

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра информационных систем**

**КУРСОВАЯ РАБОТА**  
**по дисциплине «Управление данными»**  
**Тема: Разработка и реализация базы данных**

Студент гр. 8363

\_\_\_\_\_

Нерсисян А.С.

Преподаватель

\_\_\_\_\_

Татарникова Т.М.

Санкт-Петербург

2020

## ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Нерсисян А.С.

Группа 8363

Тема работы: Разработка и реализация базы данных

Исходные данные:

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для диспетчера автобусного парка. В БД хранятся сведения о водителях, маршрутах автобусов и их характеристиках.

Каждый **водитель характеризуется**: ФИО, классом, стажем работы и окладом, причем оклад зависит от его класса и стажа работы.

**Маршрут автобуса характеризуется**: номером маршрута, временем начала и конца движения, интервалом движения и протяженностью.

**Характеристиками автобусов являются**: номер автобуса, его тип и вместимость, причем вместимость автобуса зависит от его типа.

Каждый водитель закреплен за отдельным автобусом, а каждый автобус закреплен за определенным маршрутом. Необходимо предусмотреть возможность корректировки БД в случаях поступления на работу нового водителя, списывания старого автобуса, введения нового или изменения старого маршрута и т.п.

Диспетчеру автопарка могут потребоваться следующие сведения:

- список водителей, работающих на определенном маршруте;
- номера автобусов, обслуживающих данный маршрут
- когда начинается или заканчивается движение автобусов на всех или отдельных маршрутах;
- какова протяженность всех или определенных маршрутов автобусов;
- на каких автобусах работает водитель.

Диспетчер может вносить следующие изменения:

- прием на работу нового водителя;
- списание старого автобуса;
- изменение протяженности маршрута.

Необходимо предусмотреть возможность выдачи справки о протяженности маршрута и отчета по автопарку (количество автобусов и их тип, номера маршрутов, время начала движения и интервал, ФИО водителей и их класс).

Содержание пояснительной записки:

«Задание на курсовую работу», «Содержание», «Введение», «Анализ предметной области», «Обоснование модели данных», «Логическое проектирование БД», «Обоснование выбора СУБД», «Описание функций управления данными», «Организация защиты БД», «Заключение», «Список использованных источников», «Приложение А. Руководство пользователя БД», «Приложение Б. Листинг программного кода SQL», «Приложение В. Листинг программного кода C#», «Приложение Г. Примеры сгенерированных документов».

Предполагаемый объем пояснительной записки:

Не менее 40 страниц.

Дата выдачи задания: 04.09.2020

Дата сдачи курсовой работы: 11.12.2020

Дата защиты курсовой работы: 11.12.2020

Студент

---

Нерсисян А.С.

Преподаватель

---

Татарникова Т.М.

## **АННОТАЦИЯ**

База данных «Автобусный парк» предназначена для диспетчера автопарка и обеспечивает в режиме диалога доступ к информации о водителях автопарка, маршрутах, автобусах, а также рабочих сменах водителей. Предусмотрена возможность как внесения изменений, так и получения справок с выводом информации на документ MS Word. Также БД позволяет выводить сведения о количестве автобусов в автопарке, окладе водителей с автоматическим расчетом в зависимости от стажа, а также другую полезную информацию.

База данных функционирует с ядром базы данных Microsoft SQL с интерфейсом Windows Forms на C# (Microsoft .NET Framework).

## **SUMMARY**

The "Bus fleet" database is intended for the fleet dispatcher; in the dialogue mode it provides access to information about the fleet drivers, routes, buses, as well as drivers' work shifts. It is possible to both make changes and get help with the output of information to a MS Word document. Also, the database allows you to display information about the number of buses in the fleet, the salary of drivers with automatic calculation depending on the length of service, as well as other useful information.

The database operates with a Microsoft SQL database engine with a Windows Forms interface, project development language is C# (Microsoft .NET Framework).

## СОДЕРЖАНИЕ

Введение .....	8
1. Анализ предметной области .....	9
1.1 Подробное описание объектов предметной области .....	9
1.2 Определение сущностей (объектов) .....	9
1.3 Информационные объекты (сущности) .....	10
1.4 ER-моделирование.....	10
1.5 Формулировка задачи .....	11
1.6 Ограничение на информацию БД .....	11
1.7 Краткое описание алгоритмов решения задач.....	12
1.8 Определение групп пользователей .....	12
1.9 Описание выходных документов, которые должны генерироваться в системе .....	13
1.10 Описание входных документов для заполнения БД.....	13
2. Обоснование модели данных .....	15
3. Логическое проектирование БД .....	17
3.1 Нормализация, схема базы данных.....	20
4. Обоснование выбора СУБД .....	23
4.1 Тип модели данных и её адекватность потребностям рассматриваемой предметной области.....	23
4.2 Характеристики производительности и набор функциональных возможностей .....	23
4.3 Стоимость СУБД и дополнительного ПО.....	24
4.4 Удобство и надежность СУБД в эксплуатации.....	24
5. Описание функций управления данными.....	25

5.1	Хранение (создание информационных объектов).....	25
5.2	Манипулирование (добавление, изменение, удаление, поиск данных)..	27
5.3	Доступ к данным (назначение прав доступа) .....	30
5.4	Предоставление запрашиваемых данных пользователю (генерация справок, отчетов, итогов).....	32
6.	Организация защиты БД.....	37
6.1	Описание ограничений целостности для каждого информационного объекта. ....	37
6.2	Рекомендуемые средства физической защиты (виды резервного копирования и периодичность проведения резервного копирования) .....	40
6.3	Описание процедуры подтверждения личности .....	43
6.4	Применение протоколов шифрования для защиты базы данных .....	45
	Заключение .....	47
	Список использованных источников .....	49
	Приложение А. Руководство ползователя БД.....	50
1.	Краткое описание программы .....	50
2.	Системные требования.....	50
3.	Установка программы .....	50
4.	Работа с программой .....	53
4.1	Вкладка «Водители» .....	54
4.2	Вкладка «Автобусы» .....	57
4.3	Вкладка «Маршруты» .....	59
4.4	Вкладка «Автобусы на маршруте» .....	61
4.5	Вкладка «Запросы и справки» .....	62
4.6	Вкладка «Списанные автобусы» .....	66

4.7 Контекстное меню .....	67
Приложение Б. Листинг программного кода SQL.....	69
Листинг 1. Создание базы данных и пользователя. ....	69
Листинг 2. Создание таблиц и установка зависимостей (внешних ключей)...	70
Листинг 3. Заполнение таблиц данными.....	72
Листинг 4. Поисковые запросы .....	75
Листинг 5. Резервное копирование и восстановление.....	76
Приложение В. Листинг программного кода C# .....	77
Приложение Г. Примеры сгенерированных документов.....	126
1. Сгенерированная справка о протяженности маршрута. ....	126
2. Сгенерированный полный отчет по автопарку. ....	127

## **ВВЕДЕНИЕ**

Цель выполнения курсовой работы – применение на практике знаний, полученных в процессе изучения курса «Управление данными», и получение практических навыков проектирования баз данных.

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для диспетчера автобусного парка. Необходимо обеспечить быстрый доступ к данным, а также быстрое манипулирование данными. В БД хранятся сведения о водителях, маршрутах автобусов и их характеристиках.



## **1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

### **1.1 Подробное описание объектов предметной области**

Предметной областью базы данных является информация об автопарке.

Диспетчеру автопарка доступны следующие сведения:

- информация о водителях;
- информация об автобусах;
- информация о маршрутах;
- информация о сменах;
- список водителей, работающих на определенном маршруте;
- номера автобусов, обслуживающих данный маршрут;
- когда начинается или заканчивается движение автобусов на всех или отдельных маршрутах;
- какова протяженность всех или определенных маршрутов автобусов;
- на каких автобусах работает водитель.

В виде отчета (генерируется документ Word) диспетчеру доступны следующие данные:

- справка о протяженности конкретного маршрута
- полный отчет по автопарку (количество автобусов и их тип, номера маршрутов, время начала движения и интервал, ФИО водителей и их класс).

### **1.2 Определение сущностей (объектов)**

После анализа предметной области выделим следующие сущности:

- Водитель;
- Маршрут автобуса;
- Автобус.

### 1.3 Информационные объекты (сущности)

Сущности имеют следующие атрибуты:

Водитель
ID водителя
ФИО
Класс
Стаж работы
Оклад

Маршрут автобуса
Номер маршрута
Время начала движения
Время конца движения
Интервал движения
Протяженность движения

Автобус
ID автобуса
Номер автобуса
Тип
Вместимость

### 1.4 ER-моделирование

Все 3 сущности не связаны между собой, т.к. каждый водитель может работать на нескольких автобусах, в зависимости от смены, один автобус может также обслуживать несколько маршрутов в разные смены, а один маршрут могут обслуживать несколько водителей и автобусов. Для разрешения этих отношений, вводится сущность «Водят». Добавленная сущность имеет атрибуты: номер автобуса, ID водителя, ФИО, номер маршрута, дата.

Таким образом выделим получившиеся отношения, с учетом добавления новой сущности:

Отношение «Водители – Водят» - «один ко многим», т.к. один водитель может обслуживать несколько маршрутов и работать на нескольких автобусах:

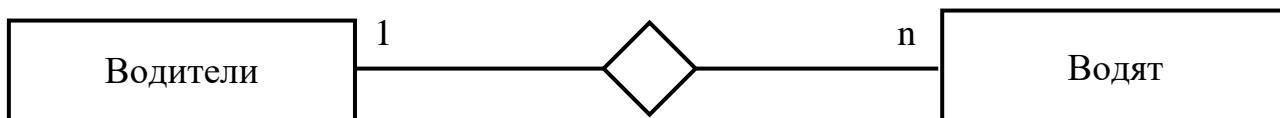


Рисунок 1.

Отношение «Автобусы – Водят» - «один ко многим», т.к. один автобус может обслуживать несколько маршрутов и на нем могут работать несколько водителей:

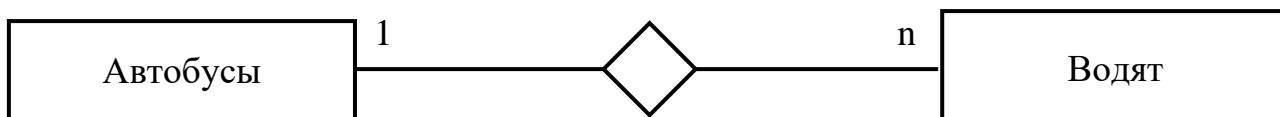


Рисунок 2.

Отношение «Маршрут автобуса – Водят» - «один ко многим», т.к. один маршрут могут обслуживать несколько автобусов и водителей:

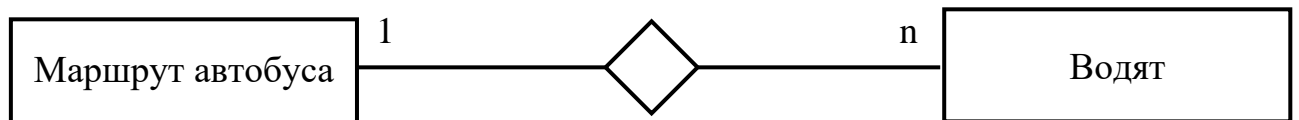


Рисунок 3.

Схема данных приведена на рисунке 4.

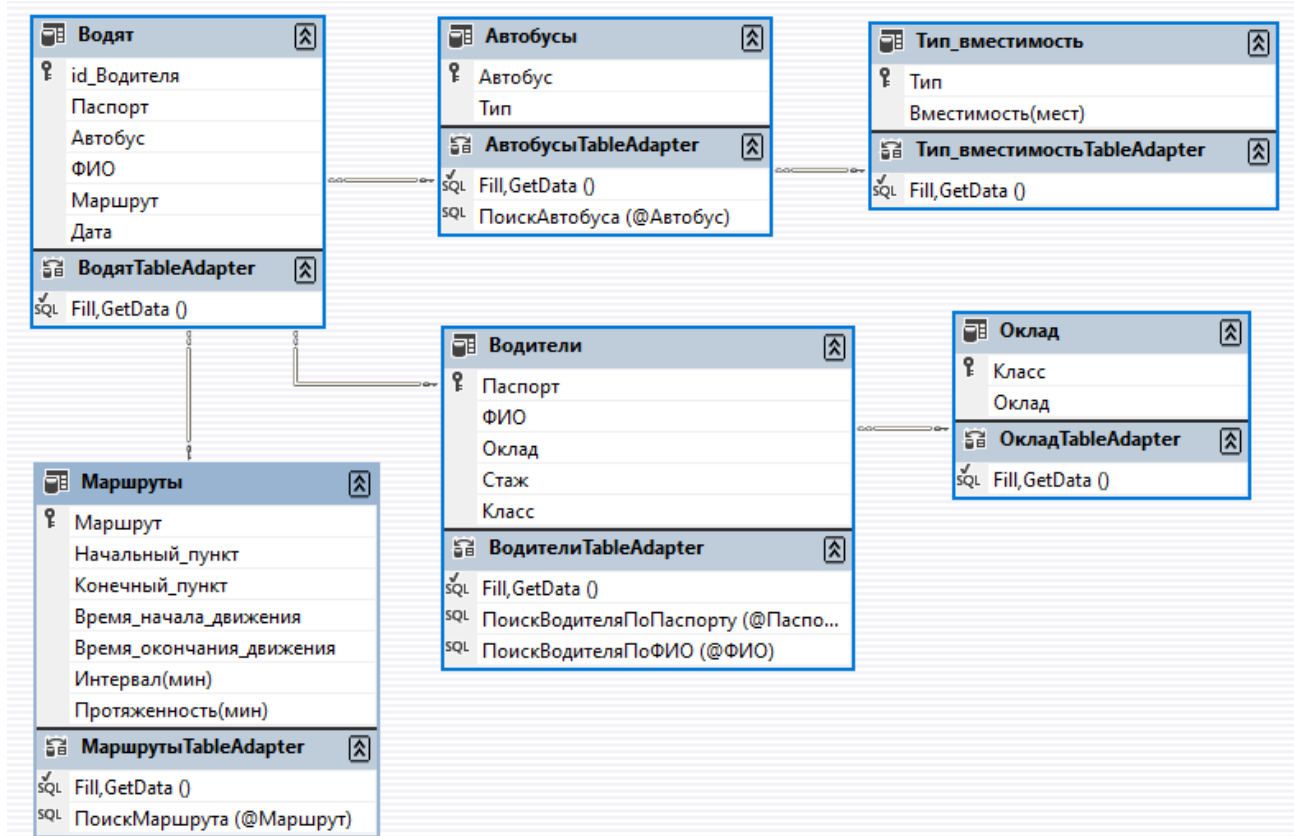


Рисунок 4 – Схема данных.

## 1.5 Формулировка задачи

Хранить информацию о водителях, маршрутах, автобусах и сменах водителей.

Печатать справки о протяженности маршрута и отчета по автопарку.

## 1.6 Ограничение на информацию БД

- Каждый водитель закреплен за отдельным автобусом,
- Каждый автобус закреплен за определенным маршрутом.
- Класс водителя должен допустить вождение данного типа автобуса.

## **1.7 Краткое описание алгоритмов решения задач**

1) Хранение информации о водителях, маршрутах, автобусах и сменах водителей.

Информация о водителях, маршрутах, автобусах и сменах храниться в 6 связанных (по внешним ключам) таблицах, связи организованы таким образом, чтобы обеспечить корректную работу системы (более подробное описание приведено в 5-ом разделе) и целостность базы данных. Например, чтобы уволить водителя (удалить данные водителя с базы данных), надо сначала снять водителя со всех смен. Такое ограничение не позволяет появления ситуации, когда смена поставлена, а водителя нет (его уволили).

2) Печать справок о протяженности маршрута и отчета по автопарку.

Для печати справок используется запрос к базе данных, а с полученными данными создается MS Word документ и заполняется (подробное описание приведено в 5-ом разделе).

## **1.8 Определение групп пользователей**

Пользователь будет один – диспетчер автопарка.

Диспетчеру автопарка доступны следующие сведения:

- информация о водителях;
- информация об автобусах;
- информация о маршрутах;
- информация о сменах;
- список водителей, работающих на определенном маршруте;
- номера автобусов, обслуживающих данный маршрут;
- когда начинается или заканчивается движение автобусов на всех или отдельных маршрутах;
- какова протяженность всех или определенных маршрутов автобусов;
- на каких автобусах работает водитель.

В виде отчета (генерируется документ Word) диспетчеру доступны следующие данные:

- справка о протяженности конкретного маршрута
- полный отчет по автопарку (количество автобусов и их тип, номера маршрутов, время начала движения и интервал, ФИО водителей и их класс).

Диспетчер может вносить следующие изменения:

- прием на работу нового водителя;
- увольнение водителя;
- добавление нового автобуса;
- списание старого автобуса;
- восстановление списанного автобуса;
- окончательное удаление списанного автобуса из базы;
- удаление маршрута
- изменение протяженности маршрута.
- создание нового маршрута;
- изменение смен водителей.

## **1.9 Описание выходных документов, которые должны генерироваться в системе**

- Справка о протяженности маршрута
- Полный отчета по автопарку (количество автобусов и их тип, номера маршрутов, время начала движения и интервал, ФИО водителей и их класс);

## **1.10 Описание входных документов для заполнения БД**

- Технический паспорт автобуса;  
(документ, где описаны номер и технические данные автобуса: тип, вместимость и др. данные).
- Информация о маршрутах автобусов;

(документ, где описаны данные маршрута: начальный и конечные остановки, протяжённость, требуемый интервал движения и др. данные).

- Водительские права водителя.

(документ, где описаны данные водителя: номер водительских прав, ФИО водителя, класс, стаж работы и др. данные)

## 2. ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ

В базе данных будут храниться структурированные данные, как следствие, выбираем модель реляционной базы данных.

Реляционная модель — это набор отношений нескольких двумерных таблиц, в каждой из которых хранится определенная информация.

Основными достоинствами этой модели является удобство, наглядность, возможность осуществления связи многие ко многим, порядок строк и столбцов не существенен, поэтому для данной предметной области эта модель наиболее приемлема.

В реляционной модели достигается гораздо более высокий уровень абстракции данных, чем в иерархической или сетевой. Реляционная модель предоставляет средства описания данных на основе только их естественной структуры, т.е. без потребности введения какой-либо дополнительной структуры для целей машинного представления. Другими словами, представление данных не зависит от способа их физической организации. Это обеспечивается за счет использования математической теории отношений (само название "реляционная" происходит от английского relation - "отношение").

Структура информации дает основание предполагать, что наиболее подходящей для даталогического проектирования будет реляционная модель данных, т.к. она способна обеспечить целостность данных при вставке, удалении и изменении записей, а так же дает возможность организации всех видов связей 1:1, 1:M и M:M (при этом связи M:M раскрываются). К недостаткам традиционных реляционных моделей данных можно отнести является избыточность по полям (из-за создания связей), а также факт того, что в качестве основного и, часто, единственного механизма, обеспечивающего быстрый поиск и выборку отдельных строк таблице (или в связанных через внешние ключи таблицах), обычно используются различные модификации индексов, основанных на B-деревьях. Такое решение оказывается эффективным только при

обработке небольших групп записей и высокой интенсивности модификации данных в базах данных.

На рисунке 4 показана реализация реляционной модели данных.

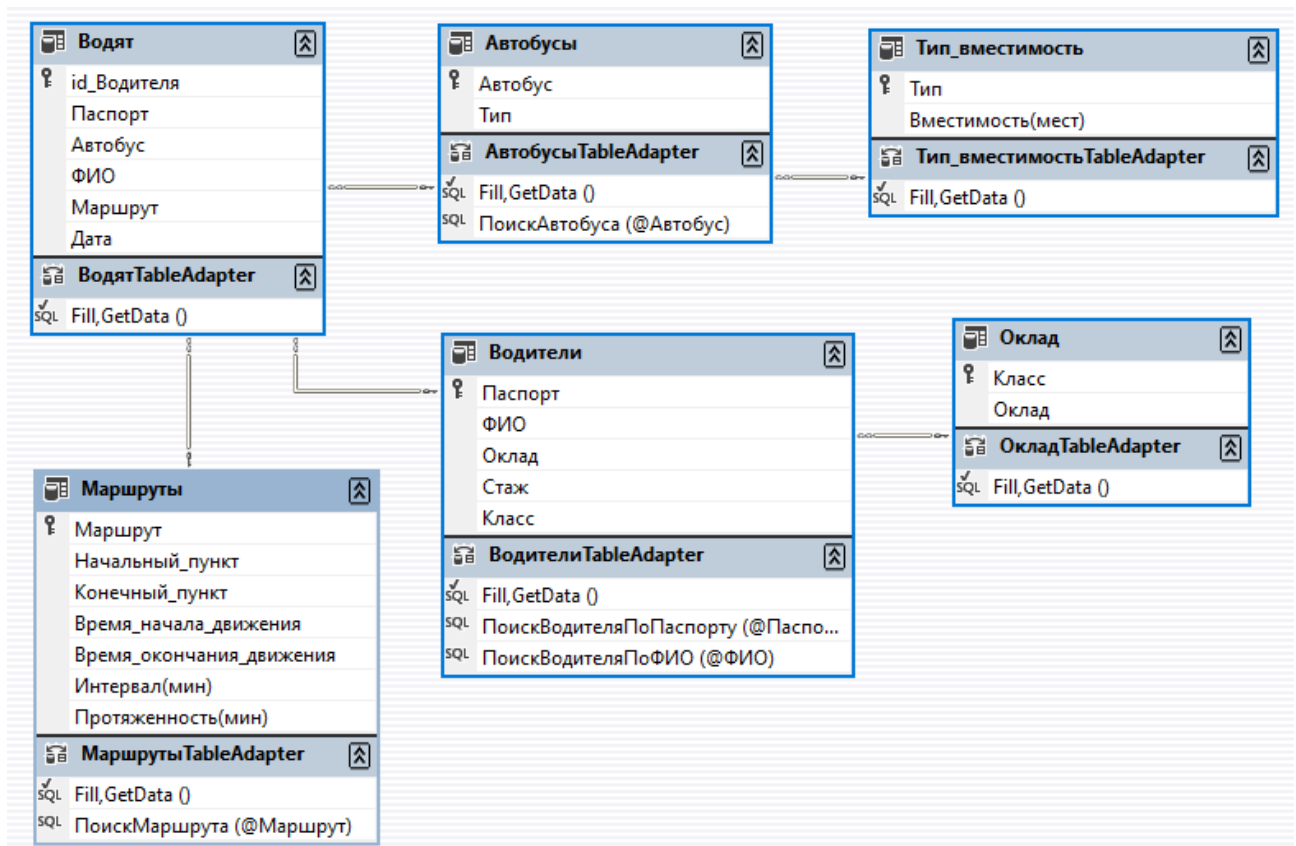


Рисунок 5. Реляционная модель данных.



### 3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БД

Для логического проектирования выбрана реляционная модель данных, т.к. она наиболее полно соответствует требованиям, предъявленным к разрабатываемой информационной системе:

- отсутствие дублируемой информации;
- поддержание целостности данных при вставке, удалении или изменении записей;
- возможность организации всех видов связи между отношениями 1:1, 1:M и M:M.

В реляционной базе данных даталогическое проектирование приводит к разработке корректной схемы базы данных, т.е. такой схемы, в которой отсутствуют нежелательные зависимости между атрибутами. При этом можно использовать процесс проектирования с помощью декомпозиции, т.е. последовательно нормализовать схему отношений, тем самым накладывая ограничения и избавляясь от нежелательных зависимостей между атрибутами.

В реляционных базах данных (РБД) даталогическое проектирование приводит к разработке схемы БД, т.е. совокупности схем отношений, адекватно моделирующих объекты ПО и семантических связей между ними.

Основой анализа корректности схемы являются функциональные зависимости между атрибутами БД. Некоторые могут быть нежелательными.

В конце этого этапа должно быть получено описание схемы БД в терминах выбранной СУБД. Целью даталогического проектирования является построение корректной схемы БД, ориентированную на реляционную модель. Корректной называется схема БД, в которой отсутствуют нежелательные зависимости между атрибутами отношений.

Процесс разработки корректной схемы РБД и является даталогическим проектированием. Возможны 2-а способа:

- Декомпозиция (разбиение).
- Синтез.

Для перехода от инфологической модели к реляционной существует специальный алгоритм:

1. Каждой сущности ставится в соответствие отношение;
2. Каждому атрибуту сущности ставится в соответствие соответствующий атрибут соответствующего отношения;
3. Первичный ключ сущности становится РК соответствующего отношения, при этом атрибуты, входящие в РК, обязательны для заполнения (NOTNULL);
4. В каждое отношение, соответствующее подчинённой сущности, добавляется набор атрибутов основной сущности, являющийся в ней первичным ключом. В отношении, соответствующее подчинённой сущности эти атрибуты становятся FK (внешним ключом);
5. По умолчанию, все атрибуты, не входящие в РК, необязательны;
6. Для отражения категоризации сущностей возможны несколько вариантов;
7. Все связи М:М должны быть раскрыты;

Воспользуемся данным алгоритмом и опишем каждую сущность инфологической модели:

*Таблица 1. Автобусы*

Название столбца	Тип данных	Атрибуты
Автобус	NVARCHAR (6)	PRIMARY KEY, NOT NULL
Тип	NVARCHAR (20)	NOT NULL

*Таблица 2. Тип\_вместимость*

Название столбца	Тип данных	Атрибуты
Тип	NVARCHAR (20)	PRIMARY KEY, NOT NULL
Вместимость(мест)	INT	NOT NULL

Таблица 3. Водители

Название столбца	Тип данных	Атрибуты
Паспорт	VARCHAR (11)	PRIMARY KEY, NOT NULL
ФИО	NVARCHAR (50)	NOT NULL
Оклад	INT	NOT NULL
Класс	NCHAR (5)	NOT NULL
Стаж_работы	INT	NOT NULL

Таблица 4. Маршруты

Название столбца.	Тип данных.	Атрибуты
Маршрут	INT	PRIMARY KEY, NOT NULL
Начальный_пункт	NVARCHAR(25)	NOT NULL
Конечный_пункт	NVARCHAR(25)	NOT NULL
Время_начала_движения	TIME(7)	NOT NULL
Время_конца_движения	TIME(7)	NOT NULL
Интервал_движения	INT	NOT NULL
Протяженность_движения	INT	NOT NULL

Таблица 5. Водят

Название столбца.	Тип данных.	Атрибуты
id_Водителя	INT	PRIMARY KEY, IDENTITY (1, 1) NOT NULL,
Паспорт	VARCHAR (11)	NOT NULL
Автобус	NVARCHAR (6)	NOT NULL
ФИО	NVARCHAR (50)	NOT NULL
Маршрут	INT	NOT NULL
Дата	DATE	NOT NULL

Таблица 6. Оклад

Название столбца	Тип данных	Атрибуты
Класс	NCHAR (5)	PRIMARY KEY, NOT NULL
Оклад	INT	NOT NULL

### 3.1 Нормализация, схема базы данных

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Нормализация – это процесс преобразования базы данных к виду, отвечающему нормальным формам. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации. Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

Понятие нормализации тесно связано с понятием функциональной зависимости. Функциональная зависимость (ФЗ) определяет отношения между объектами и их свойствами в рассматриваемой ПО.

ФЗ  $R.A \rightarrow R.B$  называется полной, если набор атрибутов В ФЗ от А и не зависит функционально от любого подмножества А.

ФЗ  $R.A \rightarrow R.B$  называется транзитивной, если существует такой набор атрибутов С, который удовлетворяет следующим свойствам:

- $C \not\subset A$ ;
- $B \not\subset C$ ;
- существует ФЗ  $R.A \rightarrow R.C$ ;
- не существует ФЗ  $R.C \rightarrow R.A$ ;
- не существует ФЗ  $R.C \rightarrow R.B$ .

