

МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ
Государственное бюджетное профессиональное образовательное
учреждение Московской области «Люберецкий техникум имени Героя
Советского Союза, летчика-космонавта Ю.А.Гагарина»

ОТЧЕТ

ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИЧЕСКОЙ ПОДГОТОВКЕ

Карабчикова Дениса Викторовича

(Фамилия, имя, отчество студента)

по профессиональному модулю

ПМ. 01 Разработка модулей программного обеспечения для
компьютерных систем

ПМ. 02 Осуществление интеграции программных модулей

Специальность

09.02.07 "Информационные системы и программирование"

Код, название

Курс 4 Группа № 195ИС

Период практической подготовки

с «16» февраля 2023 г. по «12» апреля 2023г.

Руководитель практической подготовки

от техникума преподаватель Жирнова Ю. В.

должность, ФИО

Руководитель практической подготовки

от организации Морозов И.С

должность, ФИО

Люберцы 2023

ХАРАКТЕРИСТИКА ПРАКТИКАНТА

Карабчикова Дениса Викторовича

(Ф.И.О. практиканта)

Работал в информационном отделе, разработчик WPF приложений

(подразделение , должность , сроки работы)

с «16» февраля 2023 г. по «12» апреля 2023 г.

Количество выходов на работу 39 дней.

Пропущено дней 0, из них по не уважительной причине 0

Прошел производственную (преддипломную) практическую
подготовку по специальности

09.02.07 "Информационные системы и программирование"

Качество выполнения работы

За время прохождения производственной практической подготовки
показал себя с положительной стороны. В течение всего периода
Карабчиков Денис Викторович внимательно и ответственно относился к
выполняемой работе.

В отношениях с коллегами Карабчиков Денис Викторович проявил
себя с лучшей стороны: внимательность, умение выслушать и понять,
стремление избежать конфликта, устойчивость к стрессам.

Особенно хочется отметить умение грамотно планировать свою
деятельность в соответствии со стратегией развития коллектива и
выполнение работы с максимальной эффективностью. Руководство
организации оценивает работу Карабчикова Дениса Викторовича
положительно.

Руководитель практической подготовки от организации

Руководитель информационного отдела Морозов Иван Сергеевич

(должность, ФИО)

М.П.

Описание организации

1. Название организации: ООО «Авто-сфера»
2. Адрес: г. Дзержинский, ул. Энергетиков, 24
3. Адрес для писем: г. Дзержинский, ул. Энергетиков, 24
4. Телефон: +7 495-225-23-53
5. Факсы +7-495-220-20-50
6. E-mail: kapotnya@avto-start.ru
7. Отрасль: Торговля.
8. Год основания: 2004г
9. Форма собственности: Общество с ограниченной ответственностью

10. История

Компания «Авто-Сфера» была основана в 1991 году, и начала свое развитие с Сервисного центра на Проспекте Мира.

В 1992 году там же открывается первый автосалон, Авто-Сфера становится дилером KIA.

В 1993 году компания получает статус официального дилера MITSUBISHI.

В 2002 году - открытие автосалона и современного технического центра MITSUBISHI на Дмитровском шоссе

В 2004 году Авто - Сфера под брендовым названием Авто-Старт становится официальным дилером ТаГАЗ и получает право продажи автомобилей HYUNDAI.

В 2004 году завершается строительство территории на берегу реки Яуза и начинается продажа и обслуживание автомобилей MITSUBISHI и HYUNDAI

В 2005 году основан Департамент Продажи Поддержанных Автомобилей на Дмитровском шоссе

В 2006 году открывается специализированный дилерский центр KIA на 15-м км МКАД

В 2007 году создается вторая территория по выкупу, обмену и продаже поддержанных автомобилей.

В этом же году открывается Технический Центр на Волоколамском шоссе по обслуживанию автомобилей GEELY.

Уже через 2 года (в 2009 году) компания Авто-Старт принимает в управление территорию в г. Бронницы, специализирующуюся на продаже и ремонту грузовых автомобилей FORD

В декабре 2011 началась реконструкция фасада автосалона KIA на 15-м км МКАД – здесь и открыла своё представительство фирма «Авто-сфера»

11. Дочерние предприятия и/или филиалы

- Компания «Авто-сфера» сама является дочерним предприятием фирмы «Авто-спектр»

12. Предоставляемые услуги: Техническое обслуживание и ремонт автотранспортных средств, продажа автозапчастей.

Описание подразделения, в котором была пройдена практика

1. Название подразделения: Отдел безопасности
2. Руководитель подразделения: Морозов Иван Сергеевич
3. Куратор практики: Морозов Иван Сергеевич
4. Структура и функции подразделения (в краткой форме должностные обязанности):

Цели:

Отдел безопасности автосалона участвует в организации современной системы защиты информационной системы по продаже автомобилей. Работает для того, чтобы после приобретения у нас автомобиля, покупатель с удовольствием пользовался услугами нашего автосервиса, и не переживал за утечку данных, указанных при покупке

Руководитель отдела безопасности:

4.1. Приходит на работу за 30 минут до начала рабочего дня, аккуратно одетым.

4.2. Встречает менеджеров в салоне продаж, помогает сориентироваться в предлагаемом ассортименте автомобилей.

4.3. Провожает покупателей на стоянку автомобилей, знакомит их с предлагаемыми к продаже автомобилями и предлагает возможные варианты оплаты. Если на стоянке нет того автомобиля, который желает приобрести покупатель, оформляет заявку на желаемый автомобиль и сообщает о сроке и условиях выполнения заявки.

4.4. Обеспечивает безопасность сделки. Объясняет преимущества приобретения автомобиля в авто центре, предоставляя покупателю информацию обо всех предлагаемых скидках, условиях и возможностях гарантийного обслуживания.

4.5. Сообщает каждому потенциальному покупателю о возможностях комплекса авто услуг автоцентра и приглашает воспользоваться услугами автосервиса.

4.6. Сопровождает беседу с покупателем демонстрацией каталогов, прайс-листов и всеми имеющимися в распоряжении менеджера по продажам рекламными материалами.

4.7. Во всех ситуациях действует технологично, применяя современные техники продаж и привлечения покупателей.

4.8. Передает покупателей, принявших решение о покупке продавцам-консультантам, продавцам-оформителям и контролирует оформление сделки.

5. Резюме руководителя подразделения

- Высшее образование по специальности «Программист 1С»
- стаж работы в данной сфере 5 лет
- достижения в профессиональной сфере: Является сооснователем ООО «Авто-сфера». До «Авто-сферы» работал в салоне «KIA Motors» в отделе информационной безопасности

Введение

Описание подразделения	4
Описание предметной области	7
Постановка задачи	10
Установка и настройка инструментального ПО	Error! Bookmark not defined.
Microsoft SQL Server Management Studio	Error! Bookmark not defined.
SQL EXPRESS	Error! Bookmark not defined.
Visual Studio 2022	Error! Bookmark not defined.
Создание ERD-диаграммы в Visio	12
User Story	14
Use case	15
Разработка базы данных автосалона	16
Разработка WPF приложения	21
Код приложения	29
Страница авторизации	29
Вывод данных в DataGridView	40
ListView	54
Корзина	68
Заключение	71
Список литературы	73

Описание предметной области

Автосалон - это компания, занимающаяся продажей новых и б/у автомобилей, а также предоставляющая услуги по их обслуживанию. Информационная система автосалона (IS) - это программный комплекс, который обеспечивает управление бизнес-процессами, связанными с продажей и обслуживанием автомобилей.

