

Федеральное государственное образовательное бюджетное учреждение  
высшего образования  
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ  
РОССИЙСКОЙ ФЕДЕРАЦИИ»  
(Финансовый университет)

**Колледж информатики и программирования**

ПМ.11 Разработка,  
администрирование и защита баз  
данных

УТВЕРЖДАЮ

Председатель предметно-цикловой  
комиссии информационных систем  
и программирования

Группа: ЗИСИП-522

\_\_\_\_\_/Т.Г. Аксёнова/

«\_\_\_\_» \_\_\_\_\_ 2025 г.

**КУРСОВОЙ ПРОЕКТ**

На тему: Проектирование и разработка базы данных для автосервиса

Руководитель курсового проекта

\_\_\_\_\_. Н.А. Хасанова

Исполнитель курсового проекта

\_\_\_\_\_. В.А. Ватаманюк

Оценка за проект: \_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_ 2025 г.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
Глава 1. Предпроектное исследование предметной области.....	5
1.1. Описание предметной области .....	5
1.2. Постановка задачи.....	5
1.3. Характеристика инструментальных средств разработки.....	6
Глава 2. Проектирование и разработка базы данных. ....	8
2.1. Проектирование базы данных.....	8
2.2. Разработка базы данных и интерфейса .....	13
2.3. Отладка и тестирование.....	25
2.4. Руководство администратора базы данных .....	29
ЗАКЛЮЧЕНИЕ .....	36
СПИСОК ЛИТЕРАТУРЫ (ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ) И ИНТЕРНЕТ – РЕСУРСОВ.....	37
ПРИЛОЖЕНИЕ .....	39

## **ВВЕДЕНИЕ**

Современная экономическая ситуация, характеризующаяся кризисными явлениями, оказывает значительное влияние на автомобильную отрасль. Старение автомобильного парка приводит к более сложному и затратному ремонту, а также к увеличению требований к качеству обслуживания. При этом наблюдается рост числа автосервисов, что создает высокую конкуренцию. В таких условиях компаниям необходимо проявлять гибкость и обеспечивать высокое качество сервиса, учитывая потребности клиентов.

Управление клиентами в автосервисах представляет собой сложную задачу. Для повышения уровня обслуживания важно внедрение современных информационных технологий, включая автоматизированные системы управления данными.

Оптимальным решением является разработка комплексной базы данных, содержащей актуальную информацию о клиентах, услугах, финансовых операциях и наличии запчастей. Это позволит оптимизировать внутренние процессы и улучшить взаимодействие с клиентами, что повысит уровень удовлетворенности и укрепит позиции компании на рынке.

Основной целью курсовой работы является проектирование и разработка базы данных для автосервиса, которая поможет улучшить управление информацией о клиентах, запасных частях, оказываемых услугах и проведенных ремонтных работах.

В рамках выполнения проекта необходимо решить следующие задачи:

- провести анализ потребностей автосервиса и выявить ключевые сущности, а также установить взаимосвязи между ними;
- спроектировать структуру базы данных таким образом, чтобы она могла хранить информацию о клиентах, автомобилях, заказах, выполненных работах и запасных частях;
- создать схему базы данных, включающую необходимые таблицы, поля и связи между ними;

- разработать скрипты для создания базы данных и заполнить её тестовыми данными;
- разработать интерфейс для взаимодействия с базой данных, позволяющий пользователям добавлять, редактировать и удалять записи.

Объектом исследования в данном проекте является автосервис как организация, предоставляющая услуги по обслуживанию автомобилей. Предметом исследования служит процесс разработки и проектирования базы данных для управления информацией об услугах, клиентах, заказах, запчастях и выполненных работах. В ходе выполнения проекта будут применены методы анализа требований, проектирования баз данных и разработки программного обеспечения.

Актуальность данной работы определяется необходимостью улучшения эффективности работы автосервисов посредством оптимизации управления данными. В условиях высокой конкуренции на рынке автосервисных услуг очень важно иметь систему, которая способна эффективно обрабатывать заказы, отслеживать выполнение работ и управлять запасами запчастей.

Создаваемая база данных будет использоваться для хранения информации о клиентах, автомобилях, заказах, выполненных работах и запчастях автосервиса. Это обеспечит быстрый доступ к необходимым данным, удобный ввод новых записей и возможность анализа продуктивности работы сервиса.

Для проектирования базы данных будет использован язык программирования C#. Для создания и администрирования базы данных планируется использование инструментов, таких как SQL Server Management Studio.

## **Глава 1. Предпроектное исследование предметной области**

### **1.1. Описание предметной области**

База данных разрабатывается для автосервисов, которые предоставляют разнообразные услуги по техническому обслуживанию автомобилей. В ней хранится информация о клиентах, их транспортных средствах, заказах, сотрудниках, предоставляемых услугах, запчастях, складских остатках, выполненных работах, платежах и отзывах клиентов. Клиенты оформляют заказы на обслуживание или ремонт своих машин, выбирая необходимые услуги и запчасти. Работники выполняют задания по этим заказам, включая различные типы ремонтных операций. Оплата осуществляется за оказанные услуги и использованные комплектующие. После завершения работ клиенты могут оставить свои отзывы.

### **1.2. Постановка задачи**

База данных должна учитывать категории пользователей:

- Администратор

Администратор обладает полным доступом ко всем функциям управления базой данных, включая настройку и редактирование информации.

Необходимо разработать систему отчетов, которая позволит отслеживать данные о транспортных средствах и оформленных заказах.

Система рассчитана на работу под управлением операционных систем семейства Windows, начиная от версии 10 и выше.

Первоначально информация об автомобилях и их владельцах существует независимо друг от друга. Связь между владельцем и автомобилем устанавливается только в момент создания заказа, после чего она сохраняется в базе данных. Такой подход позволяет избежать возможных сложностей, связанных с наличием у одного владельца нескольких транспортных средств.

Для удобства управления информацией будет представлен определенный список отчетов.

В таблице 1 представлена структура отчета «Заказы».

Таблица 1 – Отчет по заказам

№ Заказа	Клиент	Марка	Модель	Описание	Статус

В таблице 1.1 представлена структура отчета «Транспортные средства».

Таблица 1.1 – Отчет по транспортным средствам

Марка	Модель	VIN	Владелец	Кол-во заказов

### 1.3. Характеристика инструментальных средств разработки

Для разработки и проектирования базы данных будет использована СУБД Microsoft SQL Server Management Studio 2022. Минимальные технические требования для этой программы включают аппаратные и программные характеристики. Аппаратные требования: 64-разрядный процессор с тактовой частотой 1,4 ГГц или выше, минимум 2 ГБ оперативной памяти (рекомендуется 4 ГБ и более), не менее 6 ГБ свободного места на жестком диске для установки, а также сетевое подключение для активации и обновлений. Программные требования: поддерживаемая операционная система — Windows 10. Безопасность обеспечивается благодаря встроенным средствам аутентификации и авторизации, ролям пользователей и приложений, механизму фильтрации данных, шифрованию данных и оценке уязвимостей.

Графический интерфейс будет создан с помощью языка программирования C# и технологии WPF. C# – объектно-ориентированный язык программирования, созданный компанией Microsoft для разработки приложений, однако с появлением платформы .NET он стал кросс-платформенным. Технология WPF (Windows Presentation Foundation) входит в состав платформы .NET и используется для создания графических интерфейсов. Благодаря использованию набора API DirectX, приложения на основе WPF могут использовать вычислительные мощности графического процессора, обеспечивая аппаратное ускорение графики.

В качестве среды разработки будет применяться Microsoft Visual Studio. Эта платформа предоставляет множество продвинутых функций, таких как удобный интерфейс, поддержка различных языков программирования, помощь в написании и редактировании кода (IntelliSense) и мощные возможности отладки.

Для подготовки пояснительной записки и презентации к защите будут использованы продукты Microsoft Office: Microsoft Word и PowerPoint. Эти программы предоставляют все необходимые функции и широко распространены. Для создания схем, макетов и диаграмм выбран онлайн-сервис Draw.io, предлагающий широкий набор инструментов и возможностей.

## Глава 2. Проектирование и разработка базы данных.

### 2.1. Проектирование базы данных

Построение инфологической модели.

На рисунке 2.1 представлена инфологическая модель базы данных:

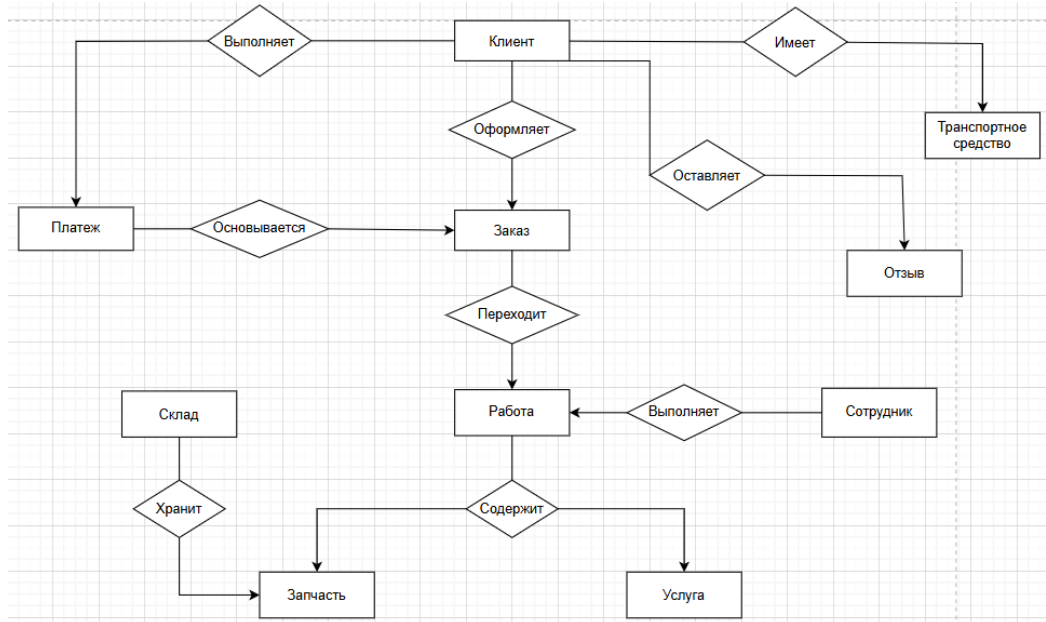


Рисунок 2.1 – Инфологическая модель базы данных

Построение даталогической модели.

Подробное описание связей между сущностями:

– Клиенты (Customers) ↔ Отзывы (Feedback):

- 1) тип связи: один ко многим (1:N). Один клиент может оставить несколько отзывов;
- 2) описание: связь между клиентом и оставленными им отзывами;
- 3) реализация: в таблице Feedback поле CustomerID является внешним ключом, ссылающимся на Customers.CustomerID;
- 4) ограничения: удаление клиента возможно, только если нет связанных с ним отзывов;
- 5) особенности: связь обеспечивает отслеживание отзывов по клиентам.

– Клиенты (Customers) ↔ Заказы (Orders):

- 1) тип связи: один ко многим (1:N). Один клиент может сделать несколько заказов;



- 2) описание: отражает историю заказов каждого клиента;
- 3) реализация: в таблице Orders поле CustomerID является внешним ключом, ссылающимся на Customers.CustomerID;
- 4) ограничения: удаление клиента невозможно, если есть связанные с ним заказы;
- 5) особенности: Связь необходима для управления историей заказов клиентов.

– Заказы (Orders) ↔ Платежи (Payments):

- 1) тип связи: один ко многим (1:N).
- 2) описание: к одному заказу может быть сделано несколько платежей.
- 3) реализация: в таблице Payments поле OrderID является внешним ключом, ссылающимся на Orders.OrderID.

– Клиенты (Customers) ↔ Транспортные средства (Vehicles):

- 1) тип связи: один ко многим (1:N). Один клиент может иметь несколько транспортных средств;
- 2) описание: указывает, какие транспортные средства принадлежат клиентам;
- 3) реализация: в таблице Vehicles поле CustomerID является внешним ключом, ссылающимся на Customers.CustomerID;
- 4) ограничения: удаление клиента невозможно, если есть связанные с ним транспортные средства;
- 5) особенности: позволяет связывать транспортные средства с их владельцами (клиентами).

