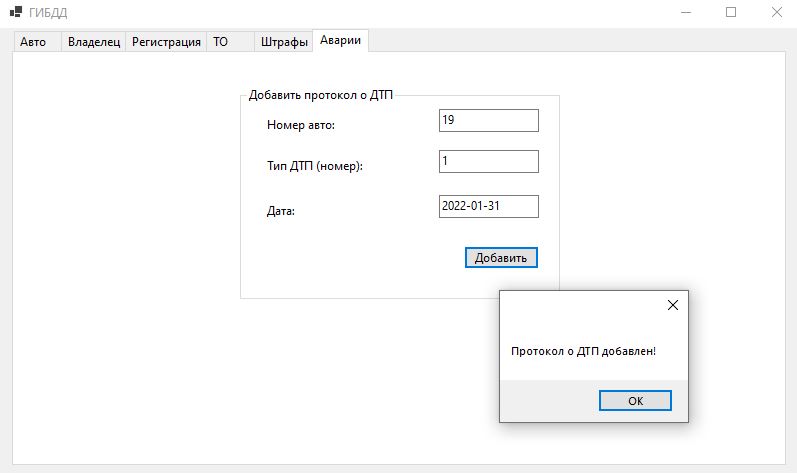


Рисунок – Добавление ДТП

Приложение позволяет формировать отчёт об автомобиле или владельце. Запрашивая отчёт по автомобилю, приложение показывает данные о владельце, о самом автомобиле, данные о налоге и авариях данного автомобиля. Для получения такого отчета переходим во вкладку «Авто», вводим уникальный номер автомобиля на нажимаем на кнопку «Поиск» (См. Рисунок 15). В целях тестирования приложения составим отчёт о тестируемой нами машиной с уникальным номером 19.

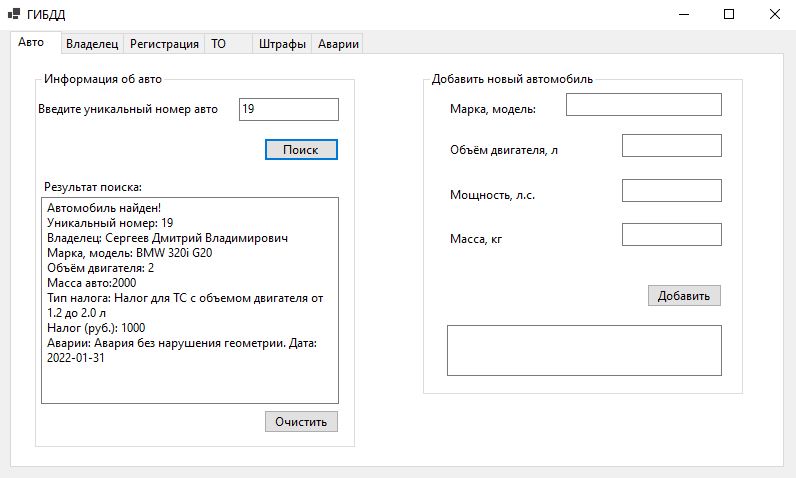


Рисунок 15 – Отчёт по автомобилю

Такой же отчёт мы можем сформировать по владельцу – приложение информирует нас об автомобилях, которыми на данный момент владеет человек и о штрафах (См. Рисунок 16):