Таблица находится в первой нормальной форме, если каждый её атрибут атомарен, то есть может содержать только одно значение. Таким образом, не существует 1НФ таблицы, в полях которых могут храниться списки значений. Для приведения таблицы к 1НФ обычно требуется разбить таблицу на несколько отдельных таблиц.

Отношение находится во второй нормальной форме, если она находится в первой нормальной форме, и при этом любой её атрибут, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа. Функционально полная зависимость означает, что атрибут функционально зависит от всего первичного составного ключа, но при этом не находится в функциональной зависимости от какой-либо из входящих в него атрибутов (частей). Или другими словами: в 2НФ нет не ключевых атрибутов, зависящих от части составного ключа.

Отношение находится в третьей нормальной форме, если она находится во второй нормальной форме 2НФ и при этом любой ее не ключевой атрибут зависит только от первичного ключа. Таким образом, отношение находится в 3НФ тогда и только тогда, когда оно находится во 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых.

Схема базы данных находится во 2НФ, так как не имеет отношений, имеющих составных первичных ключей.

3НФ – отношение находится в 3НФ, если оно находится во 2НФ и не содержит транзитивных зависимостей. Все отношения данной модели находятся в 3НФ, т.к. ни в одном из них нет транзитивных зависимостей.

При решении практических задач в большинстве случаев третья нормальная форма является достаточной. Поэтому процесс проектирования базы данных, как правило, заканчивается приведением к ней.

Разрабатываемая база данных уже удовлетворяет требованиям третьей нормальной формы. Следовательно, процесс нормализации проводить не нужно. Схема базы данных показана на рисунке 6.

Установим связь для таблиц «Водят» и «Автобусы» с помощью внешнего ключа (FOREIGN KEY) для поля данных «Автобус», связь 1:M.

```
ALTER TABLE [dbo].[Водят] WITH CHECK ADD CONSTRAINT
[FK_Водят_Автобусы] FOREIGN KEY([Автобус])
REFERENCES [dbo].[Автобусы] ([Автобус])
GO

ALTER TABLE [dbo].[Водят] CHECK CONSTRAINT
[FK_Водят_Автобусы]
GO
```

Аналогичным образом установим связь между другими таблицами.

Построим схему получившейся БД (Рисунок 6):

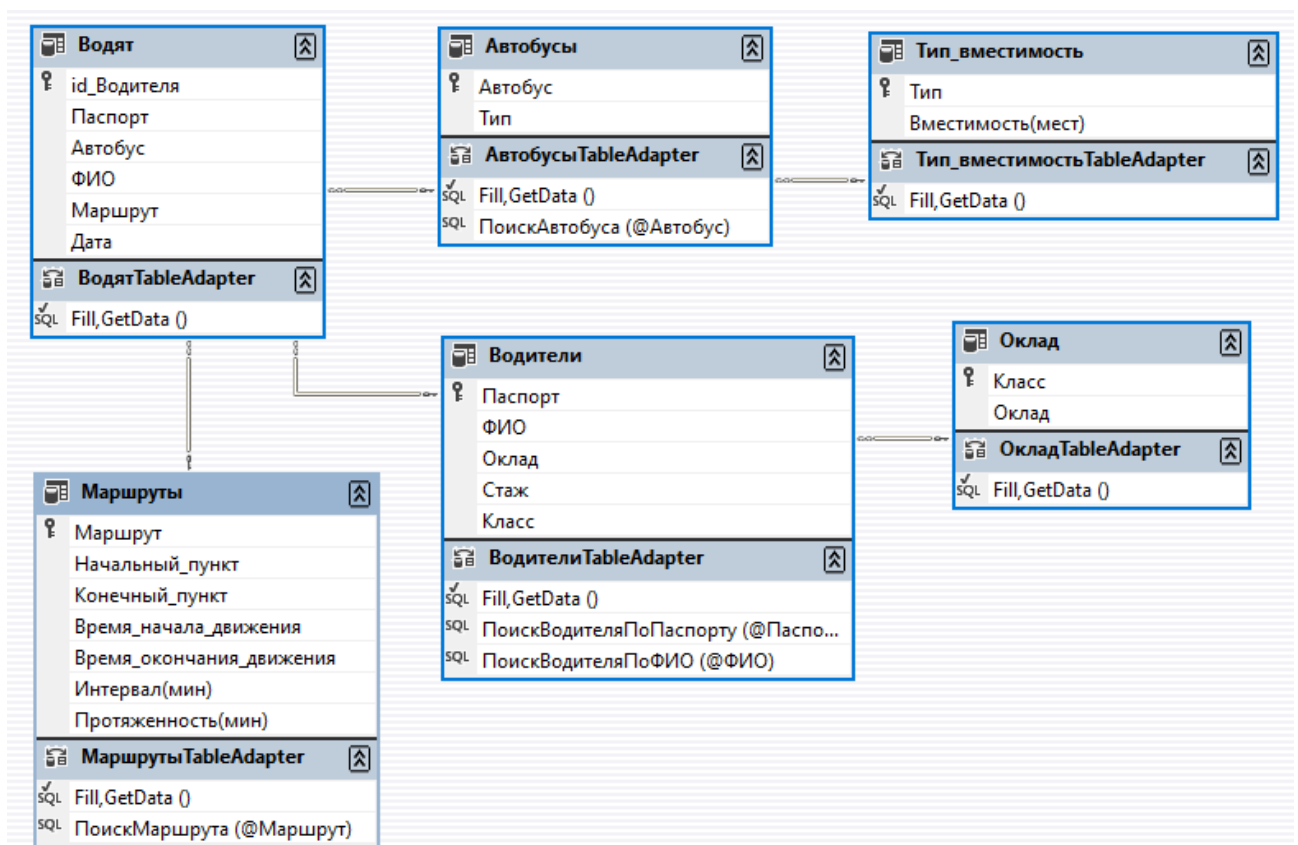


Рисунок 6. Схема реляционной модели БД «Автопарк».

## **4. ОБОСНОВАНИЕ ВЫБОРА СУБД**

### **4.1 Тип модели данных и её адекватность потребностям рассматриваемой предметной области.**

Для реализации курсового проекта я выбрал СУБД Microsoft SQL Server 2019. Microsoft SQL Server 2019 — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL (T-SQL). T-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

### **4.2 Характеристики производительности и набор функциональных возможностей**

В отличие от остальных СУБД, SQL Server обеспечивает интеграцию с Microsoft Office, гарантирует повышенную безопасность и производительность средств разработки, содержит более мощные инструменты бизнес-аналитики по сравнению с решением Oracle. Кроме того, SQL Server имеет более низкую совокупную стоимость владения.

Среди основных преимуществ MS SQL Server можно выделить следующие:

- Масштабируемость и производительность.
- База данных менее уязвима.
- Инструменты бизнес-аналитики с поддержкой самообслуживания.

MS SQL Server предоставляет такие опции, как высокая доступность, усиленная безопасность, улучшенное сжатие данных, сервисы интеграции.

### **4.3 Стоимость СУБД и дополнительного ПО**

Из представленных на рынке в данный момент решений для систем управления реляционными базами данных Microsoft SQL Server является наилучшим по соотношению цена-возможности.

Из дополнительное ПО для данного проекта требуется наличие офисного пакета Microsoft Office, в частности Microsoft Word, для генерации справок и отчетов.

Все взаимодействия с базой данных осуществляются с помощью графической оболочки программы в режиме диалогового окна Windows Forms, написанной на языке программирования C# (Microsoft .NET Framework 4.7.2).

### **4.4 Удобство и надежность СУБД в эксплуатации**

Высокая доступность обеспечивается в первую очередь за счёт отказоустойчивой кластеризации. Отказоустойчивая кластеризация обеспечивает защиту не только базы данных, но и сервера. Данная функция позволяет предотвратить любую потерю данных, что является важным аспектом для заказчика.

Кроме того, высокая доступность обеспечивается за счёт зеркалирования базы данных. В случае сбоя на главном сервере, клиенты автоматически перенаправляются на зеркальный сервер.

В MS SQL Server особое внимание уделяется безопасности. Базовые выпуски SQL Server обладают расширенными функциями обеспечения безопасности, тогда как, например, стандартные и корпоративные выпуски Oracle обеспечивают лишь базовую безопасность.



## 5. ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ

### 5.1 Хранение (создание информационных объектов)

Информационные объекты были созданы согласно таблицам 1-6 (см. раздел 3).

```
CREATE TABLE [dbo].[Автобусы] (  
    [Автобус] NVARCHAR (6) NOT NULL,  
    [Тип]      NVARCHAR (20) NOT NULL,  
    PRIMARY KEY CLUSTERED ([Автобус] ASC)  
)  
GO
```

```
CREATE TABLE [dbo].[Водители] (  
    [Паспорт] VARCHAR (11) NOT NULL,  
    [ФИО]     NVARCHAR (50) NOT NULL,  
    [Оклад]   INT          NOT NULL,  
    [Стаж]    INT          NOT NULL,  
    [Класс]   NCHAR (5)    NOT NULL,  
    PRIMARY KEY CLUSTERED ([Паспорт] ASC)  
)  
GO
```

```
CREATE TABLE [dbo].[Оклад] (  
    [Класс] NCHAR (5) NOT NULL,  
    [Оклад] NCHAR (6) NOT NULL,  
    PRIMARY KEY CLUSTERED ([Класс] ASC)  
)  
GO
```

```

CREATE TABLE [dbo].[Водят] (
    [id_Водителя] INT IDENTITY (1, 1) NOT NULL,
    [Паспорт] VARCHAR (11) NOT NULL,
    [Автобус] NVARCHAR (6) NOT NULL,
    [ФИО] NVARCHAR (50) NOT NULL,
    [Маршрут] INT NOT NULL,
    [Дата] DATE NOT NULL,
    PRIMARY KEY CLUSTERED ([id_Водителя] ASC)
)
GO

```

```

CREATE TABLE [dbo].[Маршруты] (
    [Маршрут] INT NOT NULL,
    [Начальный_пункт] NVARCHAR (25) NOT NULL,
    [Конечный_пункт] NVARCHAR (25) NOT NULL,
    [Время_начала_движения] TIME (7) NOT NULL,
    [Время_окончания_движения] TIME (7) NOT NULL,
    [Интервал(мин)] INT NOT NULL,
    [Протяженность(мин)] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([Маршрут] ASC)
)
GO

```

```

CREATE TABLE [dbo].[Списанные_автобусы] (
    [Автобус] NVARCHAR (6) NOT NULL,
    [Тип] NVARCHAR (20) NOT NULL,
    [Дата] DATE NOT NULL,
    PRIMARY KEY CLUSTERED ([Автобус] ASC)
)
GO

```

```

CREATE TABLE [dbo].[Тип_вместимость] (

```

```

        [Тип] NVARCHAR (20) NOT NULL,
        [Вместимость(мест)] INT NOT NULL,
        CONSTRAINT [PK_Тип_вместимость] PRIMARY KEY ([Тип])
    )
GO

```

## 5.2 Манипулирование (добавление, изменение, удаление, поиск данных)

Для изначального добавления данных в базу использовались запросы SQL, пример запроса показано листинге ниже (полный текст запроса в приложении Б).

```

INSERT INTO Оклад(Класс,Оклад) VALUES('А', 10000)
INSERT INTO Тип_вместимость VALUES(N'малый',40)
INSERT INTO Маршруты VALUES(1, N'Наличная ул.', N'Новосибирская ул.', '06:00:00', '00:00:00', 30, 85)
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'p258ол', N'малый')
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('2888108232', N'Кондратьев Роман Тимурович', 'С', 3, 23000)
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('6502 104936', N'п528чт', N'Михеев Власий Ефимович', 1, '2020-10-19')

```

Для добавления данных из интерфейса программы используются параметризованные запросы. Для приема нового водителя на работу его данные заносятся в базу данных следующим запросом:

```

INSERT INTO [Водители] (Паспорт, ФИО, Оклад, Класс, Стаж)
VALUES(@Паспорт, @ФИО, @Оклад, @Класс, @Стаж)

```

С помощью встроенных средств языка С# вместо параметров @Паспорт, @ФИО, @Класс, @Стаж подставляются соответствующие данные из определенных текстовых полей из интерфейса программы. Учитывая стаж и класс работы водителя считается оклад, подставляются параметры и

выполняется запрос. В случае удачного выполнения в строке состояния программы выводится соответствующее сообщение (см. рис. 7).

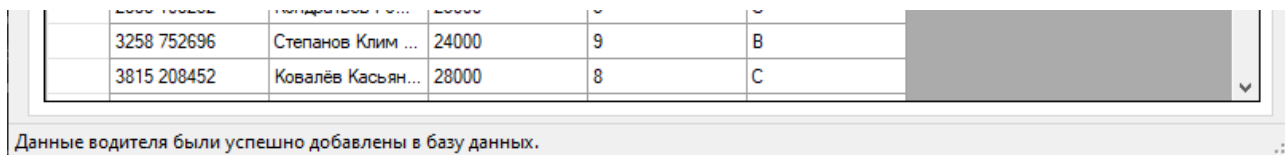


Рисунок 7 – Строка состояния программы

В случае возникновения какой-либо ошибки программа корректно обработает и сообщит об ошибке. Например, при попытке добавить нового водителя, если водитель с таким паспортом уже есть в базе, программа выдаст ошибку (см рис. 8).

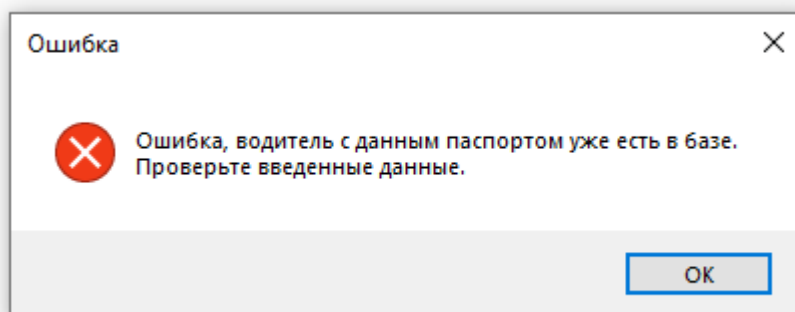


Рисунок 8 – Диалоговое окно ошибки

Запросы удаления данных из базы также реализованы с помощью параметров. Параметризированный запрос удаления аналогичен запросу добавления по части получения данных от пользователя.

При попытке увольнения водителя (удаление данных водителя из базы), показывается диалоговое окно с подтверждением (см. рис. 9).

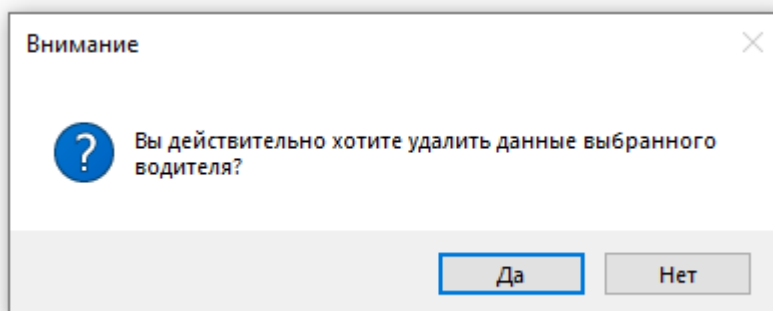


Рисунок 9 – Диалоговое окно подтверждения удаления

Если водитель на смене, т.е. он закреплен за некоторым автобусом в таблице «Водят» (вкладка «Автобусы на маршруте» в интерфейсе программы), то его нельзя уволить. Сначала нужно будет снять водителя с маршрута, а потом уже уволить. В случае, если водитель на смене и диспетчер подтвердит удаление данных водителя, программа корректно обработает ошибку и выведет на экран диалоговое окно с описанием ошибки и действий по ее устранению (см. рис. 10).

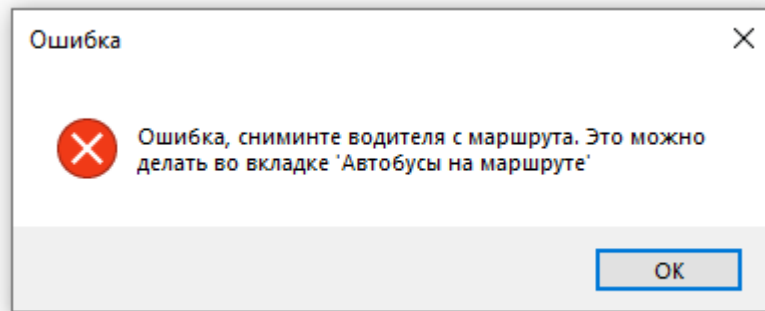


Рисунок 10 – Диалоговое окно ошибки удаления водителя на смене

Пример запроса удаления данных:

```
DELETE FROM [Водители]
WHERE [Паспорт] = @Паспорт
```

Пример запроса обновления данных:

```
UPDATE [Маршруты]
SET [Протяженность(мин)] = @Протяженность
WHERE Маршрут = @Маршрут
```

Пример поискового запроса:

```
SELECT [Автобус], [ФИО]
FROM [Водят]
WHERE [Паспорт] = @Паспорт
```

Значения параметров @Паспорт, @Протяженность и @Маршрут берутся из соответствующих текстовых полей из интерфейса программы.

### 5.3 Доступ к данным (назначение прав доступа)

При запуске программы, запускается форма авторизации пользователя, в данном случае – диспетчера. Диспетчер вводит свой логин и пароль и в случае правильного ввода запускается основное окно программы (см. рис 11, 12).

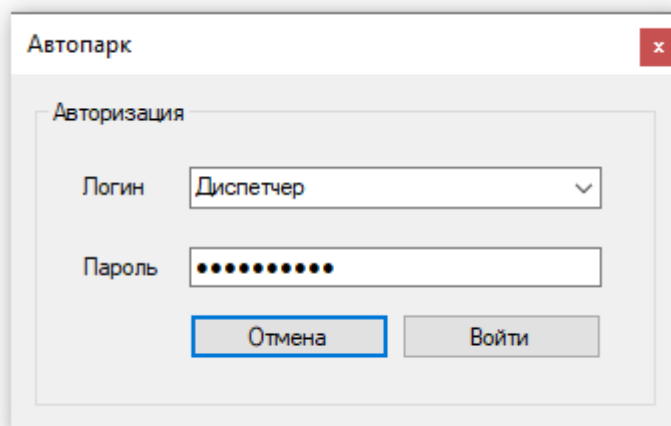


Рисунок 11 – Окно авторизации

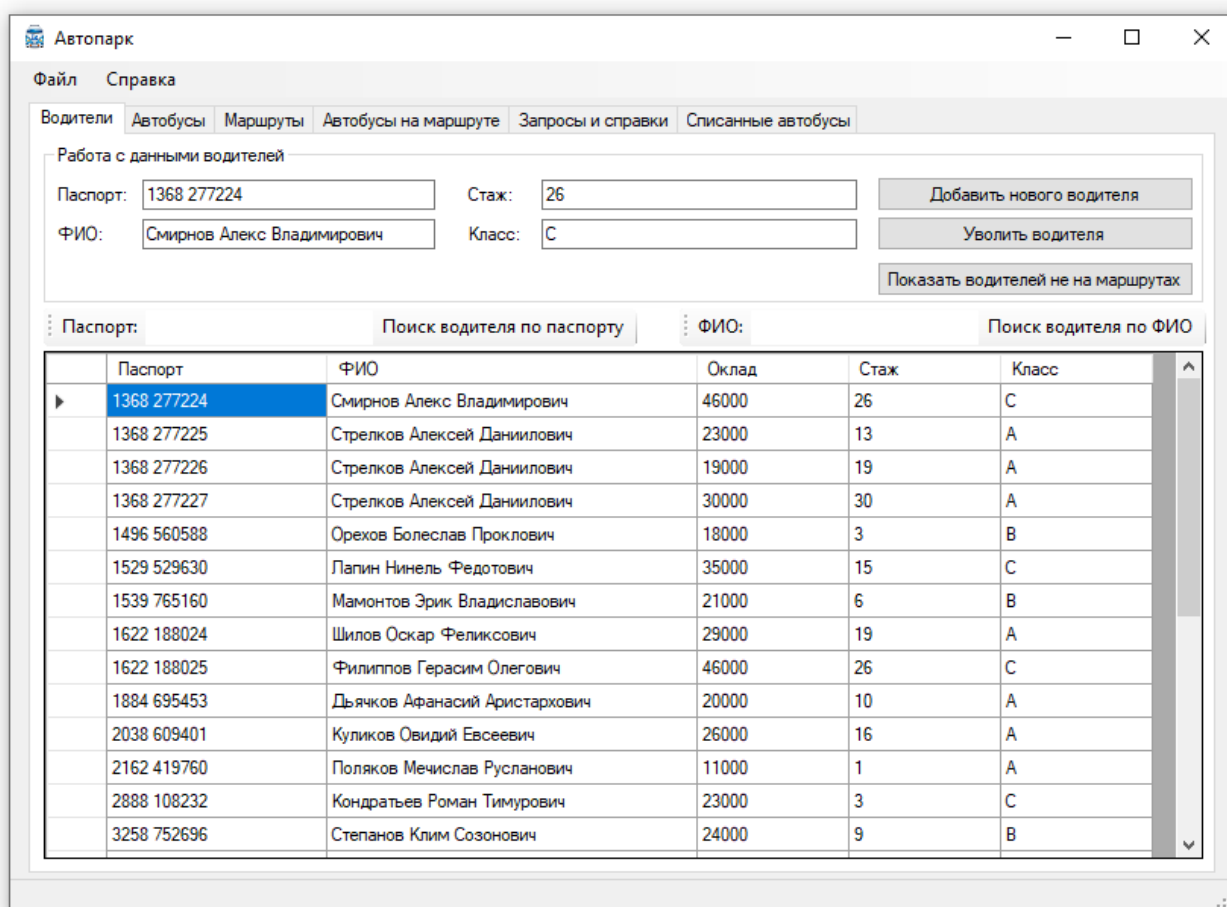


Рисунок 12 – Основное окно программы

Диспетчер использует свою учетную запись на сервере. Программа создает строку подключения к базе данных, заносит введенные данные для авторизации и пытается установить связь с базой данных, в случае успеха загружается основное окно программы.

Запрос для создания пользователя Диспетчер на сервере:

```
USE [master]
GO
CREATE LOGIN [Dispetcher]
WITH PASSWORD='dispetcher',
DEFAULT_DATABASE=[Avtopark],
CHECK_EXPIRATION=OFF,
CHECK_POLICY=OFF
GO

USE [Avtopark]
GO
CREATE USER [Dispetcher] FOR LOGIN [Dispetcher]
GO
USE [Avtopark]
GO
EXEC sp_addrolemember N'db_datawriter', N'Dispetcher'
GO
```

Запрос на предоставление прав этому пользователю к тем же базам:

```
USE [Avtopark]
GO
GRANT DELETE TO [Dispetcher]
GO
USE [Avtopark]
GO
GRANT INSERT TO [Dispetcher]
GO
```

```
USE [Avtopark]
GO
GRANT SELECT TO [Dispatcher]
GO
USE [Avtopark]
GO
GRANT UPDATE TO [Dispatcher]
GO
```

#### 5.4 Предоставление запрашиваемых данных пользователю (генерация справок, отчетов, итогов)

В программе все справки и отчеты можно получить во вкладке «Запросы и справки» (см. рис. 13).

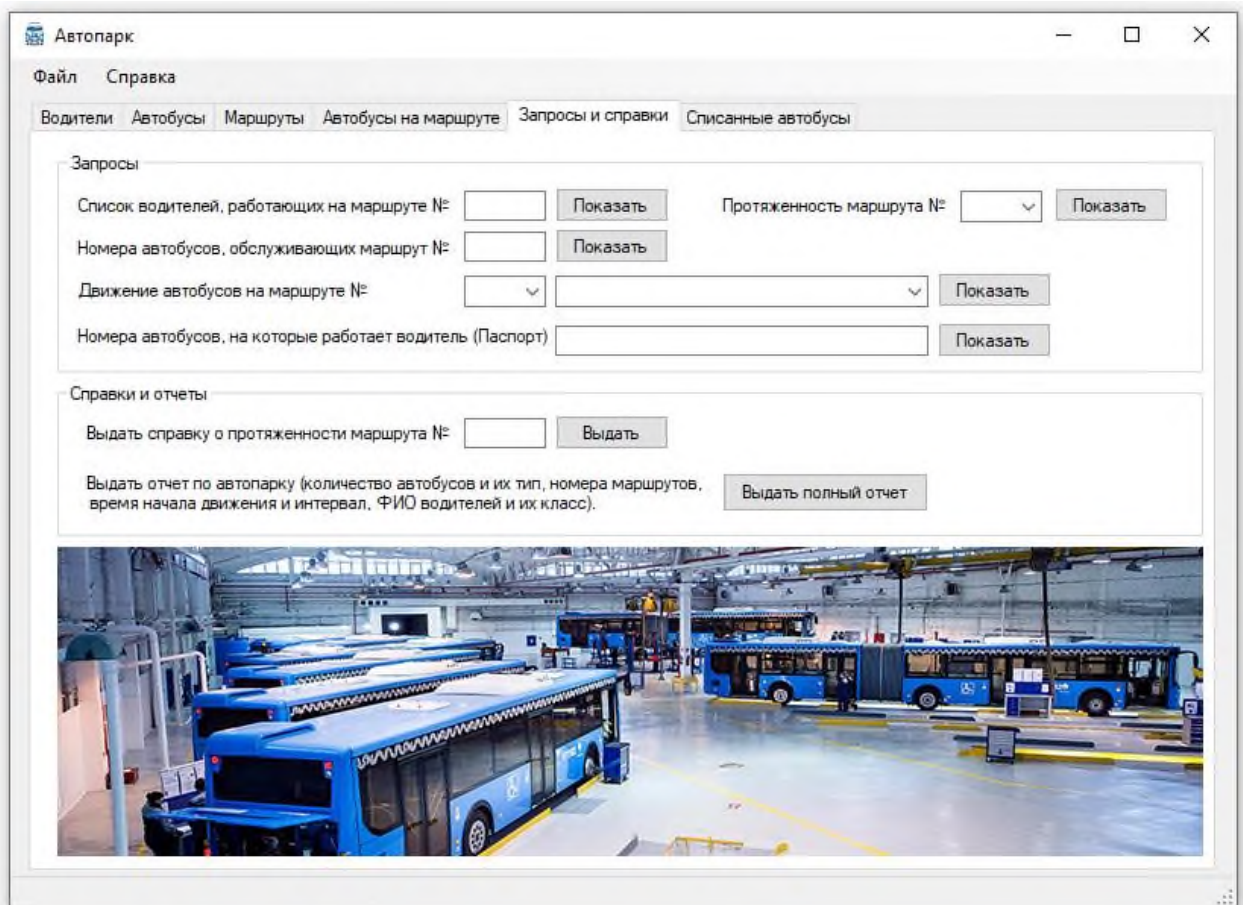


Рисунок 13 – Вкладка «Запросы и отчеты»

В разделе «Запросы» собраны все короткие запросы, которые выводятся на экран в режиме диалогового окна и показывают результат запроса (см. рис. 14).



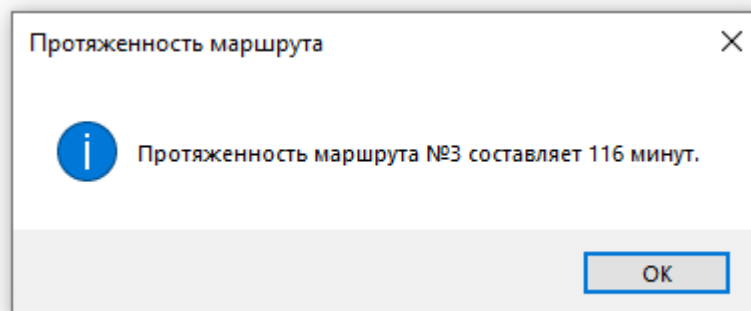


Рисунок 14 – Результат короткого запроса

Если текстовое поле «№» имеет значение INT (целое), то вызывается новое диалоговое окно, выполняется запрос:

```
SELECT [Протяженность(мин)]  
FROM [Маршруты]  
WHERE [Маршрут] = @Маршрут
```

где @Маршрут принимает значение поля «№»,

все нужные значения получаются и встроенными средствами языка C# выводятся на диалоговое окно.