– Транспортные средства (Vehicles) ↔ Заказы (Orders):

- 1) тип связи: один ко многим (1:N). Одно транспортное средство может быть связано с несколькими заказами;
- 2) описание: отображает информацию по каждому заказу, относящееся к конкретному транспортному средству;

- 3) реализация: в таблице Orders поле VehicleID является внешним ключом, ссылающимся на Vehicles.VehicleID;
- 4) ограничения: удаление транспортного средства невозможно, если есть связанные с ним заказы;
- 5) особенности: отображает историю использования транспортных средств в заказах.

– Сотрудники (Employees) ↔ Работа (Work):

- 1) тип связи: один ко многим (1:N). Один сотрудник может выполнять много работ;
- 2) описание: связывает сотрудников с выполняемой работой;
- 3) реализация: В таблице Work поле EmployeeID является внешним ключом, ссылающимся на Employees.EmployeeID;
- 4) ограничения: удаление сотрудника невозможно, если есть связанные с ним записи в Work;
- 5) особенности: связь позволяет отслеживать производительность сотрудников.

– Склад (Inventory) ↔ Запчасти (Parts):

- 1) тип связи: один ко многим (1:N). Один склад может содержать несколько запчастей (Parts);
- 2) описание: показывает, какие запчасти хранятся на каком складе;
- 3) реализация: в таблице Parts поле InventoryID является внешним ключом, ссылающимся на Inventory.InventoryID;
- 4) ограничения: удаление склада невозможно, если в нем есть связанные запчасти;
- 5) особенности: позволяет отслеживать количество и размещение запчастей.

– Работа (Work) ↔ Заказы (Orders):

- 1) тип связи: один ко многим (1:N). Одна работа может быть связана с несколькими заказами;

- 2) описание: связывает работу с заказами;
- 3) реализация: в таблице Orders поле WorkID является внешним ключом, ссылающимся на Work.WorkID;
- 4) ограничения: удаление работы невозможно, если есть связанные с ней заказы;
- 5) особенности: позволяет отслеживать работы, выполняемые в рамках конкретного заказа.

– Работа (Work) ↔ Платежи (Payments):

- 1) тип связи: один ко многим (1:N). Одна работа может быть связана с несколькими платежами;
- 2) описание: связывает работу с платежами;
- 3) реализация: в таблице Payments поле WorkID является внешним ключом, ссылающимся на Work.WorkID;
- 4) ограничения: удаление работы невозможно, если есть связанные с ней платежи;
- 5) особенности: позволяет отслеживать оплаты по каждой выполненной работе.

– Работа (Work) ↔ Запчасти (Parts):

- 1) тип связи: один ко многим (1:N). Одна работа может использовать много запчастей (Parts);
- 2) описание: связывает работу с запчастями;
- 3) реализация: в таблице Parts поле WorkID является внешним ключом, ссылающимся на Work.WorkID;
- 4) ограничения: удаление работы невозможно, если есть связанные с ней запчасти;
- 5) особенности: позволяет отслеживать расход запчастей на каждую работу.

– Работа (Work) ↔ Услуги (Services):

- 1) тип связи: один ко многим (1:N). Одна работа может быть связана с несколькими услугами (Services);
- 2) описание: связывает работу с услугами;
- 3) реализация: в таблице Services поле WorkID является внешним ключом, ссылающимся на Work.WorkID;
- 4) ограничения: удаление работы невозможно, если есть связанные с ней услуги;
- 5) особенности: позволяет отслеживать, какие услуги были оказаны при выполнении работы.

– Клиенты (Customers) ↔ Платежи (Payments):

- 1) тип связи: один ко многим (1:N);
- 2) описание: один клиент может совершить несколько платежей;
- 3) реализация: в таблице Payments поле CustomerID является внешним ключом, ссылающимся на Customers.CustomerID;
- 4) особенности: позволяет связать платежи с конкретным клиентом.

На рисунке 2.2 изображена даталогическая модель:

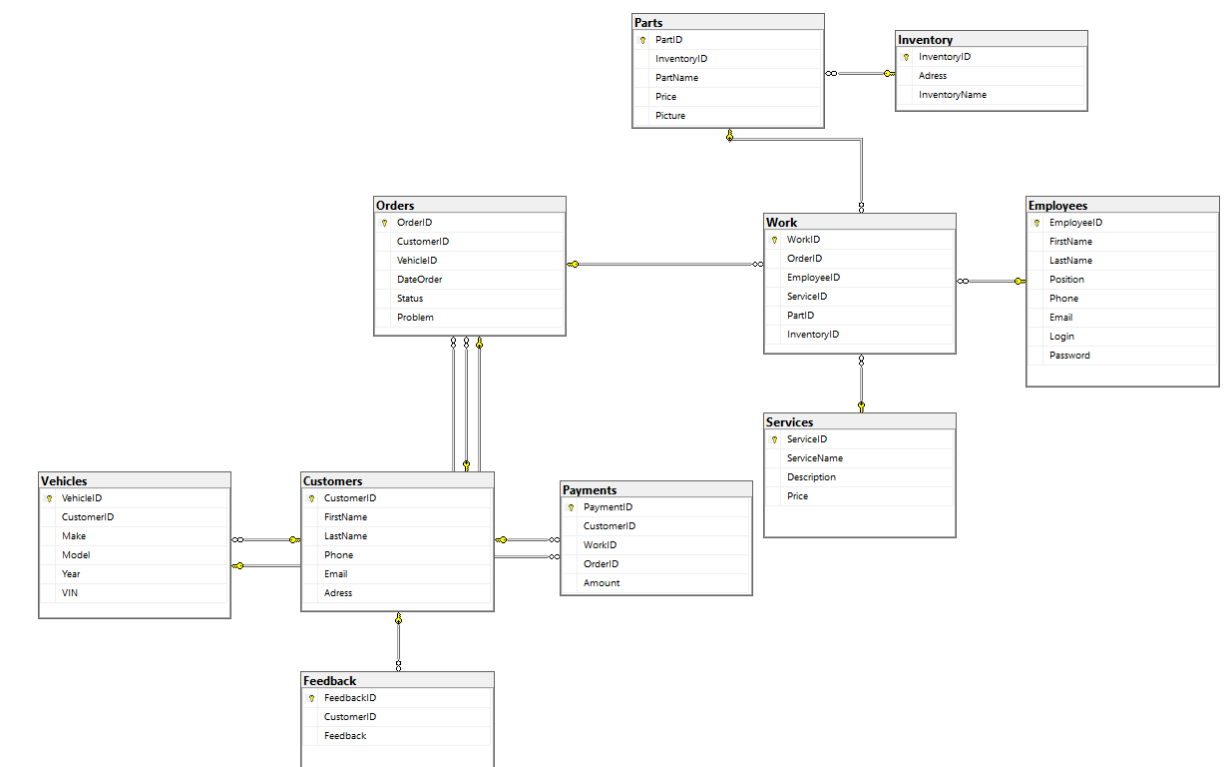


Рисунок 2.2 – Даталогическая модель базы данных

## Моделирование бизнес-процессов.

Рисунки 2.3 и 2.4 демонстрируют основные события при создании заказа и выполнении работы.

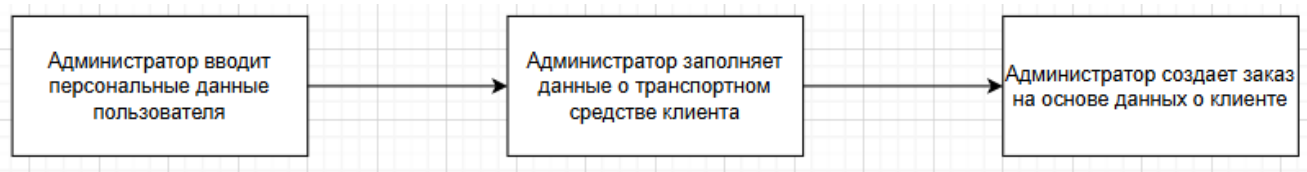


Рисунок 2.3 – Бизнес-процессы создания заказа

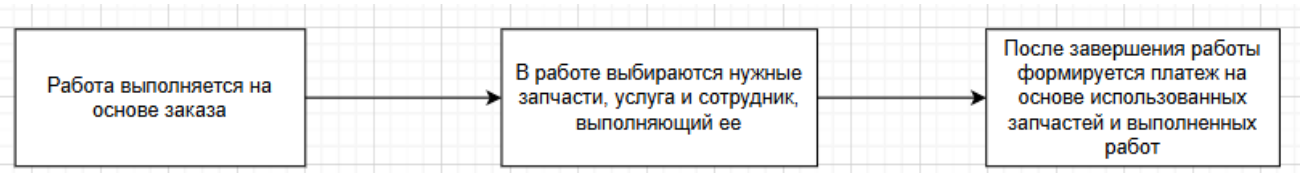


Рисунок 2.4 – Бизнес-процессы выполнения работы

## 2.2. Разработка базы данных и интерфейса

В ходе разработки базы данных было создано 10 таблиц.

В таблице «Customers» (рисунок 2.5) указаны персональные данные клиента. В таблице есть: CustomerID (первичный ключ), FirstName (имя клиента), LastName (фамилия клиента), Phone (номер телефона клиента), Email (электронная почта клиента), Address (адрес клиента).

LAPTOP-23ORVT5S....e - dbo.Customers			
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	CustomerID	int	<input type="checkbox"/>
	FirstName	nvarchar(50)	<input type="checkbox"/>
	LastName	nvarchar(50)	<input type="checkbox"/>
	Phone	nvarchar(50)	<input type="checkbox"/>
	Email	nvarchar(50)	<input type="checkbox"/>
	Address	nvarchar(100)	<input type="checkbox"/>

Рисунок 2.5 – Таблица «Customers»

В таблице «Employees» (рисунок 2.6) хранится информация о сотрудниках. В таблице есть: EmployeeID (первичный ключ), FirstName (имя сотрудника), LastName (фамилия сотрудника), Position (должность), Phone (номер телефона сотрудника), Email (электронная почта сотрудника), Login (логин для входа), Password (пароль для входа).

LAPTOP-23ORVT5S....e - dbo.Employees			
LAPTOP-23ORVT5S....e - dbo.Customers			
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	EmployeeID	int	<input type="checkbox"/>
	FirstName	nvarchar(50)	<input type="checkbox"/>
	LastName	nvarchar(50)	<input type="checkbox"/>
	Position	nvarchar(50)	<input type="checkbox"/>
	Phone	nvarchar(50)	<input type="checkbox"/>
	Email	nvarchar(50)	<input type="checkbox"/>
	Login	nvarchar(50)	<input checked="" type="checkbox"/>
	Password	nvarchar(50)	<input checked="" type="checkbox"/>

Рисунок 2.6 – Таблица «Employees»

В таблице «Feedback» (рисунок 2.7) есть: FeedbackID (первичный ключ), CustomerID (внешний ключ из таблицы «Customers»), Feedback (отзыв клиента).

LAPTOP-23ORVT5S....ce - dbo.Feedback			
LAPTOP-23ORVT5S....e - dbo.Employees			
LAPTOP-23ORVT5S....ce - dbo.Feedback			
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	FeedbackID	int	<input type="checkbox"/>
	CustomerID	int	<input type="checkbox"/>
	Feedback	nvarchar(300)	<input type="checkbox"/>

Рисунок 2.7 – Таблица «Feedback»

В таблице «Inventory» (рисунок 2.8) есть: InventoryID (первичный ключ), Address (адрес склада), InventoryName (название склада).

LAPTOP-23ORVT5S.C...e - dbo.Inventory			
LAPTOP-23ORVT5S....ce - dbo.Feedback			
LAPTOP-23ORVT5S.C...e - dbo.Inventory			
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	InventoryID	int	<input type="checkbox"/>
	Address	nvarchar(100)	<input type="checkbox"/>
	InventoryName	nvarchar(50)	<input type="checkbox"/>

Рисунок 2.8 – Таблица «Inventory»

В таблице «Orders» (рисунок 2.9) хранится информация о заказах. В таблице есть: OrderID (первичный ключ), CustomerID (внешний ключ из таблицы «Customers»), VehicleID (внешний ключ из таблицы «Vehicles»), DateOrder (дата заказа), Status (статус выполнения), Problem (описание проблемы).