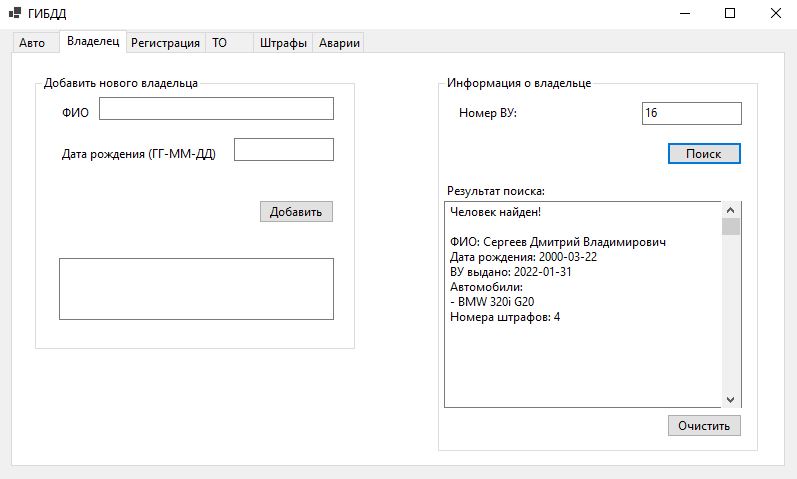


Рисунок 16 – Отчёт по владельцу

# Тестирование приложения

Итак, после того, как наш веб-сервис был реализован, мы можем приступить к тестированию. Работать с сервером мы будем с помощью Swagger UI – встроенной внешней оболочки, облегчающий отправку HTTP запросов. Итак, запустим приложение (Рисунок 17).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 17 – Запуск приложения

Интерфейс Swagger предоставляет нам все функции, которые мы реализовали в контроллерах. Тип запроса POST означает, что данный метод добавляет данные в базу. GET – получение данных из базы. PUT – изменение, и DELETE – удаление.

Пройдём с самого начала процесс регистрации нового авто на нового человека. Первым делом, добавим автомобиль (См. Рисунок 18):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 18 – Добавление автомобиля

Сервер прислал нам информацию о запросе и результат (См. Рисунок 19):

Изображение выглядит как текст, программное обеспечение, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок – Результат добавления авто

Проверяем наличие нового авто в базе данных (См. Рисунок20):



Рисунок – Добавленный авто

Далее регистрируем владельца и его права (См. Рисунок 21):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок –Добавление владельца

После того, как автомобиль, владелец и водительское удостоверение добавлены в базу данных, мы можем приступить к регистрации. В процессе регистрации на автомобиль автоматически накладывается соответствующий налог. Мы указываем Ид владельца и Ид автомобиля, а также уникальный гос. номер (См. Рисунок22):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 22 – Регистрация

Сервер вернет статус 200, если регистрация пройдена успешно. В процессе регистрации также произошло автоматическое страхование автомобиля сроком на 1 год.

Далее над автомобилем необходимо провести технический осмотр. Добавим ТО. Для этого необходимо указать Ид авто, и пробег (См. Рисунок23):

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 23 – Добавление ТО

Для полного тестирования нашей системы добавим штраф. Для этого перейдем в контроллер Tickets и выпишем штраф (См. Рисунок 24)

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Операционная система

Автоматически созданное описание

Рисунок 24 – Добавление штрафа.

Теперь посмотрим результаты, с помощью методов GET.

Проверяем наш автомобиль: кто его владелец, какой у него налог (См. Рисунок 25) и (См. Рисунок 26):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 25 – Ответ сервера

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 26 – Ответ сервера

Всё работает правильно. Сервер присылает JSON ответы, которые мы можем обрабатывать в своих приложениях.

# ЗАКЛЮЧЕНИЕ

В ходе проделанной курсовой работы было создано подробное описание процесса разработки программного обеспечения для дорожной инспекции с использованием средств СУБД.

Были применены следующие средства разработки: MS SQL Visual Studio, MS SQL Server, MS SQL Management Studio.

В результате были смоделированы: концептуальная модель данных, ER-диаграмма, которая легла в основу приложения в части обработки данных. Также было смоделировано и само приложение для работы в данной автоматизированной системе.

Сама же программа оказалась устойчивой к «вылетам» за счёт обработки исключений, появляющихся вследствие попытки внести и сохранить данные, имеющие либо неверное введённые поля, либо имеющие точно такие же значения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Белоногов, Г. Г., Автоматизация процессов накопления, поиска и обобщения информации / Г. Г. Белоногов, А.П. Новоселов – НАУКА-М, 2017. – 256 с. - ISBN 978-5-534-00874-6
2. Пинягина, О.В., Базы данных: учебное пособие: практикум / О.В.Пинягина, И.А. Фукин. – Казань: Казанский федеральный университет, 2018. – 91 с. – ISBN 978-5-517-02575-3
3. Ульман Дж. Основы систем баз данных / Дж. Ульман // Финансы и статистика. – 2017. – 292 с. – ISBN 5-85582-069-6
4. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7: сайт. – 2023. - URL: <http://metanit.com/sharp/tutorial/> (дата обращения 02.03.2023)