Если текстовое поле «№» имеет значение «всех» (см. рис. 15), то вызывается новое окно, выполняется запрос:

```
SELECT [Маршрут], [Протяженность(мин)]  
FROM [Маршруты]
```

создается контейнер список массивов строк `List<string[]> data`  
контейнер заполняется возвращенными значениями из таблиц SQL  
потом выводятся все данные из контейнера в `dataGridView`.

Запросы возвращающие таблицы, выводятся в отдельном окне. (см. рис. 15, 16). Окно работает в отдельном потоке, что позволяет, не закрывая результат запроса, продолжать работу в основной окне программы, что несомненно является большим плюсом.

Запросы

Список водителей, работающих на маршруте №

Протяженность маршрута №

Номера автобусов, обслуживающих маршрут №

Движение автобусов на маршруте №

Номера автобусов, на которые работает водитель (Паспорт)

Рисунок 15 – Раздел «Запросы»

Начало и окончание движений автобусов на всех маршрутах

	Маршрут	Время_начала_дв	Время_окончания
▶	1	06:00:00	00:00:00
	2	08:00:00	23:00:00
	3	08:00:00	23:30:00
	4	06:00:00	23:00:00
	5	08:00:00	23:30:00
	6	06:00:00	23:00:00
	7	08:00:00	23:00:00
	8	06:00:00	23:30:00
	9	06:00:00	00:00:00
*			

Рисунок 16 – Раздел «Запросы»

В разделе «Справки и отчеты» собраны все документы на печать, которые генерируются в программе (см. рис. 13).

Для получения справки о протяженности конкретного маршрута, необходимо заполнить соответствующее поле и нажать кнопку «Выдать» (см. рис. 17).

Справки и отчеты

Выдать справку о протяженности маршрута №

Выдать отчет по автопарку (количество автобусов и их тип, номера маршрутов, время начала движения и интервал, ФИО водителей и их класс).

Рисунок 17 – Раздел «Справки и отчеты»

В результате открывается новый документ Microsoft Office Word, заполняется и форматируется документ (см. рис. 18).

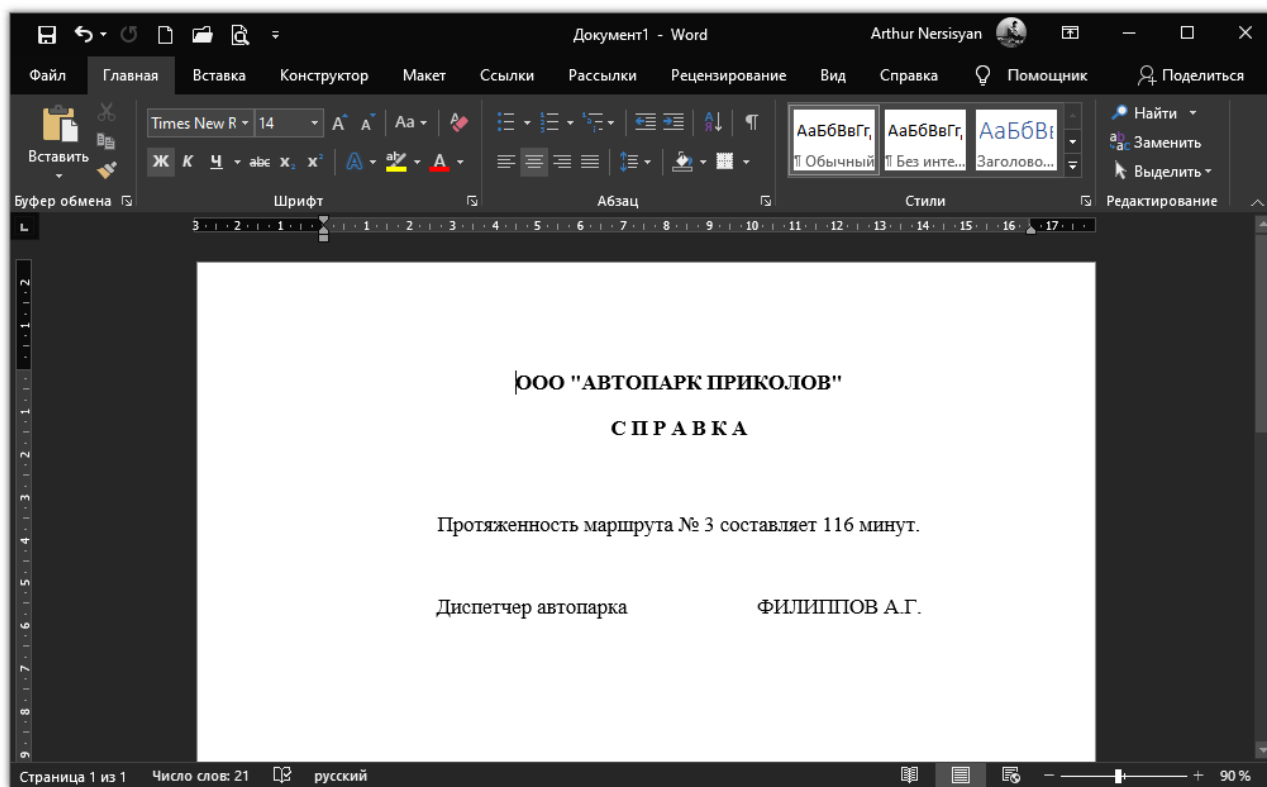


Рисунок 18 – Раздел «Справки о протяжённости маршрута»

Создание MS Word документов происходит в несколько этапов. Сначала выполняется запрос в базу, для получения нужных данных:

```
SELECT [Протяженность(мин)]
FROM [Маршруты]
WHERE [Маршрут] = @Маршрут
```

где @Маршрут принимает значение поля «№». Затем открывается MS Word, создается новый документ и включается отображение окна MS Word.

```
Word._Application oWord;  
Word._Document oDoc;  
oWord = new Word.Application();  
oWord.Visible = true;
```

Далее начинается передача текста в MS Word и его форматирование (в данном случае: жирный, , интервал после строки бпт, выровнен по центру, перевод строки).

```
Word.Paragraph oPara1;  
oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);  
oPara1.Range.Text = "ООО \"АВТОПАРК ПРИКОЛОВ\"";  
oPara1.Range.Font.Bold = 1;  
oPara1.Format.SpaceAfter = 6;  
oPara1.Range.ParagraphFormat.Alignment =  
Word.WdParagraphAlignment.wdAlignParagraphCenter;  
oPara1.Range.InsertParagraphAfter();
```

Аналогичными средствами строится остальная часть документа.

В разделе «Справки и отчеты» есть возможность генерации полного отчета по автопарку (количество автобусов и их тип, номера маршрутов, время начала движения и интервал, ФИО водителей и их класс). До генерации полного отчета выполняются множество запросов по базе, все данные нужные данные собираются, потом создается документ.

## 6. ОРГАНИЗАЦИЯ ЗАЩИТЫ БД

### 6.1 Описание ограничений целостности для каждого информационного объекта.

Контроль достоверности данных осуществляется с помощью ограничений целостности. Для ограничения полей данных используются следующие виды ограничений:

- Тип и формат поля. Тип поля определяет допустимые для данного поля символы, а иногда и более жесткие ограничения на допустимые значения (как, например, для полей типа дата или логическое). В случае несоответствия, выводится диалоговое окно с ошибкой.

- Задание диапазона значений. Используется для числовых полей. Признак непустого поля, поля, состоящей только из пробелов, табуляций и null-значений. Характеризует недопустимость пустого значения поля в БД.

- На уровне программы выполняется проверка правильности введенных данных, попытки ввести строку в поле, предназначенное для целых значений, сопровождаются выводом ошибок в диалоговом окне с указанием на проблему и методы его решения.

- Задание домена. Поле может принимать значение из заданного множества. Например, нельзя добавить автобус в базу с типом, который не определен в базе.

- Специфическим ограничением на значение поля является признак его уникальности. Это ограничение проверяет допустимость значения данного поля, но при этом просматривается вся таблица (файл). При попытке добавить нового водителя в базу выдается ошибка, если водитель с таким паспортом уже есть базе.

- Очень важным ограничением целостности являются функциональные зависимости. Внешние ключи (*Foreign Key*) являются ограничением для поддержания логической целостности БД, гарантирующие непротиворечивость информации. Наличие внешних ключей не позволит пользователю удалить

критически важные элементы базы данных. Например, до того, как списать старый автобус, нужно снять его с маршрута, тоже самое справедливо для водителей – прежде чем уволить водителя, нужно снять его со смены.

Ограничение **CHECK** ограничивает данные, которые пользователь может ввести в определенную колонку указанными значениями. Следующий пример добавляет ограничение, чтобы гарантировать, что автобус в таблице «Водят» (таблица смен водителей) соответствует зарегистрированному в базе данных автобусу в таблице «Автобусы»:

Создание внешних ключей для таблицы *Водят*:

```
ALTER TABLE [dbo].[Водят]
WITH CHECK ADD
CONSTRAINT [FK_Водят_Автобусы]
FOREIGN KEY([Автобус])
REFERENCES [dbo].[Автобусы] ([Автобус])
GO

ALTER TABLE [dbo].[Водят]
CHECK CONSTRAINT [FK_Водят_Автобусы]
GO
```

```
ALTER TABLE [dbo].[Водят]
WITH CHECK ADD
CONSTRAINT [FK_Водят_Водители]
FOREIGN KEY([Паспорт])
REFERENCES [dbo].[Водители] ([Паспорт])
GO

ALTER TABLE [dbo].[Водят]
CHECK CONSTRAINT [FK_Водят_Водители]
GO
```

```
ALTER TABLE [dbo].[Водят]
WITH CHECK ADD
```

```

CONSTRAINT [FK_Водят_Маршруты]
FOREIGN KEY([Маршрут])
REFERENCES [dbo].[Маршруты] ([Маршрут])
GO

ALTER TABLE [dbo].[Водят]
CHECK CONSTRAINT [FK_Водят_Маршруты]
GO

```

Создание внешних ключей для таблицы *Водители*:

```

ALTER TABLE [dbo].[Водители]
WITH CHECK ADD
CONSTRAINT [FK_Водители_Оклад]
FOREIGN KEY([Класс])
REFERENCES [dbo].[Оклад] ([Класс])
GO

ALTER TABLE [dbo].[Водители]
CHECK CONSTRAINT [FK_Водители_Оклад]
GO

```

Создание внешних ключей для таблицы *Автобусы*:

```

ALTER TABLE [dbo].[Автобусы]
WITH CHECK ADD
CONSTRAINT [FK_Автобусы_Тип_вместимость]
FOREIGN KEY([Тип])
REFERENCES [dbo].[Тип_вместимость] ([Тип])
GO

ALTER TABLE [dbo].[Автобусы]
CHECK CONSTRAINT [FK_Автобусы_Тип_вместимость]
GO

```

## **6.2 Рекомендуемые средства физической защиты (виды резервного копирования и периодичность проведения резервного копирования)**

Резервное копирование баз данных является самым простым и дешевым средством обеспечения сохранности данных. Резервная копия базы данных представляет собой копию данных, структур и объектов безопасности, содержащихся в базе данных. База данных резервируется по своему графику в зависимости от количества выполняемых за день транзакций записи. Для минимизации потерь при сбое базы данных выполняется резервное копирование базы данных. Чтобы убедиться, что резервная копия работоспособна, проверяется его работа после операции восстановления.

В данной работе применяется стратегия полного резервирования, что является самой простой для понимания и реализации. В конце каждого рабочего дня просто запускается процедура полного резервирования базы данных. При этом не нужно выполнять отдельное резервирование журналов и не требуется использовать дополнительные параметры. Управление файлами в таком режиме резервирования также не требует особого внимания, так как речь идет о единственном файле полной резервной копии. Восстановление из полной резервной копии тоже очень простое: необходимо просто восстановление из единственного файла. Использование полных резервных копий – хороший выбор для такого маленького проекта.

Когда SQL Server осуществляет полное резервирование базы данных, сначала выполняется сохранение на диск всех экстендов (экстент представляет собой восемь идущих последовательно страниц, размер каждой составляет 8 Кбайт). Затем SQL Server резервирует журнал транзакций, чтобы все изменения базы данных, которые могли произойти за время резервирования, также были сохранены в файле полной резервной копии.

Когда используется только полное резервирование, то в случае краха системы часть данных может быть потеряна – в первую очередь изменения, выполненные с момента последнего резервирования.



Для выполнения полного резервирования базы данных выполняется следующий код:

```
string name = "Avtopark-BackUP";
string path = @"C:\SQLdata\BACKUPS\";

// если каталог не существует то создать
DirectoryInfo dirInfo = new DirectoryInfo(path);
if (!dirInfo.Exists)
{
    Directory.CreateDirectory(path);
}

// "собираем" имя для резервной копии
var backupFileName = String.Format("{0}{1}-{2}.bak",
    path, name, DateTime.Now.ToString("yyyy-MM-dd-hh-mm-ss"));

// "собираем" запрос
var query = String.Format("BACKUP DATABASE [Avtopark] TO DISK
= '{0}'", backupFileName);

// выполняем запрос
SqlCommand command = new SqlCommand(query, sqlConnection);
command.ExecuteNonQuery();
```

В итоге запрос на создание резервной копии выглядит так

```
BACKUP DATABASE [Avtopark]
TO DISK =
    'C:\SQLdata\BACKUPS\Avtopark-BackUP-yyyy-MM-dd-hh-mm-ss.bak'
```

где **yyyy-MM-dd-hh-mm-ss** текущая дата и время на компьютере.

Для резервного копирования базы данных в контекстном меню «Файл» нужно выбрать пункт «Выполнить резервное копирование» (см. рис. 19).

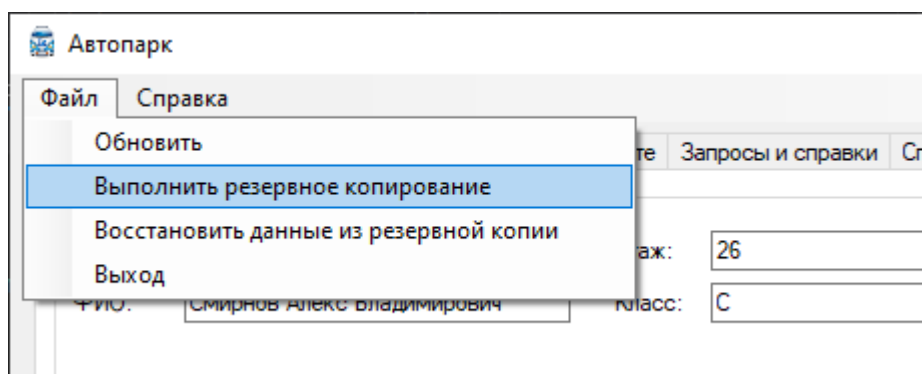


Рисунок 19 – Контекстное меню «Файл»

В случае успешного выполнения резервного копирования базы данных выводится соответствующее сообщение в диалоговом окне (см. рис. 20).

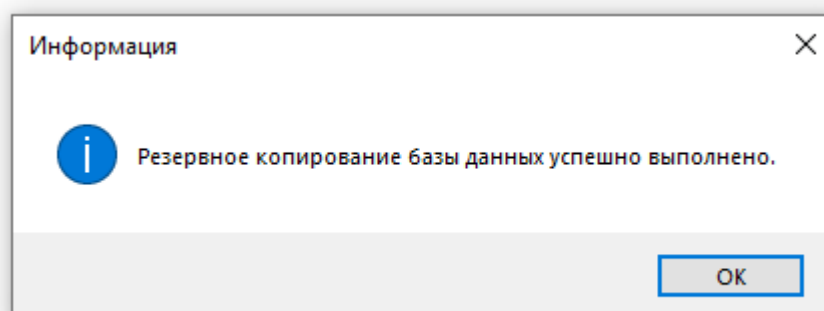


Рисунок 20 – Диалоговое окно «Информация» об успешном выполнении резервного копирования

Для восстановления данных в контекстном меню «Файл» нужно выбрать пункт «Восстановление данных из резервной копии» (см. рис. 19). Откроется диалоговое окно для выбора файла резервной копии. На основе данных выбранного файла в программа получает имя и путь к файлу резервной копии.

Для восстановления базы данных выполняется следующая команда:

```
RESTORE DATABASE [Avtopark]
FROM DISK = "BACKUP FILE PATH"
WITH REPLACE;
```

где “BACKUP FILE PATH” генерируется в ходе выполнение программы, результат помещается в запрос после определения имени и пути к файлу резервной копии. WITH REPLACE означает перезапись любого существующего файла с тем же именем.

### 6.3 Описание процедуры подтверждения личности

Программа запускается с окна авторизации (см. рис. 21)

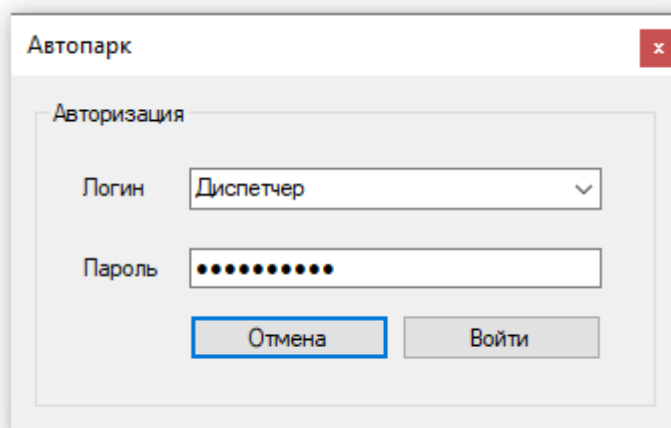


Рисунок 21 – Окно авторизации

Данной программой по взаимодействию с базой данных будет управлять только один пользователь – диспетчер автопарка. Для всех запросов, указанных в задании на курсовой проект, которые должны быть доступны диспетчеру автопарка, есть соответствующая кнопка в интерфейсе приложения. Операции, которые не должны быть доступны диспетчеру, например удаление таблицы из базы, или всей базы, не могут быть выполнены, потому что для таких действий нет кнопок в программе.

Пользователю Диспетчер доступны все операции по манипуляции данным во всех таблицах. Это целесообразно и безопасно, так как установленные ограничения логической целостности БД обеспечивают достаточный уровень защиты. Кроме того, диспетчеру автопарка должны быть доступны операции по добавлению, удалению, обновлению и поиску данных, ибо его работа в этом и заключается.

Программа запускается с окна входа, проверяется правильность логина и пароля, если все верно, запускается основное окно, а форма авторизации закрывается автоматически.

Диспетчер (единственный пользователь БД) введет свои учетные данные. В программе генерируется строка подключения к базе данных на основе учетных данных, полученных из текстовых полей программы.

Строка подключения к базе данных выглядит так:

```
Data Source=(LocalDB)\MSSQLLocalDB;  
AttachDbFilename=%Avtopark%\Avtopark.mdf;  
User Id=Dispatcher;  
Password=dispatcher;  
Integrated Security=True
```

При условии правильно введенных учетных данных устанавливается соединение с базой данных и загружается основное окно программы, в противном случае выводится ошибка о неправильно введенных учетных данных (см. рис. 22).

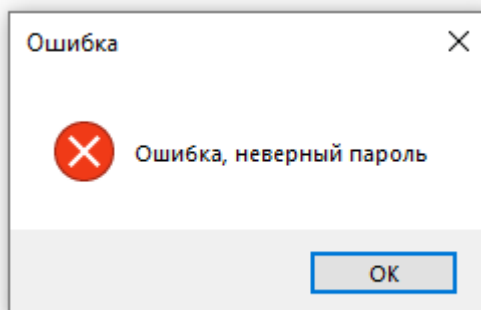


Рисунок 22 – Диалоговое окно о неправильных учетных данных

Количество групп пользователей: 1;

Группа №1 – Диспетчер автопарка.

Права доступа к каждому информационному объекту задаются согласно таблице 7:

*Таблица 7. Назначение прав доступа к информационным объектам*

<b>Объект</b>	<b>Группа №1</b>
Водители	SIUD
Автобусы	SIUD
Маршруты	SIUD
Водят	SIUD

Используемые сокращения:

S – чтение данных (SELECT);                      U – модификация данных (UPDATE);  
 I – добавление данных (INSERT);                D – удаление данных (DELETE).

#### **6.4 Применение протоколов шифрования для защиты базы данных**

Шифрование представляет собой способ скртия данных с помощью ключа или пароля. Это делает данные бесполезными без соответствующего ключа или пароля для дешифрования. Шифрование не решает проблемы управления доступом. Однако оно повышает защиту за счет ограничения потери данных даже при обходе системы управления доступом. Например, если компьютер, на котором установлена база данных, был настроен неправильно, и злоумышленник смог получить конфиденциальные данные, то украденная информация будет бесполезна, если она была предварительно зашифрована.

В SQL Server можно шифровать соединения, данные и хранимые процедуры, и позволяет использовать алгоритмы шифрования вплоть до AES с 256-разрядным ключом.

SQL Server шифрует данные, используя иерархическую структуру средств шифрования и управления ключами. На каждом уровне данные низшего уровня шифруются на основе комбинации сертификатов, асимметричных ключей и симметричных ключей. Асимметричные и симметричные ключи можно хранить вне модуля расширенного управления ключами SQL Server.

Шифрование может потребоваться, если пользователи получают доступ к данным через открытую сеть. Использование шифрования включает политику управления паролями, ключами и сертификатами.

Несмотря на то, что шифрование является полезным средством обеспечения безопасности, в рамках данной курсовой работы шифрование не применено.

## ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта была спроектирована база данных и построена программа, обеспечивающая взаимодействие с базой данных в режиме диалога, для диспетчера автобусного парка. В БД хранятся сведения о водителях, маршрутах автобусов и их характеристиках.

База данных функционирует в среде реляционной СУБД Microsoft SQL Server 2019, используемый язык запросов — Transact-SQL. Все взаимодействия с базой данных осуществляются с помощью графической оболочки программы в режиме диалогового окна Windows Forms, написанной на языке программирования C# (Microsoft .NET Framework 4.7.2).

Проведен анализ предметной области, логическое проектирование, нормализация схем, физическое проектирование, назначение прав доступа, определения и реализация ограничений целостности и другие процедуры для корректного функционирования и защиты базы данных.

Решены следующие задачи:

- хранение информации о водителях, маршрутах, автобусах и сменах;
- выдача справки о протяженности маршрута и отчета по автопарку;
- предусмотрена возможность корректировки БД в случаях поступления на работу нового водителя, списывания старого автобуса, введения нового или изменения старого маршрута;
- разработана реляционная СУБД, содержащая элементы автоматизации и обработки данных.

Предусмотрена возможность выдачи следующих сведений:

- список водителей, работающих на определенном маршруте;
- номера автобусов, обслуживающих данный маршрут
- когда начинается или заканчивается движение автобусов на всех или отдельных маршрутах;
- какова протяженность всех или определенных маршрутов автобусов;

- на каких автобусах работает водитель.

Предусмотрена возможность внесения следующих изменений:

- прием на работу нового водителя;
- списание старого автобуса;
- изменение протяженности маршрута.

В ходе выполнения данной курсовой работы были приобретены практические навыки работы и создания баз данных в среде Microsoft SQL Server 2019 и программ в среде Microsoft .NET Framework 4.7.2.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шехтман В. Е. Базы данных, SQL и все такое: курс лекций / В. Е. Шехтман; НФИ КемГУ. - Новокузнецк, 2006. – 195 с.
2. Базы данных: учеб. пособие для студ. высших учебн. заведений / А.В. Кузин, С.В. Левонисова. – 2 – е изд. стер. – М.: Издательский центр «Академия», 2008.
3. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. — СПб.: Питер, 2013. — 896 с.
4. Техническая документация по SQL Server. // Техническая документация Майкрософт для разработчиков. URL: <https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15> (дата обращения: 04.12.2020).
5. Документация по .NET. // Техническая документация Майкрософт для разработчиков. URL: <https://docs.microsoft.com/ru-ru/dotnet/> (дата обращения: 04.12.2020).
6. Документация по C# // Техническая документация Майкрософт для разработчиков. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 04.12.2020).
7. Документация по Visual Studio // Техническая документация Майкрософт для разработчиков. URL: <https://docs.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2019&preserve-view=true> (дата обращения: 04.12.2020).

## ПРИЛОЖЕНИЕ А. РУКОВОДСТВО ПОЛЗОВАТЕЛЯ БД

### 1. Краткое описание программы

Программа «Автопарк» предназначена для обеспечения взаимодействия с базой данных в режиме диалога, для диспетчера автобусного парка. В БД хранятся сведения о водителях, маршрутах автобусов и их характеристиках.

### 2. Системные требования

**Операционная система:** Windows 10 или более поздние версии

**Видеоадаптер:** DirectX версии не ниже 9 с драйвером WDDM 1.1.

**Дисплей:** не меньше 1024 x 768.

**Дополнительное ПО:** Microsoft .NET Framework версии 4.7.2

**Дополнительное ПО:** Microsoft Office Word 2019

### 3. Установка программы

Программа представляется в виде архивного файла «Avtopark.zip». Необходимо распаковать архив и запустить установочный файл «setup.exe» (см. рис. 23).

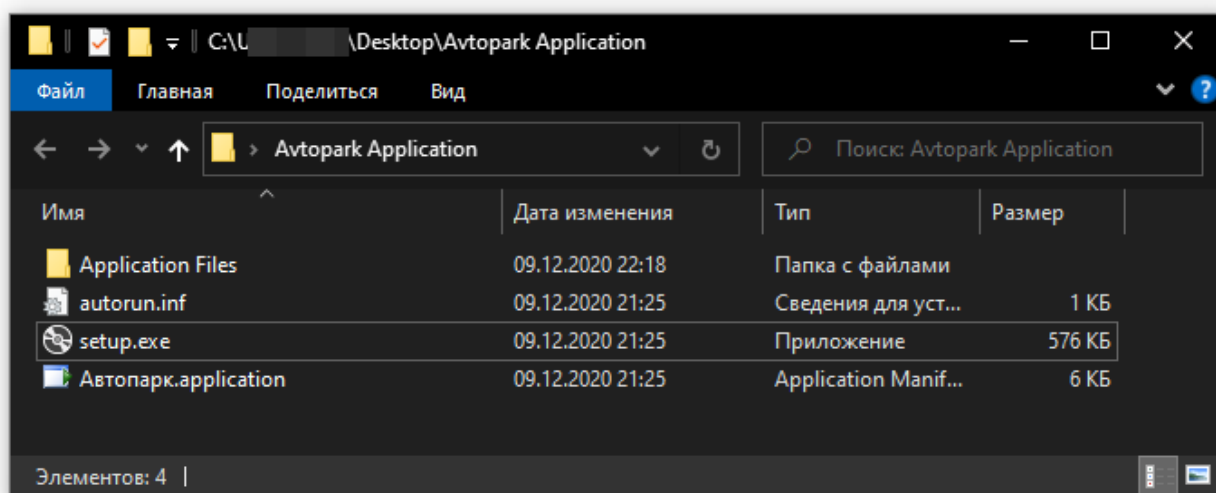


Рисунок 23 – файл «setup.exe»

После двойного нажатия левой кнопки мыши происходит запуск установочного файла (см. рис. 24).