LAPTOP-23ORVT5S.C...rvice - dbo.Orders		LAPTOP-23ORVT5S.C...e - dbo.Inventory	
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	OrderID	int	<input type="checkbox"/>
	CustomerID	int	<input type="checkbox"/>
	VehicleID	int	<input type="checkbox"/>
	DateOrder	datetime	<input type="checkbox"/>
	Status	nvarchar(50)	<input type="checkbox"/>
	Problem	text	<input type="checkbox"/>

Рисунок 2.9 – Таблица «Orders»

В таблице «Parts» (рисунок 2.10) хранится информация о запчастях. В таблице есть: PartID (первичный ключ), InventoryID (внешний ключ из таблицы «Inventory»), PartName (название запчасти), Price (стоимость), Picture (картинка).

LAPTOP-GUIKJIL.Car_service - dbo.Parts			
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	PartID	int	<input type="checkbox"/>
	InventoryID	int	<input type="checkbox"/>
	PartName	nvarchar(50)	<input type="checkbox"/>
	Price	decimal(18, 0)	<input type="checkbox"/>
	Picture	nvarchar(255)	<input checked="" type="checkbox"/>

Рисунок 2.10 – Таблица «Parts»

В таблице «Payments» (рисунок 2.11) есть: PaymentID (первичный ключ), CustomerID (внешний ключ из таблицы «Customers»), WorkID (внешний ключ из таблицы «Work»), OrderID (внешний ключ из таблицы «Orders»), Amount (сумма оплаты).

LAPTOP-23ORVT5S....ce - dbo.Payments			
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	PaymentID	int	<input type="checkbox"/>
	CustomerID	int	<input type="checkbox"/>
	WorkID	int	<input type="checkbox"/>
	OrderID	int	<input type="checkbox"/>
	Amount	decimal(18, 0)	<input type="checkbox"/>

Рисунок 2.11 – Таблица «Payments»

В таблице «Services» (рисунок 2.12) хранится информация об услугах. В таблице есть: ServiceID (первичный ключ), ServiceName (название услуги), Description (описание), Price (стоимость).

LAPTOP-23ORVT5S.C...ice - dbo.Services		LAPTOP-23ORVT5S.C...ervice - dbo.Parts		LAPTOP-23ORVT5S.C...ice - dbo.Vehicles
	Имя столбца	Тип данных	Разрешить значения NULL	
🔑	ServiceID	int	<input type="checkbox"/>	
	ServiceName	nvarchar(50)	<input type="checkbox"/>	
	Description	text	<input type="checkbox"/>	
	Price	decimal(18, 0)	<input type="checkbox"/>	

Рисунок 2.12 – Таблица «Services»

В таблице «Vehicles» (рисунок 2.13) есть: VehicleID (первичный ключ), CustomerID (внешний ключ из таблицы «Customers»), Make (марка транспортного средства), Model (модель), Year (год выпуска), VIN (VIN- номер).

LAPTOP-23ORVT5S.C...ice - dbo.Vehicles		LAPTOP-23ORVT5S.C...ice - dbo.Services		LAPTOP-23ORVT5S.C...ice - dbo.Vehicles
	Имя столбца	Тип данных	Разрешить значения NULL	
🔑	VehicleID	int	<input type="checkbox"/>	
	CustomerID	int	<input type="checkbox"/>	
	Make	nvarchar(50)	<input type="checkbox"/>	
	Model	nvarchar(50)	<input type="checkbox"/>	
	Year	int	<input type="checkbox"/>	
	VIN	nvarchar(50)	<input type="checkbox"/>	

Рисунок 2.13 – Таблица «Vehicles»

В таблице «Work» (рисунок 2.14) есть: WorkID (первичный ключ), OrderID (внешний ключ из таблицы «Orders»), EmployeeID (внешний ключ из таблицы «Employees»), ServiceID (внешний ключ из таблицы «Services»), PartID (внешний ключ из таблицы «Parts»), InventoryID (внешний ключ из таблицы «Inventory»).

LAPTOP-23ORVT5S.C...ervice - dbo.Work			
	Имя столбца	Тип данных	Разрешить значения NULL
🔑	WorkID	int	<input type="checkbox"/>
	OrderID	int	<input type="checkbox"/>
	EmployeeID	int	<input type="checkbox"/>
	ServiceID	int	<input type="checkbox"/>
	PartID	int	<input type="checkbox"/>
	InventoryID	int	<input type="checkbox"/>

Рисунок 2.14 – Таблица «Work»

База данных была подключена к WPF-приложению. При подключении все таблицы были перенесены из SQL в C# со всеми данными (листинг 2.1).



## Листинг 2.1. Подключение к WPF-приложению

```
<connectionStrings>
  <add name="Entities"
connectionString="metadata=res://*/Model.csdl|res://*/Model.ssdl|res://*/Model
.msl;provider=System.Data.SqlClient;provider connection string="data
source=LAPTOP-GUIKJJIL;initial catalog=Car_service;integrated
security=True;trustservercertificate=True;MultipleActiveResultSets=True;App=En
tityFramework"" providerName="System.Data.EntityClient" />
</connectionStrings>
```

Для взаимодействия с реляционной базой данных через проект используется объектно-ориентированное представление для базы, в частности, класс-контекст (листинг 2.2).

## Листинг 2.2 – Контекст для работы с базой данных

```
namespace Car_Service
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;

    public partial class Entities : DbContext
    {
        public Entities()
            : base("name=Entities")
        {
        }

        private static Entities _context;
        public static Entities GetContext()
        {
            if (_context == null) _context = new Entities(); return _context;
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public virtual DbSet<Customers> Customers { get; set; }
        public virtual DbSet<Employees> Employees { get; set; }
        public virtual DbSet<Feedback> Feedback { get; set; }
        public virtual DbSet<Inventory> Inventory { get; set; }
        public virtual DbSet<Orders> Orders { get; set; }
        public virtual DbSet<Parts> Parts { get; set; }
        public virtual DbSet<Payments> Payments { get; set; }
        public virtual DbSet<Services> Services { get; set; }
        public virtual DbSet<Vehicles> Vehicles { get; set; }
        public virtual DbSet<Work> Work { get; set; }
    }
}

select p.ID, p.Title, count(ps.ProductID) as SalesCount
from dbo.Product p
left join dbo.ProductSale ps on p.ID = ps.ProductID
group by p.ID, p.Title;
```

Для разработки интерфейса использован С# – современный, инновационный, с открытым исходным кодом, объектно-ориентированный язык программирования.

При запуске приложения открывается форма авторизации. Также можно перейти на форму регистрации для добавления нового администратора (рисунок 2.15 – 2.16).

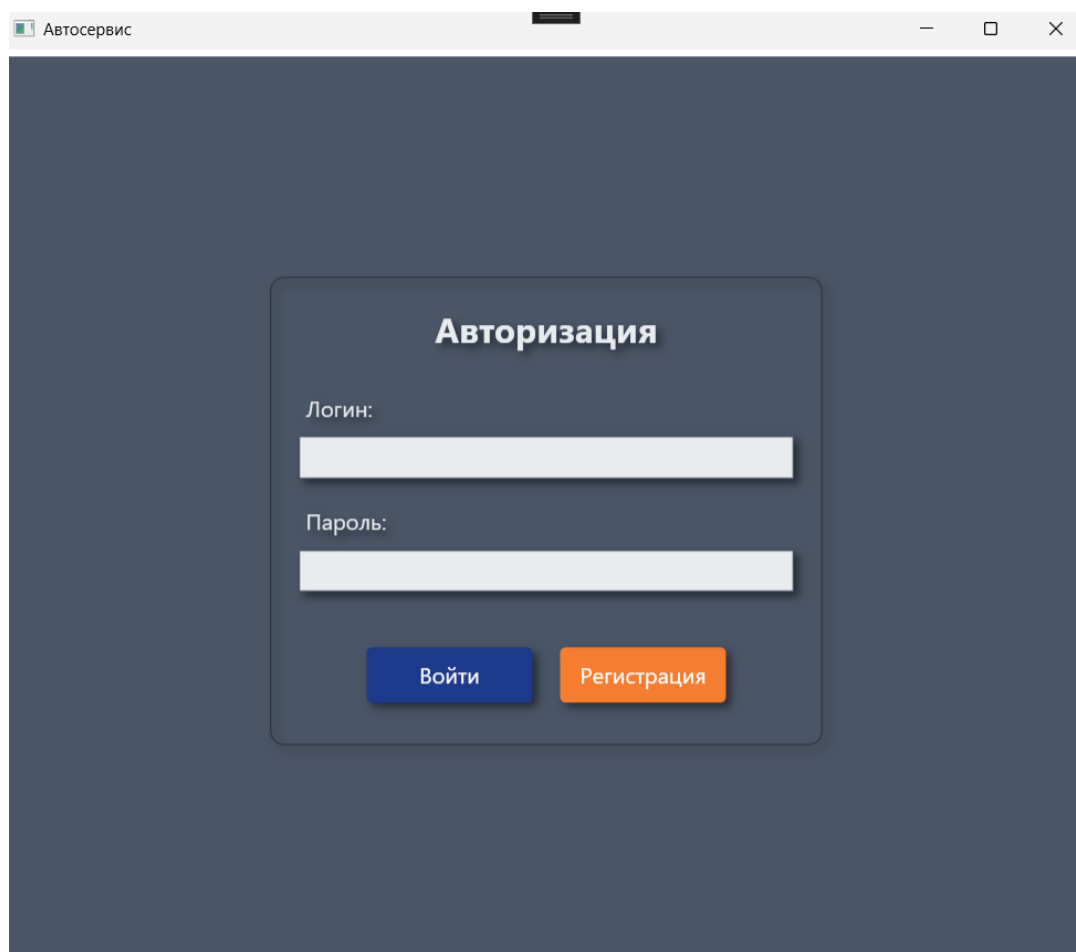
The image shows a screenshot of a web application window titled 'Автосервис'. The window has a dark blue background. In the center, there is a white rounded rectangle containing the title 'Авторизация' in bold. Below the title, there are two input fields: 'Логин:' followed by a white text box, and 'Пароль:' followed by a white text box. At the bottom of the white rectangle, there are two buttons: a blue button labeled 'Войти' and an orange button labeled 'Регистрация'.

Рисунок 2.15 – Форма авторизации в приложении

The image shows a web application window titled "Автосервис" (Car Service). Inside the window is a registration form titled "Регистрация" (Registration). The form contains the following fields and labels:

- ФИО сотрудника: (Employee's full name)
- Должность: (Position)
- Номер телефона: +7( ) - - (Phone number)
- Email: example@gmail.ru
- Логин: (Login)
- Пароль: (Password)

At the bottom of the form are two buttons: "Зарегистрироваться" (Register) in blue and "Назад" (Back) in orange.

Рисунок 2.16 – Форма регистрации в приложении

После нажатия кнопки «Вход» возможны следующие варианты:

- отображение сообщения об ошибке при отсутствии логина или пароля, либо при вводе неверных данных;
- переход в интерфейс администратора (рисунок 2.17).