Разработка WPF (Windows Presentation Foundation) приложения для IS автосалона позволит создать графический интерфейс пользователя, который облегчит работу с системой. WPF - это технология разработки клиентских приложений под Windows, позволяющая создавать современные и эффективные пользовательские интерфейсы.

Функциональные требования к WPF приложению могут включать в себя:

- возможность просмотра и редактирования информации о автомобилях в наличии;
- управление заказами на покупку автомобилей и оформление документов;
- ведение учета транспортных средств, находящихся в сервисном центре;
- планирование работ по обслуживанию автомобилей и учет выполненных работ;
- мониторинг финансовых показателей автосалона.

Для разработки WPF приложения может использоваться язык программирования C# и среда разработки Microsoft Visual Studio. Также необходимо обеспечить интеграцию приложения с базой данных автосалона, что позволит хранить и обрабатывать большой объем информации.

Теперь разберём программное обеспечение, которое будет использоваться во время нашей разработки

Программное обеспечéние — программа или множество программ, используемых для управления компьютером.

Microsoft разработала Visual Studio 2022, интегрированную среду разработки (IDE), которая предназначена для создания приложений для операционных систем Windows, мобильных приложений и веб-разработки. В IDE поддерживаются множество языков программирования, включая C#, C++, Visual Basic, Python, JavaScript и многие другие.

С помощью Visual Studio разработчики получают доступ к широкому набору инструментов для создания, тестирования, отладки и развертывания приложений, что позволяет ускорить процесс разработки и облегчить взаимодействие между членами команды. В IDE также доступны множество плагинов и расширений, которые облегчают задачи разработчиков и расширяют функциональность IDE.

Visual Studio 2022 - это интегрированная среда разработки (IDE), которую разработала компания Microsoft для создания приложений для операционных систем Windows, а также для веб-разработки и разработки мобильных приложений. Среда поддерживает множество языков программирования, включая C#, C++, Visual Basic, Python, JavaScript и многие другие.

Visual Studio обеспечивает разработчикам широкий набор инструментов для создания, отладки, тестирования и развертывания приложений. Это позволяет ускорить процесс разработки и упростить взаимодействие между разработчиками в команде. В Visual Studio также есть множество плагинов и расширений, которые позволяют дополнительно расширить функциональность IDE для удобства разработки и облегчения задач.

Microsoft SQL Server Management Studio (SSMS) – это интегрированная среда для управления базами данных Microsoft SQL Server. Она предоставляет графический интерфейс для управления и настройки экземпляров SQL Server, а также позволяет создавать, изменять и удалять базы данных, таблицы, представления, процедуры и другие объекты баз данных.

Постановка задачи

Автосалон - это как магазин, в котором представлены новые и поддержанные автомобили, а также их технические характеристики. Основными элементами предметной области являются поставляемые автомобили с различными признаками, а также информация о покупателях, продавцах и заказах.

База данных будет использоваться для учета продаж и поставок автомобилей и будет относиться к классу баз данных управления предприятием. Созданная база данных поможет упростить процедуру поиска информации о товарах и ценах на них. Созданная база данных также включает информацию о покупателях, сотрудниках машинах, заказах.

Приложение для работы с информационной системой автосалона упрощает процедуру поиска необходимой информации о машинах и ценах на них. С ее помощью сотрудник может легко узнать информацию о поставленном или проданном автомобиле, клиент может просматривать существующий в наличии модельный ряд авто, руководитель отслеживать деятельность сотрудников и динамику продаж.

Как у администратора, так и у клиента есть возможность найти интересующий его автомобиль, выставить нужные фильтры, отсортировать список автомобилей по наличию, по возрастанию цены и по определённой марке автомобиля.

На рисунке 1 определим задачи которые должны быть решены на производственной практической подготовке:

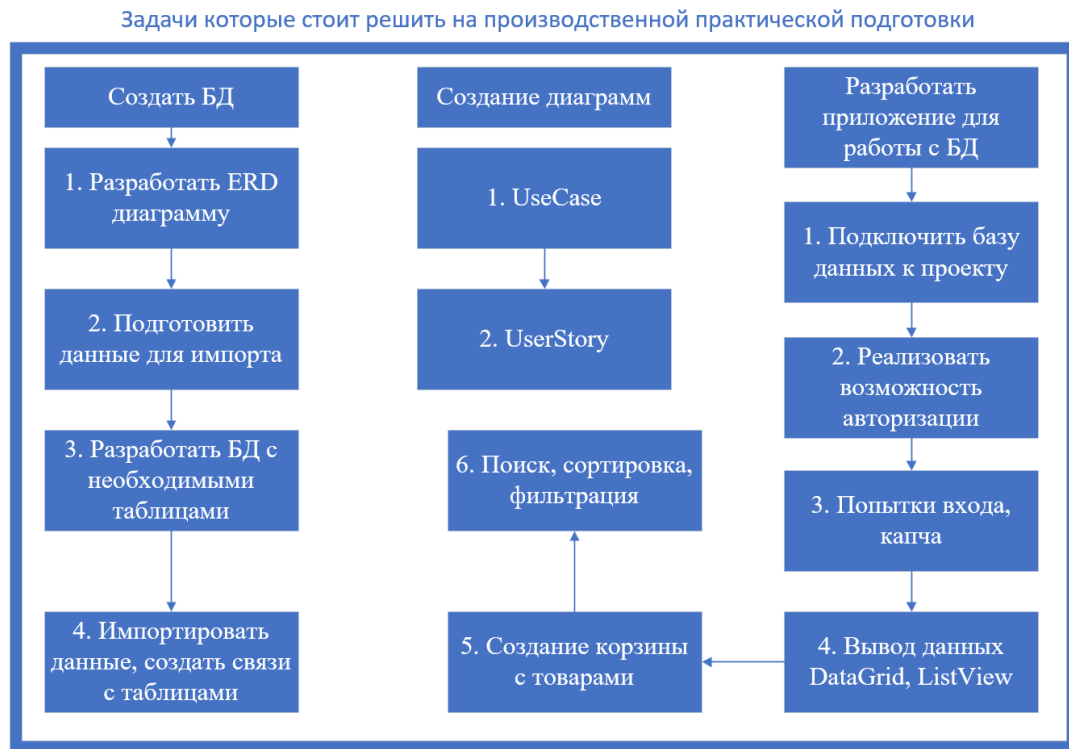


Рисунок 1. Набор задач на производственную практическую подготовку

Необходимо выполнить следующие задачи:

1. Создать БД → Разработать ERD диаграмму → Подготовить данные для импорта → Разработать БД с нужными таблицами → Импортировать данные.
2. Создание диаграмм → UseCase → UserStory
3. Разработать WPF приложение → Подключить БД к проекту → Реализовать возможность авторизации → Попытки входа и капча → Вывод данных, DataGridView, ListView → Создание корзины с товарами → Поиск, сортировка, фильтрация.