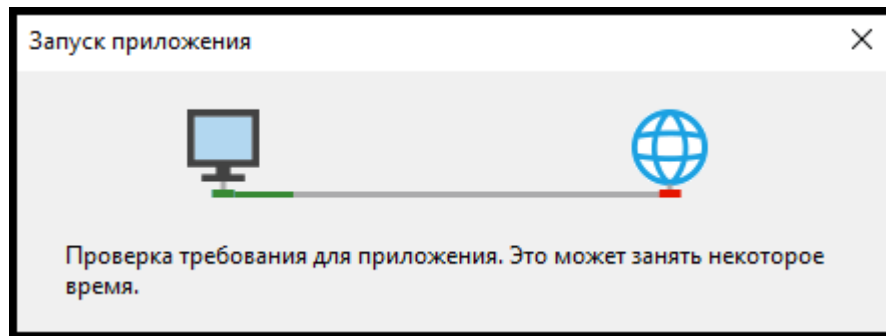


Рисунок 24 – запуск файла «setup.exe»

На экран выводится диалоговое окно «Установка приложения – Предупреждение о безопасности». Для подтверждения установки нажмите кнопку «Установить» (см. рис. 25).

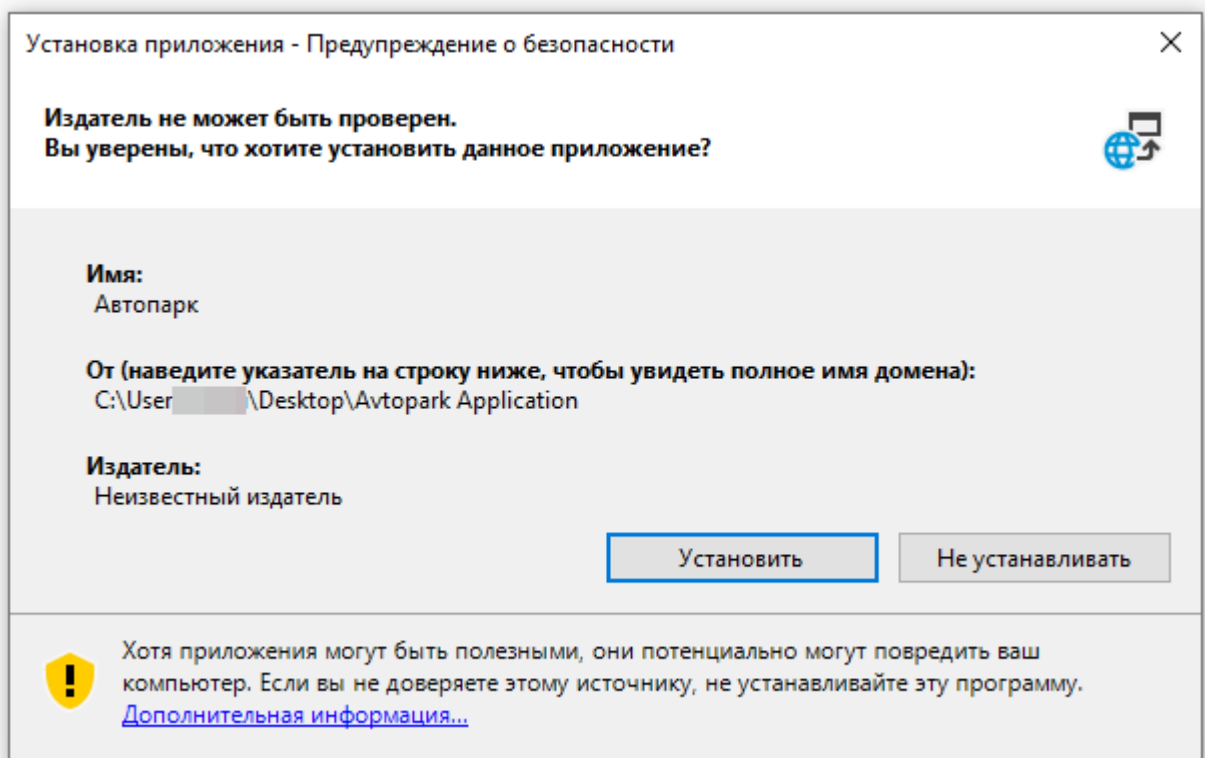


Рисунок 25 – Диалоговое окно «Установка приложения – Предупреждение о безопасности»

Начнется установка приложения, дождитесь его окончания (см. рис. 26).

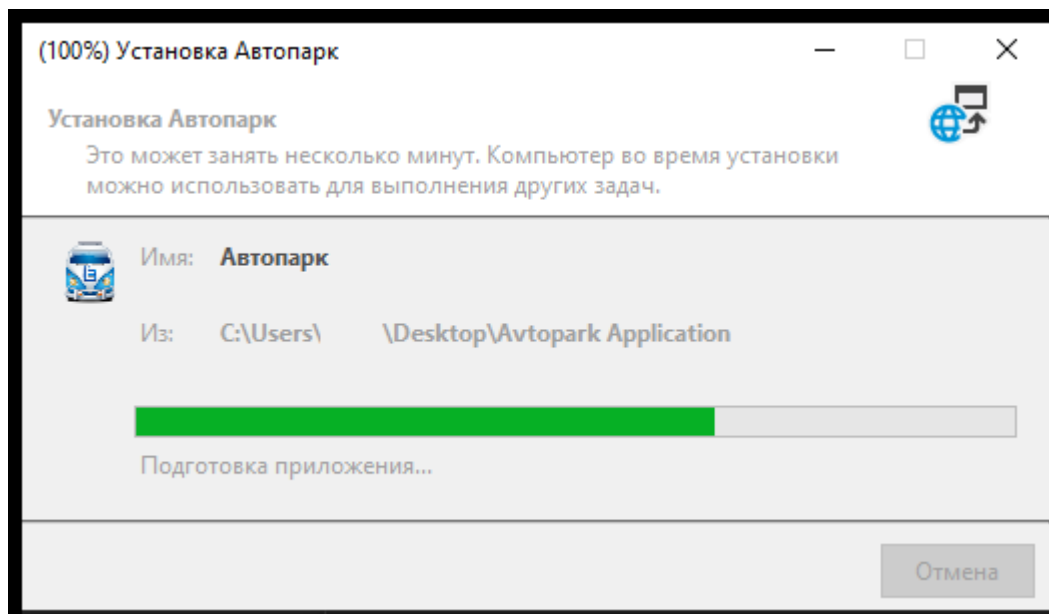


Рисунок 26 – Процесс установки приложения «Автопарк»

После окончания процесса установки программа запустится и выведет на экран форму авторизации (см. рис. 27). Кроме того, программа добавится к списку приложений и его можно будет найти в меню Пуск (см. рис. 28).

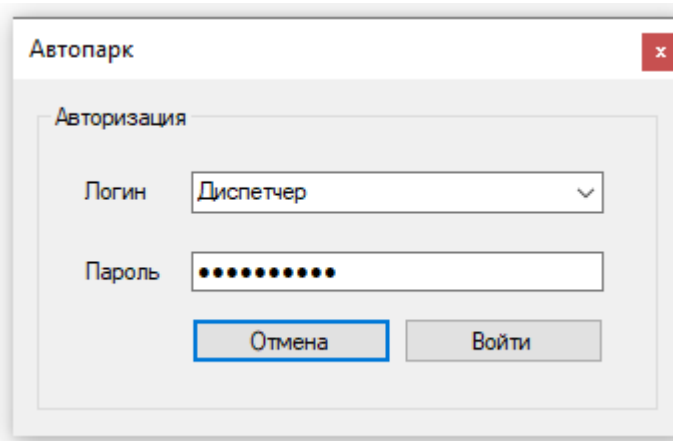


Рисунок 27 – Окно авторизации приложения «Автопарк»

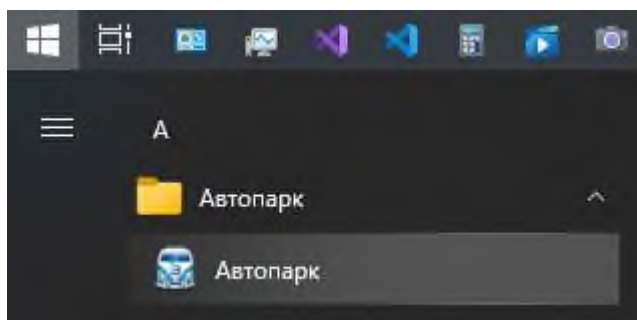


Рисунок 28 – Приложение «Автопарк» в меню Пуск

#### 4. Работа с программой

При открытии программы на экран выводится окно авторизации (см. рис. 28). Введите свои учетные данные нажмите кнопку «Войти». Логин и пароль по умолчанию представлены ниже, и они загружаются автоматически. В вы можете изменить логин и пароль, а также отключить автозаполнение в коде программы.

Логин: Диспетчер

Пароль: dispatcher

При правильно введенных учетных данных откроется основное окно программы (см. рис. 29).

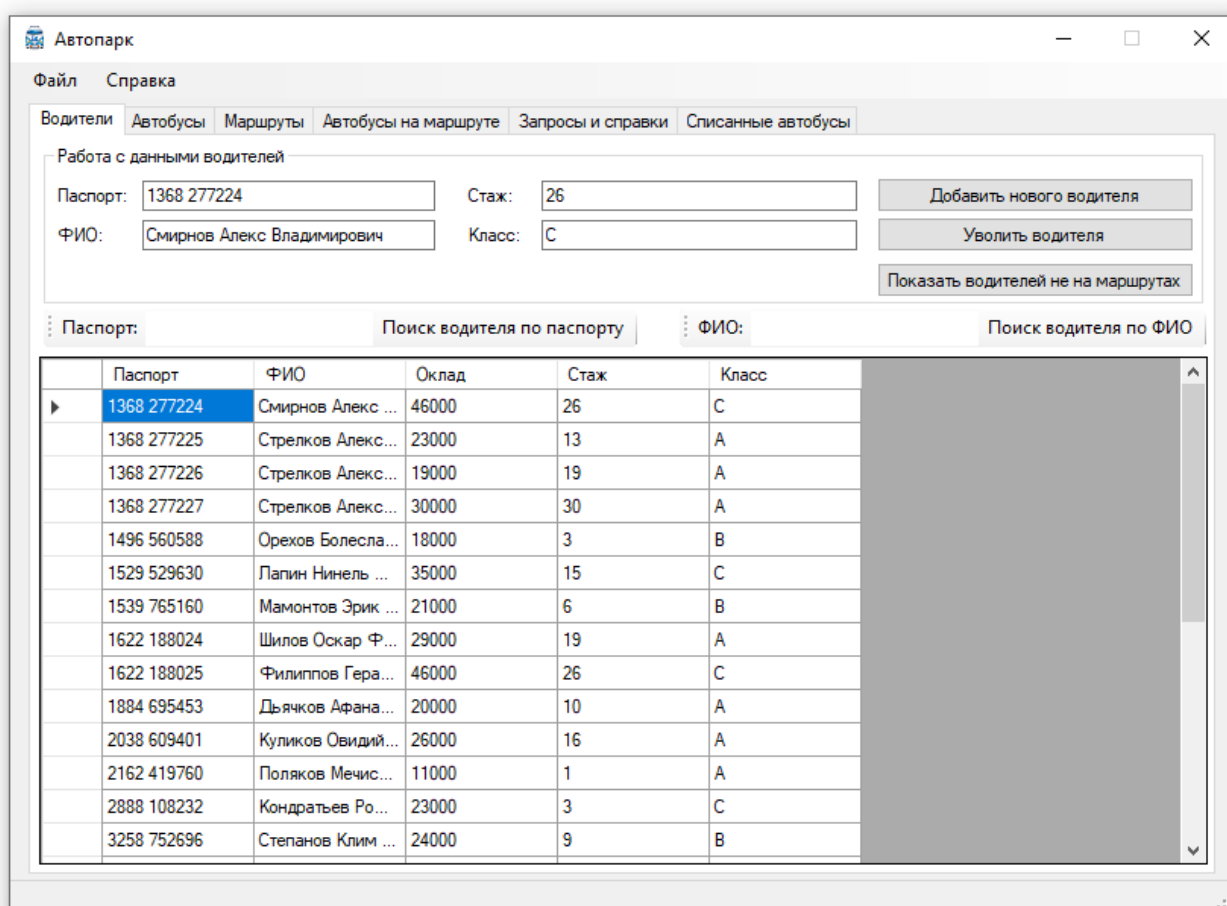


Рисунок 29 – Основное окно программы

Главное окно программы «Автопарк» имеет 6 вкладок: «Водители», «Автобусы», «Маршруты», «Автобусы на маршруте», «Запросы и справки» и «Списанные автобусы». Подробно рассмотрим функционал каждой из них.

## 4.1 Вкладка «Водители»

В области «Работа с данными водителей» есть три кнопки. Рассмотрим каждый из них.

Кнопка «Добавить нового водителя» предназначен для добавления данных водителя в базу. Прежде чем нажимать на кнопку нужно заполнить поля «Паспорт», «ФИО», «Стаж» и «Класс». Если данные введены корректно, то при нажатии кнопки «Добавить нового водителя» операция успешно выполняется и в строке состояния программы появится соответствующее сообщение (см.рис. 30).

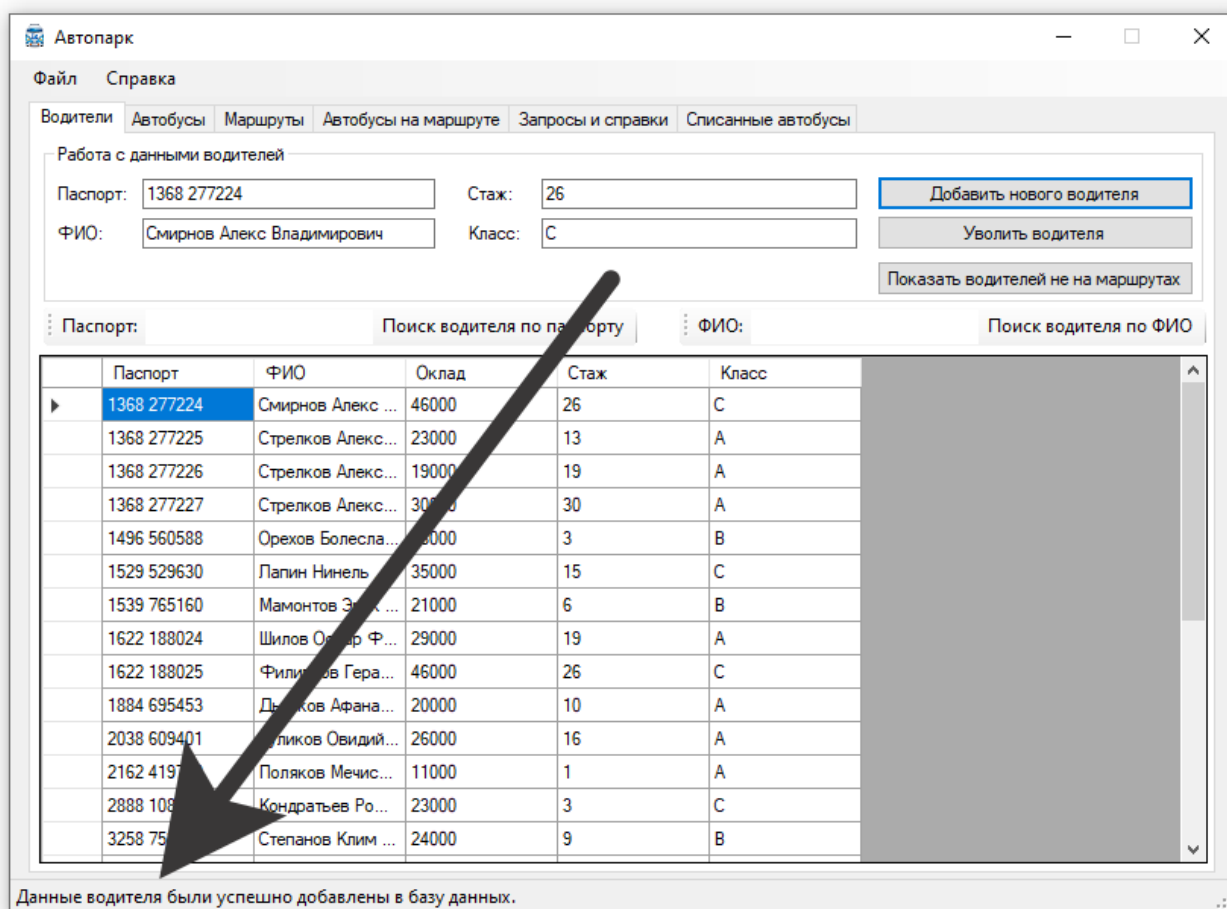


Рисунок 30 – Сообщение об успешном выполнении запроса добавления водителя в строке состояния программы

Если пользователь попытается добавить водителя, который уже есть в базе данных (проверяется по номеру паспорта), то программа выдаст ошибку в новом диалоговом окне с описание проблемы (см. рис. 31). Также текст ошибки будет дублироваться в строке состояния.

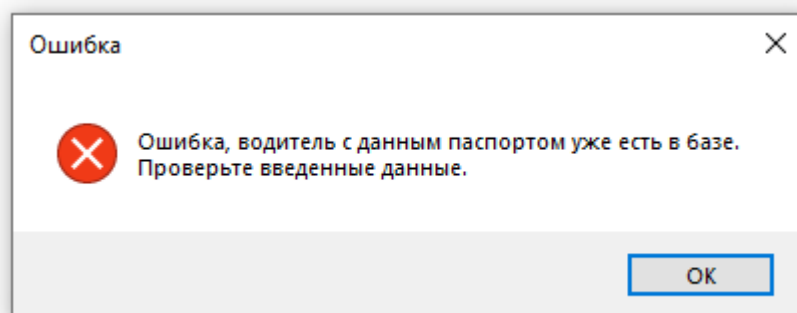


Рисунок 31 – Диалоговое окно ошибки

Кнопка «Уволить водителя» предназначен для увольнения водителя. Для увольнения водителя необходимо заполнить текстовое поле «Паспорт» и нажать кнопку «Уволить водителя». Программа выведет диалоговое окно для подтверждения удаления (см. рис. 32).

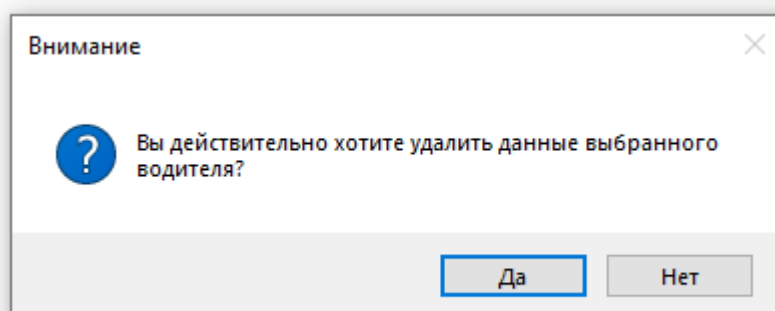


Рисунок 32 – Диалоговое окно подтверждения увольнения водителя

При подтверждении нажатием кнопки «Да», данные водителя удаляются из базы данных. В случае, если водитель закреплен за автобусом и назначен на смену, удаление не выполняется и выводится соответствующая ошибка в новом диалоговом окне с возможным решением появившийся проблемы (см. рис. 33).

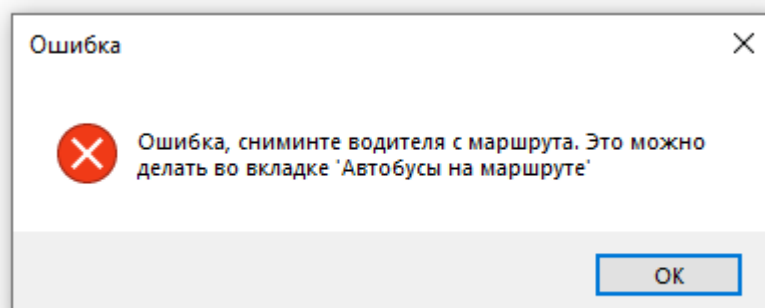
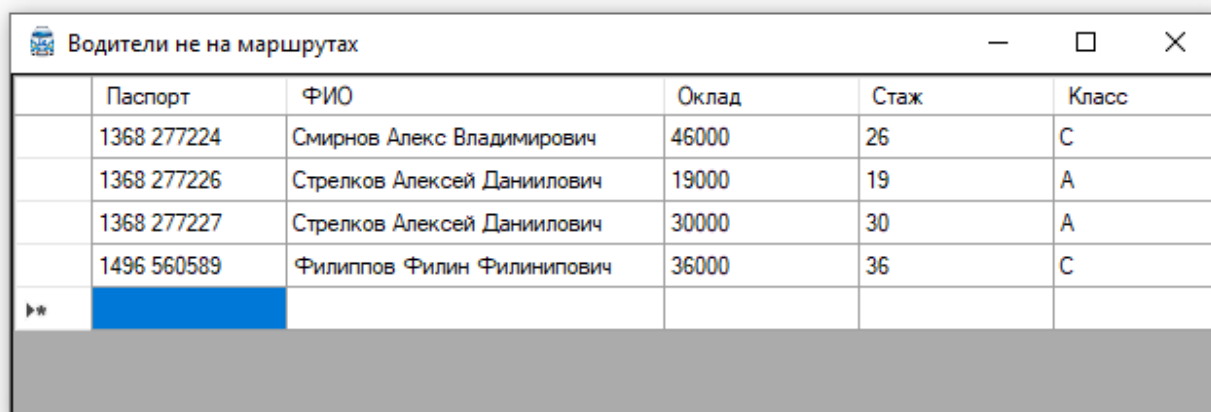


Рисунок 33 – Диалоговое окно ошибки при попытке удаления данных водителя

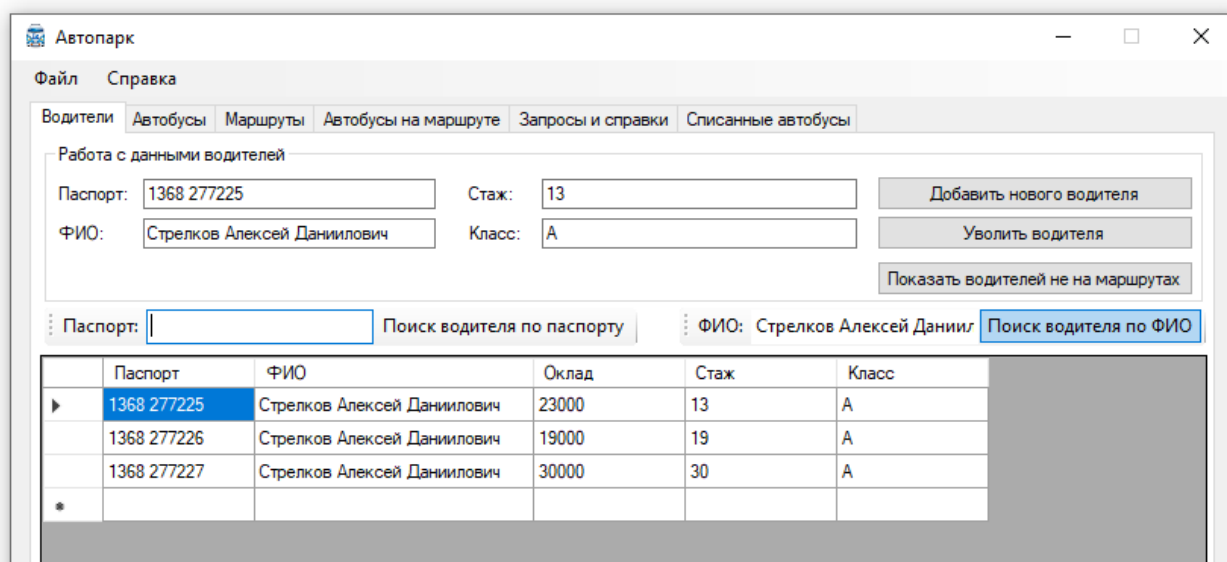
Кнопка «Показать водителей не на маршрутах», в новом диалоговом окне в виде таблицы выводит на экран список водителей, которым не назначены смены (см. рис. 34).



	Паспорт	ФИО	Оклад	Стаж	Класс
	1368 277224	Смирнов Алекс Владимирович	46000	26	С
	1368 277226	Стрелков Алексей Даниилович	19000	19	А
	1368 277227	Стрелков Алексей Даниилович	30000	30	А
	1496 560589	Филиппов Филин Филинипович	36000	36	С
▶*					

Рисунок 34 – Диалоговое окно «Водители не на маршрутах»

Кнопки «Поиск водителя по паспорту» и «Поиск водителя по ФИО» предназначены для поиска водителей по базе. Результаты поиска показываются на таблице в основной окне программы (см. рис. 35).



Автопарк

Файл Справка

Водители Автобусы Маршруты Автобусы на маршруте Запросы и справки Списанные автобусы

Работа с данными водителей

Паспорт: 1368 277225 Стаж: 13 Добавить нового водителя

ФИО: Стрелков Алексей Даниилович Класс: А Уволить водителя

Показать водителей не на маршрутах

Паспорт: Поиск водителя по паспорту ФИО: Стрелков Алексей Даниил Поиск водителя по ФИО

	Паспорт	ФИО	Оклад	Стаж	Класс
▶	1368 277225	Стрелков Алексей Даниилович	23000	13	А
	1368 277226	Стрелков Алексей Даниилович	19000	19	А
	1368 277227	Стрелков Алексей Даниилович	30000	30	А
*					

Рисунок 35 – Результат кнопок поиска

Для обновления таблицы нажмите Файл => Обновить. В нижней части вкладки «Водители» выводится непосредственно весь список водителей, номера паспортов, ФИО, оклад, стаж и класс работы.



## 4.2 Вкладка «Автобусы»

В области «Работа с данными автобусов» есть три кнопки. Рассмотрим каждый из них.

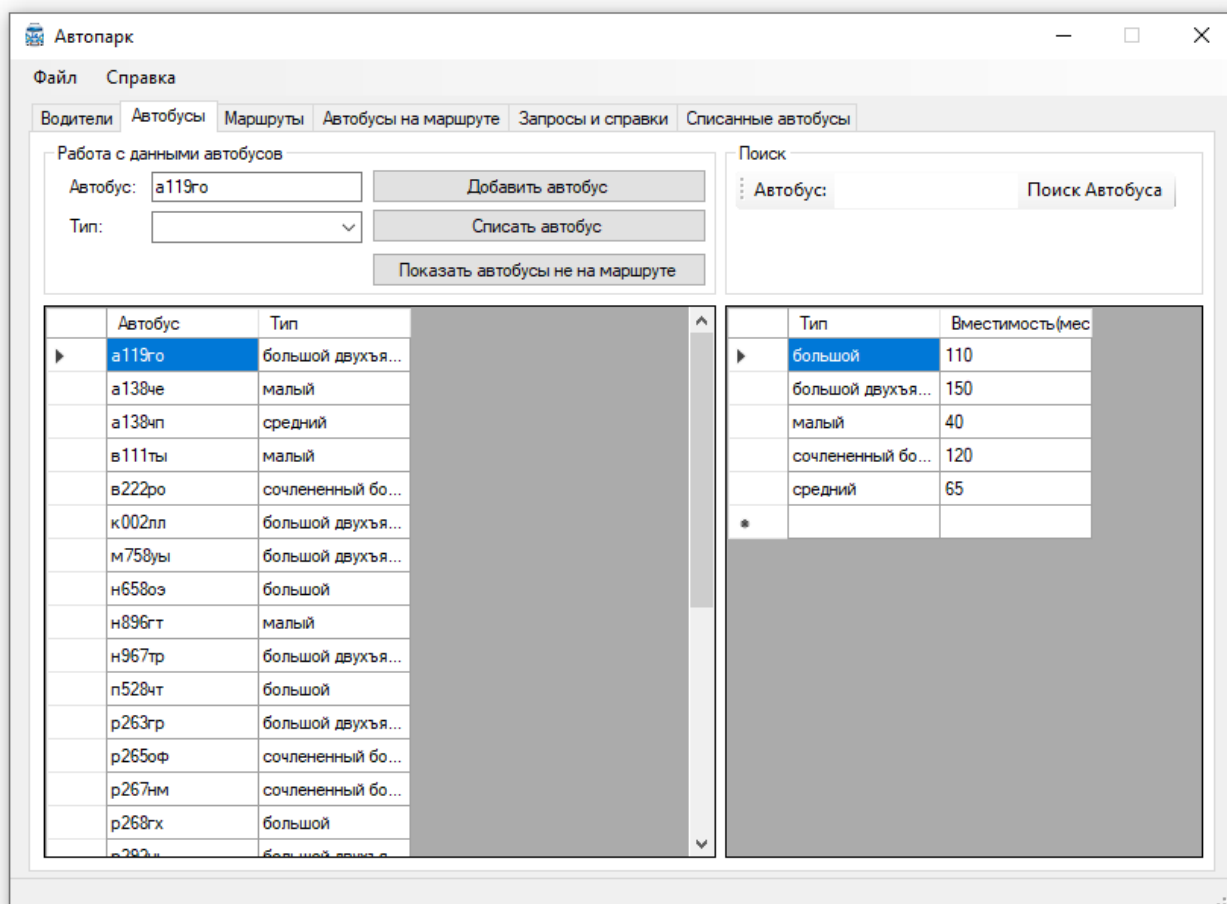


Рисунок 36 – Вкладка «Автобусы»

Кнопка «Добавить автобус» предназначена для добавления автобусов в базу данных. Прежде чем нажимать на кнопку нужно заполнить поля «Автобус» и «Тип». В поле «Тип» интегрирован выпадающий список (см. рис. 37).