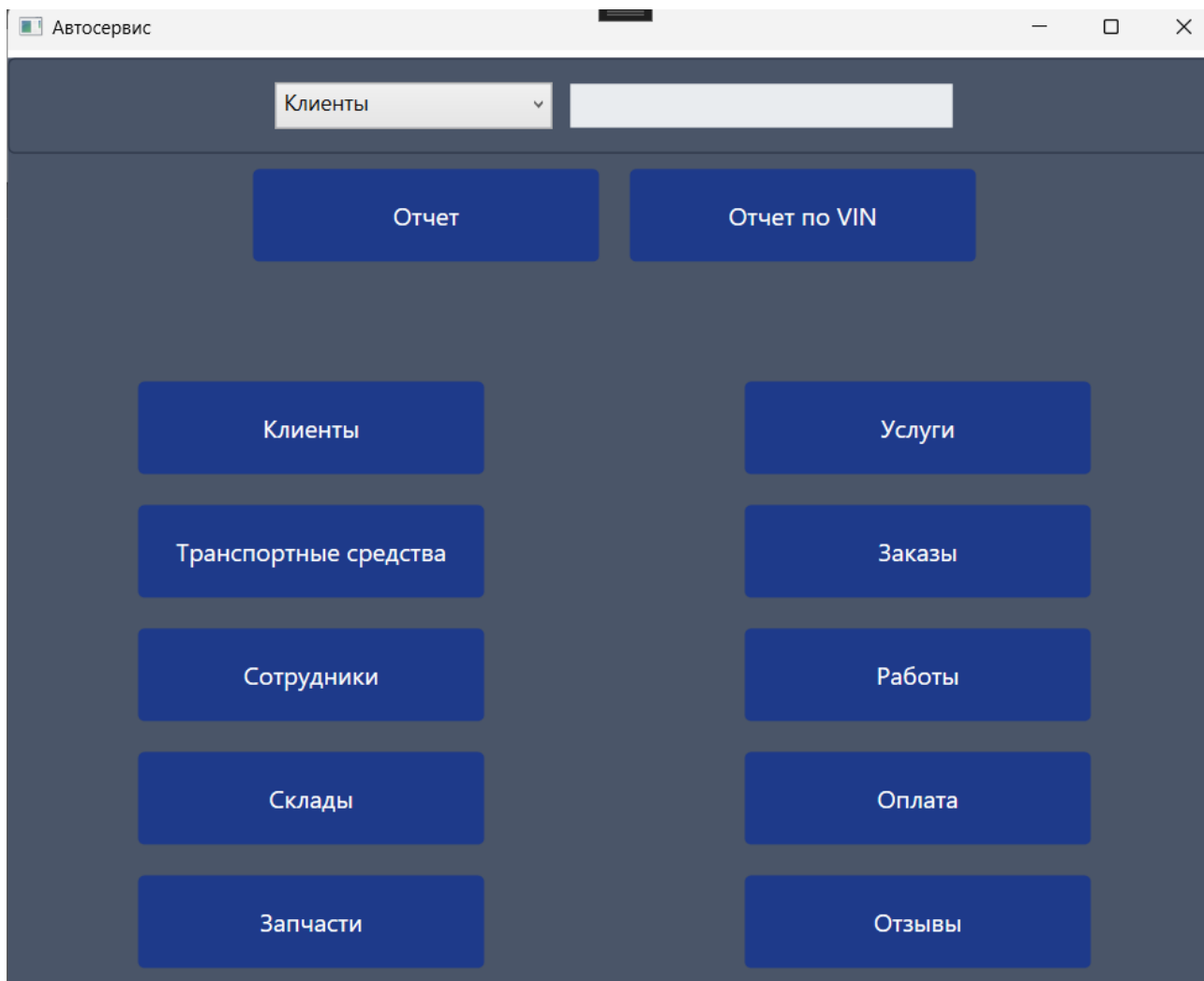


Рисунок 2.17 – Интерфейс администратора

Администратор на главной странице может выполнить поиск данных и установить для него фильтр (рисунок 2.18). Фрагмент реализации поиска по страницам показан в листинге 2.3.

Листинг 2.3 – Фрагмент реализации поиска

```
private List<dynamic> GetSearchResults(string pageType, string searchText)
{
    if (string.IsNullOrEmpty(pageType) || string.IsNullOrEmpty(searchText))
        return new List<dynamic>();

    switch (pageType)
    {
        case "Клиенты":
            return Entities.GetContext().Customers
                .Where(c => c.FullName.ToLower().Contains(searchText) ||
                    c.Phone.Contains(searchText))
                .Select(c => new { c.FullName, c.Phone, c.Email, c.Adress })
                .ToList<dynamic>();

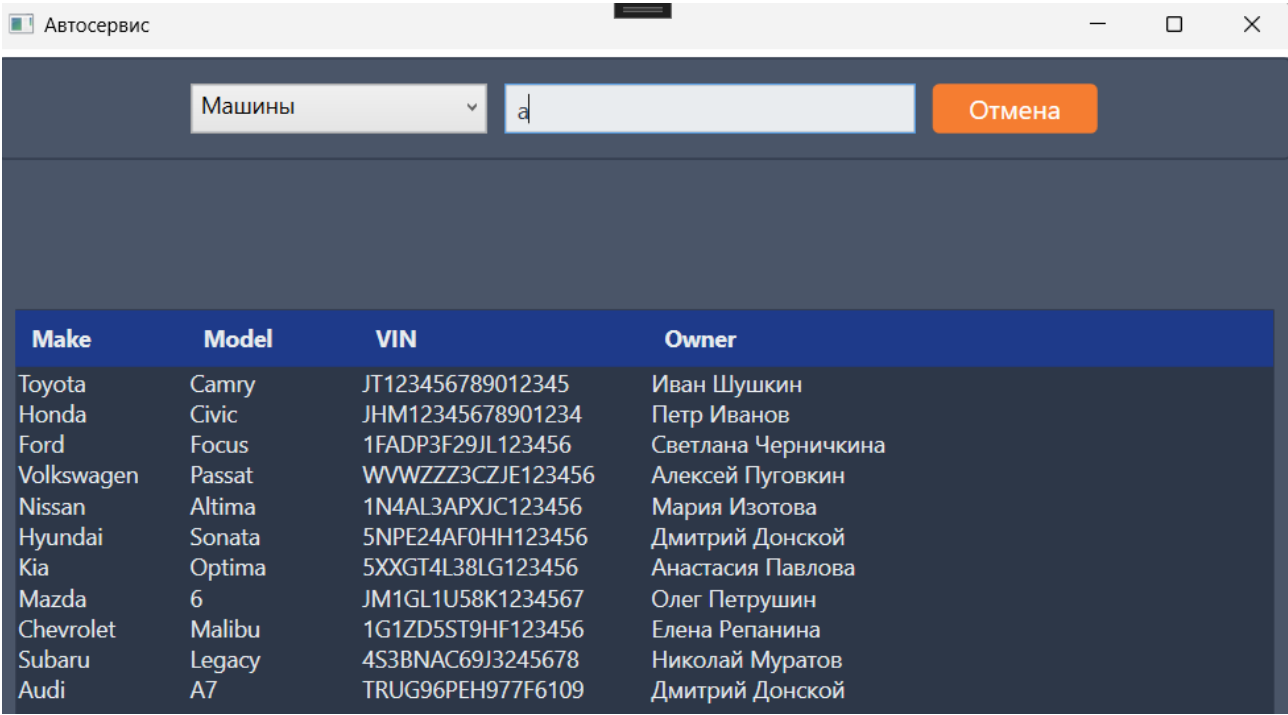
        case "Машины":
            return Entities.GetContext().Vehicles
```

```

        .Where(v => v.Make.ToLower().Contains(searchText) ||
                   v.Model.ToLower().Contains(searchText) ||
                   v.VIN.Contains(searchText))
        .Select(v => new { v.Make, v.Model, v.VIN, Owner =
v.Customers.FullName })
        .ToList<dynamic>();

    default:
        return new List<dynamic>();
    }
}

```



Make	Model	VIN	Owner
Toyota	Camry	JT123456789012345	Иван Шушкин
Honda	Civic	JHM12345678901234	Петр Иванов
Ford	Focus	1FADP3F29JL123456	Светлана Чернишкина
Volkswagen	Passat	WWZZZ3CZJE123456	Алексей Пуговкин
Nissan	Altima	1N4AL3APXJC123456	Мария Изотова
Hyundai	Sonata	5NPE24AF0HH123456	Дмитрий Донской
Kia	Optima	5XXGT4L38LG123456	Анастасия Павлова
Mazda	6	JM1GL1U58K1234567	Олег Петрушин
Chevrolet	Malibu	1G1ZD5ST9HF123456	Елена Репанина
Subaru	Legacy	4S3BNAC69J3245678	Николай Муратов
Audi	A7	TRUG96PEH977F6109	Дмитрий Донской

Рисунок 2.18 – Поиск и фильтрация

При выборе страницы администратор попадает на вкладку, где доступны следующие функции: просмотр, добавление, редактирование и удаление данных. Рассмотрим на примере страницы «Транспортные средства» (рисунок 2.19)

Марка	Модель	VIN	ФИО клиента
Toyota	Camry	JT123456789012345	Иван Шушкин
Honda	Civic	JHM12345678901234	Петр Иванов
Ford	Focus	1FADP3F29JL123456	Светлана Чернишкина
Volkswagen	Passat	WVWZZZ3CZJE123456	Алексей Пуговкин
Nissan	Altima	1N4AL3APXJC123456	Мария Изотова
Hyundai	Sonata	5NPE24AF0HH123456	Дмитрий Донской
Kia	Optima	5XXGT4L38LG123456	Анастасия Павлова
Mazda	6	JM1GL1U58K1234567	Олег Петрушин
Chevrolet	Malibu	1G1ZD5ST9HF123456	Елена Репанина
Subaru	Legacy	4S3BNAC69J3245678	Николай Муратов
Audi	A7	TRUG96PEN977F6109	Дмитрий Донской

Добавить
Редактировать
Удалить
Назад

Рисунок 2.19 – Страница «Транспортные средства»

При добавлении нового транспортного средства необходимо указать марку, модель, VIN-номер и выбрать клиента (рисунок 2.20). Код, отвечающий за сохранение данных, приведен в листинге 2.4.

Листинг 2.4 – Добавление транспортного средства

```
private void bSave_Click(object sender, RoutedEventArgs e)
{
    StringBuilder errors = new StringBuilder();
    if (string.IsNullOrEmpty(_vehicles.Make))
        errors.AppendLine("Введите марку транспортного средства");
    if (string.IsNullOrEmpty(_vehicles.Model))
        errors.AppendLine("Введите модель транспортного средства");
    if (string.IsNullOrEmpty(_vehicles.VIN)) errors.AppendLine("Введите VIN номер транспортного средства");
    else if (!Regex.IsMatch(_vehicles.VIN, @"^[A-HJ-NPR-Z0-9]{17}$")) errors.AppendLine("Стандартный VIN состоит из 17 символов (цифры и латинские буквы, за исключением I, O и Q)");
    if (fName.SelectedItem == null) errors.AppendLine("Выберите клиента");

    if (errors.Length > 0) { MessageBox.Show(errors.ToString()); return; }
    if (_vehicles.VehicleID == 0)
        Entities.GetContext().Vehicles.Add(_vehicles);

    if (fName.SelectedItem is Customers selectedCustomer)
    {

```

```

        _vehicles.CustomerID = selectedCustomer.CustomerID;
    }

    try
    {
        Entities.GetContext().SaveChanges();
        MessageBox.Show("Данные успешно сохранены");
        NavigationService.GoBack();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}

```

Рисунок 2.20 – Страница для добавления транспортного средства

При выборе страницы «Отчет по заказам», администратор перейдет на вкладку, где доступны функции: формирование отчета, импорт данных в Word и Excel. (Рисунок 2.21). Формирование отчета представлено в листинге 2.5.