Создание ERD-диаграммы в Visio

В Visio необходимо создать представление информационной системы автосалона, для этого нужно запустить программу и выбрать нужный шаблон (Рисунок 2).

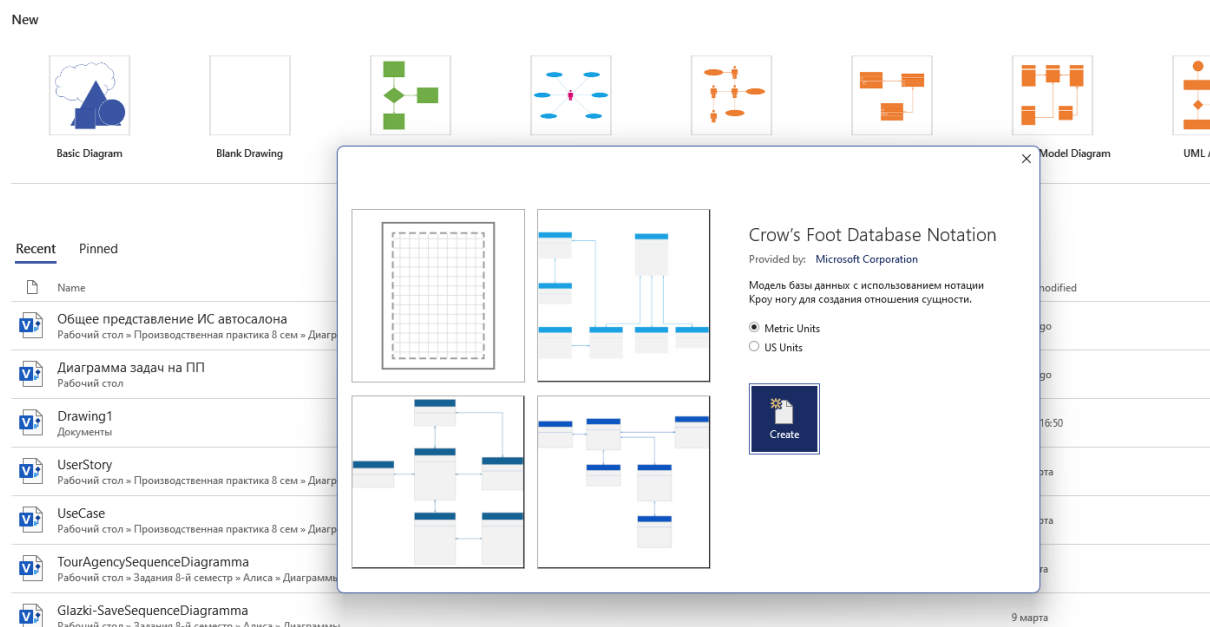


Рисунок 2. Выбор шаблона.

С помощью представленных наборов фигур составим таблицы, необходимые для работы с информационной системой Автосалона (Рисунок 3).

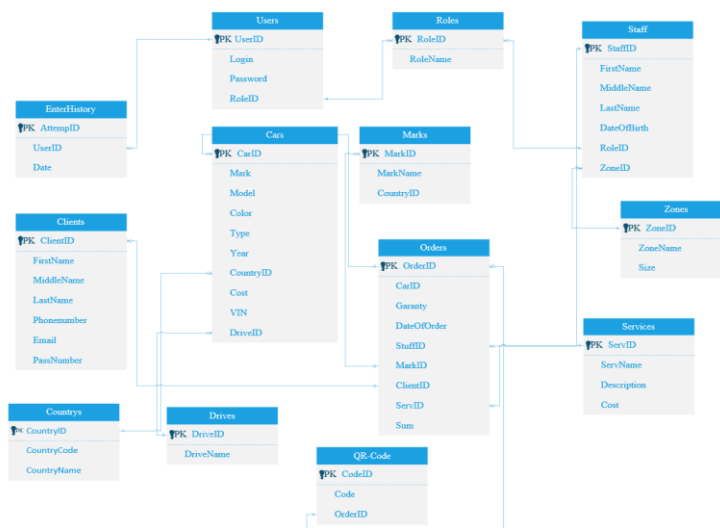


Рисунок 3. Готовая ERD диаграмма автосалона.

Таблица приводов (передний, задний, полный): Таблица приводов используется для хранения информации о типах приводов автомобилей, которые предлагаются в автосалоне. Эта таблица обычно содержит список всех возможных типов приводов, таких как передний, задний и полный, и используется для классификации автомобилей и поиска по типу привода.

Таблица заказов: Таблица заказов используется для хранения информации о заказах клиентов на автомобили. Эта таблица может содержать информацию о дате и времени заказа, автомобиле, на который сделан заказ, статусе заказа и информации о клиенте.

Таблица машин: Таблица машин используется для хранения информации о каждом автомобиле, который продается в автосалоне. Эта таблица может содержать информацию о марке и модели автомобиля, годе выпуска, типе кузова, типе привода, количестве мест и другой информации, которая поможет описать автомобиль.

Таблица пользователей: Таблица пользователей используется для хранения информации о пользователях информационной системы автосалона. Эта таблица может содержать информацию о имени пользователя, электронной почте, пароле и других аутентификационных данных.

Таблица сотрудников: Таблица сотрудников используется для хранения информации о сотрудниках автосалона. Эта таблица может содержать информацию о имени, фамилии, должности, зарплате и другой информации, которая поможет управлять персоналом автосалона.

Таблица зон: Таблица зон используется для хранения информации о зонах работы сотрудников. Эта таблица может содержать информацию о названии зоны и описании ее границ.

Таблица ролей: Таблица ролей используется для хранения информации о ролях пользователей в информационной системе автосалона. Эта таблица может содержать информацию о ролях, таких как

администратор, менеджер и продавец, и об их правах доступа к функционалу системы.

Таблица истории входа: Таблица истории входа используется для хранения информации о входах пользователей в информационную систему автосалона. Эта таблица может содержать информацию о времени входа, имени пользователя и IP-адресе, с которого был сделан вход.

Таблица клиентов: Таблица клиентов используется для хранения информации о клиентах автосалона.

User Story

User Story (продуктовая история) - это короткая и простая форма описания требований к функциональности продукта или сервиса, написанная в виде истории из глаз пользователя. User Story помогает команде разработчиков продукта понять, какую задачу нужно выполнить и для какой целевой аудитории.

При разработке User Stories вам необходимо использовать технологию разработки программного обеспечения BDD (сокр. от англ. Behavior-driven development, дословно «разработка через поведение») — это методология разработки программного обеспечения, являющаяся ответвлением от методологии разработки через тестирование (TDD)).

User story вкратце описывает:

- человека, использующего систему (заказчик);
- то, что должно содержаться в данной системе (примечание);
- то, для чего она нужна пользователю (цель).

Для разработки используйте следующий шаблон:

История: Авторизация

Как Клиент

Я хочу иметь возможность авторизоваться как клиент

Чтобы я мог увидеть список автомобилей и заказать тест драйв

Сценарий: Авторизация пройдена в качестве клиента

Дано: клиент открыл программу и ввёл верные данные для входа – логин и пароль

Когда Авторизация пройдена успешно

Тогда клиент получит информацию об автомобилях и получит возможность заказать тест драйв

Сценарий: Авторизация не пройдена

Дано: клиент открыл программу и ввёл не верные данные для входа

Когда было произведено 3 неудачных попытки

Тогда Происходит блокировка входа на 10 секунд

Use case

Use Case (случай использования) - это сценарий или описание того, как конкретный пользователь может использовать определенное приложение, систему или продукт для достижения определенной цели или решения определенной проблемы.