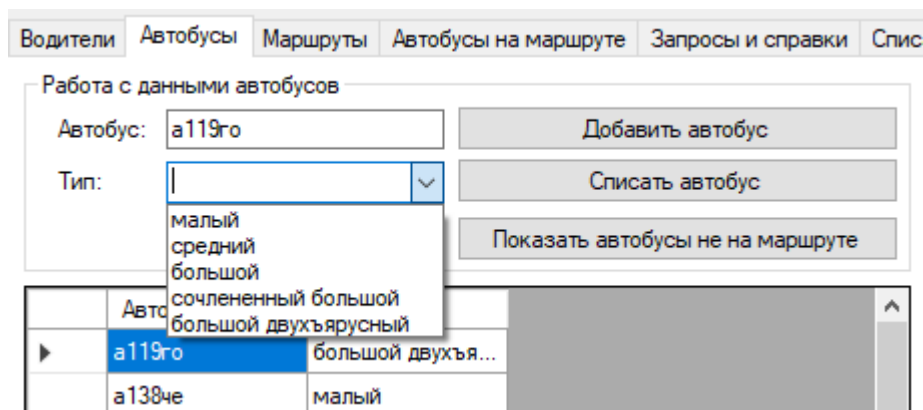


Рисунок 37 – Выпадающий список поля «Тип»

Значение поля «Тип» обязательно должен выбраться из выпадающего списка, значение поля «Автобус» не может повторяться, в противном случае программа выдаст ошибку (см. рис. 38). В случае успешного выполнения запроса на добавление данных автобуса в базу, в строке состояния программы появится соответствующее сообщение.

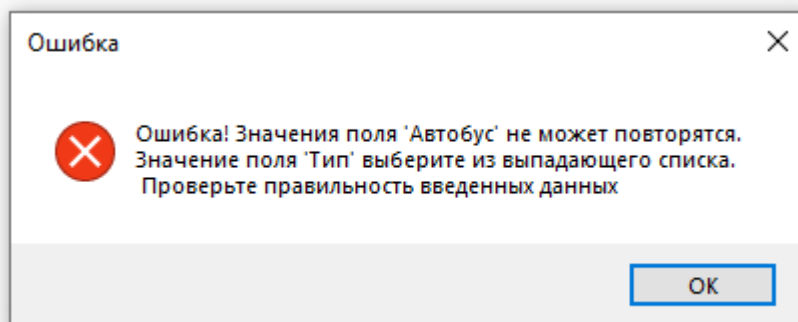


Рисунок 38 – Диалоговое окно ошибки

Кнопка «Списать автобус» предназначен для перемещения автобуса из таблицы Автобусов в таблицу списанных автобусов. Данная кнопка работает такой же логикой, как и кнопка «Уволить водителя», т.е. нельзя списать автобус если он находится на смене и за ним закреплен водитель. Чтобы списать автобус нужно в поле Автобус ввести номер автобуса и нажать на кнопку «Списать автобус». Если автобус все же был закреплен за водителем и находился на смене, то на экране появится диалоговое окно ошибки с описанием проблемы, а также с инструкцией по его решению (см. рис. 39).

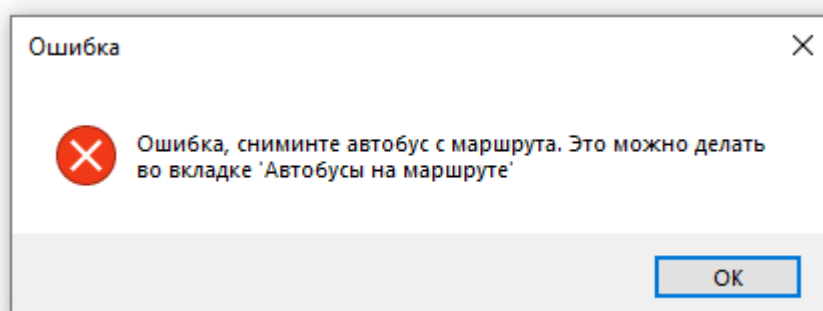
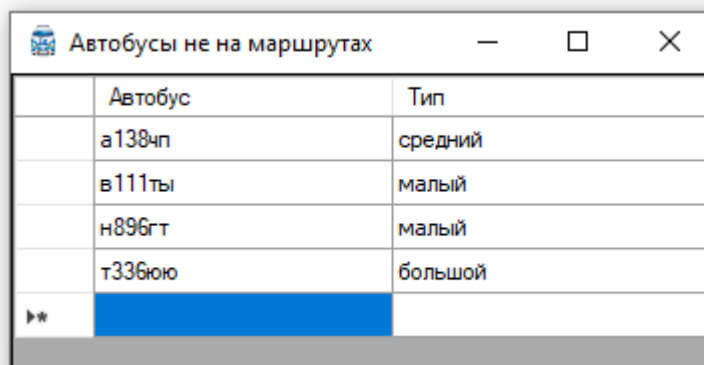


Рисунок 39 – Диалоговое окно ошибки

Кнопка Кнопка «Показать автобусы не на маршрутах», в новом диалоговом окне в виде таблицы выводит на экран список автобусов, которым не назначены водители и маршруты (см. рис. 40).



Автобус	Тип
a138чп	средний
в111ты	малый
н896гт	малый
т336юю	большой
»*	

Рисунок 40– Диалоговое окно «Автобусы не на маршрутах»

В области «Поиск» есть кнопка «Поиск автобуса», для поиска необходима заполнить поле искомым номером автобуса и нажать на кнопку. Результат отобразится на таблице в основной окне программы.

В левой нижней части вкладки «Автобусы» выводится весь список автобусов, а в правой части окна располагается таблица тип-вместимость.

### 4.3 Вкладка «Маршруты»

В области «Работа с данными маршрутов автобусов» есть 4 кнопки, рассмотрим каждый из них (см. рис. 41).

Кнопка «Изменить протяженность маршрута» изменяет протяженность маршрута. До нажатия на эту кнопку необходимо заполнить поля «Маршрут» и «Протяженность».

Кнопка «Удалить маршрут» позволяет удалить маршрут, номер которого введен в поле «Маршрут». Если на данном маршруте закреплены смены, при попытке удаления программа выдаст ошибку с описанием проблемы и с инструкцией по его устранению (см. рис. 42, 43). Если на маршруте не закреплены смены, то маршрут удалиться из базы данных и в строке состояния программы появится соответствующее сообщение.

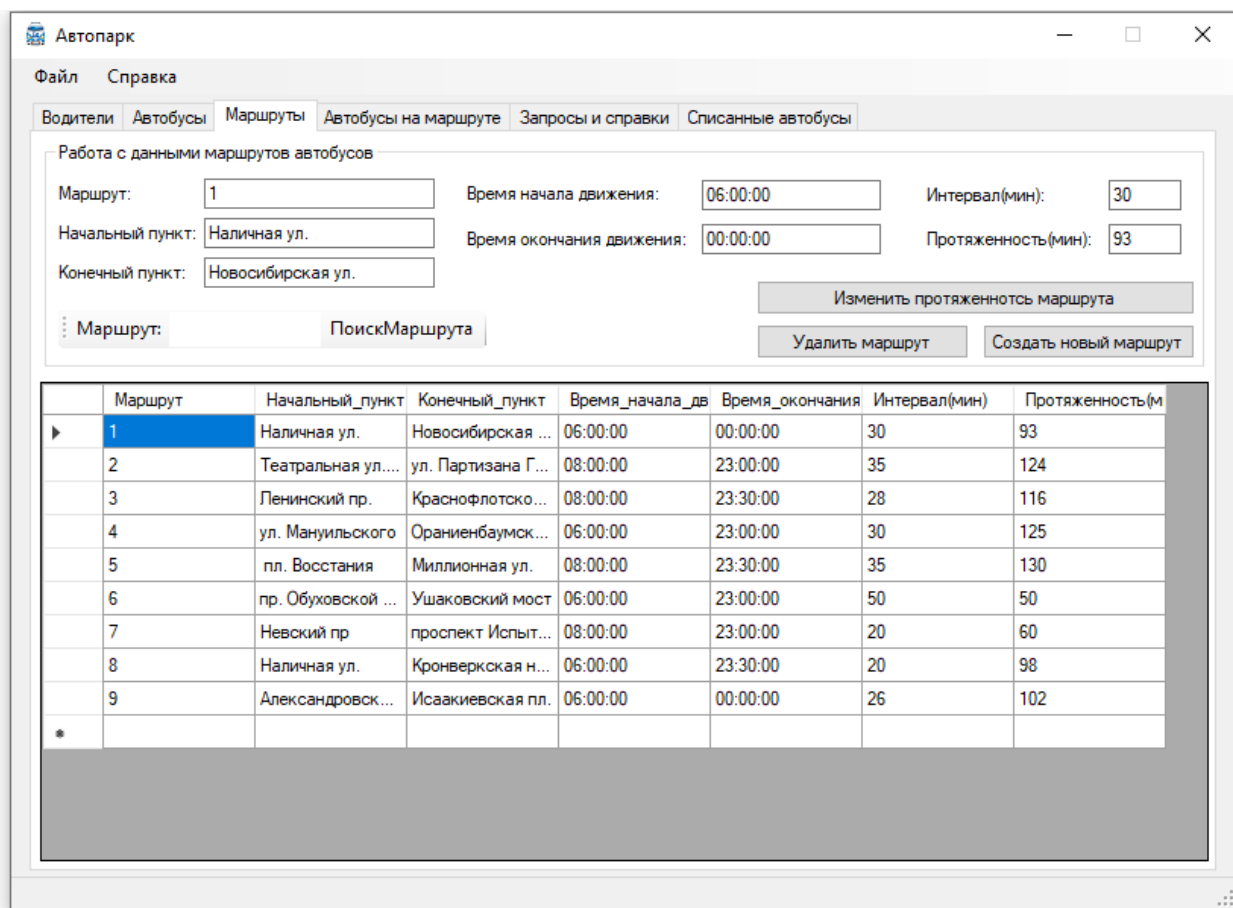


Рисунок 41– Вкладка «Маршруты»

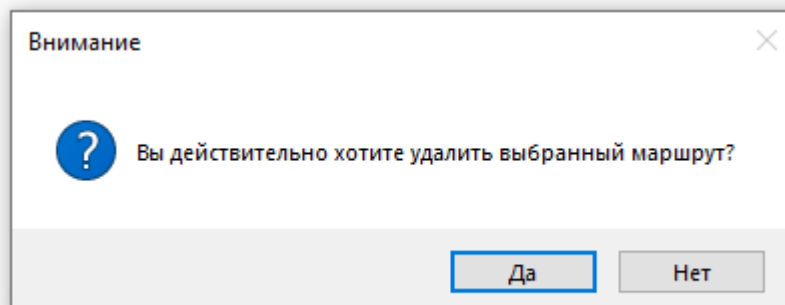


Рисунок 42– Диалоговое окно подтверждения удаления маршрута

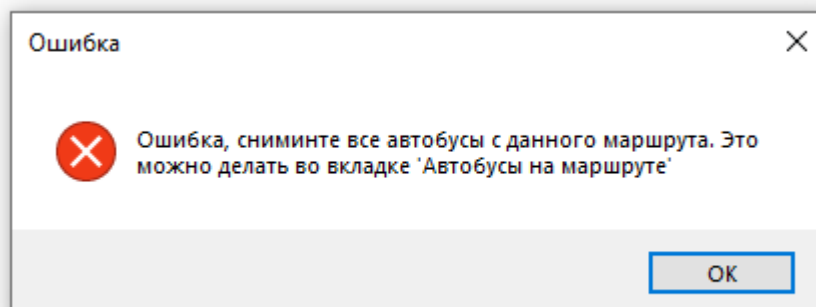


Рисунок 43– Диалоговое окно ошибки удаления маршрута

Кнопка «Создать новый маршрут» предназначен для создания новых маршрутов. Для создания нового маршрута необходима заполнить все поля в области работы с данными маршрутов, потом нажать на кнопку «Создать новый маршрут». В случае успешного выполнения запроса в строке состояния программы появится соответствующее сообщение, в противном случае появится диалоговое окно с описанием ошибки и инструкций по его устранению.

Кнопка поиск маршрута позволяет искать маршруты по номеру. Результат поиска отобразится в таблице в главной окне программы.

#### 4.4 Вкладка «Автобусы на маршруте»

В области «Работа с автобусами на маршруте» расположены 3 кнопки, рассмотрим их по очереди.

Кнопка «Снять водителя с маршрута» снимает водителя с маршрута. Перед нажатием кнопки необходимо заполнить поле «Паспорт» (см. рис. 44).

Автопарк

Файл Справка

Водители Автобусы Маршруты Автобусы на маршруте Запросы и справки Списанные автобусы

Работа с автобусами на маршруте

Паспорт:  Автобус:  Дата: 8 декабря 2020 г. ▾

ФИО:  Маршрут:

Снять водителя с маршрута Снять автобус с маршрута Новый автобус на маршруте

	id_Водителя ▲	Паспорт	Автобус	ФИО	Маршрут	Дата
	28	6502 104936	п528чт	Михеев Власий Ефимович	1	19.10.2020
	30	1539 765160	ф789би	Мамонтов Эрик Владиславович	3	18.10.2020
	31	3258 752696	р963де	Степанов Клим Созонович	3	18.10.2020
	33	6146 835286	с125но	Новиков Влас Протасьевич	5	18.10.2020
	34	2162 419760	а138че	Поляков Мечислав Русланович	5	18.10.2020
	35	1368 277225	ф236за	Стрелков Алексей Данилович	9	18.10.2020
	36	5622 188804	ф253др	Ширяев Алан Владиславович	6	18.10.2020
	37	4541 089924	р263гр	Рябов Владлен Георгиевич	7	18.10.2020
	38	1622 188024	р267нм	Шилов Оскар Феликсович	8	18.10.2020
	39	1884 695453	р268гх	Дьячков Афанасий Аристархович	9	18.10.2020
	40	8833 680121	у289ьф	Быков Рубен Егорович	1	18.10.2020
	41	7583 148161	р292ль	Осипов Флор Андреевич	2	18.10.2020
	42	2038 609401	т425ер	Куликов Овидий Евсеевич	6	19.10.2020
	44	6872 720006	т226уы	Самсонов Егор Мэлсович	8	19.10.2020
	45	9835 642111	н658оз	Хохлов Остап Лаврентьевич	7	19.10.2020

Рисунок 44 – Вкладка «Автобусы на маршруте»

При нажатии кнопки «Снять водителя с маршрута» выводится диалоговое окно для подтверждения действия (см. рис. 45). При нажатии «Да» водитель и автобус снимаются с маршрута. Кнопка «Снять автобус с маршрута» действует аналогичным образом по отношению автобусов, в итоге автобус и водитель сменяются с маршрута, разница только в том, какой именно информацией обладает диспетчер автопарка в конкретное время.

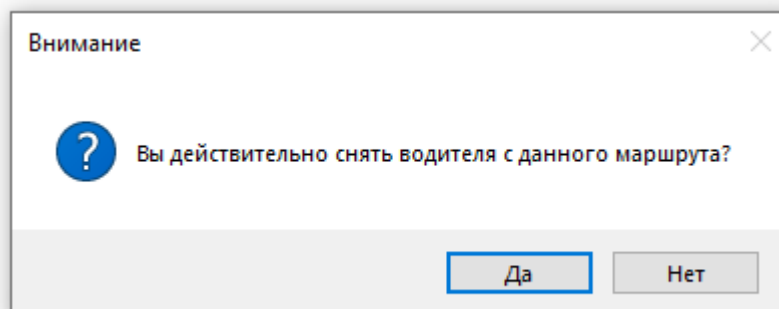


Рисунок 45 – Диалоговое окно подтверждения удаления смены

Кнопка «Новый автобус на маршруте» добавляет введенные в соответствующие поля данные водителя и автобуса, а также дату, установленную в поле дата, в таблицу смен.

Внизу вкладки «Автобусы на маршруте» закреплена таблица смен водителей.

#### 4.5 Вкладка «Запросы и справки»

В области запросы собраны все запросы информативного характера (см. рис. 46).

«Список водителей, работающих на маршруте № \_\_\_\_» – при вводе номера маршрута в поле данных и нажатии кнопки «Показать» в новом окне выводится информация в виде таблицы (см. рис. 47).

«Номера автобусов, обслуживающих маршрут № \_\_\_\_» – работает также как и предыдущий запрос, в диалоговом окне выводится информация в виде таблицы (см. рис. 48).

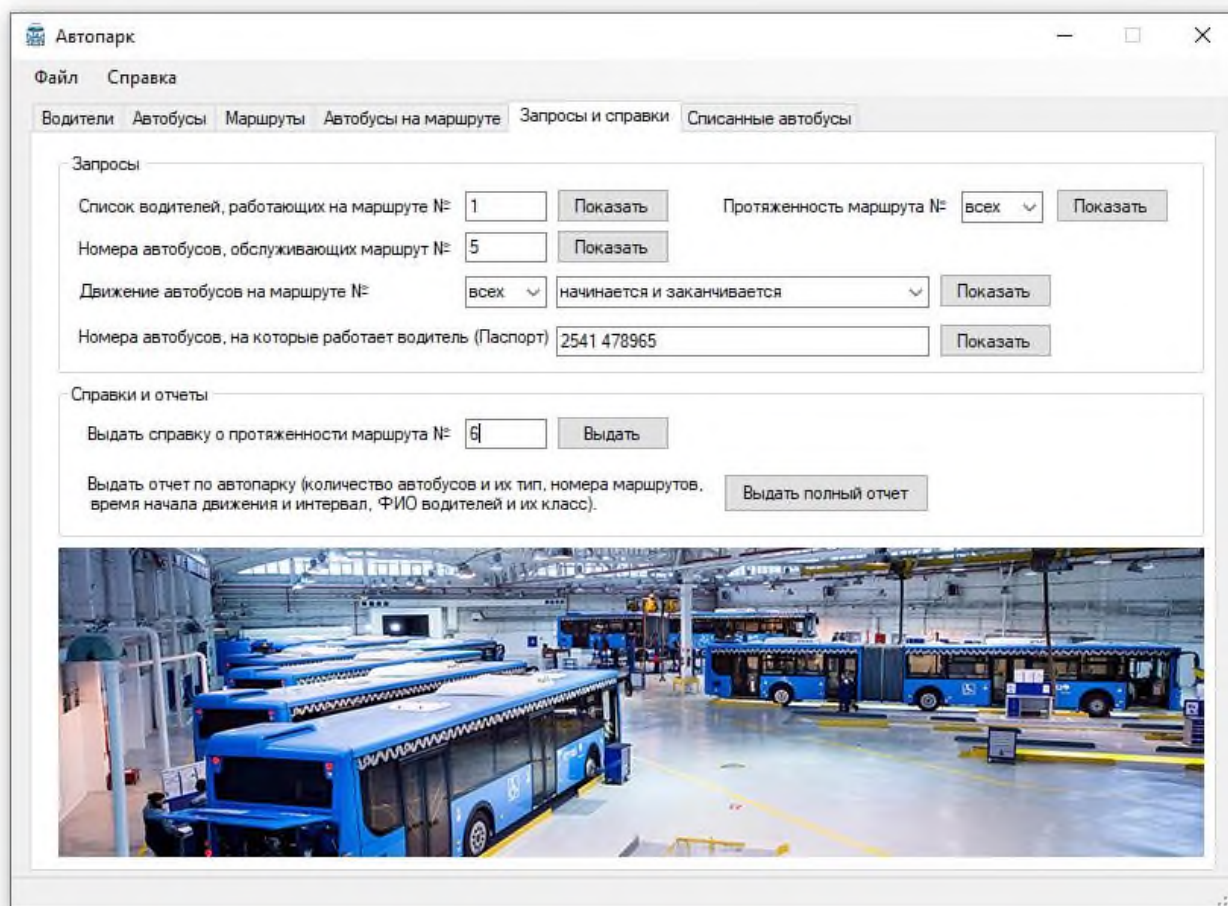


Рисунок 46 – Вкладка «Запросы и справки»

Список водителей на выбранном маршруте		
	Паспорт	ФИО
▶	8833 680121	Быков Рубен Егорович
	2888 108232	Кондратьев Роман Тимурович
	6502 104936	Михеев Власий Ефимович
*		

Рисунок 47 – Запрос списка водителей на выбранном маршруте

Автобусы выбранного маршрута		
	Автобус	Тип
▶	а138че	малый
	в222ро	сочлененный большой
	с125но	сочлененный большой
	я743ор	большой
*		

Рисунок 48 – Запрос списка автобусов на выбранном маршруте

«Движение автобусов на маршруте № \_\_А\_\_ \_\_\_\_\_Б\_\_\_\_\_» – в зависимости от параметров запроса на экран выводится разное количество информации в разном виде. При выборе параметра «А» как целое числовое значение маршрута, вне зависимости от параметра Б на экран выводится диалоговое окно с информацией (см. рис. 49).

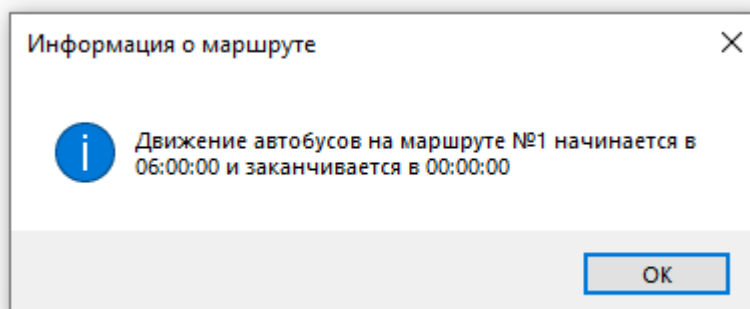


Рисунок 49 – Диалоговое окно с информацией

При указании параметра А как «всех» вне зависимости от выбранного параметра Б, данные будут выводиться в новом окне в табличном виде (см. рис. 50).

Маршрут	Время_начала_движения	Время_окончания_движения
1	06:00:00	00:00:00
2	08:00:00	23:00:00
3	08:00:00	23:30:00
4	06:00:00	23:00:00
5	08:00:00	23:30:00
6	06:00:00	23:00:00
7	08:00:00	23:00:00
8	06:00:00	23:30:00
9	06:00:00	00:00:00
*		

Рисунок 50 – Диалоговое окно с информацией

«Номера автобусов на которые работает водитель (Паспорт) \_\_\_\_\_» – в режиме диалогового окна как информацию на экран выводит информацию о водителе и автобусах, на которые он работает (см. рис. 51).



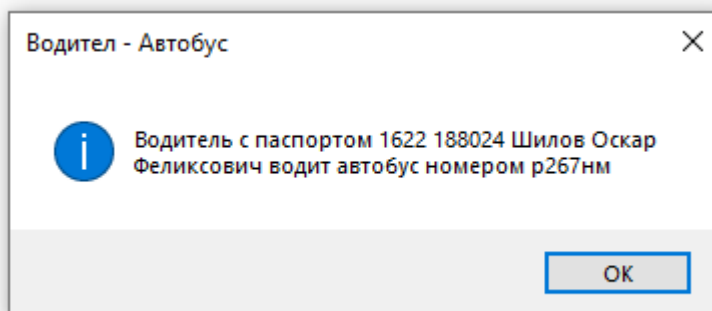


Рисунок 51 – Диалоговое окно с информацией

В области справки и отчеты собраны все документы на печать.

Справка о протяженности маршрута.

«Выдать справку о протяженности маршрута № \_\_\_\_» – после ввода номера маршрута и нажатии кнопки «Выдать», программа создаст документ MS Word, откроет его и начнет заполнять данными. После создания справки программа выводит диалоговое окно и сообщает, что справка готова (см. рис. 52).

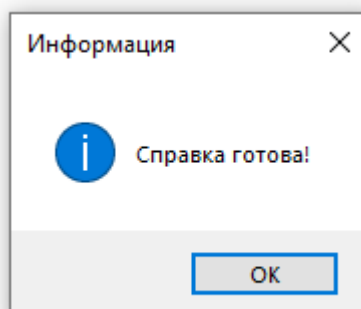


Рисунок 52 – Диалоговое окно с информацией

При любых попытках нажать все вышеперечисленные кнопки из вкладки «Запросы и справки» без заполнения соответствующих полей данных, программа выдаст ошибку (см. рис. 53)

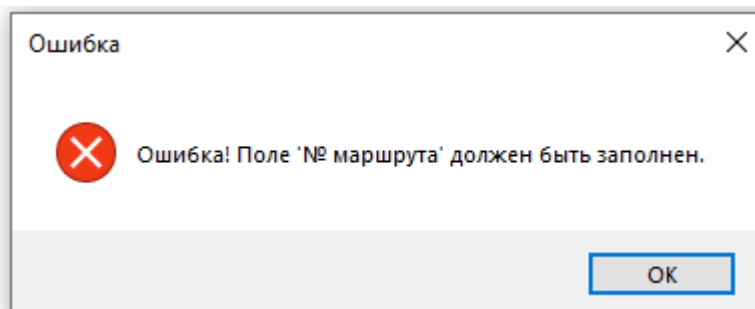


Рисунок 53 – Диалоговое окно с ошибкой

По нажатию кнопки «Выдать полный отчет», без заполнения каких-либо полей, начинается генерация полного отчета по автопарку. Программа запустит MS Word, создаст документ, откроет его и начнет заполнять данными. После создания отчета на экран выводится диалоговое окно с информацией о завершении генерации отчета (см. рис. 54).

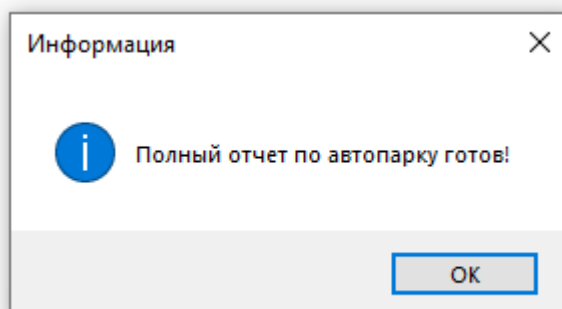


Рисунок 54 – Диалоговое окно с информацией

Примеры сгенерированных документов в приложении Г.