Листинг 2.5 – Формирование отчета

```

private void InitializeChart()
{
    if (wfhChart.Child is Chart chart)
    {
        chart.Series.Clear();
        chart.ChartAreas.Clear();
        chart.Legends.Clear();

        ChartArea chartArea = new ChartArea("MainArea");
        chartArea.AxisX.Title = "Статусы заказов";
        chartArea.AxisY.Title = "Количество";
        chartArea.AxisX.MajorGrid.Enabled = false;
        chartArea.AxisY.MajorGrid.LineColor = Color.LightGray;
        chartArea.BackColor = Color.White;
    }
}

```

```

        chart.ChartAreas.Add(chartArea);

        Legend legend = new Legend();
        legend.BackColor = Color.Transparent;
        chart.Legends.Add(legend);
    }
}

private void UpdateChart()
{
    if (!(wfhChart.Child is Chart chart)) return;
    if (OrdersDataGrid.ItemsSource == null) return;

    chart.Series.Clear();

    var ordersInProgress = _context.Orders.Count(o => o.Status == "В
процессе");
    var ordersCompleted = _context.Orders.Count(o => o.Status ==
"Выполнен");

    string chartType = (cbChartType.SelectedItem as
ComboBoxItem)?.Tag?.ToString() ?? "Column";

    Series series = new Series("Заказы");
    series.ChartType = (SeriesChartType)Enum.Parse(typeof(SeriesChartType),
chartType);
    series.IsValueShownAsLabel = true;
    series.LabelFormat = "{0}";
    series.LabelForeColor = Color.White;

    series.Points.AddXY("В процессе", ordersInProgress);
    series.Points.AddXY("Выполненные", ordersCompleted);

    series.Points[0].Color = Color.FromArgb(65, 140, 240);
    series.Points[1].Color = Color.FromArgb(70, 200, 120);

    if (chartType == "Pie" || chartType == "Doughnut")
    {
        series["PieLabelStyle"] = "Outside";
        chart.ChartAreas[0].Area3DStyle.Enable3D = true;
    }

    chart.Series.Add(series);
}

```



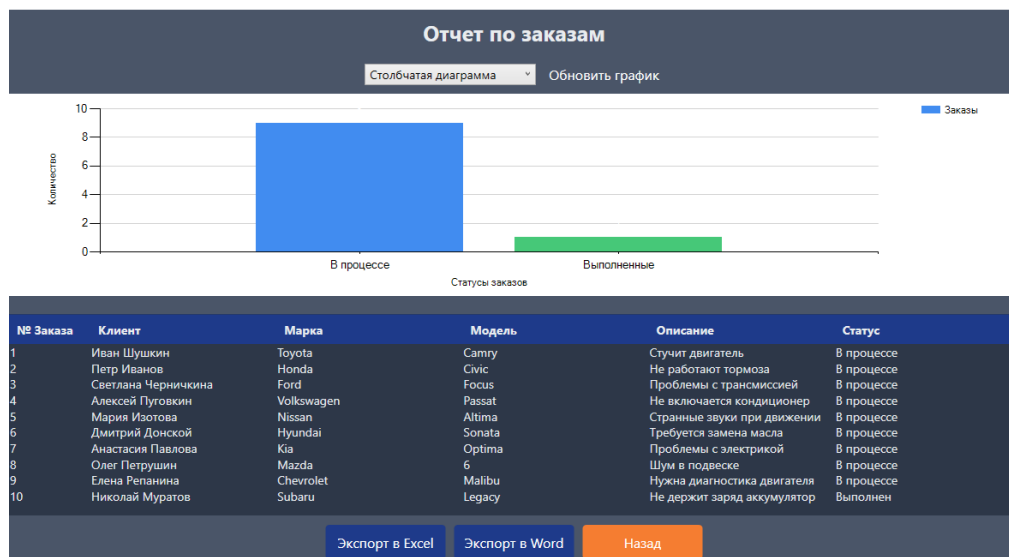


Рисунок 2.21 – Страница «Отчет по заказам»

Также создан второй отчет с теми же функциями по транспортным средствам (Рисунок 2.22)

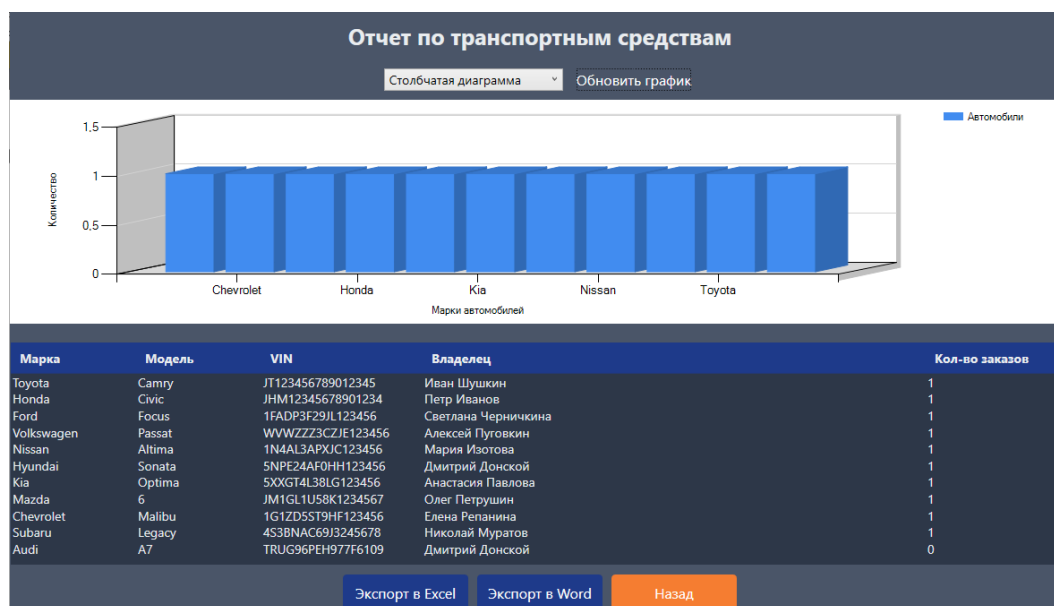


Рисунок 2.22 – Отчет по транспортным средствам

## 2.3. Отладка и тестирование

Тестирование – это исследование программного обеспечения, имеющий своей целью проверку соответствия между действительным поведением программы и ее ожидаемым поведением на конечном наборе испытаний.

Результаты тестирования и отладки приложения показаны в таблице 2.1

Таблица 2.1 – Результаты тестирования и отладки

№ теста	Входные данные	Вводимое значение	Ожидаемая реакция программы	Фактическая реакция программы	Ошибка выявлена
1	Логин и пароль	Test test	Сообщение «Пользователь с такими данными не найден!»	На рисунке 2.23	Нет
2	ФИО Должность Телефон Email Логин Пароль	«Иванов Иван Иванович» «Директор склада» « » « » «ivan» «ivan123»	Сообщения «Укажите номер телефона в формате +7XXXXXXXXXX», «Введите корректный email»	На рисунке 2.24	Нет
3	ФИО Номер телефона Email Адрес	«Петров Петр Петрович» «+7(924)371 23-62» «Petya@gmail.com» «г. Москва, ул. Горная 32-19»	Сообщение «Данные успешно сохранены»	На рисунке 2.25	Нет
4	№ заказа ФИО клиента Итоговая стоимость	«8» « » «13744»	Сообщение «Выберите клиента»	На рисунке 2.26	Нет
5	Удаление записи из любой таблицы	Выбор нужной записи	Сообщение «Вы точно хотите удалить записи в количестве 1 элементов?». Если «да», то сообщение «Данные успешно удалены»	На рисунке 2.27 – 2.28	Нет

The screenshot shows a dark-themed authorization form titled "Авторизация". It contains two input fields: "Логин:" (Login) with the value "test" and "Пароль:" (Password) with masked characters. Below the fields are two buttons: "Войти" (Login) in blue and "Регистрация" (Registration) in orange. A modal dialog box is displayed in the foreground with the message "Пользователь с такими данными не найден!" (User with such data not found!) and an "OK" button.

Рисунок 2.23 – Ввод неверных логина и пароля

The screenshot shows a dark-themed registration form titled "Регистрация". It contains several input fields: "ФИО сотрудника:" (Employee's full name) with "Иванов Иван Иванович", "Должность:" (Position) with "Директор склада", "Номер телефона:" (Phone number) with "+7( ) \_ \_ \_", "Email:" with "example@gmail.ru", "Логин:" (Login) with "ivan", and "Пароль:" (Password) with masked characters. At the bottom are two buttons: "Зарегистрироваться" (Register) in blue and "Назад" (Back) in orange. A modal dialog box is displayed in the foreground with the message "Укажите номер телефона в формате +7XXXXXXXXXX Введите корректный email" (Specify the phone number in the format +7XXXXXXXXXX. Enter a correct email) and an "OK" button.

Рисунок 2.24 – Регистрация с пустыми полями

Форму для добавления клиента. Поля:

- ФИО клиента: Петров Петр Петрович
- Номер телефона: +7(924)371 23-62
- Email: Petya@gmail.com
- Адрес: г. Москва, ул. Горная 32-19

Кнопки: Сохранить (синяя), Назад (оранжевая).

Всплывающее окно: Данные успешно сохранены. Кнопка: ОК.

Рисунок 2.25 – Добавление клиента

Форму для проверки данных:

- № заказа: 8
- ФИО клиента: Выберите клиента
- Итоговая стоимость: 13744

Кнопки: Сохранить (синяя), Назад (оранжевая).

Всплывающее окно: Выберите клиента. Кнопка: ОК.

Рисунок 2.26 – Проверка заполнения данных

Всплывающее окно с заголовком "Внимание!" и значком вопроса:

Вы точно хотите удалить записи в количестве 1 элементов?

Кнопки: Да, Нет.

Рисунок 2.27 – Удаление записи

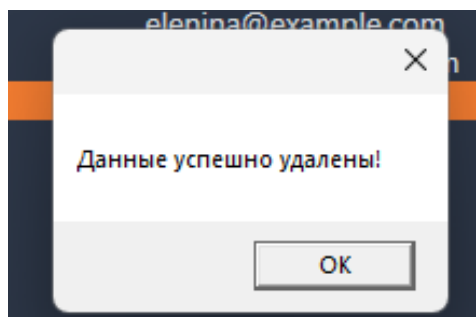


Рисунок 2.28 – Успешное удаление записи

Для проверки работоспособности приложения использован метод «черного ящика», согласно которому проверяющий не использует знание о реализации функций приложения и проверяет работу приложения с точки зрения внешнего пользователя. Выполненные тесты показали, что приложение соответствует предъявляемым к нему требованиям.

#### 2.4. Руководство администратора базы данных

Данное руководство адресовано администраторам базы данных, которые будут работать с базой данных автосервиса. В нем содержатся последовательные инструкции по решению основных задач, связанных с установкой и настройкой MS SQL Server, восстановлением базы данных, а также установкой ролей, привилегий и запретов.

##### Установка MS SQL Server

- 1) Загрузите установочный файл MS SQL Server с официального сайта Microsoft и выберите нужный язык (Рисунок 2.29)