В рамках разработки программного обеспечения, Use Case является частью методологии разработки, которая позволяет описывать, как система будет использоваться в реальной жизни. Use Case может содержать информацию о том, кто будет использовать систему, как они будут ее использовать, какую информацию они вводят и получают из системы, и какие результаты они ожидают.

Use Case также может быть использован для описания бизнес-процессов, чтобы определить, какие роли и акторы будут участвовать в процессе, какие шаги они будут предпринимать и какие решения они будут принимать. Это позволяет бизнес-аналитикам и разработчикам создать систему, которая эффективно поддерживает бизнес-процессы и требования пользователей.

Ниже приведён пример Use Case диаграммы, на которой показаны возможности администратора, менеджера и клиента

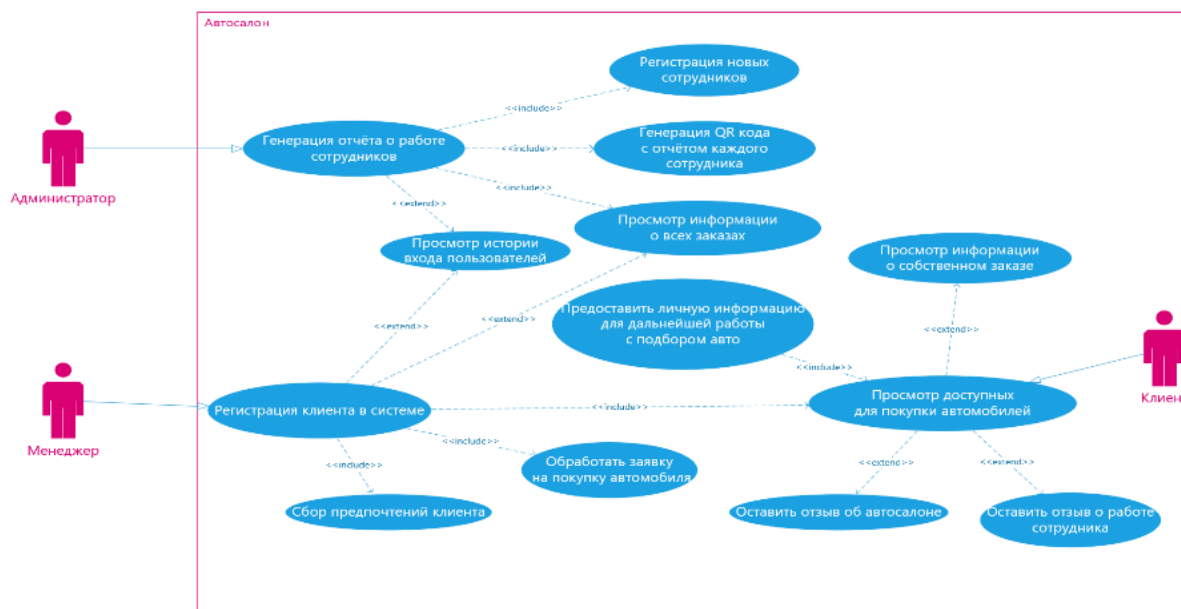


Рисунок 4. Use case автосалона.

Разработка базы данных автосалона

База данных автосалона представляет собой систему хранения и организации информации о продаваемых автомобилях, клиентах, заказах и других связанных с продажей автомобилей данных.

У нас уже есть ERD диаграмма которая является важным ключём в создании БД. На её основе мы создадим таблицы, поставим нужные ключи и типы данных, для этого запустим Microsoft SQL Server Management Studio с помощью специального значка (Рисунок 5).

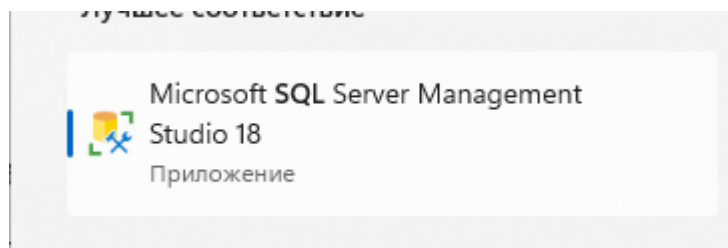


Рисунок 5. Значёк запуска приложения

Вводим данные для подключения, которые мы получили при установке SQL Server Express (Рисунок 6).

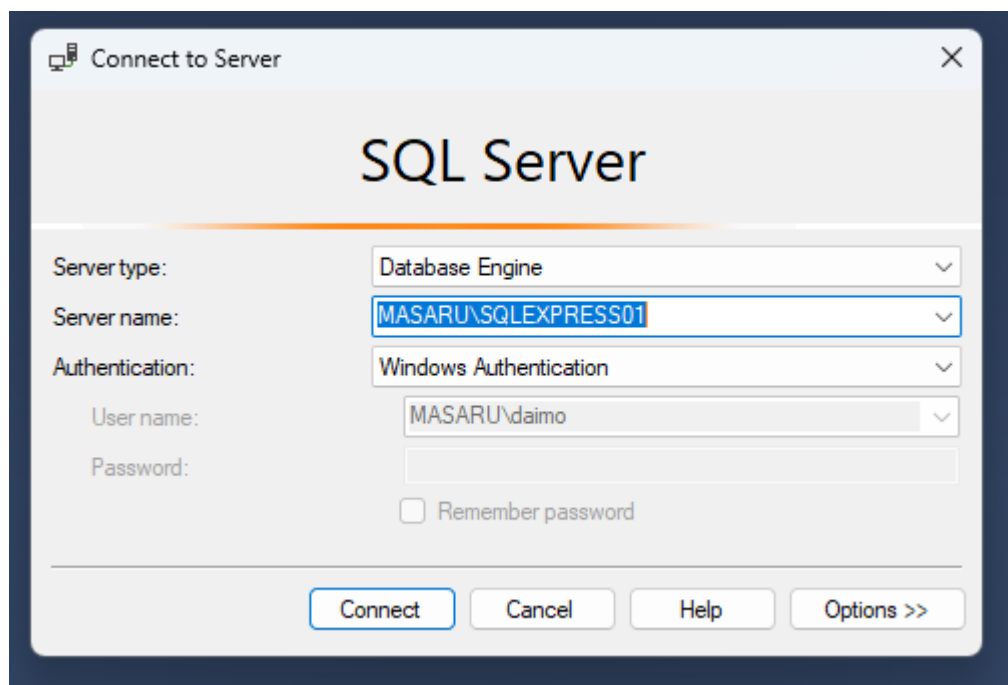


Рисунок 6. Подключение к ядру БД.

После подключения нажимаем правой кнопкой мыши по папке «DataBases» и выбираем «New Database» (Рисунок 7).