#### 4.6 Вкладка «Списанные автобусы»

Во вкладке «Списанные автобусы» всего 2 кнопки и таблица списанных автобусов. Кнопка «Вернуть автобус» возвращает списанный автобус в базу данных, а кнопка «Удалить навсегда» удаляет списанный автобус из базы данных навсегда (см. рис. 55).

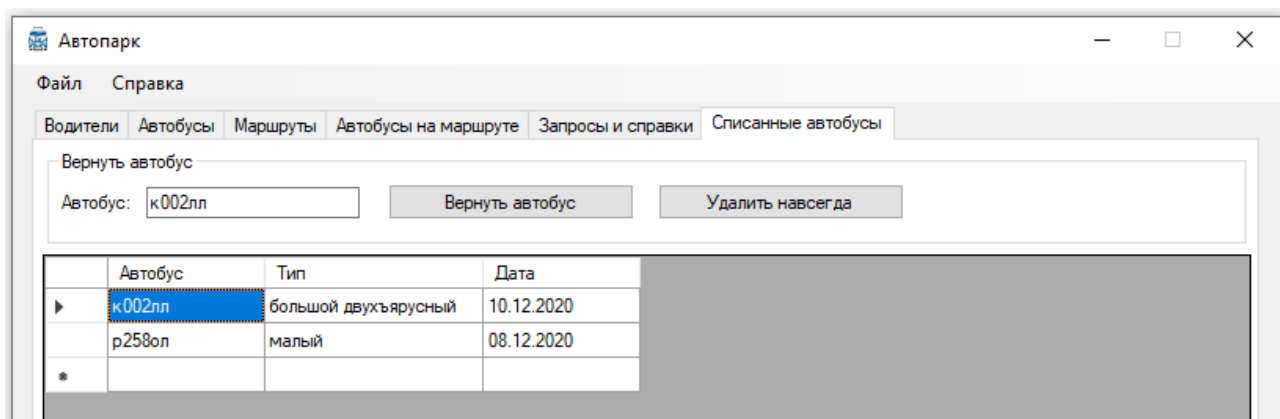


Рисунок 55 – Вкладка «Списанные автобусы»

## 4.7 Контекстное меню

Контекстное меню «Файл» содержит следующие элементы (см. рис. 56):

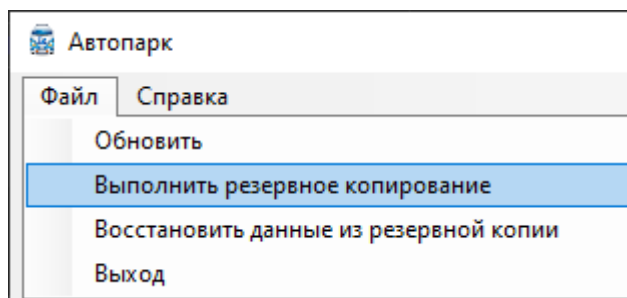


Рисунок 56 – Контекстное меню «Файл»

- Обновить – обновляет содержимое вкладки.
- Выполнить резервное копирование – создает резервную копию базы данных.
- Восстановить данные из резервной копии – открывается диалоговое окно выбора файла резервной копии БД и по нажатию кнопки ОК восстанавливает БД из резервной копии.
- Выход – Закрывает программу.

Контекстное меню «Справка» содержит всего один элемент (см. рис. 57):

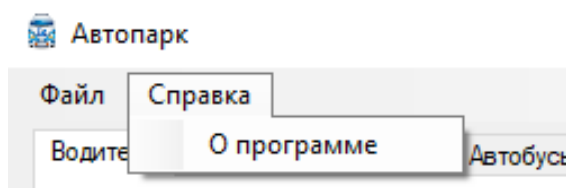


Рисунок 57 – Контекстно меню «Справка»

О программе – на отдельном диалоговом окне открывается форма «О программе», где есть информация о названии и версии программы, а также приписаны авторские права, год разработки программного обеспечения, название организации и приведено описание программы. Левую часть окна «О программе» занимает логотип онлайн курса по дисциплине «Управление данными» (см. рис. 58).

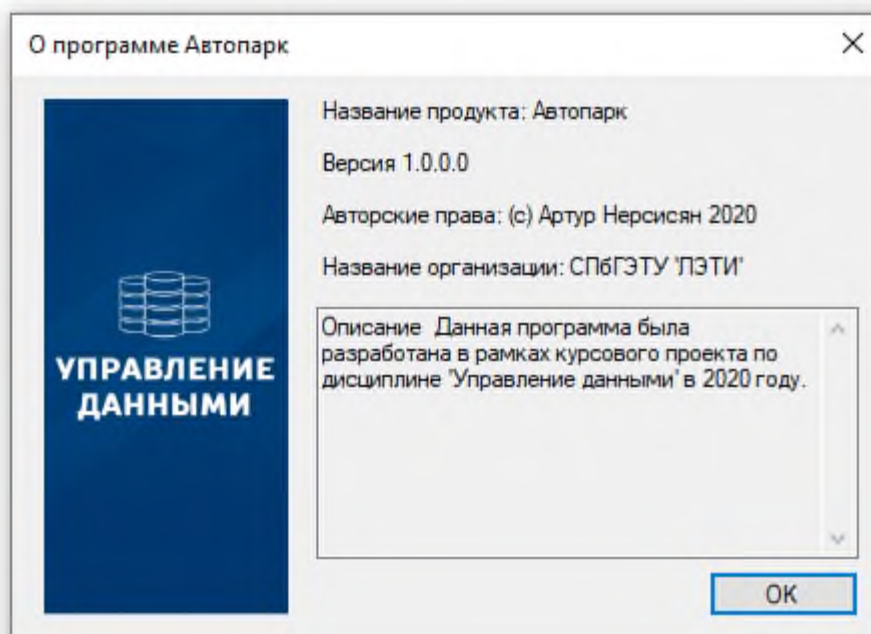


Рисунок 58 – Диалоговое окно «О программе»

## ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММНОГО КОДА SQL

### Листинг 1. Создание базы данных и пользователя.

/\*Запрос на создание имени входа на сервер\*/

```
USE [master]
GO
```

```
CREATE DATABASE [Avtopark]
GO
```

```
CREATE LOGIN [Dispetcher]
WITH PASSWORD='dispetcher',
DEFAULT_DATABASE=[Avtopark],
CHECK_EXPIRATION=OFF,
CHECK_POLICY=OFF
GO
```

```
USE [Avtopark]
GO
CREATE USER [Dispetcher] FOR LOGIN [Dispetcher]
GO
USE [Avtopark]
GO
EXEC sp_addrolemember N'db_datawriter', N'Dispetcher'
GO
```

/\*запрос на предоставление прав этому пользователю к тем же базам\*/

```
USE [Avtopark]
GO
GRANT DELETE TO [Dispetcher]
GO
USE [Avtopark]
GO
GRANT INSERT TO [Dispetcher]
GO
USE [Avtopark]
GO
GRANT SELECT TO [Dispetcher]
GO
USE [Avtopark]
GO
GRANT UPDATE TO [Dispetcher]
GO
```

## Листинг 2. Создание таблиц и установка зависимостей (внешних ключей)

```
USE [Avtopark]
GO

CREATE TABLE [dbo].[Автобусы] (
    [Автобус] NVARCHAR (6) NOT NULL,
    [Тип] NVARCHAR (20) NOT NULL,
    PRIMARY KEY CLUSTERED ([Автобус] ASC)
)
GO

CREATE TABLE [dbo].[Тип_вместимость] (
    [Тип] NVARCHAR (20) NOT NULL,
    [Вместимость(мест)] INT NOT NULL,
    CONSTRAINT [PK_Тип_вместимость] PRIMARY KEY ([Тип])
)
GO

CREATE TABLE [dbo].[Водители] (
    [Паспорт] VARCHAR (11) NOT NULL,
    [ФИО] NVARCHAR (50) NOT NULL,
    [Оклад] INT NOT NULL,
    [Стаж] INT NOT NULL,
    [Класс] NCHAR (5) NOT NULL,
    PRIMARY KEY CLUSTERED ([Паспорт] ASC)
)
GO

CREATE TABLE [dbo].[Оклад] (
    [Класс] NCHAR (5) NOT NULL,
    [Оклад] NCHAR (6) NOT NULL,
    PRIMARY KEY CLUSTERED ([Класс] ASC)
)
GO

CREATE TABLE [dbo].[Водят] (
    [id_Водителя] INT IDENTITY (1, 1) NOT NULL,
    [Паспорт] VARCHAR (11) NOT NULL,
    [Автобус] NVARCHAR (6) NOT NULL,
    [ФИО] NVARCHAR (50) NOT NULL,
    [Маршрут] INT NOT NULL,
    [Дата] DATE NOT NULL,
    PRIMARY KEY CLUSTERED ([id_Водителя] ASC)
)
GO
```

```

CREATE TABLE [dbo].[Маршруты] (
    [Маршрут] INT NOT NULL,
    [Начальный_пункт] NVARCHAR (25) NOT NULL,
    [Конечный_пункт] NVARCHAR (25) NOT NULL,
    [Время_начала_движения] TIME (7) NOT NULL,
    [Время_окончания_движения] TIME (7) NOT NULL,
    [Интервал(мин)] INT NOT NULL,
    [Протяженность(мин)] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([Маршрут] ASC)
)
GO

CREATE TABLE [dbo].[Списанные_автобусы] (
    [Автобус] NVARCHAR (6) NOT NULL,
    [Тип] NVARCHAR (20) NOT NULL,
    [Дата] DATE NOT NULL,
    PRIMARY KEY CLUSTERED ([Автобус] ASC)
)
GO

ALTER TABLE [dbo].[Водят] WITH CHECK ADD CONSTRAINT [FK_Водят_Автобусы]
FOREIGN KEY([Автобус])
REFERENCES [dbo].[Автобусы] ([Автобус])
GO
ALTER TABLE [dbo].[Водят] CHECK CONSTRAINT [FK_Водят_Автобусы]
GO
ALTER TABLE [dbo].[Водят] WITH CHECK ADD CONSTRAINT [FK_Водят_Водители]
FOREIGN KEY([Паспорт])
REFERENCES [dbo].[Водители] ([Паспорт])
GO
ALTER TABLE [dbo].[Водят] CHECK CONSTRAINT [FK_Водят_Водители]
GO
ALTER TABLE [dbo].[Водят] WITH CHECK ADD CONSTRAINT [FK_Водят_Маршруты]
FOREIGN KEY([Маршрут])
REFERENCES [dbo].[Маршруты] ([Маршрут])
GO
ALTER TABLE [dbo].[Водят] CHECK CONSTRAINT [FK_Водят_Маршруты]
GO
ALTER TABLE [dbo].[Водители] WITH CHECK ADD CONSTRAINT [FK_Водители_Оклад]
FOREIGN KEY([Класс])
REFERENCES [dbo].[Оклад] ([Класс])
GO
ALTER TABLE [dbo].[Водители] CHECK CONSTRAINT [FK_Водители_Оклад]
GO
ALTER TABLE [dbo].[Автобусы] WITH CHECK ADD CONSTRAINT [FK_Автобусы_Тип_
_вместимость] FOREIGN KEY([Тип])
REFERENCES [dbo].[Тип_вместимость] ([Тип])
GO
ALTER TABLE [dbo].[Автобусы] CHECK CONSTRAINT [FK_Автобусы_Тип_вместимост
ь]
GO

```

### Листинг 3. Заполнение таблиц данными

```
INSERT INTO Оклад(Класс,Оклад) VALUES('A', 10000)
INSERT INTO Оклад(Класс,Оклад) VALUES('B', 15000)
INSERT INTO Оклад(Класс,Оклад) VALUES('C', 20000)

INSERT INTO Тип_вместимость VALUES(N'малый',40)
INSERT INTO Тип_вместимость VALUES(N'средний',65)
INSERT INTO Тип_вместимость VALUES(N'большой',110)
INSERT INTO Тип_вместимость VALUES(N'сочлененный большой',120)
INSERT INTO Тип_вместимость VALUES(N'большой двухъярусный',150)

INSERT INTO Маршруты VALUES(1, N'Наличная ул.', N'Новосибирская ул.', '06:00:00','00:00:00', 30, 85)
INSERT INTO Маршруты VALUES(2, N'Театральная ул. пл.', N'ул. Партизана Германа', '08:00:00','23:00:00', 35, 124)
INSERT INTO Маршруты VALUES(3, N'Ленинский пр.', N'Краснофлотское шоссе', '08:00:00','23:30:00', 28, 116)
INSERT INTO Маршруты VALUES(4, N'ул. Мануильского', N'Ораниенбаумский пр', '06:00:00','23:00:00', 30, 125)
INSERT INTO Маршруты VALUES(5, N'пл. Восстания', N'Миллионная ул.', '08:00:00','23:30:00', 35, 130)
INSERT INTO Маршруты VALUES(6, N'пр. Обуховской Обороны', N'Ушаковский мост', '06:00:00','23:00:00', 50, 50)
INSERT INTO Маршруты VALUES(7, N'Невский пр', N'проспект Испытателей', '08:00:00','23:00:00', 20, 60)
INSERT INTO Маршруты VALUES(8, N'Наличная ул.', N'Кронверкская наб.', '06:00:00','23:30:00', 20, 98)
INSERT INTO Маршруты VALUES(9, N'Александровская ул.', N'Исаакиевская пл.', '06:00:00','00:00:00', 26, 102)

INSERT INTO Автобусы(Автобус, Тип) VALUES(N'p258ол', N'малый')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'п528чт', N'большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'к002лл', N'большой двухъярусный')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'ф789би', N'сочлененный большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'p963де', N'большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'в111ты', N'малый')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'с125но', N'сочлененный большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'а138че', N'малый')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'ф236за', N'большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'ф253др', N'малый')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'p263гр', N'большой двухъярусный')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'p267нм', N'сочлененный большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'p268гх', N'большой')
```



```

INSERT INTO Автобусы(Автобус, Тип) VALUES(N'у289ьф', N'большой двухъярусный')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'р292ць', N'большой двухъярусный')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'т425ер', N'малый')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'т336юю', N'большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'т226уы', N'сочлененный большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'н658оэ', N'большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'м758уы', N'большой двухъярусный')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'н896гт', N'малый')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'н967тр', N'большой двухъярусный')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'х487гф', N'малый')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'р265оф', N'сочлененный большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'я743ор', N'большой')
INSERT INTO Автобусы(Автобус, Тип) VALUES(N'в222ро', N'сочлененный большой')

```

```

INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('2888 10823 2', N'Кондратьев Роман Тимурович', 'С', 3, 23000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('6502 10493 6', N'Михеев Власий Ефимович', 'А', 1, 11000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1529 52963 0', N'Лапин Нинель Федотович', 'С', 15, 35000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1539 76516 0', N'Мамонтов Эрик Владиславович', 'В', 6, 21000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('3258 75269 6', N'Степанов Клим Созонович', 'В', 9, 24000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1605 74603 4', N'Агафонов Зиновий Дмитриевич', 'В', 12, 27000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('6146 83528 6', N'Новиков Влас Протасьевич', 'С', 6, 26000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('2162 41976 0', N'Поляков Мечислав Русланович', 'А', 1, 11000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1368 27722 5', N'Стрелков Алексей Даниилович', 'А', 13, 23000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('5622 18880 4', N'Ширяев Алан Владиславович', 'А', 9, 19000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('4541 08992 4', N'Рябов Владлен Георгиевич', 'А', 8, 18000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1622 18802 4', N'Шилов Оскар Феликсович', 'А', 19, 29000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1884 69545 3', N'Дьячков Афанасий Аристархович', 'А', 10, 20000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('8833 68012 1', N'Быков Рубен Егорович', 'А', 12, 22000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('7583 14816 1', N'Осипов Флор Андреевич', 'В', 2, 17000)

```

```

INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('2038 60940
1', N'Куликов Овидий Евсеевич', 'А', 16, 26000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1677 20436
1', N'Анисимов Аверьян Богуславович', 'А', 4, 14000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('6872 72000
6', N'Самсонов Егор Мэлсович', 'В', 18, 33000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('9835 64211
1', N'Хохлов Остап Лаврентьевич', 'С', 14, 34000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1119 71152
7', N'Силин Платон Даниилович', 'С', 2, 22000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('3158 97529
6', N'Зыков Моисей Тарасович', 'В', 5, 20000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('1496 56058
8', N'Орехов Болеслав Проклович', 'В', 3, 18000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('3815 20845
2', N'Ковалёв Касьян Яковович', 'С', 8, 28000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('5574 92731
6', N'Шилов Филипп Эльдарович', 'А', 2, 12000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('7258 84172
2', N'Фомичёв Эдуард Вениаминович', 'А', 5, 15000)
INSERT INTO Водители(Паспорт, ФИО, Класс, Стаж, Оклад) VALUES('5429 86734
5', N'Петухов Василий Серапионович', 'В', 5, 20000)

INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('2888 1082
32', N'р258ол', N'Кондратьев Роман Тимурович', 1, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('6502 1049
36', N'п528чт', N'Михеев Власий Ефимович', 1, '2020-10-19')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1529 5296
30', N'к002лл', N'Лапин Нинель Федотович', 2, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1539 7651
60', N'ф789би', N'Мамонтов Эрик Владиславович', 3, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('3258 7526
96', N'р963де', N'Степанов Клим Созонович', 3, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1605 7460
34', N'в111ты', N'Агафонов Зиновий Дмитриевич', 4, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('6146 8352
86', N'с125но', N'Новиков Влас Протасьевич', 5, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('2162 4197
60', N'а138че', N'Поляков Мечислав Русланович', 5, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1368 2772
25', N'ф236за', N'Стрелков Алексей Даниилович', 9, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('5622 1888
04', N'ф253др', N'Ширяев Алан Владиславович', 6, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('4541 0899
24', N'р263гр', N'Рябов Владлен Георгиевич', 7, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1622 1880
24', N'р267нм', N'Шилов Оскар Феликсович', 8, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1884 6954
53', N'р268гх', N'Дьячков Афанасий Аристархович', 9, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('8833 6801
21', N'у289ьф', N'Быков Рубен Егорович', 1, '2020-10-18')

```

```

INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('7583 1481
61', N'р292ць', N'Осипов Флор Андреевич', 2, '2020-10-18')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('2038 6094
01', N'т425ер', N'Куликов Овидий Евсеевич', 6, '2020-10-19')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1677 2043
61', N'т336юю', N'Анисимов Аверьян Богуславович', 4, '2020-10-19')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('6872 7200
06', N'т226уы', N'Самсонов Егор Мэлсович', 8, '2020-10-19')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('9835 6421
11', N'н658оэ', N'Хохлов Остап Лаврентьевич', 7, '2020-10-19')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1119 7115
27', N'м758уы', N'Силин Платон Даниилович', 5, '2020-10-19')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('3158 9752
96', N'н896гт', N'Зыков Моисей Тарасович', 1, '2020-10-22')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('1496 5605
88', N'н967тр', N'Орехов Болеслав Проклович', 3, '2020-10-25')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('3815 2084
52', N'х487гф', N'Ковалёв Касьян Яковович', 8, '2020-10-27')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('5574 9273
16', N'р265оф', N'Шилов Филипп Эльдарович', 2, '2020-10-25')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('7258 8417
22', N'я743ор', N'Фомичёв Эдуард Вениаминович', 5, '2020-10-23')
INSERT INTO Водят(Паспорт, Автобус, ФИО, Маршрут, Дата) VALUES('5429 8673
45', N'в222ро', N'Петухов Василий Серапионович', 5, '2020-10-25')

```

#### Листинг 4. Поисковые запросы

```

SELECT *
FROM [Водители]
WHERE [Паспорт] = @Паспорт
SELECT *
FROM [Водители]
WHERE [ФИО] = @ФИО
/*Водители вне смен*/
SELECT*
FROM [Водители]
WHERE Водители.Паспорт NOT IN (SELECT Паспорт
                                FROM Водят)

SELECT *
FROM [Автобусы]
WHERE [Автобус] = @Автобус
/*Автобусы вне смен*/
SELECT*
FROM [Автобусы]
WHERE Автобусы.Автобус NOT IN (SELECT Автобус
                                FROM Водят)

```

```

SELECT *
FROM [Маршруты]
WHERE [Маршрут] = @Маршрут
/*Водители маршрута*/
SELECT [Автобус], [ФИО]
FROM [Водят]
WHERE [Маршрут] = @Маршрут
ORDER BY [ФИО] ASC
/*Автобусы маршрута*/
SELECT Автобус, Тип
FROM [Автобусы]
WHERE Автобусы.Автобус IN (SELECT Автобус
                             FROM Водят
                             WHERE Маршрут = @Маршрут)
/*Движение на маршруте начинается, заканчивается*/
SELECT [Маршрут], [Время_начала_движения], [Время_окончания_движения]
FROM [Маршруты]
/*Движение на маршруте начинается*/
SELECT [Маршрут], [Время_начала_движения]
FROM [Маршруты]
/*Движение на маршруте заканчивается*/
SELECT [Маршрут], [Время_окончания_движения]
FROM [Маршруты]
/*Протяженность всех маршрутов*/
SELECT [Маршрут], [Протяженность(мин)]
FROM [Маршруты]
/*Протяженность конкретного маршрута*/
SELECT [Маршрут], [Протяженность(мин)]
WHERE [Маршрут] = @Маршрут
/*Автобус на котором работает водитель*/
SELECT [Автобус], [ФИО]
FROM [Водят]
WHERE [Паспорт] = @Паспорт

```

## Листинг 5. Резервное копирование и восстановление

```

BACKUP DATABASE Avtopark
TO DISK = 'C:\SQLdata\BACKUPS\Avtopark_FullDbBkup.bak'

RESTORE DATABASE Avtopark
FROM DISK = 'C:\SQLdata\BACKUPS\Avtopark_FullDbBkup.bak'
WITH REPLACE

```

## ПРИЛОЖЕНИЕ В. ЛИСТИНГ ПРОГРАММНОГО КОДА C#

### Листинг 1. FILE: Program.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        public static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new loginForm());
        }
    }
}
```

### Листинг 2. FILE: mainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Word = Microsoft.Office.Interop.Word;
using System.Reflection;
using System.IO;
```

```

namespace Avtopark
{
    public partial class MainForm : Form
    {
        SqlConnection sqlConnection;
        string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +
            @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Security=True";
        string masterConnString = @"Data Source = (LocalDB)\MSSQLLocalDB;AttachDbFilename=" +
            @"C:\winformsDB\Avtopark\Database1.mdf; Initial Catalog = master; Integrated Security = True";

        public MainForm()
        {
            InitializeComponent();

            //загрузка интерфейса программы
            private async void Form1_Load(object sender, EventArgs e)
            {
                this.водятTableAdapter.Fill(this.database1DataSet.Водят);
                this.списанные_автобусыTableAdapter.Fill(this.database1DataSet.Списанные_автобусы);
                this.маршрутыTableAdapter.Fill(this.database1DataSet.Маршруты);
                this.тип_вместимостьTableAdapter.Fill(this.database1DataSet.Тип_вместимость);
                this.автобусыTableAdapter.Fill(this.database1DataSet.Автобусы);
                this.водителиTableAdapter.Fill(this.database1DataSet.Водители);

                sqlConnection = new SqlConnection(connectionString);

                await sqlConnection.OpenAsync();
            }

            //поиск водителя по паспорту
            private void поискВодителяПоПаспортуToolStripButton_Click(object sender, EventArgs e)
            {
                try
                {
                    this.водителиTableAdapter.ПоискВодителяПоПаспорту(this.database1DataSet.Водители, паспортToolStripTextBox.Text);
                }
                catch (System.Exception ex)
                {

```

```

        System.Windows.Forms.MessageBox.Show(ex.Message);
    }

}

//поиск водителя по ФИО
private void поискВодителяПоФИОToolStripButton_Click(object sender, EventArgs e)
{
    try
    {
        this.водителиTableAdapter.ПоискВодителяПоФИО(this.database1DataSet.Водители, ФИОToolStripTextBox.Text);
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}

//добавить водителя
private async void button1_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(паспортTextBox.Text) && !string.IsNullOrEmpty(ФИОTextBox.Text) && !string.IsNullOrEmpty(классTextBox.Text) && !string.IsNullOrEmpty(стажTextBox.Text))
    {
        string cmd = "INSERT INTO [Водители] (Паспорт, ФИО, Оклад, Класс, Стаж) " +
            "VALUES(@Паспорт, @ФИО, @Оклад, @Класс, @Стаж)";
        SqlCommand command = new SqlCommand(cmd, sqlConnection);
        command.Parameters.AddWithValue("Паспорт", паспортTextBox.Text);
        command.Parameters.AddWithValue("ФИО", ФИОTextBox.Text);

        int staj = Convert.ToInt32(стажTextBox.Text);
        int klass = 0;
        if (классTextBox.Text == "A") klass = 10000;
        if (классTextBox.Text == "B") klass = 15000;
        if (классTextBox.Text == "C") klass = 20000;
        int oklad = (staj * 1000) + klass;
        string Oklad = Convert.ToString(oklad);
    }
}

```



```

        command.Parameters.AddWithValue("Оклад", Oklad);
        command.Parameters.AddWithValue("Класс", классTextBox.Text);
    t);
        command.Parameters.AddWithValue("Стаж", стажTextBox.Text)
    ;

    try
    {
        await command.ExecuteNonQueryAsync();

        toolStripStatusLabel1.Text = "Данные водителя были успешно добавлены в базу данных.";
        toolStripStatusLabel1.Visible = true;
    }
    catch (System.Exception)
    {
        MessageBox.Show("Ошибка, водитель с данным паспортом уже есть в базе. Проверьте введенные данные.",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
else
{
    toolStripStatusLabel1.Text = "Поля 'id_Водителя', 'ФИО', 'Класс' и " +
        "'Стаж_работы' должны быть заполнены.";
    toolStripStatusLabel1.Visible = true;
}
this.водителиTableAdapter.Fill(this.database1DataSet.водители);
);

}

private void выходToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (sqlConnection != null && sqlConnection.State != ConnectionState.Closed)
        sqlConnection.Close();
    Application.Exit();
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (sqlConnection != null && sqlConnection.State != ConnectionState.Closed)
        sqlConnection.Close();
    Application.Exit();
}
}

```



```

//показать водителей не на маршрутах
private void button20_Click(object sender, EventArgs e)
{
    VoditeliNeVodyat netNeVodyat = new VoditeliNeVodyat();
    netNeVodyat.Show();
}

//показать автобусы не на маршрутах
private void button21_Click(object sender, EventArgs e)
{
    AvtobusiNeEzdyat netNeEzdyat = new AvtobusiNeEzdyat();
    netNeEzdyat.Show();
}

//уволить водителя
private async void button2_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(паспортTextBox.Text) && !string.IsN
ullOrWhiteSpace(паспортTextBox.Text))
    {
        if (MessageBox.Show("Вы действительно хотите удалить данн
ые выбранного водителя?", "Внимание", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == System.Windows.Forms.DialogR
esult.Yes)
        {
            string cmd = "DELETE FROM [Водители] WHERE [Паспорт]
= @Паспорт";
            SqlCommand command = new SqlCommand(cmd, sqlConnectio
n);
            command.Parameters.AddWithValue("Паспорт", паспортTex
tBox.Text);

            try
            {
                await command.ExecuteNonQueryAsync();

                toolStripStatusLabel1.Text = "Данные водителя был
и успешно удалены.";
                toolStripStatusLabel1.Visible = true;
            }
            catch (System.Exception)
            {
                MessageBox.Show("Ошибка, снимите водителя с марш
ruta. Это можно делать во вкладке 'Автобусы на маршруте'",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon
n.Error);
            }
        }
    }
}