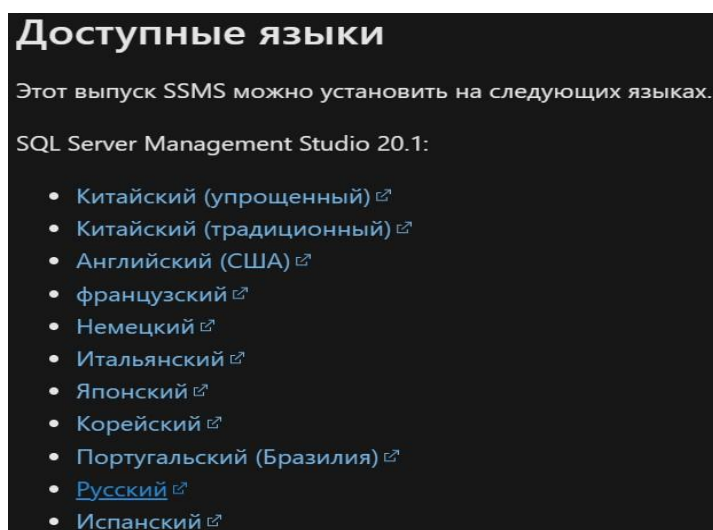
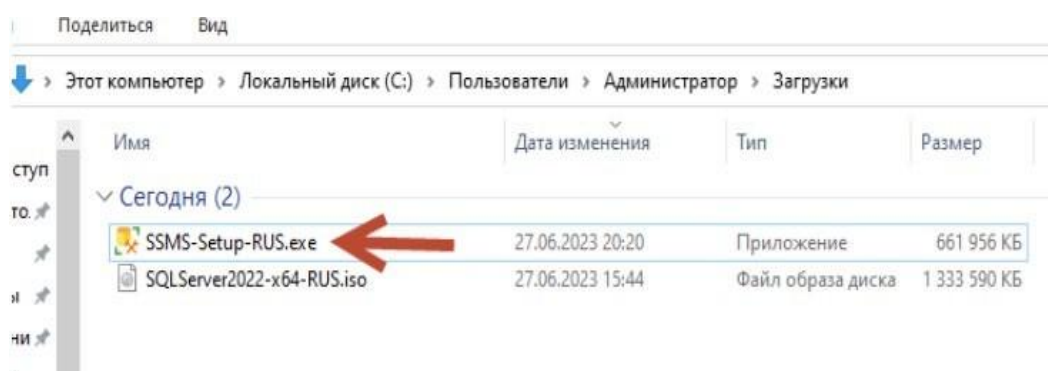
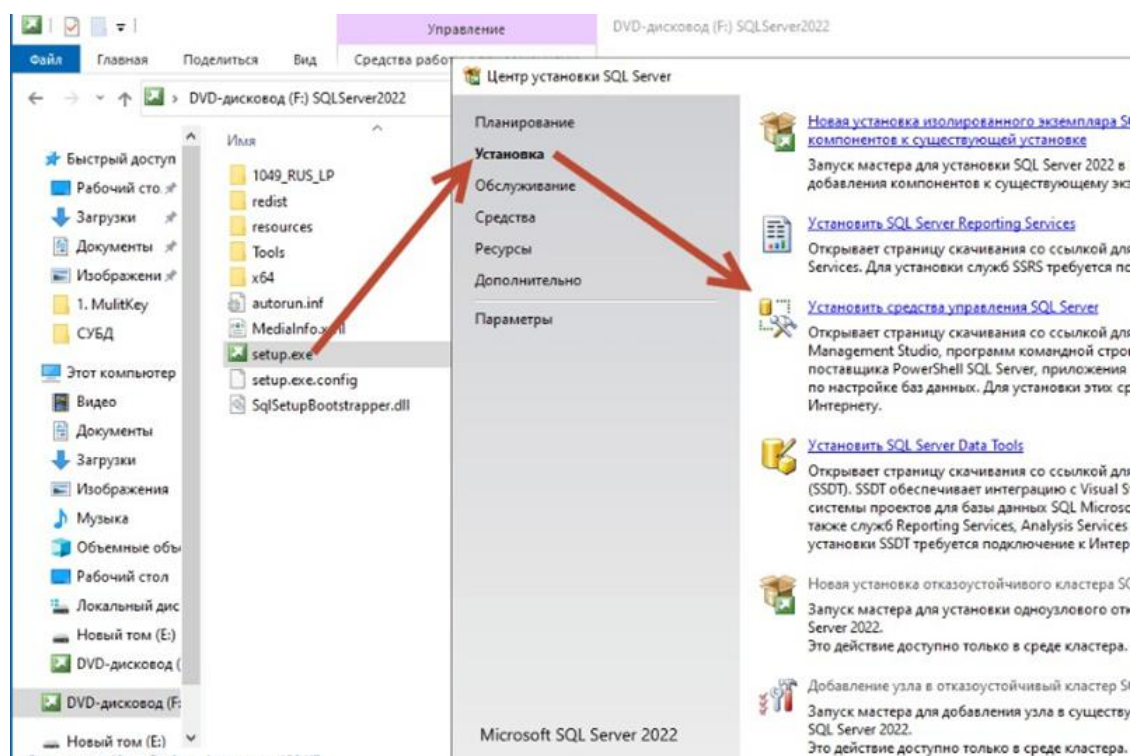


Рисунок 2.29 – Выбор установочного файла

- 2) Запустите установочный файл и следуйте инструкциям на рисунках 2.30 – 2.31.



- 3) После того, как установщик скачался, необходимо его запустить. В результате запустится программа, где нужно нажать на кнопку «Установить». В случае необходимости вы можете изменить каталог, в который будет установлена среда SQL Server Management Studio (Рисунок 2.32)

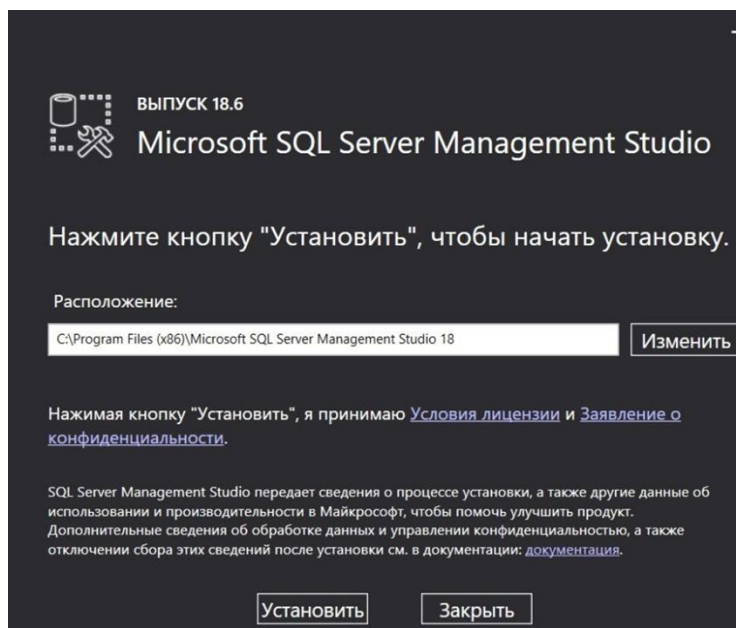


Рисунок 2.32 – Выбор каталога, в который будет установлена среда SQL Server Management Studio

- 4) После сообщения о завершении загрузки нужно перезагрузить компьютер. Чтобы подключиться к Microsoft SQL Server, запускаем среду SSMS и в окне «Соединение с сервером» вводим данные для подключения (Рисунок 2.33)

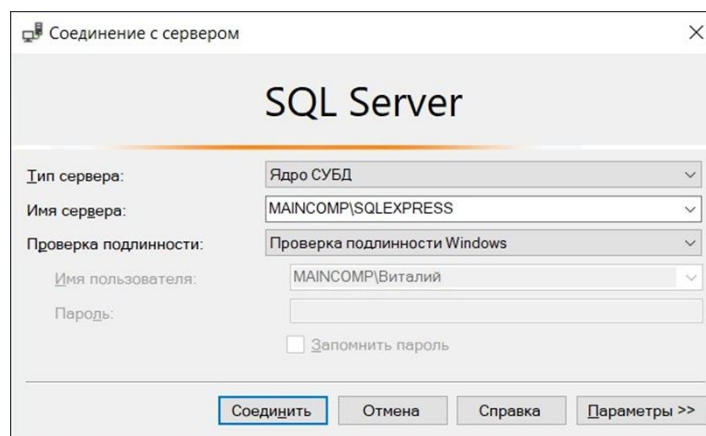


Рисунок 2.33 – Ввод данных для подключения

- 5) Выберите требуемые компоненты (Database Engine Services, SQL Server Replication, Full-Text Search и т.д.).
- 6) Укажите параметры для Database Engine, включая тип аутентификации (Windows Authentication или Mixed Mode) и учетные данные для системного администратора (sa).

7) Завершите установку и перезагрузите компьютер.

### Инструкция по восстановлению БД

- 1) Запустите SQL Server Management Studio и подключитесь к экземпляру SQL Server.
- 2) Слева, в обозревателе объектов (Object Explorer), раскрываем вкладку «Базы данных» (Server Objects), находим в списке базу данных, из которой (или в которую) необходимо восстановить данные, кликаем по ней ПКМ, затем в появившемся контекстном меню выбираем «Задачи» (Tasks) — «Восстановить» (Restore) — «База данных...» (Database...) (Рисунок 2.34)

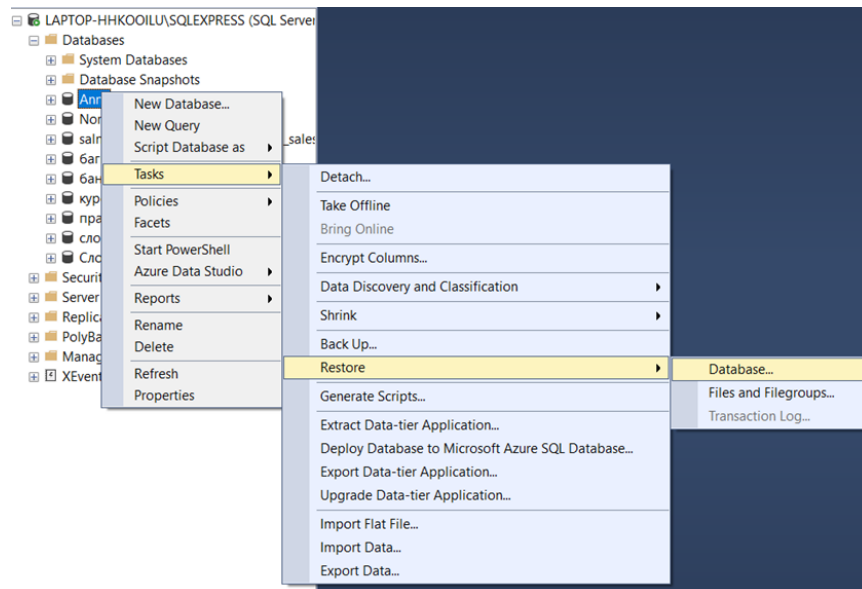


Рисунок 2.34 – Восстановление базы данных

- 3) Запустите мастер восстановления базы данных. Выберите источник резервной копии, при этом мастер автоматически подберет файлы для восстановления (Рисунок 2.35)



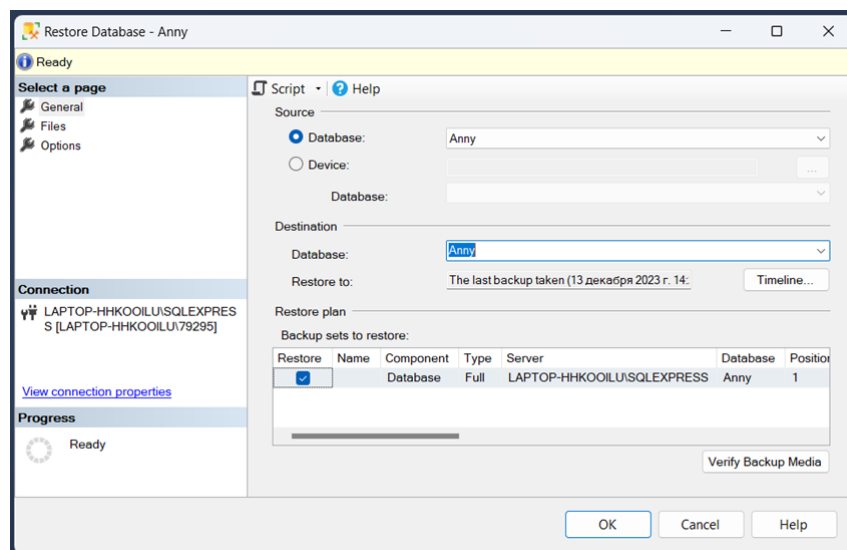


Рисунок 2.35 – Восстановление базы данных

Если нужно загрузить данные из конкретного файла, установите переключатель на «Устройство» и укажите источник (Рисунок 2.36)

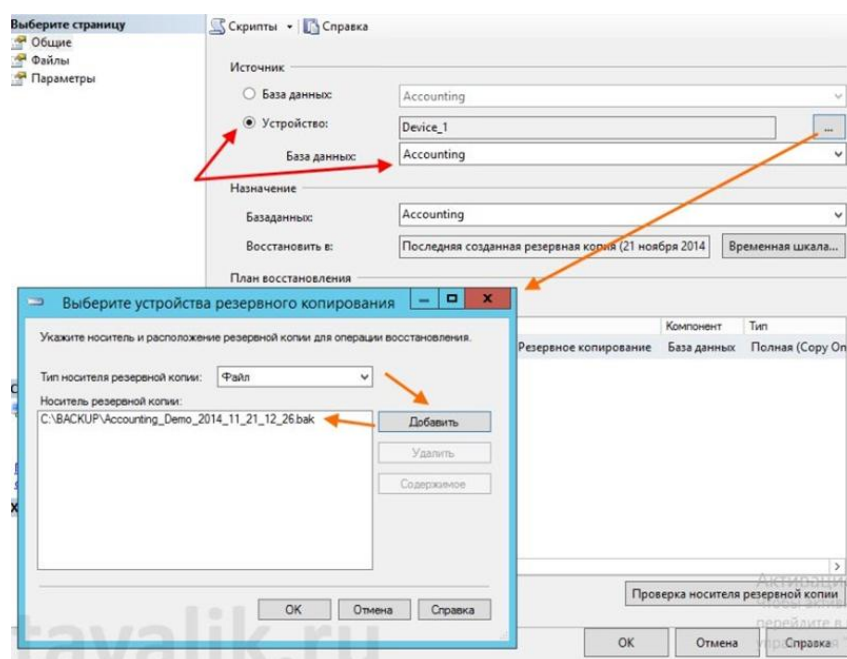


Рисунок 2.36 – Восстановление базы данных

Затем выберите базу данных назначения для загрузки данных (Рисунок 2.37)