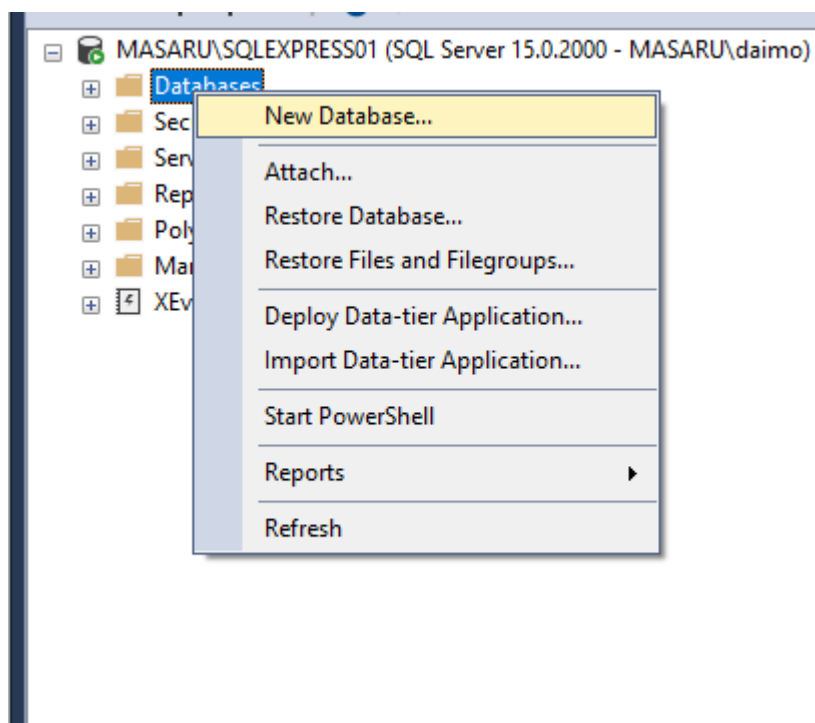


Рисунок 7. Создание новой базы данных.

Вводим название БД и переходим к созданию таблиц (Рисунок 8).

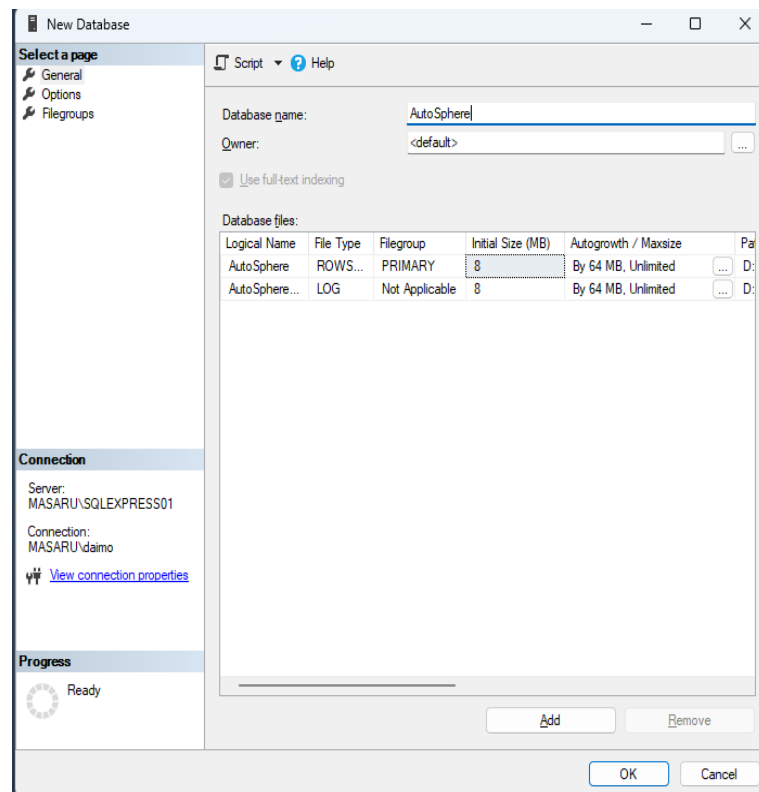


Рисунок 8. Выбор имени базы данных.

После успешного создания БД нажимаем правой кнопкой мыши по папке «Tables» и выбираем пункт «New Table» (Рисунок 9).

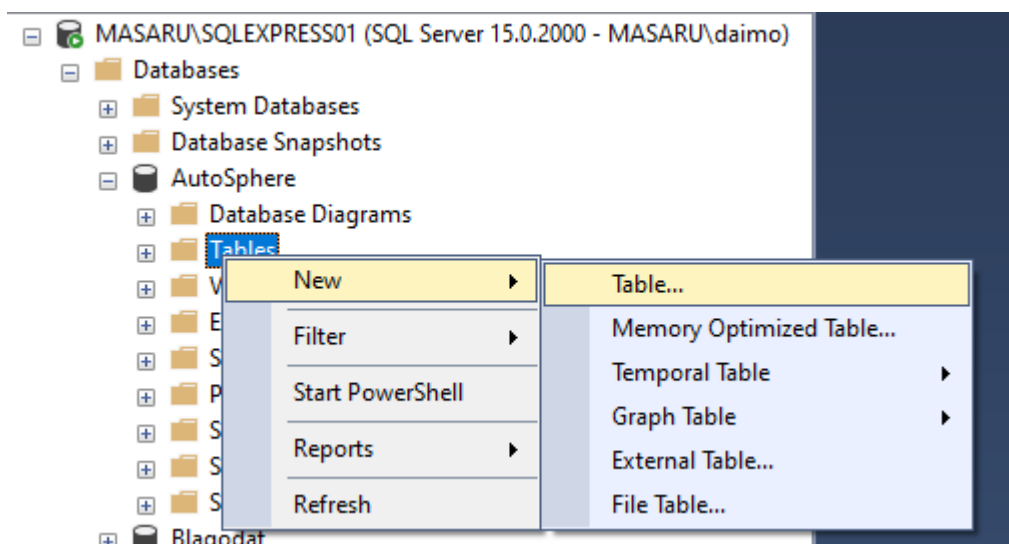


Рисунок 9. Создание новой таблицы.

Перед нами открывается окно с созданием таблицы, в которой нам необходимо выбрать название столбца и его тип данных (Рисунок 10).

The screenshot shows a window titled 'MASARU\SQLEXPRES...ere - dbo.Table_1'. It displays a table with the following structure:

Column Name	Data Type	Allow Nulls
		<input type="checkbox"/>

Рисунок 10. Создание таблицы.

Создадим таблицу сотрудников, для этого определим названия столбцов и их типы данных. Первый столбец в основном используется для нумерации объекта и этому столбцу присваивается первичный ключ (Рисунок 11).

The screenshot shows a table with the following structure:

Column Name	Data Type	Allow Nulls
StaffID	int	<input type="checkbox"/>
FirstName	nvarchar(50)	<input type="checkbox"/>
MiddleName	nvarchar(50)	<input type="checkbox"/>
LastName	nvarchar(50)	<input type="checkbox"/>
DateOfBirth	date	<input type="checkbox"/>
RoleID	int	<input type="checkbox"/>
ZoneID	int	<input type="checkbox"/>
		<input type="checkbox"/>

Рисунок 11. Таблица сотрудников.