```

```

        }
        else
        {
            MessageBox.Show("Ошибка! Поле 'Паспорт' должен быть заполнен.",
                            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            toolStripStatusLabel1.Text = "Поле 'Паспорт' должен быть заполнен.";
            toolStripStatusLabel1.Visible = true;
        }
        this.водителиTableAdapter.Fill(this.database1DataSet.водители);
    }

    //кнопка обновить
    private void обновитьToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //
        this.водителиTableAdapter.Fill(this.database1DataSet.водители);
        //this.водителиTableAdapter.Update(this.database1DataSet.водители);
        this.автобусыTableAdapter.Fill(this.database1DataSet.Автобусы);
        this.тип_местимостьTableAdapter.Fill(this.database1DataSet.Тип_местимость);
        this.маршрутыTableAdapter.Fill(this.database1DataSet.Маршруты);
        this.списанные_автобусыTableAdapter.Fill(this.database1DataSet1.Списанные_автобусы);
    }

    //окно о программе
    private void оПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
    {
        AboutBox1 form2 = new AboutBox1();
        form2.Show();
    }

    //удалить автобус
    private async void button3_Click(object sender, EventArgs e)
    {
        if (toolStripStatusLabel1.Visible)
            toolStripStatusLabel1.Visible = false;
        if (!string.IsNullOrEmpty(автобусTextBox.Text) && !string.IsNullOrWhiteSpace(автобусTextBox.Text))
        {

```

```

        string cmd1 = "SELECT * FROM [Автобусы] WHERE Автобус = @
Автобус";
        SqlCommand command0 = new SqlCommand(cmd1, sqlConnection)
;
        SqlDataReader sdaReader;

        command0.Parameters.AddWithValue("Автобус", автобусTextBo
x.Text);

        sdaReader = await command0.ExecuteReaderAsync();
        string avtobusType = "";

        while (sdaReader.Read())
            avtobusType = sdaReader["Тип"].ToString();
        sdaReader.Close();

        string cmd2 = "INSERT INTO [Списанные_автобусы] (Автобус,
Тип, Дата)" +
            "VALUES(@Автобус, @Тип, @Дата)";
        SqlCommand command1 = new SqlCommand(cmd2, sqlConnection)
;

        command1.Parameters.AddWithValue("Автобус", автобусTextBo
x.Text);

        command1.Parameters.AddWithValue("Тип", avtobusType);
        command1.Parameters.AddWithValue("Дата", DateTime.Now.Dat
e);

        try
        {
            await command1.ExecuteNonQueryAsync();
        }
        catch (System.Exception ex)
        {
            MessageBox.Show(Convert.ToString(ex), "Ошибка", Messa
geBoxButtons.OK, MessageBoxIcon.Error);
        }

        string cmd3 = "DELETE FROM [Автобусы] WHERE [Автобус] = @
Автобус";
        SqlCommand command2 = new SqlCommand(cmd3, sqlConnection)
;

        command2.Parameters.AddWithValue("Автобус", автобусTextBo
x.Text);

        try
        {
            await command2.ExecuteNonQueryAsync();

```

```

        toolStripStatusLabel1.Text = "Данные автобуса были ус
пешно удалены из базы данных.";
        toolStripStatusLabel1.Visible = true;
    }
    catch (System.Exception)
    {
        MessageBox.Show("Ошибка, снимите автобус с маршрута.
Это можно делать во вкладке 'Автобусы на маршруте'",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.E
rror);
    }
}
else
{
    toolStripStatusLabel1.Text = "Поле 'Автобус' должен быть
заполнен.";
    toolStripStatusLabel1.Visible = true;
}
this.автобусыTableAdapter.Fill(this.database1DataSet.Автобусы
);
}

private void поискАвтобусаToolStripButton_Click(object sender, Ev
entArgs e)
{
    try
    {
        this.автобусыTableAdapter.ПоискАвтобуса(this.database1Dat
aSet.Автобусы, автобусToolStripTextBox.Text);
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}

private async void button4_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(автобусToolStripTextBox.Text) && !string.IsN
ullOrWhiteSpace(автобусToolStripTextBox.Text) &&
        !string.IsNullOrEmpty(comboBox1.Text) && !string.IsNullOrW
hiteSpace(comboBox1.Text))
    {
        string cmd = "INSERT INTO [Автобусы] (Автобус, Тип) VALUE
S(@Автобус, @Тип)";
        SqlCommand command = new SqlCommand(cmd, sqlConnection);
        command.Parameters.AddWithValue("Автобус", автобусToolStrip
.Text);
    }
}

```

```

        command.Parameters.AddWithValue("Тип", comboBox1.Text);
        try
        {
            await command.ExecuteNonQueryAsync();

            toolStripStatusLabel1.Text = "Данные автобуса были успешно добавлены в базу данных.";
            toolStripStatusLabel1.Visible = true;
        }
        catch
        {
            MessageBox.Show("Ошибка! Значения поля 'Автобус' не может повторяться. Значение поля 'Тип' выберите из выпадающего списка. \n Проверьте правильность введенных данных",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            toolStripStatusLabel1.Text = "Ошибка! Значения поля 'Автобус' не может повторяться. Значение поля 'Тип' выберите из выпадающего списка.";
            toolStripStatusLabel1.Visible = true;
        }
    }
    else
    {
        MessageBox.Show("Ошибка, Поля 'Автобус' и 'Тип' должны быть заполнены.",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

        toolStripStatusLabel1.Text = "Поля 'Автобус' и 'Тип' должны быть заполнены.";
        toolStripStatusLabel1.Visible = true;
    }
    this.автобусыTableAdapter.Fill(this.database1DataSet.Автобусы);
}

//=====

//маршруты
//=====

//создать новый маршрут
private async void button6_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    // если непусты и не состоят из пробелов, а правильность введенных данных проверит сама форма и БД

```

```

        if (!string.IsNullOrEmpty(маршрутTextBox.Text) && !string.IsN
ullOrWhiteSpace(маршрутTextBox.Text) &&
            !string.IsNullOrEmpty(начальный_пунктTextBox.Text) && !str
ing.IsNullOrEmpty(начальный_пунктTextBox.Text) &&
            !string.IsNullOrEmpty(конечный_пунктTextBox.Text) && !stri
ng.IsNullOrEmpty(конечный_пунктTextBox.Text) &&
            !string.IsNullOrEmpty(время_начала_движенияTextBox.Text) &
& !string.IsNullOrEmpty(время_начала_движенияTextBox.Text) &&
            !string.IsNullOrEmpty(время_окончания_движенияTextBox.Text
) && !string.IsNullOrEmpty(время_окончания_движенияTextBox.Text) &&
            !string.IsNullOrEmpty(интервал_мин_TextBox.Text) && !strin
g.IsNullOrEmpty(интервал_мин_TextBox.Text) &&
            !string.IsNullOrEmpty(протяженность_мин_TextBox.Text) && !
string.IsNullOrEmpty(протяженность_мин_TextBox.Text))
        {
            string cmd = "INSERT INTO [Маршруты] (Маршрут, Начальный_
пункт, Конечный_пункт, Время_начала_движения, " +
                "Время_окончания_движения, [Интервал(мин)], [Протяжен
ность(мин)]) VALUES(@Маршрут, @Начальный_пункт, " +
                "@Конечный_пункт, @Время_начала_движения, @Время_око
нчания_движения, @Интервал, @Протяженность)";
            SqlCommand command = new SqlCommand(cmd, sqlConnection);
            command.Parameters.AddWithValue("Маршрут", маршрутTextBox
.Text);
            command.Parameters.AddWithValue("Начальный_пункт", началь
ный_пунктTextBox.Text);
            command.Parameters.AddWithValue("Конечный_пункт", конечны
й_пунктTextBox.Text);
            command.Parameters.AddWithValue("Время_начала_движения",
время_начала_движенияTextBox.Text);
            command.Parameters.AddWithValue("Время_окончания_движения
", время_окончания_движенияTextBox.Text);
            command.Parameters.AddWithValue("Интервал", интервал_мин_
TextBox.Text);
            command.Parameters.AddWithValue("Протяженность", протяжен
ность_мин_TextBox.Text);

            try
            {
                await command.ExecuteNonQueryAsync();

                toolStripStatusLabel1.Text = "Данные маршрута были ус
пешно добавлены в базу данных.";
                toolStripStatusLabel1.Visible = true;
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }
        }
    }

```

```

        else
        {
            MessageBox.Show("Ошибка! Поля 'Маршрут', 'Начальный пункт', 'Конечный пункт', 'Время начала движения', " +
                "'Время окончания движения', 'Интервал(мин)' и 'Протяженность(мин)' должны быть заполнены.",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            toolStripStatusLabel1.Text = "Поля 'Маршрут', 'Начальный пункт', 'Конечный пункт', 'Время начала движения', " +
                "'Время окончания движения', 'Интервал(мин)' и 'Протяженность(мин)' должны быть заполнены.";
            toolStripStatusLabel1.Visible = true;
        }
        this.маршрутыTableAdapter.Fill(this.database1DataSet.Маршруты);
    };
}

//изменить протяженность маршрута
private async void button5_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    // если непусты и не состоят из пробелов, а правильность введенных данных проверит сама форма и БД
    if (!string.IsNullOrEmpty(маршрутTextBox.Text) && !string.IsNullOrEmpty(протяженность_мин_TextBox.Text) && !string.IsNullOrEmpty(протяженность_мин_TextBox.Text))
    {
        string cmd = "UPDATE [Маршруты] SET [Протяженность(мин)]=@Протяженность WHERE Маршрут=@Маршрут";
        SqlCommand command = new SqlCommand(cmd, sqlConnection);
        command.Parameters.AddWithValue("Маршрут", маршрутTextBox.Text);
        command.Parameters.AddWithValue("Протяженность", протяженность_мин_TextBox.Text);

        try
        {
            await command.ExecuteNonQueryAsync();

            toolStripStatusLabel1.Text = "Протяженность маршрута была успешно изменена.";
            toolStripStatusLabel1.Visible = true;
        }
        catch (System.Exception ex)
        {
            System.Windows.Forms.MessageBox.Show(ex.Message);
        }
    }
}

```

```

    }
    else
    {
        toolStripStatusLabel1.Text = "Поля 'Маршрут' и 'Протяженн
ость(мин)' должны быть заполнены.";
        toolStripStatusLabel1.Visible = true;
    }
    this.маршрутыTableAdapter.Fill(this.database1DataSet.Маршруты
);
}

private void поискМаршрутаToolStripButton_Click(object sender, Ev
entArgs e)
{
    try
    {
        this.маршрутыTableAdapter.ПоискМаршрута(this.database1Dat
aSet.Маршруты, ((int)(System.Convert.ChangeType(маршрутToolStripText
Box.Text, typeof(int)))));
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message, "Ошибка"
, MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

//удалить маршрут
private async void button7_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(маршрутTextBox.Text) && !string.IsN
ullOrWhiteSpace(маршрутTextBox.Text))
    {
        if (MessageBox.Show("Вы действительно хотите удалить выбр
анный маршрут?", "Внимание", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == System.Windows.Forms.DialogR
esult.Yes)
        {
            string cmd = "DELETE FROM [Маршруты] WHERE [Маршрут]
= @Маршрут";
            SqlCommand command = new SqlCommand(cmd, sqlConnectio
n);
            command.Parameters.AddWithValue("Маршрут", маршрутTex
tBox.Text);

            try
            {
                await command.ExecuteNonQueryAsync();
            }
        }
    }
}

```



```

        toolStripStatusLabel1.Text = "Маршрут был успешно
удален из базы данных.";
        toolStripStatusLabel1.Visible = true;
    }
    catch (System.Exception)
    {
        MessageBox.Show("Ошибка, снимите все автобусы с
данного маршрута. Это можно делать во вкладке 'Автобусы на маршруте'",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon
n.Error);
    }
}
else
{
    MessageBox.Show("Поле 'Маршрут' должен быть заполнен", "О
шибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    toolStripStatusLabel1.Text = "Поле 'Маршрут' должен быть
заполнен.";
    toolStripStatusLabel1.Visible = true;
}
this.маршрутыTableAdapter.Fill(this.database1DataSet.Маршруты
);
}

//вернуть автобус
private async void button8_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(автобусTextBox1.Text) && !string.Is
NullOrWhiteSpace(автобусTextBox1.Text))
    {
        string cmd1 = "SELECT * FROM [Списанные_автобусы] WHERE А
втобус = @Автобус";
        SqlCommand command0 = new SqlCommand(cmd1, sqlConnection)
;

        SqlDataReader sdaReader;

        command0.Parameters.AddWithValue("Автобус", автобусTextBo
x1.Text);

        sdaReader = await command0.ExecuteReaderAsync();
        string avtobusType = "";

        while (sdaReader.Read())
            avtobusType = sdaReader["Тип"].ToString();
        sdaReader.Close();
    }
}

```

```

        string cmd2 = "INSERT INTO [Автобусы] (Автобус, Тип) VALU
ES(@Автобус, @Тип)";
        SqlCommand command1 = new SqlCommand(cmd2, sqlConnection)
;

        command1.Parameters.AddWithValue("Автобус", автобусTextBo
x1.Text);
        command1.Parameters.AddWithValue("Тип", avtobusType);

        try
        {
            await command1.ExecuteNonQueryAsync();
        }
        catch (System.Exception ex)
        {
            MessageBox.Show(Convert.ToString(ex), "Ошибка", Messa
geBoxButtons.OK, MessageBoxIcon.Error);
        }

        string cmd3 = "DELETE FROM [Списанные_автобусы] WHERE [Ав
тобус] = @Автобус";
        SqlCommand command2 = new SqlCommand(cmd3, sqlConnection)
;

        command2.Parameters.AddWithValue("Автобус", автобусTextBo
x1.Text);

        try
        {
            await command2.ExecuteNonQueryAsync();

            toolStripStatusLabel1.Text = "Данные автобуса были во
сстановлены.";
            toolStripStatusLabel1.Visible = true;
        }
        catch (System.Exception ex)
        {
            MessageBox.Show(Convert.ToString(ex), "Ошибка", Messa
geBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    else
    {
        toolStripStatusLabel1.Text = "Поле 'Автобус' должен быть
заполнен.";
        toolStripStatusLabel1.Visible = true;
    }
    this.списанные_автобусыTableAdapter.Fill(this.database1DataSe
t1.Списанные_автобусы);
}

```

```

//удалить навсегда
private async void button9_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(автобусTextBox1.Text) && !string.Is
NullOrWhiteSpace(автобусTextBox1.Text))
    {
        if (MessageBox.Show("Вы действительно хотите удалить выбр
анный элемент?", "Внимание", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == System.Windows.Forms.DialogR
esult.Yes)
        {
            string cmd = "DELETE FROM [Списанные_автобусы] WHERE
[Автобус] = @Автобус";
            SqlCommand command = new SqlCommand(cmd, sqlConnectio
n);
            command.Parameters.AddWithValue("Автобус", автобусTex
tBox1.Text);

            try
            {
                await command.ExecuteNonQueryAsync();

                toolStripStatusLabel1.Text = "Автобус успешно уда
лен из базы данных.";
                toolStripStatusLabel1.Visible = true;
            }
            catch (System.Exception)
            {
                MessageBox.Show("Ошибка! Поле 'Автобус' должен бы
ть заполнен.",
                                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.
Error);
            }
        }

        this.маршрутыTableAdapter.Fill(this.database1DataSet.Марш
руты);
    }
    else
    {
        toolStripStatusLabel1.Text = "Поле 'Автобус' должен быть
заполнен.";
        toolStripStatusLabel1.Visible = true;
    }
}

//Снять водителя с маршрута
private async void button11_Click(object sender, EventArgs e)

```

```

    {
        if (toolStripStatusLabel1.Visible)
            toolStripStatusLabel1.Visible = false;
        if (!string.IsNullOrEmpty(паспортTextBox1.Text) && !string.Is
NullOrWhiteSpace(паспортTextBox1.Text))
        {
            if (MessageBox.Show("Вы действительно снять водителя с да
нного маршрута?", "Внимание", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == System.Windows.Forms.DialogR
esult.Yes)
            {
                string cmd = "DELETE FROM [Водят] WHERE [Паспорт] = @
Паспорт";
                SqlCommand command = new SqlCommand(cmd, sqlConnectio
n);
                command.Parameters.AddWithValue("Паспорт", паспортTex
tBox1.Text);

                try
                {
                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Автобус и водитель
сняты выбранного маршрута.";
                    toolStripStatusLabel1.Visible = true;
                }
                catch (System.Exception)
                {
                    MessageBox.Show("Ошибка! Проверьте правильность в
веденных данных.",
                                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.
Error);
                }
            }
        }
        else
        {
            MessageBox.Show("Ошибка! Поле 'Паспорт' должен быть запол
нен.",
                            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.
Error);
            toolStripStatusLabel1.Text = "Поле 'Паспорт' должен быть
заполнен.";
            toolStripStatusLabel1.Visible = true;
        }
        this.водятTableAdapter.Fill(this.database1DataSet.Водят);
    }

    //Снять автобус с маршрута
    private async void button12_Click(object sender, EventArgs e)
    {

```

```

        if (toolStripStatusLabel1.Visible)
            toolStripStatusLabel1.Visible = false;
        if (!string.IsNullOrEmpty(автобусTextBox2.Text) && !string.Is
NullOrWhiteSpace(автобусTextBox2.Text))
        {
            if (MessageBox.Show("Вы действительно снять автобус с дан
ного маршрута?", "Внимание", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == System.Windows.Forms.DialogR
esult.Yes)
            {
                string cmd = "DELETE FROM [Водят] WHERE [Автобус] = @
Автобус";
                SqlCommand command = new SqlCommand(cmd, sqlConnectio
n);
                command.Parameters.AddWithValue("Автобус", автобусTex
tBox2.Text);

                try
                {
                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Автобус и водитель
сняты выбранного маршрута.";
                    toolStripStatusLabel1.Visible = true;
                }
                catch (System.Exception)
                {
                    MessageBox.Show("Ошибка! Проверьте правильность в
веденных данных.",
                                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.
Error);
                }
            }
        }
        else
        {
            MessageBox.Show("Ошибка! Поле 'Автобус' должен быть запол
нен.",
                            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.
Error);
            toolStripStatusLabel1.Text = "Поле 'Автобус' должен быть
заполнен.";
            toolStripStatusLabel1.Visible = true;
        }
        this.водятTableAdapter.Fill(this.database1DataSet.Водят);
    }

    //Новый автобус на маршруте
    private async void button10_Click(object sender, EventArgs e)
    {
        if (toolStripStatusLabel1.Visible)

```

```

        toolStripStatusLabel1.Visible = false;
        if (!string.IsNullOrEmpty(паспортTextBox1.Text) && !string.Is
NullOrWhiteSpace(паспортTextBox1.Text) &&
            !string.IsNullOrEmpty(фИОTextBox1.Text) && !string.IsNul
lOrWhiteSpace(фИОTextBox1.Text) &&
            !string.IsNullOrEmpty(автобусTextBox2.Text) && !string.Is
NullOrWhiteSpace(автобусTextBox2.Text) &&
            !string.IsNullOrEmpty(маршрутTextBox1.Text) && !string.Is
NullOrWhiteSpace(маршрутTextBox1.Text) &&
            !string.IsNullOrEmpty(датаDateTimePicker.Text) && !string
.IsNullOrEmpty(датаDateTimePicker.Text))
        {
            string cmd = "INSERT INTO [Водят] (Паспорт, Автобус, ФИО,
Маршрут, Дата) " +
                "VALUES(@Паспорт, @Автобус, @ФИО, @Маршрут, @Дата)";
            SqlCommand command = new SqlCommand(cmd, sqlConnection);
            command.Parameters.AddWithValue("Паспорт", паспортTextBox
1.Text);
            command.Parameters.AddWithValue("Автобус", автобусTextBox
2.Text);
            command.Parameters.AddWithValue("ФИО", фИОTextBox1.Text);
            command.Parameters.AddWithValue("Маршрут", маршрутTextBox
1.Text);
            command.Parameters.AddWithValue("Дата", DateTime.Parse(да
таDateTimePicker.Text));

            try
            {
                await command.ExecuteNonQueryAsync();

                toolStripStatusLabel1.Text = "Данные маршрута были ус
пешно добавлены в базу данных.";
                toolStripStatusLabel1.Visible = true;
            }
            catch (System.Exception ex)
            {
                string errMsg = "Ошибка! Водитель и/или автобус с так
ими данными в базе не найдено." +
                    " Проверьте правильность введенных данных. Прежде
чем добавить новый автобус на маршруте," +
                    " убедитесь что водитель и автобус добавлены в ба
зу.";
                MessageBox.Show(errMsg + "\n\n" + ex.Message, "Ошибка
", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show("Ошибка! Поля 'Паспорт', 'ФИО', 'Автобус'
, 'Маршрут' и 'Дата' должны быть заполнены.",

```

```

        "Ошибка", MessageBoxButtons.OK, MessageBoxIcon
        .Error);
        toolStripStatusLabel1.Text = "Поля 'Паспорт', 'ФИО', 'Авт
обус', 'Маршрут' и 'Дата' должны быть заполнены.";
        toolStripStatusLabel1.Visible = true;
    }
}

//=====ОТЧЕТЫ===ЗАПРОСЫ===СПРАВКИ=====
==
//=====
==

private void button13_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(textBox1.Text) && !string.IsNullOrW
hiteSpace(textBox1.Text))
    {
        VoditeliMarshruta newVod = new VoditeliMarshruta(textBox1
.Text);
        newVod.Show();
    }
    else
    {
        MessageBox.Show("Ошибка! Поле '№ маршрута' должен быть за
полнен.",
        "Ошибка", MessageBoxButtons.OK, MessageBoxIcon
        .Error);
        toolStripStatusLabel1.Text = "Поле '№ маршрута' должен бы
ть заполнен.";
        toolStripStatusLabel1.Visible = true;
    }
}

private void button14_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(textBox2.Text) && !string.IsNullOrW
hiteSpace(textBox2.Text))
    {
        AvtobusiMarshruta newAvt = new AvtobusiMarshruta(textBox2
.Text);
        newAvt.Show();
    }
    else
    {
        MessageBox.Show("Ошибка! Поле '№ маршрута' должен быть за
полнен.",

```

```

        "Ошибка", MessageBoxButtons.OK, MessageBoxIcon
n.Error);
        toolStripStatusLabel1.Text = "Поле '№ маршрута' должен бы
ть заполнен.";
        toolStripStatusLabel1.Visible = true;
    }
}

private void button16_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(comboBox4.Text) && !string.IsNullOrWhiteSpace(comboBox4.Text))
    {
        if (Convert.ToString(comboBox4.Text) == "всех")
        {
            MarshrutProtejennost newMarsh = new MarshrutProtejenn
ost();
            newMarsh.Show();
        }
        else
        {
            string query = "SELECT [Протяженность(мин)] " +
"FROM [Маршруты] " +
"WHERE [Маршрут] = @Маршрут";

            SqlCommand command = new SqlCommand(query, sqlConnect
ion);
            command.Parameters.AddWithValue("Маршрут", comboBox4.
Text);

            try
            {
                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();

                while (reader.Read())
                {
                    data.Add(new string[1]);

                    data[data.Count - 1][0] = reader[0].ToString(
);
                }

                reader.Close();
                MessageBox.Show("Протяженность маршрута №" + Conv
ert.ToString(comboBox4.Text) +
" составляет " + data[0][0] + " минут.", "Про
тяженность маршрута",

```



```

        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    ;
    // вывести для одного в мессидж боксе
}
}
else
{
    MessageBox.Show("Ошибка! Поле '№ маршрута' должен быть заполнен.",
        "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    toolStripStatusLabel1.Text = "Поле '№ маршрута' должен быть заполнен.";
    toolStripStatusLabel1.Visible = true;
}

}

private void button17_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if (!string.IsNullOrEmpty(textBox3.Text) && !string.IsNullOrWhiteSpace(textBox3.Text))
    {
        string query = "SELECT [Автобус], [ФИО] " +
            "FROM [Водят] " +
            "WHERE [Паспорт] = @Паспорт";

        SqlCommand command = new SqlCommand(query, sqlConnection);
        ;
        command.Parameters.AddWithValue("Паспорт", textBox3.Text);
        ;

        try
        {
            SqlDataReader reader = command.ExecuteReader();

            List<string[]> data = new List<string[]>();

            while (reader.Read())
            {
                data.Add(new string[2]);
            }
        }
    }
}

```

```

        data[data.Count - 1][0] = reader[0].ToString();
        data[data.Count - 1][1] = reader[1].ToString();
    }

    reader.Close();
    MessageBox.Show("Водитель с паспортом " + Convert.ToS
tring(textBox3.Text) +
        " " + data[0][1] + " водит автобус номером " + da
ta[0][0], "Водител - Автобус",
        MessageBoxButtons.OK, MessageBoxIcon.Information)
;

    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    }
    else
    {
        MessageBox.Show("Ошибка! Поле 'Паспорт' должен быть запол
нен.",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon
n.Error);
        toolStripStatusLabel1.Text = "Поле 'Паспорт' должен быть
заполнен.";
        toolStripStatusLabel1.Visible = true;
    }
}

private void button15_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
    if ((!string.IsNullOrEmpty(comboBox2.Text) && !string.IsNullo
rWhiteSpace(comboBox2.Text) &&
        !string.IsNullOrEmpty(comboBox3.Text) && !string.IsNullo
rWhiteSpace(comboBox3.Text)))
    {
        if (Convert.ToString(comboBox2.Text) == "всех" &&
            Convert.ToString(comboBox3.Text) == "начинается и зак
анчивается")
        {
            DvijCombo newCombo = new DvijCombo();
            newCombo.Show();
        }

        if (Convert.ToString(comboBox2.Text) == "всех" &&
            Convert.ToString(comboBox3.Text) == "начинается")
        {
            DvijenieStart newStartDvij = new DvijenieStart();

```

```

        newStartDvij.Show();
    }

    if (Convert.ToString(comboBox2.Text) == "всех" &&
        Convert.ToString(comboBox3.Text) == "заканчивается")
    {
        DvijenieEnd newEndDvij = new DvijenieEnd();
        newEndDvij.Show();
    }

    if (Convert.ToString(comboBox2.Text) != "всех" &&
        Convert.ToString(comboBox3.Text) == "заканчивается")
    {
        string query = "SELECT [Время_окончания_движения] " +
            "FROM [Маршруты] " +
            "WHERE [Маршрут] = @Маршрут";

        SqlCommand command = new SqlCommand(query, sqlConnect
ion);
        command.Parameters.AddWithValue("Маршрут", comboBox2.
Text);

        try
        {
            SqlDataReader reader = command.ExecuteReader();

            List<string[]> data = new List<string[]>();

            while (reader.Read())
            {
                data.Add(new string[1]);

                data[data.Count - 1][0] = reader[0].ToString(
);
            }

            reader.Close();
            MessageBox.Show("Движение автобусов на маршруте №
" + Convert.ToString(comboBox2.Text) +
                " заканчивается в " + data[0][0], "Информация
о маршруте", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошиб
ка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    if (Convert.ToString(comboBox2.Text) != "всех" &&
        Convert.ToString(comboBox3.Text) == "начинается")

```

```

        {
            string query = "SELECT [Время_начала_движения] " +
                           "FROM [Маршруты] " +
                           "WHERE [Маршрут] = @Маршрут";

            SqlCommand command = new SqlCommand(query, sqlConnect
ion);
            command.Parameters.AddWithValue("Маршрут", comboBox2.
Text);

            try
            {
                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();

                while (reader.Read())
                {
                    data.Add(new string[1]);

                    data[data.Count - 1][0] = reader[0].ToString(
);
                }

                reader.Close();
                MessageBox.Show("Движение автобусов на маршруте №
" + Convert.ToString(comboBox2.Text) +
                             " начинается в " + data[0][0], "Информация о
маршруте", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошиб
ка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        if (Convert.ToString(comboBox2.Text) != "всех" &&
            Convert.ToString(comboBox3.Text) == "начинается и зак
анчивается")
        {
            string query = "SELECT [Время_начала_движения], [Врем
я_окончания_движения] " +
                           "FROM [Маршруты] " +
                           "WHERE [Маршрут] = @Маршрут";

            SqlCommand command = new SqlCommand(query, sqlConnect
ion);
            command.Parameters.AddWithValue("Маршрут", comboBox2.
Text);

```

```

try
{
    SqlDataReader reader = command.ExecuteReader();

    List<string[]> data = new List<string[]>();

    while (reader.Read())
    {
        data.Add(new string[2]);

        data[data.Count - 1][0] = reader[0].ToString(
);
        data[data.Count - 1][1] = reader[1].ToString(
);
    }

    reader.Close();
    MessageBox.Show("Движение автобусов на маршруте №
" + Convert.ToString(comboBox2.Text) +
        " начинается в " + data[0][0] + " и заканчива
ется в " + data[0][1],
        "Информация о маршруте", MessageBoxButtons.OK
, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошиб
ка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

else
{
    MessageBox.Show("Ошибка! Поля '№ маршрута' 'начало/конец'
и должны быть заполнены.",
        "Ошибка", MessageBoxButtons.OK, MessageBoxIcon
n.Error);
    toolStripStatusLabel1.Text = "Поля '№ маршрута' 'начало/к
онец' должны быть заполнены.";
    toolStripStatusLabel1.Visible = true;
}
}

//справка о протяженности маршрута
private void button18_Click(object sender, EventArgs e)
{
    if (toolStripStatusLabel1.Visible)
        toolStripStatusLabel1.Visible = false;
}

```

```

        if (!string.IsNullOrEmpty(textBox4.Text) && !string.IsNullOrEmptySpace(textBox4.Text))
        {
            string query = "SELECT [Протяженность(мин)] " +
                           "FROM [Маршруты] " +
                           "WHERE Маршрут=@Маршрут";

            SqlCommand command = new SqlCommand(query, sqlConnection)
;
            command.Parameters.AddWithValue("Маршрут", textBox4.Text)
;

            try
            {
                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();

                while (reader.Read())
                {
                    data.Add(new string[1]);

                    data[data.Count - 1][0] = reader[0].ToString();
                }

                reader.Close();

                // Создаем документ Word.
                object oMissing = System.Reflection.Missing.Value;
                object oEndOfDoc = "\\endofdoc"; /* \endofdoc is a pr
redefined bookmark */

                //Start Word and create a new document.
                Word._Application oWord;
                Word._Document oDoc;
                oWord = new Word.Application();
                oWord.Visible = true;
                oDoc = oWord.Documents.Add(ref oMissing, ref oMissing
,
                ref oMissing, ref oMissing);

                //Insert a paragraph at the beginning of the document
.

                Word.Paragraph oPara1;
                oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);
                oPara1.Range.Text = "ООО \"АВТОПАРК ПРИКОЛОВ\"";
                oPara1.Range.Font.Bold = 1;
                oPara1.Format.SpaceAfter = 6;      //24 pt spacing afte
r paragraph.