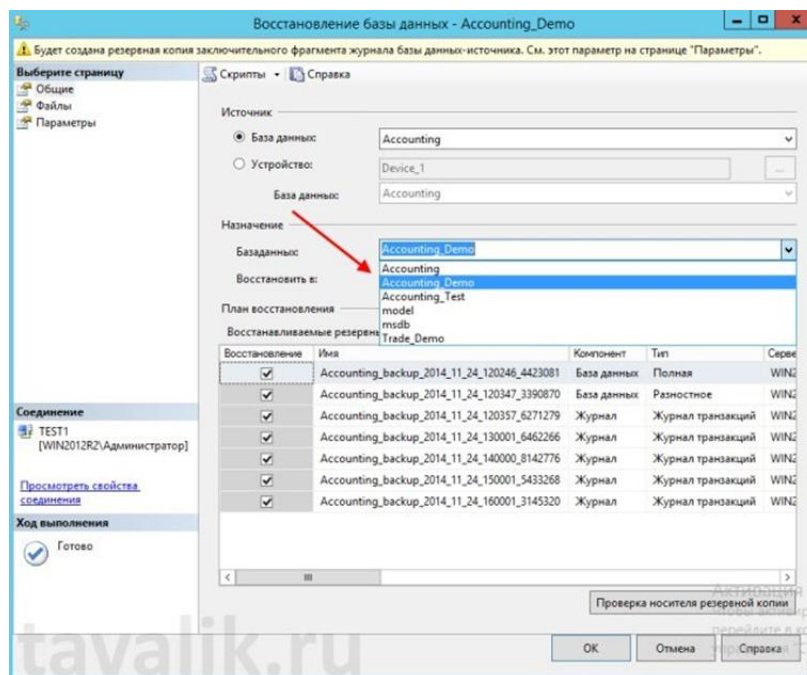


Рисунок 2.37 – Восстановление базы данных через файл

- 4) Нажав кнопку «Временная шкала...», можно выбрать время восстановления данных с точностью до секунды (Рисунок 2.38)

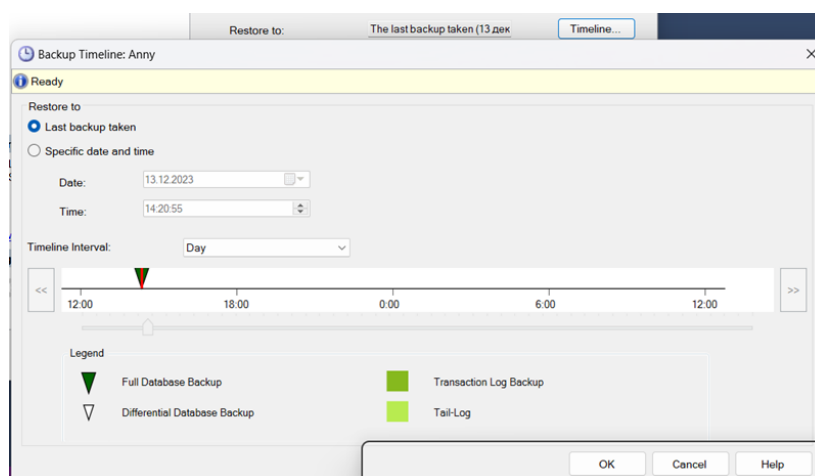


Рисунок 2.38 – Выбор времени

Важно помнить, что при восстановлении в другую информационную базу нужно указать путь к файлам этой базы на вкладке «Файлы» (Рисунок 2.39)

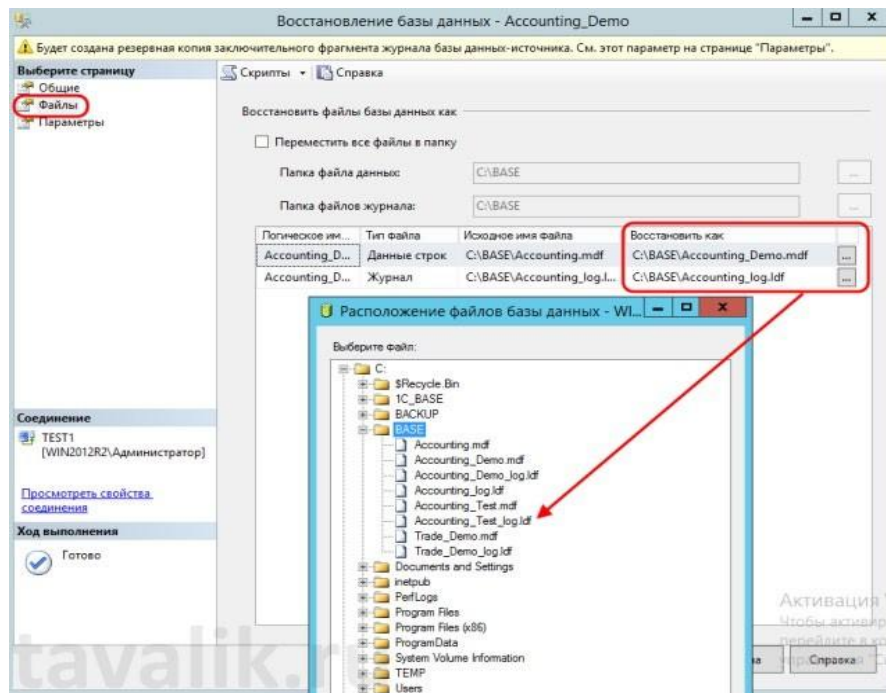


Рисунок 2.39 – Восстановление данных в информационную базу

- 5) На вкладке «Параметры» (Options) можно указать дополнительные параметры резервного копирования (Рисунок 2.40)

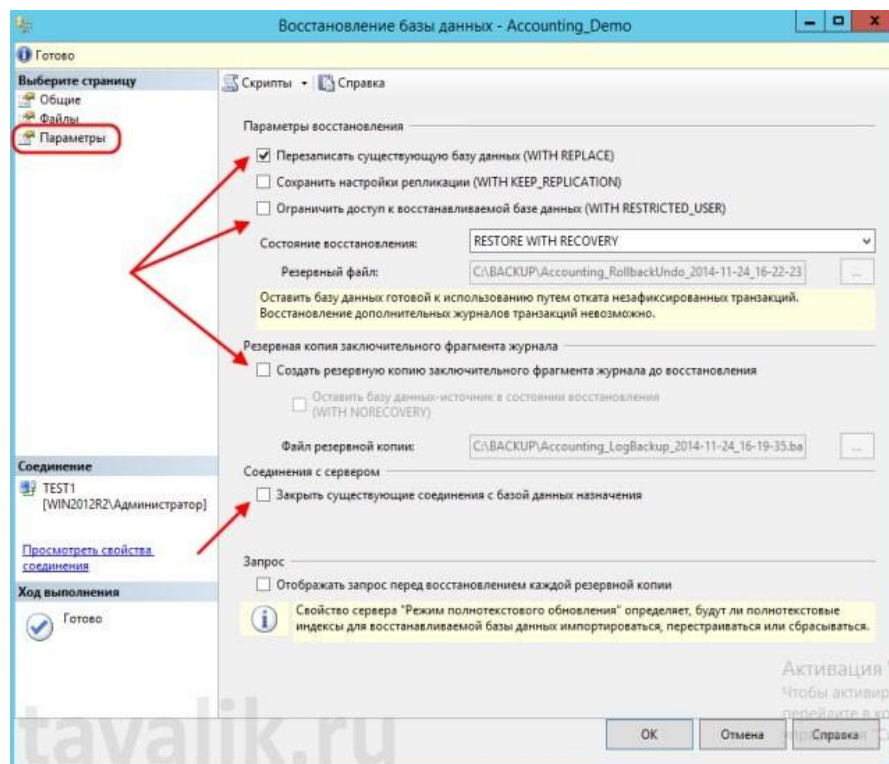


Рисунок 2.40 – Флаги

- 6) Когда все необходимые параметры установлены, нажимаем «ОК» для запуска процесса восстановления базы данных.

## **ЗАКЛЮЧЕНИЕ**

Цели курсового проектирования базы данных для автосервиса были достигнуты. Система управления автосервисом была разработана с учетом всех заданных требований. Она позволяет управлять автомобилями, клиентами, заказами, работами, услугами, запчастями, сотрудниками, отзывами и оплатой. Все задачи курсового проектирования были выполнены.

Система полностью реализована и работает корректно. Администраторы могут взаимодействовать с системой через удобный интерфейс, который обеспечивает удобство использования.

Достоинствами разработанного программного средства являются:

- Функциональность;
- Удобство использования;
- Гибкость;
- Расширяемость системы.

Недостатком является недостаточная оптимизация для больших объемов данных, однако это можно устранить путем решения проблемы с производительностью.

Самостоятельно были изучены методы работы с базами данных в SQL Server, архитектурой приложений, а также методологии разработки ПО. Это помогло в понимании принципов работы и выбора подходящих инструментов для разработки приложения.

## **СПИСОК ЛИТЕРАТУРЫ (ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ) И ИНТЕРНЕТ – РЕСУРСОВ**

1. ГОСТ 7.32-2017 «Отчет о научно-исследовательской работе. Структура и правила оформления»
2. ГОСТ Р 7.0.100-2018 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления»
3. ГОСТ 7.80-2000 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления»
4. ГОСТ 7.82-2001 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов»
5. ГОСТ 7.0.12-2011 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила»
6. ГОСТ 7.11-2004 «Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках»
7. Троелсен Э. Язык программирования C# 10 и .NET 6. - М.: Диалектика, 2022. - 1152 с.
8. Прайс, М.Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М.Дж. Прайс – Москва: Питер, 2023. – 848 с.
9. Паттерны проектирования в современном C#/C. Тепляков - М.: ДМК Пресс, 2022. - 320 с.
10. Вендел, С. Дизайн и поведение пользователей. Применение психологии и поведенческой экономики в разработке и UX / С. Вендел. – Москва: Sprint Book, 2025. – 400 с.