Мы определили такие столбцы как:

- Номер сотрудника
- Имя
- Фамилия
- Отчество
- Дата рождения
- Номер роли
- Номер зоны

В созданной нами таблице будет храниться информации о сотрудниках, номер сотрудника, имя, фамилия и так далее. В дальнейшем, администратор или менеджер сможет просматривать записи из этой таблицы, а так же производить выборку по фамилиям, например вывести список сотрудников с зарплатой выше тридцати тысяч, заполним таблицу (Рисунок 12).

	StaffID	FirstName	MiddleName	LastName	DateOfBirth	RoleID	ZoneID
▶	1	Иван	Комаров	Николаевич	1997-07-17	2	1
	2	Даниил	Козлов	Алексеевич	1994-03-29	2	1
	3	Валентина	Морозова	Викторовна	1992-12-06	2	1
	4	Дарья	Лебедева	Николаевна	1989-09-28	1	2
	5	Татьяна	Савицкая	Сергеевна	1987-08-02	2	2
	6	Юлия	Макарова	Викторовна	1991-05-11	2	2
	7	Татьяна	Новикова	Олеговна	1988-02-01	2	1
	8	Егор	Зайцев	Дмитриевич	1996-11-19	2	3
	9	Виктор	Иванов	Олегович	1998-06-07	2	1
	10	Денис	Соколов	Александрович	1993-03-24	2	1
	11	Никита	Лебедев	Викторович	1984-12-10	2	3
	12	Сергей	Сергеев	Романович	1986-08-08	2	3
	13	Антон	Баженов	Валентинович	1983-01-14	1	4
	14	Дмитрий	Смирнов	Юрьевич	1985-09-03	1	4
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 12. Заполненная таблица

Аналогичным образом были созданы оставшиеся таблицы в БД, которые были представлены на ERD диаграмме (Рисунок 13).

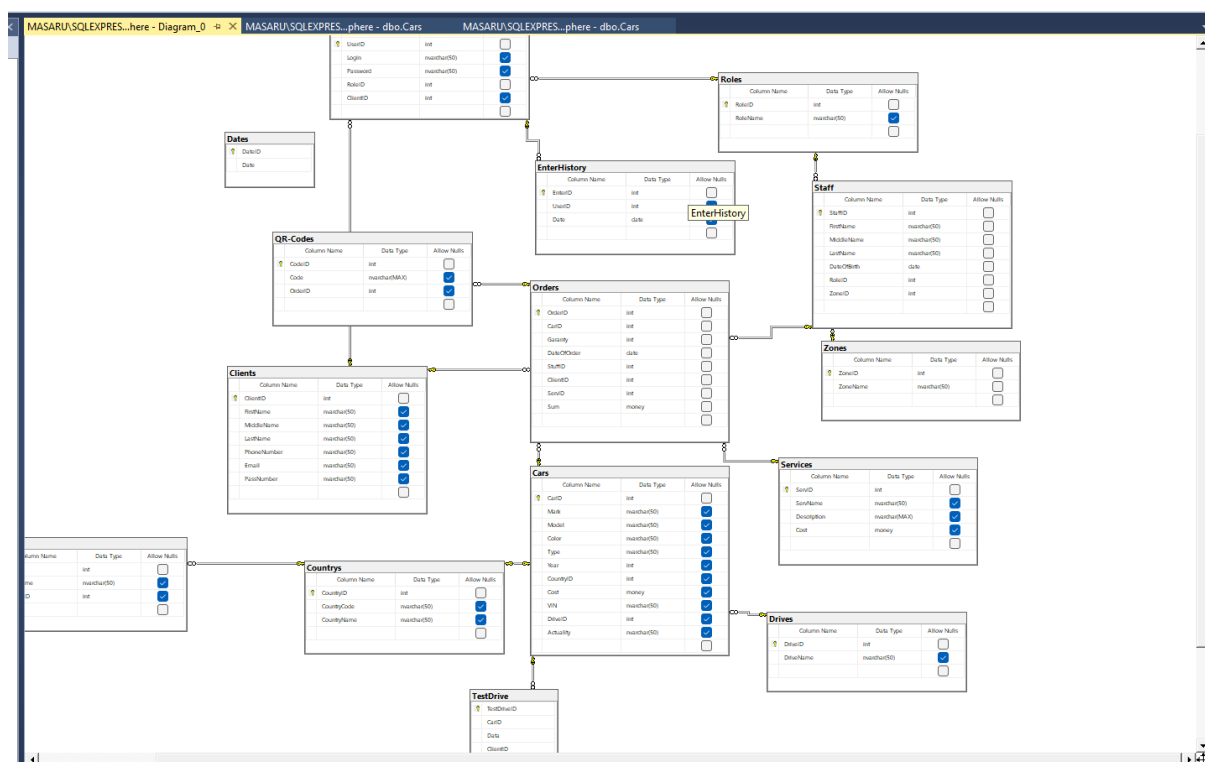


Рисунок 13. Диаграмма базы данных созданная в SQL Server Management Studio.

База данных автосалона успешно создана

Разработка WPF приложения

WPF (Windows Presentation Foundation) - это технология разработки графического интерфейса пользователя (GUI) для Windows приложений, которая предоставляет широкие возможности для создания интерактивных и multifunctional приложений.

1. Для работы с базой данных автосалона, WPF приложение может быть использовано для реализации следующих функций:
2. Отображение списка автомобилей и их характеристик из базы данных, таких как модель, год выпуска, цена и т.д.
3. Добавление новых автомобилей в базу данных.
4. Редактирование и обновление информации об уже существующих автомобилях в базе данных.
5. Удаление автомобилей из базы данных.

6. Поиск и фильтрация автомобилей по различным параметрам, таким как модель, год выпуска, цена и т.д.

Для создания WPF приложения для работы с базой данных автосалона можно использовать Visual Studio, интегрированную с Microsoft SQL Server для хранения и управления данными.

Для начала необходимо создать проект WPF в Visual Studio и настроить подключение к базе данных автосалона. Для этого можно использовать инструменты Server Explorer и Data Source Configuration Wizard в Visual Studio.

Затем можно создать различные элементы управления, такие как таблицы, списки, формы и кнопки, для отображения и редактирования данных из базы данных. В коде приложения можно использовать язык SQL для выполнения запросов к базе данных, а также LINQ (Language Integrated Query) для работы с данными и объектами приложения.

Для улучшения пользовательского интерфейса и удобства работы с приложением можно использовать стили и темы оформления WPF, а также добавить функции автозаполнения, валидации вводимых данных и диалоговые окна для подтверждения действий пользователя.

После создания приложения и проверки его работоспособности можно развернуть его на целевом устройстве, таком как ПК или планшет, для использования сотрудниками автосалона.

Перейдём в ранее установленное приложение Visual Studio 2022 для создания приложения, создадим новый проект с шаблоном WPF (Рисунок 14).