```

```

        oPara1.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;
        oPara1.Range.InsertParagraphAfter();

        //Insert a paragraph at the end of the document.
        Word.Paragraph oPara2;
        object oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).
Range;

        oPara2 = oDoc.Content.Paragraphs.Add(ref oRng);
        oPara2.Range.Text = "С П Р А В К А";
        oPara2.Range.Font.Bold = 1;
        oPara2.Format.SpaceAfter = 40;
        oPara2.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;
        oPara2.Range.InsertParagraphAfter();

        //Insert another paragraph.
        Word.Paragraph oPara3;
        oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
        oPara3 = oDoc.Content.Paragraphs.Add(ref oRng);
        oPara3.Range.Text = "Протяженность маршрута № " + textBox4.Text + " составляет " + data[0][0] + " минут.";
        oPara3.Range.Font.Bold = 0;
        oPara3.Format.SpaceAfter = 30;
        oPara3.Range.InsertParagraphAfter();

        //Insert another paragraph.
        Word.Paragraph oPara4;
        oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
        oPara4 = oDoc.Content.Paragraphs.Add(ref oRng);
        oPara4.Range.Text = "Диспетчер автопарка\t\t\tФИЛИППОВ А.Г.";

        oPara4.Range.Font.Bold = 0;
        oPara4.Format.SpaceAfter = 24;
        oPara4.Range.InsertParagraphAfter();

        MessageBox.Show("Справка готова!", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
else
{

```

```

        MessageBox.Show("Ошибка! Поле '№ маршрута' должен быть з
аполнен.",
                        "Ошибка", MessageBoxButtons.0
К, MessageBoxIcon.Error);
        toolStripStatusLabel1.Text = "Поле '№ маршрута' должен б
ыть заполнен.";
        toolStripStatusLabel1.Visible = true;
    }
}

//полный отчет
private async void button19_Click(object sender, EventArgs e)
{
    int i = 0, avtobusCount = 0;
    int[] typesCount = new int[5];
    string[] types = new string[5];
    types[0] = "малый"; types[1] = "средний"; types[2] = "большой
";
    types[3] = "сочлененный большой"; types[4] = "большой двухъяр
усный";

    foreach (string vs in types)
    {
        string query2 = "SELECT [Автобус], [Тип] " +
            "FROM [Автобусы] " +
            "WHERE Тип = @Тип";

        SqlCommand command1 = new SqlCommand(query2, sqlConnectio
n);
        command1.Parameters.AddWithValue("Тип", vs);

        try
        {
            SqlDataReader reader = await command1.ExecuteReaderAs
ync();

            List<string[]> data = new List<string[]>();

            while (reader.Read())
            {
                data.Add(new string[2]);

                data[data.Count - 1][0] = reader[0].ToString();
                data[data.Count - 1][1] = reader[1].ToString();
            }

            reader.Close();
            typesCount[i] = data.Count();
            i++;
        }
    }
}

```



```

        catch (Exception ex)
        {
            MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    //считать количество автобусов //всех
    foreach (int vs in typesCount)
    {
        avtobusCount += vs;
    }

    //номера маршрутов, время начала движения и интервал
    string query = "SELECT [Маршрут], [Время_начала_движения], [И
        нтервал(мин)] " +
        "FROM [Маршруты]";

    SqlCommand command = new SqlCommand(query, sqlConnection);
    List<string[]> dataMarshruts = new List<string[]>();
    try
    {
        SqlDataReader reader = await command.ExecuteReaderAsync()
;

        while (reader.Read())
        {
            dataMarshruts.Add(new string[3]);

            dataMarshruts[dataMarshruts.Count - 1][0] = reader[0]
.ToString();
            dataMarshruts[dataMarshruts.Count - 1][1] = reader[1]
.ToString();
            dataMarshruts[dataMarshruts.Count - 1][2] = reader[2]
.ToString();
        }

        reader.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
            sageBoxButtons.OK, MessageBoxIcon.Error);
    }

    //ФИО Водителей и их класс
    string query1 = "SELECT [ФИО], [Класс] FROM [Водители]";

```

```

SqlCommand command3 = new SqlCommand(query1, sqlConnection);
List<string[]> dataDrivers = new List<string[]>();
try
{
    SqlDataReader reader = await command3.ExecuteReaderAsync(
);

    while (reader.Read())
    {
        dataDrivers.Add(new string[2]);

        dataDrivers[dataDrivers.Count - 1][0] = reader[0].ToS
tring();
        dataDrivers[dataDrivers.Count - 1][1] = reader[1].ToS
tring();
    }

    reader.Close();

}
catch (Exception ex)
{
    MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
}

// Создаем документ Word.
object oMissing = System.Reflection.Missing.Value;
object oEndOfDoc = "\\endofdoc"; /* \endofdoc is a predefined
bookmark */

//Start Word and create a new document.
Word._Application oWord;
Word._Document oDoc;
oWord = new Word.Application();
oWord.Visible = true;
oDoc = oWord.Documents.Add(ref oMissing, ref oMissing,
ref oMissing, ref oMissing);

//Insert a paragraph at the beginning of the document.
Word.Paragraph oPara1;
oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);
oPara1.Range.Text = "ООО \"АВТОПАРК ПРИКОЛОВ\"";
oPara1.Range.Font.Bold = 1;
oPara1.Format.SpaceAfter = 6;    //6 pt spacing after paragra
ph.
oPara1.Range.ParagraphFormat.Alignment = Word.WdParagraphAlig
nment.wdAlignParagraphCenter;
oPara1.Range.InsertParagraphAfter();

```

```

//Insert a paragraph at the end of the document.
Word.Paragraph oPara2;
object oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara2 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara2.Range.Text = "ПОЛНЫЙ ОТЧЕТ";
oPara2.Range.Font.Bold = 1;
oPara2.Format.SpaceAfter = 40;
oPara2.Range.ParagraphFormat.Alignment = Word.WdParagraphAlign
ment.WdAlignParagraphCenter;
oPara2.Range.InsertParagraphAfter();

//Insert another paragraph.
Word.Paragraph oPara3;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara3 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara3.Range.Text = "Автобусов всего: " + avtobusCount + " из
которых:";
oPara3.Range.Font.Bold = 0;
oPara3.Format.SpaceAfter = 0;
oPara3.Range.ParagraphFormat.Alignment = Word.WdParagraphAlign
ment.WdAlignParagraphLeft;
oPara3.Range.InsertParagraphAfter();

//Insert another paragraph.
Word.Paragraph oPara4;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara4 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara4.Range.Text = typesCount[0] + " " + types[0] + ";";
oPara4.Range.Font.Bold = 0;
oPara4.Format.SpaceAfter = 0;
oPara4.Range.InsertParagraphAfter();

//Insert another paragraph.
Word.Paragraph oPara5;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara5 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara5.Range.Text = typesCount[1] + " " + types[1] + ";";
oPara5.Range.Font.Bold = 0;
oPara5.Format.SpaceAfter = 0;
oPara5.Range.InsertParagraphAfter();

//Insert another paragraph.
Word.Paragraph oPara6;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara6 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara6.Range.Text = typesCount[2] + " " + types[2] + ";";
oPara6.Range.Font.Bold = 0;
oPara6.Format.SpaceAfter = 0;
oPara6.Range.InsertParagraphAfter();

```

```

//Insert another paragraph.
Word.Paragraph oPara7;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara7 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara7.Range.Text = typesCount[3] + " " + types[3] + ";";
oPara7.Range.Font.Bold = 0;
oPara7.Format.SpaceAfter = 0;
oPara7.Range.InsertParagraphAfter();

//Insert another paragraph.
Word.Paragraph oPara8;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara8 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara8.Range.Text = typesCount[4] + " " + types[4] + ";";
oPara8.Range.Font.Bold = 0;
oPara8.Format.SpaceAfter = 24;
oPara8.Range.InsertParagraphAfter();

//Insert another paragraph.
Word.Paragraph oPara9;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara9 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara9.Range.Text = "Парк обслуживает всего " + dataMarshruts
.Count() + " маршрутов. " +
    "Номера маршрутов, время начала движения и интервал в таб
лице ниже.";
oPara9.Range.Font.Bold = 0;
oPara9.Format.SpaceAfter = 6;
oPara9.Range.InsertParagraphAfter();

//маршруты таблица
//Insert a table, fill it with data, and make the first row
//bold.

Word.Table oTable;
Word.Range wrdRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Ra
nge;

oTable = oDoc.Tables.Add(wrdRng, dataMarshruts.Count() + 1, d
ataMarshruts[0].Count(), ref oMissing, ref oMissing);
oTable.Range.ParagraphFormat.SpaceAfter = 0;
oTable.Cell(1, 1).Range.Text = "Номер маршрута";
oTable.Cell(1, 2).Range.Text = "Время начала движения";
oTable.Cell(1, 3).Range.Text = "Интервал(мин)";
int r, c;
string strText;
for (r = 1; r <= dataMarshruts.Count(); r++)
    for (c = 1; c <= dataMarshruts[0].Count(); c++)

```

```

        {
            strText = dataMarshruts[r - 1][c - 1].ToString();
            oTable.Cell(r + 1, c).Range.Text = strText;
        }
oTable.Rows[1].Range.Font.Bold = 1;

//ФИО Водителей и их класс
//Insert another paragraph.
Word.Paragraph oPara10;
oRng = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oPara10 = oDoc.Content.Paragraphs.Add(ref oRng);
oPara10.Range.Text = "\nВ парке работают " + dataDrivers.Count() + " водителей. " +
    "Данные водителей в таблице ниже.";
oPara10.Range.Font.Bold = 0;
oPara10.Format.SpaceAfter = 6;
oPara10.Range.InsertParagraphAfter();

//водители таблица
//Insert a table, fill it with data, and make the first row
//bold.

Word.Table oTable1;
Word.Range wrdRng1 = oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
oTable1 = oDoc.Tables.Add(wrdRng1, dataDrivers.Count() + 1, dataDrivers[0].Count(), ref oMissing, ref oMissing);
oTable1.Range.ParagraphFormat.SpaceAfter = 0;
oTable1.Cell(1, 1).Range.Text = "ФИО водителя";
oTable1.Cell(1, 2).Range.Text = "Класс";
int r1, c1;
string strText1;
for (r1 = 1; r1 <= dataDrivers.Count(); r1++)
    for (c1 = 1; c1 <= dataDrivers[0].Count(); c1++)
    {
        strText1 = dataDrivers[r1 - 1][c1 - 1].ToString();
        oTable1.Cell(r1 + 1, c1).Range.Text = strText1;
    }
oTable1.Rows[1].Range.Font.Bold = 1;

MessageBox.Show("Полный отчет по автопарку готов!", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

private void выполнитьРезервноеКопированиеToolStripMenuItem_Click
(object sender, EventArgs e)
{
    try

```

```

{
    string name = "Avtopark-BackUP";
    string path = @"C:\SQLdata\BACKUPS\";
    // если каталог не существует то создать
    DirectoryInfo dirInfo = new DirectoryInfo(path);
    if (!dirInfo.Exists)
    {
        Directory.CreateDirectory(path);
    }

    // задаем имя для резервной копии
    var backupFileName = String.Format("{0}{1}-{2}.bak",
        path, name, DateTime.Now.ToString("yyyy-MM-dd-hh-mm-ss"));

    var query = String.Format("BACKUP DATABASE [master] TO DI
SK = '{0}'",
        backupFileName);

    SqlCommand command = new SqlCommand(query, sqlConnection)
;
    command.ExecuteNonQuery();

    MessageBox.Show("Резервное копирование базы данных успешн
о выполнено.",
        "Информация", MessageBoxButtons.OK, MessageBo
xIcon.Information);
    toolStripStatusLabel1.Text = "Резервное копирование базы
данных успешно выполнено.";
    toolStripStatusLabel1.Visible = true;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.O
K, MessageBoxIcon.Error);
}
}

private async void восстановитьДанныеИзРезервнойКопииToolStripMen
uItemClick(object sender, EventArgs e)
{
    // Тут откроется форма и если будет нажата кнопка ОК
    // тогда путь запишется в переменную path, а имя файла в back
upFileName.
    OpenFileDialog openFileDialog = new OpenFileDialog();
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        string mainPath = sqlConnection.Database;

        try
        {

```



```

{
    public partial class loginForm : Form
    {
        public loginForm()
        {
            InitializeComponent();
            //автозагрузка логина и пароля))
            comboBox1.Text = "Диспетчер";
            textBox2.Text = "dispetcher";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if ((!string.IsNullOrEmpty(comboBox1.Text) && !string.IsNullOrEmpty(
rWhiteSpace(comboBox1.Text)) &&
                (!string.IsNullOrEmpty(textBox2.Text) && !string.IsNullOrEmpty(
WhiteSpace(textBox2.Text))))
            {
                if (Convert.ToString(comboBox1.Text) == "Диспетчер")
                {
                    if (Convert.ToString(textBox2.Text) == "dispetcher")
                    {
                        this.Hide();
                        MainForm form = new MainForm();
                        form.Show();
                    }
                    else
                    {
                        MessageBox.Show("Ошибка, неверный пароль", "Ошибк
a", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
            else
            {
                MessageBox.Show("Ошибка, поля 'Логин' и 'Пароль' не должн
ы быть пустыми.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}

```



#### Листинг 4. FILE: DvijCombo.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class DvijCombo : Form
    {
        public DvijCombo()
        {
            InitializeComponent();
            LoadData();
        }

        private void LoadData()
        {
            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;A
ttachDbFilename=" +
                @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Securit
y=True";

            SqlConnection myConnection = new SqlConnection(connectionString)
;

            myConnection.Open();

            string query = "SELECT [Маршрут], [Время_начала_движения], [В
ремя_окончания_движения] " +
                "FROM [Маршруты]";

            SqlCommand command = new SqlCommand(query, myConnection);

            try
            {
                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();

                while (reader.Read())
                {
                    data.Add(new string[3]);
                }
            }
        }
    }
}
```

```

        data[data.Count - 1][0] = reader[0].ToString();
        data[data.Count - 1][1] = reader[1].ToString();
        data[data.Count - 1][2] = reader[2].ToString();
    }

    reader.Close();

    myConnection.Close();

    foreach (string[] s in data)
        dataGridView1.Rows.Add(s);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
    }
    }
}

```

## Листинг 5. FILE: DvijEnd.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class DvijenieEnd : Form
    {
        public DvijenieEnd()
        {
            InitializeComponent();
            LoadData();
        }
        private void LoadData()
        {
            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;A
ttachDbFilename=" +

```

```

        @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Security=True";

        SqlConnection myConnection = new SqlConnection(connectionString);

        myConnection.Open();

        string query = "SELECT [Маршрут], [Время_окончания_движения]
        " +
            "FROM [Маршруты]";

        SqlCommand command = new SqlCommand(query, myConnection);

        try
        {
            SqlDataReader reader = command.ExecuteReader();

            List<string[]> data = new List<string[]>();

            while (reader.Read())
            {
                data.Add(new string[2]);

                data[data.Count - 1][0] = reader[0].ToString();
                data[data.Count - 1][1] = reader[1].ToString();
            }

            reader.Close();

            myConnection.Close();

            foreach (string[] s in data)
                dataGridView1.Rows.Add(s);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

## Листинг 6. FILE: DvijenieStart.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class DvijenieStart : Form
    {
        public DvijenieStart()
        {
            InitializeComponent();
            LoadData();
        }

        private void LoadData()
        {
            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;A
ttachDbFilename=" +
                @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Securit
y=True";

            SqlConnection myConnection = new SqlConnection(connectionString)
;

            myConnection.Open();

            string query = "SELECT [Маршрут], [Время_начала_движения] " +
                "FROM [Маршруты]";

            SqlCommand command = new SqlCommand(query, myConnection);

            try
            {
                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();

                while (reader.Read())
                {
                    data.Add(new string[2]);
                }
            }
        }
    }
}
```

```

        data[data.Count - 1][0] = reader[0].ToString();
        data[data.Count - 1][1] = reader[1].ToString();
    }

    reader.Close();

    myConnection.Close();

    foreach (string[] s in data)
        dataGridView1.Rows.Add(s);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
    }
    }
}
}

```

## Листинг 7. FILE: MarshrutProtejennost.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class MarshrutProtejennost : Form
    {
        public MarshrutProtejennost()
        {
            InitializeComponent();
            LoadData();
        }

        private void LoadData()
        {
            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;A
ttachDbFilename=" +

```

```

        @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Security=True";

        SqlConnection myConnection = new SqlConnection(connectString)
;

        myConnection.Open();

        string query = "SELECT [Маршрут], [Протяженность(мин)] " +
            "FROM [Маршруты]";

        SqlCommand command = new SqlCommand(query, myConnection);

        try
        {
            SqlDataReader reader = command.ExecuteReader();

            List<string[]> data = new List<string[]>();

            while (reader.Read())
            {
                data.Add(new string[2]);

                data[data.Count - 1][0] = reader[0].ToString();
                data[data.Count - 1][1] = reader[1].ToString();
            }

            reader.Close();

            myConnection.Close();

            foreach (string[] s in data)
                dataGridView1.Rows.Add(s);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
}

```

## Листинг 8. FILE: VoditeliMarshruta.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class VoditeliMarshruta : Form
    {
        public VoditeliMarshruta(string marshrut)
        {
            InitializeComponent();
            LoadData(marshrut);
        }

        private void LoadData(string marshrut)
        {
            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +
                @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Security=True";

            SqlConnection myConnection = new SqlConnection(connectionString);

            myConnection.Open();

            string query = "SELECT [Паспорт], [ФИО] " +
                "FROM [Водят] " +
                "WHERE [Маршрут] = @Маршрут " +
                "ORDER BY [ФИО] ASC";

            SqlCommand command = new SqlCommand(query, myConnection);
            command.Parameters.AddWithValue("Маршрут", marshrut);

            try
            {
                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();
            }
        }
    }
}
```

```

        while (reader.Read())
        {
            data.Add(new string[2]);

            data[data.Count - 1][0] = reader[0].ToString();
            data[data.Count - 1][1] = reader[1].ToString();
        }

        reader.Close();

        myConnection.Close();

        foreach (string[] s in data)
            dataGridView1.Rows.Add(s);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
}

```

## Листинг 9. FILE: VoditeliNeVodyat.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class VoditeliNeVodyat : Form
    {
        public VoditeliNeVodyat()
        {
            InitializeComponent();
            LoadData();
        }
    }
}

```



```

private async void LoadData()
{
    string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;A
ttachDbFilename=" +
        @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Securit
y=True";

    SqlConnection myConnection = new SqlConnection(connectionString)
;
    myConnection.Open();

    string query = "SELECT* " +
        "FROM [Водители] " +
        "WHERE Водители.Паспорт NOT IN (SELECT Паспорт "
+
        "FROM Водят)";

    SqlCommand command = new SqlCommand(query, myConnection);

    try
    {
        SqlDataReader reader = await command.ExecuteReaderAsync()
;

        List<string[]> data = new List<string[]>();

        while (reader.Read())
        {
            data.Add(new string[5]);

            data[data.Count - 1][0] = reader[0].ToString();
            data[data.Count - 1][1] = reader[1].ToString();
            data[data.Count - 1][2] = reader[2].ToString();
            data[data.Count - 1][3] = reader[3].ToString();
            data[data.Count - 1][4] = reader[4].ToString();
        }

        reader.Close();
        myConnection.Close();

        foreach (string[] s in data)
            dataGridView1.Rows.Add(s);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
}

```

## Листинг 10. FILE: AvtobusiMarshruta.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class AvtobusiMarshruta : Form
    {
        public AvtobusiMarshruta(string marshrut)
        {
            InitializeComponent();
            LoadData(marshrut);
        }

        private void LoadData(string marshrut)
        {
            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;A
ttachDbFilename=" +
                @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Securit
y=True";

            SqlConnection myConnection = new SqlConnection(connectionString)
;

            myConnection.Open();

            string query = "SELECT Автобус, Тип " +
                "FROM [Автобусы] " +
                "WHERE Автобусы.Автобус IN (SELECT Автобус " +
                "FROM Водят " +
                "WHERE Маршрут = @Мар
шрут)";

            SqlCommand command = new SqlCommand(query, myConnection);
            command.Parameters.AddWithValue("Маршрут", marshrut);

            try
            {
                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();
            }
        }
    }
}
```

```

        while (reader.Read())
        {
            data.Add(new string[2]);

            data[data.Count - 1][0] = reader[0].ToString();
            data[data.Count - 1][1] = reader[1].ToString();
        }

        reader.Close();

        myConnection.Close();

        foreach (string[] s in data)
            dataGridView1.Rows.Add(s);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
}

```

#### Листинг 11. FILE: AvtobusiNeEzdyat.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    public partial class AvtobusiNeEzdyat : Form
    {
        public AvtobusiNeEzdyat()
        {
            InitializeComponent();
            LoadData();
        }
    }
}

```

```

private async void LoadData()
{
    string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;A
ttachDbFilename=" +
        @"C:\winformsDB\Avtopark\Database1.mdf;Integrated Securit
y=True";

    SqlConnection myConnection = new SqlConnection(connectionString)
;

    myConnection.Open();

    string query = "SELECT* " +
        "FROM [Автобусы] " +
        "WHERE Автобусы.Автобус NOT IN (SELECT Автобус "
+
        "FROM Водят)";

    SqlCommand command = new SqlCommand(query, myConnection);

    try
    {
        SqlDataReader reader = await command.ExecuteReaderAsync()
;

        List<string[]> data = new List<string[]>();

        while (reader.Read())
        {
            data.Add(new string[2]);

            data[data.Count - 1][0] = reader[0].ToString();
            data[data.Count - 1][1] = reader[1].ToString();

        }

        reader.Close();

        myConnection.Close();

        foreach (string[] s in data)
            dataGridView1.Rows.Add(s);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка", Mes
sageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}

```

## Листинг 12. FILE: AboutBox1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Linq;
using System.Reflection;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avtopark
{
    partial class AboutBox1 : Form
    {
        public AboutBox1()
        {
            InitializeComponent();
            this.Text = "О программе Автопарк";
            this.labelProductName.Text = "Название продукта: Автопарк";
            this.labelVersion.Text = "Версия 1.0.0.0";
            this.labelCopyright.Text = "Авторские права: (с) Артур Нерсис
ян 2020";
            this.labelCompanyName.Text = "Название организации: СПбГЭТУ '
ЛЭТИ'";
            this.textBoxDescription.Text = "Описание \n Данная программа
была разработана в рамках курсового проекта по дисциплине 'Управление дан
ными' в 2020 году.";
        }

        private void okButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

## ПРИЛОЖЕНИЕ Г. ПРИМЕРЫ СГЕНЕРИРОВАННЫХ ДОКУМЕНТОВ

### 1. Сгенерированная справка о протяженности маршрута.

===== Граница документа =====

**ООО "АВТОПАРК ПРИКОЛОВ"**

**С П Р А В К А**

Протяженность маршрута № 6 составляет 50 минут.

Диспетчер автопарка

ФИЛИППОВ А.Г.

===== Граница документа =====

## 2. Сгенерированный полный отчет по автопарку.

===== Граница документа =====

### ООО "АВТОПАРК ПРИКОЛОВ"

#### ПОЛНЫЙ ОТЧЕТ

Автобусов всего: 27 из которых:

6 малый;

1 средний;

7 большой;

6 сочлененный большой;

7 большой двухъярусный;

Парк обслуживает всего 9 маршрутов. Номера маршрутов, время начала движения и интервал в таблице ниже.

Номер маршрута	Время начала движения	Интервал(мин)
1	06:00:00	30
2	08:00:00	35
3	08:00:00	28
4	06:00:00	30
5	08:00:00	35
6	06:00:00	50
7	08:00:00	20
8	06:00:00	20
9	06:00:00	26

В парке работают 26 водителей. Данные водителей в таблице ниже.

<b>ФИО водителя</b>	<b>Класс</b>
Смирнов Алекс Владимирович	С
Стрелков Алексей Даниилович	А
Стрелков Алексей Даниилович	А
Стрелков Алексей Даниилович	А
Орехов Болеслав Проклович	В
Лапин Нинель Федотович	С
Мамонтов Эрик Владиславович	В
Шилов Оскар Феликсович	А
Филиппов Герасим Олегович	С
Дьячков Афанасий Аристархович	А
Куликов Овидий Евсеевич	А
Поляков Мечислав Русланович	А
Кондратьев Роман Тимурович	С
Степанов Клим Созонович	В
Ковалёв Касьян Яковович	С
Рябов Владлен Георгиевич	А
Петухов Василий Серапионович	В
Шилов Филипп Эльдарович	А
Ширяев Алан Владиславович	А
Новиков Влас Протасьевич	С
Михеев Власий Ефимович	А
Самсонов Егор Мэлсович	В
Фомичёв Эдуард Вениаминович	А
Осипов Флор Андреевич	В
Быков Рубен Егорович	А
Хохлов Остап Лаврентьевич	С