11. Гамма, Э. Паттерны объектно-ориентированного проектирования / Э. Гамма, Р. Джонсон. – Москва: Питер, 2022. – 448 с.
12. Microsoft Docs. Официальная документация по SQL Server 2022 [Электронный ресурс]. – URL: <https://docs.microsoft.com/sql>
13. Microsoft Learn. Entity Framework Core 7: Performance Tuning [Электронный ресурс]. – URL: <https://learn.microsoft.com/ef/core/>
14. Официальный сайт Microsoft Developer Network [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/dotnet/>
15. Microsoft. WPF Documentation for .NET 7 [Электронный ресурс]. – URL: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/>
16. Введение в ADO.Net – [Электронный ресурс]. – URL: <https://metanit.com/sharp/adonet/1.1.php>
17. Восстановление базы данных из резервной копии – [Электронный ресурс]. – URL: Восстановление базы данных из резервной копии в MS SQL Server 2012 | Tavalik.ru
18. Интернет-сервис для построения схем и диаграмм Draw.io. – [Электронный ресурс]. – URL: <https://www.draw.io/>
19. Интернет-сервис для генерации VIN-номеров. – [Электронный ресурс]. – URL: Генератор номеров VIN - Raspolozhenie-vin.com/
20. Интернет-сервис для поиска иконок. – [Электронный ресурс]. – URL: Material Design Icons - Icon Library - Pictogrammers

## ПРИЛОЖЕНИЕ

### ПРИЛОЖЕНИЕ А

#### Листинг А.1 – Скрипт создания базы данных

```
CREATE DATABASE Car_service;

CREATE TABLE Customers (
CustomerID INT PRIMARY KEY NOT NULL,
FirstName VARCHAR(50) NOT NULL,
LastName VARCHAR(50) NOT NULL,
Phone VARCHAR(20) NOT NULL,
Email VARCHAR(50) NOT NULL,
Adress VARCHAR(100) NOT NULL
);

CREATE TABLE Vehicles (
VehicleID INT PRIMARY KEY NOT NULL,
CustomerID INT,
Make VARCHAR(50) NOT NULL,
Model VARCHAR(50) NOT NULL,
Year INT NOT NULL,
VIN VARCHAR(17) NOT NULL,
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY NOT NULL,
FirstName VARCHAR(50) NOT NULL,
LastName VARCHAR(50) NOT NULL,
Position VARCHAR(50) NOT NULL,
Phone VARCHAR(20) NOT NULL,
Email VARCHAR(50) NOT NULL,
Login VARCHAR(50),
Password VARCHAR(50)
);

CREATE TABLE Services (
ServiceID INT PRIMARY KEY NOT NULL,
ServiceName VARCHAR(50) NOT NULL,
Description TEXT NOT NULL,
Price DECIMAL(18,0) NOT NULL
);

CREATE TABLE Orders (
OrderID INT PRIMARY KEY NOT NULL,
CustomerID INT,
DateOrder datetime default getdate() not null,
Status VARCHAR(50) NOT NULL CHECK (Status IN ('В процессе', 'Выполнен')),
VehicleID INT,
Problem TEXT NOT NULL,
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID)
);

CREATE TABLE Inventory (
InventoryID INT PRIMARY KEY NOT NULL,
Adress VARCHAR(100) NOT NULL,
InventoryName VARCHAR(50) NOT NULL
);
```

```

CREATE TABLE Parts (
PartID INT PRIMARY KEY NOT NULL,
InventoryID INT,
PartName VARCHAR(50) NOT NULL,
Price DECIMAL(18,0) NOT NULL,
FOREIGN KEY (InventoryID) REFERENCES Inventory(InventoryID)
);

CREATE TABLE Work (
WorkID INT PRIMARY KEY NOT NULL,
OrderID INT,
EmployeeID INT,
ServiceID INT,
PartID INT,
InventoryID INT,
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),
FOREIGN KEY (ServiceID) REFERENCES Services(ServiceID),
FOREIGN KEY (PartID) REFERENCES Parts(PartID),
FOREIGN KEY (InventoryID) REFERENCES Inventory(InventoryID)
);

CREATE TABLE Payments (
PaymentID INT PRIMARY KEY NOT NULL,
WorkID INT,
CustomerID INT,
Amount DECIMAL(18,0) NOT NULL,
FOREIGN KEY (WorkID) REFERENCES Work(WorkID),
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE Feedback (
FeedbackID INT PRIMARY KEY NOT NULL,
CustomerID INT,
Feedback VARCHAR(300) NOT NULL,
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

```

## Листинг А.2 – Импорт данных отчета в Word

```

private void ExportToWord_Click(object sender, RoutedEventArgs e)
{
    var orders = Entities.GetContext().Orders
        .Where(o => o.Status == "В процессе" || o.Status == "Выполнен")
        .OrderBy(x => x.Status)
        .ThenBy(x => x.OrderID)
        .ToList();

    var customers = Entities.GetContext().Customers.ToList();
    var vehicles = Entities.GetContext().Vehicles.ToList();

    var application = new Word.Application();
    Word.Document document = application.Documents.Add();

    Word.Paragraph titleParagraph = document.Paragraphs.Add();
    Word.Range titleRange = titleParagraph.Range;
    titleRange.Text = $"Отчет по заказам на {DateTime.Now.ToString("dd-MM-yyyy")}";
    titleRange.ParagraphFormat.Alignment =
    Word.WdParagraphAlignment.wdAlignParagraphCenter;
    titleRange.Font.Name = "Times New Roman";
}

```



```

titleRange.Font.Size = 16;
titleRange.Font.Bold = 1;
titleRange.InsertParagraphAfter();

var inProcessCount = orders.Count(o => o.Status == "В процессе");
var completedCount = orders.Count(o => o.Status == "Выполнен");

Word.Paragraph statsParagraph = document.Paragraphs.Add();
Word.Range statsRange = statsParagraph.Range;
statsRange.Text = $"Заказов в процессе: {inProcessCount}\nВыполненных
заказов: {completedCount}";
statsRange.ParagraphFormat.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphLeft;
statsRange.Font.Name = "Times New Roman";
statsRange.Font.Size = 14;
statsRange.Font.Bold = 1;
statsRange.InsertParagraphAfter();

Word.Paragraph tableParagraph = document.Paragraphs.Add();
Word.Range tableRange = tableParagraph.Range;
Word.Table ordersTable = document.Tables.Add(tableRange, orders.Count +
1, 5);

ordersTable.Borders.InsideLineStyle =
ordersTable.Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle;
ordersTable.Range.Cells.VerticalAlignment =
Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

ordersTable.Cell(1, 1).Range.Text = "Статус";
ordersTable.Cell(1, 2).Range.Text = "№ Заказа";
ordersTable.Cell(1, 3).Range.Text = "Клиент";
ordersTable.Cell(1, 4).Range.Text = "Автомобиль";
ordersTable.Cell(1, 5).Range.Text = "Описание проблемы";

Word.Range headerRange = ordersTable.Rows[1].Range;
headerRange.Font.Bold = 1;
headerRange.Font.Name = "Times New Roman";
headerRange.Font.Size = 14;
headerRange.ParagraphFormat.Alignment =
Word.WdParagraphAlignment.wdAlignParagraphCenter;
headerRange.Shading.BackgroundPatternColor = Word.WdColor.wdColorGray15;

for (int i = 0; i < orders.Count; i++)
{
    var order = orders[i];
    var customer = customers.FirstOrDefault(c => c.CustomerID ==
order.CustomerID);
    var vehicle = vehicles.FirstOrDefault(v => v.VehicleID ==
order.VehicleID);

    ordersTable.Cell(i + 2, 1).Range.Text = order.Status;
    ordersTable.Cell(i + 2, 2).Range.Text = order.OrderID.ToString();
    ordersTable.Cell(i + 2, 3).Range.Text = customer?.FullName ?? "Не
указан";
    ordersTable.Cell(i + 2, 4).Range.Text = $"{vehicle?.Make ?? "Не
указана"} {vehicle?.Model ?? ""}";
    ordersTable.Cell(i + 2, 5).Range.Text = order.Problem ?? "Не
указана";

    if (order.Status == "В процессе")
        ordersTable.Rows[i + 2].Shading.BackgroundPatternColor =
Word.WdColor.wdColorYellow;
}

```

```

        else
            ordersTable.Rows[i + 2].Shading.BackgroundPatternColor =
Word.WdColor.wdColorGreen;

        for (int j = 1; j <= 5; j++)
        {
            ordersTable.Cell(i + 2, j).Range.Font.Name = "Times New Roman";
            ordersTable.Cell(i + 2, j).Range.Font.Size = 12;
        }
    }

    ordersTable.AutoFitBehavior(Word.WdAutoFitBehavior.wdAutoFitContent);

    string desktop =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    string filePath = System.IO.Path.Combine(desktop,
$"OrdersReport_All_{DateTime.Now.ToString("dd-MM-yyyy")}.docx");

    int counter = 1;
    while (System.IO.File.Exists(filePath))
    {
        filePath = System.IO.Path.Combine(desktop,
$"OrdersReport_All_{DateTime.Now.ToString("dd-MM-yyyy")}__{counter}.docx");
        counter++;
    }

    document.SaveAs2(filePath);
    document.Close();
    application.Quit();

    MessageBox.Show($"Отчет сохранен в: {filePath}", "Экспорт завершен",
        MessageBoxButton.OK, MessageBoxImage.Information);
}

```

### Листинг А.3 – Импорт данных отчета в Excel

```

private void ExportToExcel_Click(object sender, RoutedEventArgs e)
{
    var orders = Entities.GetContext().Orders
        .Where(o => o.Status == "В процессе" || o.Status == "Выполнен")
        .OrderBy(x => x.Status)
        .ThenBy(x => x.OrderID)
        .ToList();

    var customers = Entities.GetContext().Customers.ToList();
    var vehicles = Entities.GetContext().Vehicles.ToList();

    var application = new Excel.Application();
    Excel.Workbook workbook = application.Workbooks.Add();
    Excel.Worksheet worksheet = workbook.Worksheets[1];

    worksheet.Cells[1, 1] = $"Отчет по заказам на
{DateTime.Now.ToString("dd-MM-yyyy")}";
    Excel.Range headerRange = worksheet.Range["A1:E1"];
    headerRange.Merge();
    headerRange.Font.Bold = true;
    headerRange.Font.Size = 14;
    headerRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

    var inProcessCount = orders.Count(o => o.Status == "В процессе");
    var completedCount = orders.Count(o => o.Status == "Выполнен");
}

```

```

worksheet.Cells[2, 1] = $"Заказов в процессе: {inProcessCount}";
worksheet.Cells[3, 1] = $"Выполненных заказов: {completedCount}";
worksheet.Range["A2:A3"].Font.Bold = true;

worksheet.Cells[5, 1] = "Статус";
worksheet.Cells[5, 2] = "№ Заказа";
worksheet.Cells[5, 3] = "Клиент";
worksheet.Cells[5, 4] = "Автомобиль";
worksheet.Cells[5, 5] = "Описание проблемы";
worksheet.Cells[5, 6] = "Статус выполнения";

Excel.Range tableHeader = worksheet.Range["A5:E5"];
tableHeader.Font.Bold = true;
tableHeader.Interior.Color = Excel.XlRgbColor.rgbLightGray;

for (int i = 0; i < orders.Count; i++)
{
    var order = orders[i];
    var customer = customers.FirstOrDefault(c => c.CustomerID ==
order.CustomerID);
    var vehicle = vehicles.FirstOrDefault(v => v.VehicleID ==
order.VehicleID);

    worksheet.Cells[i + 6, 1] = order.Status;
    worksheet.Cells[i + 6, 2] = order.OrderID;
    worksheet.Cells[i + 6, 3] = customer?.FullName ?? "Не указан";
    worksheet.Cells[i + 6, 4] = $"{vehicle?.Make ?? "Не указана"}
{vehicle?.Model ?? ""}";
    worksheet.Cells[i + 6, 5] = order.Problem ?? "Не указана";
    worksheet.Cells[i + 6, 5] = order.Status ?? "Не указан";

    var rowRange = worksheet.Range[$"A{i + 6}:E{i + 6}"];
    if (order.Status == "В процессе")
        rowRange.Interior.Color = Excel.XlRgbColor.rgbLightYellow;
    else
        rowRange.Interior.Color = Excel.XlRgbColor.rgbLightGreen;
}

Excel.Range dataRange = worksheet.Range[$"A5:E{5 + orders.Count}"];
dataRange.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;
dataRange.Borders.Weight = Excel.XlBorderWeight.xlThin;
dataRange.Columns.AutoFit();

string desktop =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
string fileName = $"OrdersReport_All_{DateTime.Now.ToString("dd-MM-
yyyy")}.xlsx";
string filePath = System.IO.Path.Combine(desktop, fileName);

int counter = 1;
while (System.IO.File.Exists(filePath))
{
    fileName = $"OrdersReport_All_{DateTime.Now.ToString("dd-MM-
yyyy")}_{counter}.xlsx";
    filePath = System.IO.Path.Combine(desktop, fileName);
    counter++;
}

workbook.SaveAs(filePath);
workbook.Close();
application.Quit();

```

```
        MessageBox.Show($"Отчет успешно сохранен:\n{filePath}", "Экспорт  
завершен",  
                        MessageBoxButtons.OK, MessageBoxIcon.Information);  
    }
```