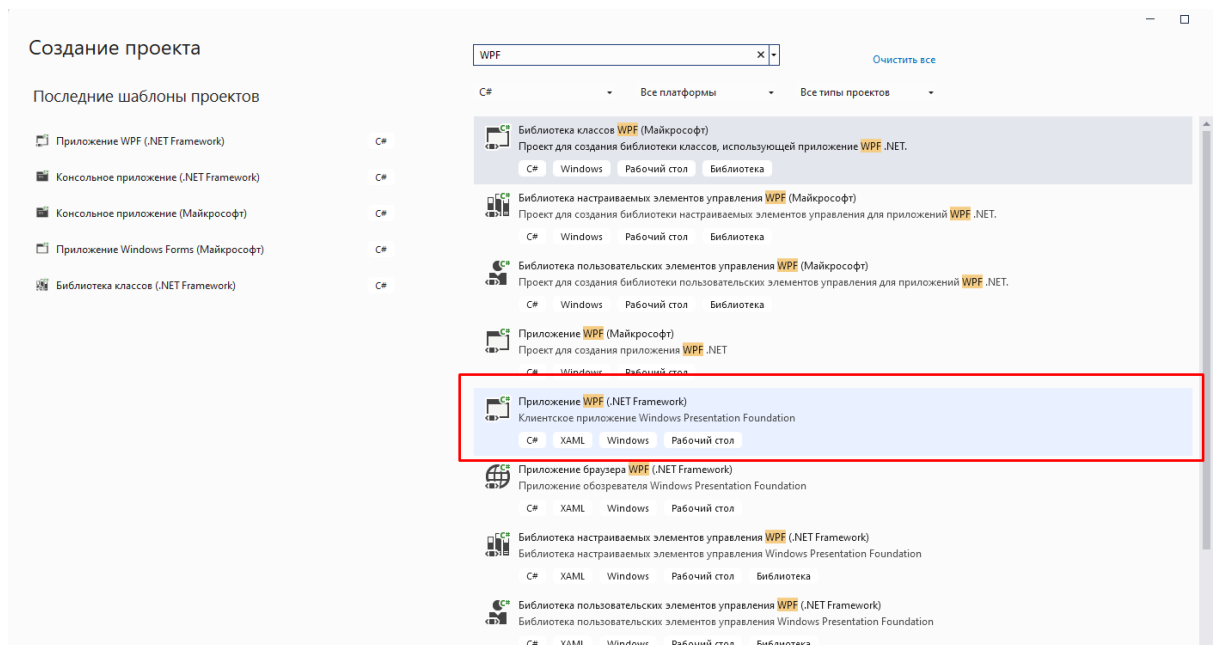


Рисунок 14. Создание нового WPF проекта

Вводим название проекта и жмём кнопку «Создать» (Рисунок 15).

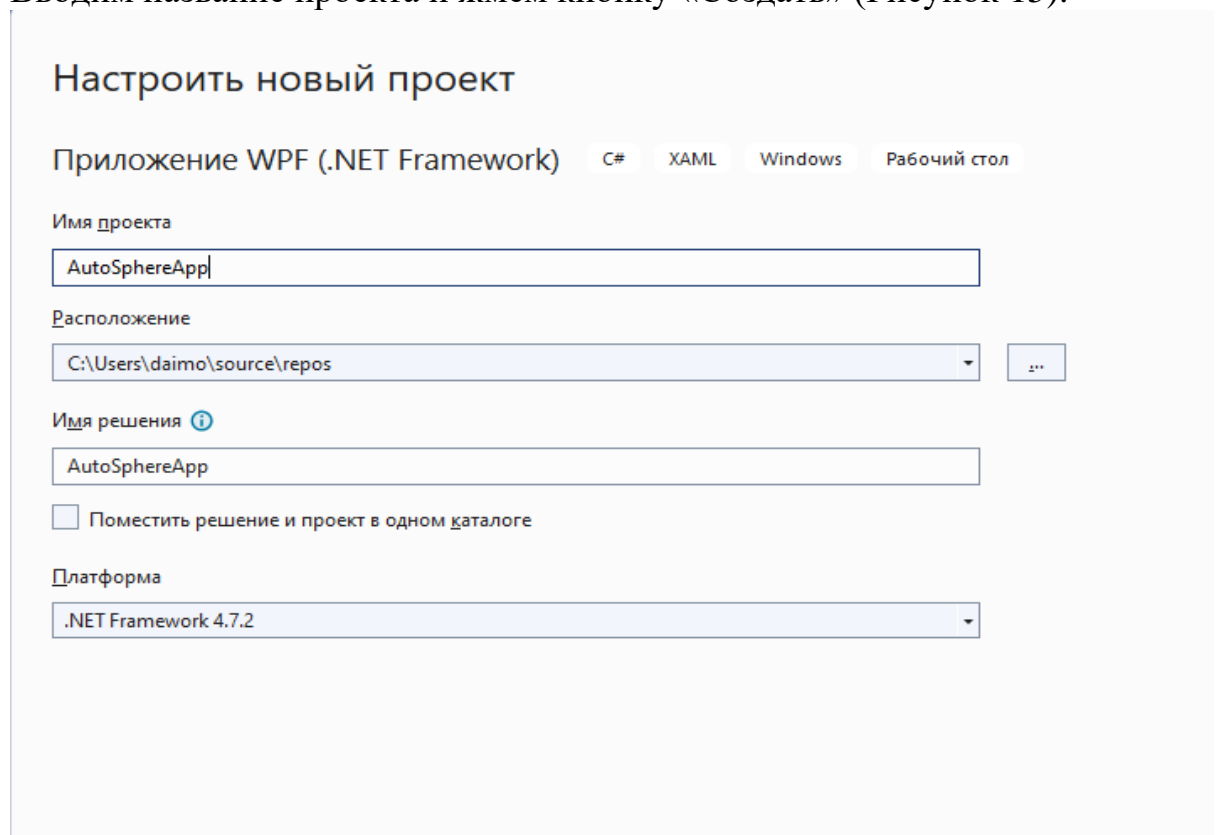


Рисунок 15. Название проекта

После загрузки нужных компонентов перед нами откроется главное окно приложения для разработки пользовательского интерфейса. С

помощью XAML разметки создадим главное окно приложения, «шапку» и «футер», как на сайтах (Рисунок 16).

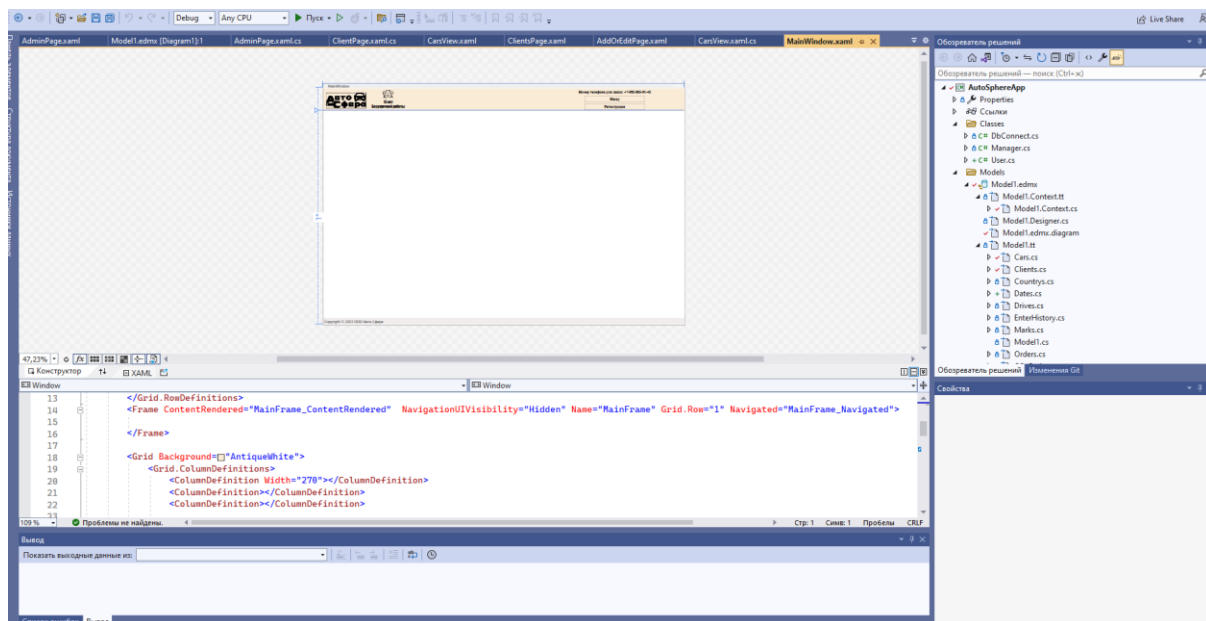


Рисунок 16. Главное окно Visual Studio 2022

С помощью XAML разметки создаём Grid и Frame, нужные кнопки, добавляем логотип кампании. Ниже приведён полный код страницы:

```
<Window x:Class="AutoSphereApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:AutoSphereApp"
        mc:Ignorable="d"
        Title="Main Window" Height="800" Width="1200">
    <Grid Background="White">
        <Grid.RowDefinitions>
            <RowDefinition Height="75" x:Name="Head"></RowDefinition>
            <RowDefinition Height="*"></RowDefinition>
        </Grid.RowDefinitions>
        <Frame ContentRendered="MainFrame_ContentRendered" NavigationUIVisibility="Hidden" Name="MainFrame" Grid.Row="1" Navigated="MainFrame_Navigated">
        </Frame>

        <Grid Background="AntiqueWhite">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="270"></ColumnDefinition>
                <ColumnDefinition></ColumnDefinition>
                <ColumnDefinition></ColumnDefinition>
            </Grid.ColumnDefinitions>
        </Grid>
    </Grid>
```



```

        <Button Name="QuitButton" Visibility="Collapsed" Content="Выход"
Grid.Column="1" Click="Button_Click"></Button>
        <Image HorizontalAlignment="Left" Source="\Resources\Autosphere-
Logotip.png" Width="150" RenderTransformOrigin="0.548,-0.008" Margin="0,0,0,-
7"/>

        <StackPanel Height="auto" HorizontalAlignment="Right" Grid.Col-
umn="0">

            <Image Source="Resources\guaranty.png" HorizontalAlignment="Cen-
ter" Height="37" Width="60" Stretch="Fill"></Image>
            <TextBlock HorizontalAlignment="Center" Grid.Row="1" TextWrap-
ping="Wrap" Text="15 лет" VerticalAlignment="Top" FontFamily="Impact"/>
            <TextBlock HorizontalAlignment="Left" Grid.Row="1" TextWrap-
ping="Wrap" Text="Безупречной работы" VerticalAlignment="Top" FontFamily="Im-
pact"/>

        </StackPanel>

        <StackPanel Grid.Column="3">
            <Label Grid.Column="3" VerticalAlignment="Top" Width="NaN" Hori-
zontalAlignment="Center" Content="Номер телефона для связи: +7-985-803-91-45"
FontFamily="Times New Roman" Height="25" FontWeight="Bold"/>
            <Button FontWeight="Bold" FontFamily="Times New Roman"
Name="BackButton" Background="AntiqueWhite" Width="200" Content="Назад"
Height="20" Click="BackButton_Click" ></Button>
            <Button FontWeight="Bold" FontFamily="Times New Roman" Back-
ground="AntiqueWhite" Name="RegistrationButton" Width="200" Con-
tent="Регистрация" Height="20" Click="RegistrationButton_Click" Margin="5"
></Button>
        </StackPanel>
    </Grid>
    <DockPanel Grid.Row="1" VerticalAlignment="Bottom" Background="Antique-
White">
        <StatusBar DockPanel.Dock="Bottom">
            <TextBlock Text="Copyright © 2023 ООО Авто-Сфера" />

        </StatusBar>
    </Grid>
    <!-- Основное содержимое окна -->
    </Grid>
</DockPanel>

</Grid>
</Window>

```

Этот код является частью XAML-файла главного окна (MainWindow) WPF-приложения для работы с автосалоном.

На первых строках определяются пространства имен (xmlns), которые будут использоваться в дальнейшем, включая пространства имен для основных элементов управления в WPF (PresentationFramework) и для локальных ресурсов приложения.

Затем определяется Grid-контейнер (элемент управления, который упорядочивает другие элементы в виде сетки), содержащий две строки.

Первая строка содержит Grid.RowDefinition для определения размера первой строки, содержащей элементы управления, такие как логотип автосалона.

Вторая строка имеет высоту "*", что означает, что она займет всю оставшуюся высоту окна.

В этой строке находятся три элемента управления:

1. **Frame** - элемент управления, который позволяет встраивать страницы в приложение.
2. **Grid** - элемент управления, который содержит другие элементы управления, включая логотип, кнопку выхода, номер телефона для связи и кнопки навигации.

Теперь, нам необходимо настроить навигацию, для этого мы будем использовать класс «Manager», в котором мы определим ранее созданную в XAML разметке рамку, в которой будет происходить «прорисовка» контента с других страниц. Для создания класса выберем правой кнопкой мыши наш проект и создадим новую папку «Classes» и уже в неё добавим новый класс «Manager» (Рисунок 17).

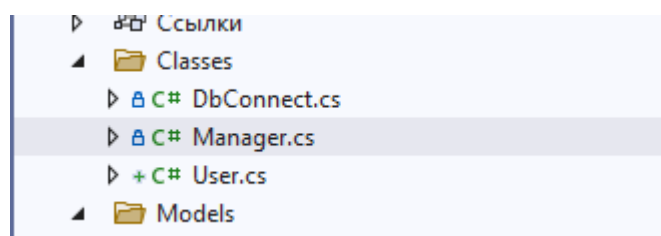


Рисунок 17. Папка Classes и класс Manager

Напишем следующий код в класс Manager:

```
class Manager
{
    public static Frame MainFrame { get; set; }
}
```

Этот код определяет класс Manager, в котором объявляется статическое свойство MainFrame типа Frame. Это свойство позволяет получить доступ к Frame в различных частях приложения, таких как окна и страницы, и использовать его для навигации между различными страницами в приложении. Благодаря использованию статического свойства, MainFrame может быть доступен для всех экземпляров класса Manager без необходимости создания новых экземпляров.

Теперь нам необходимо создать подключение к БД и класс подключения. Добавляем класс Dbconnect в папку Classes и создаём публичный метод:

```
public static Models.AutoSphereEntities modeldb;
```

Эта строка отвечает за подключение добавленной модели к нашему проекту. Теперь нам необходимо создать папку Models и добавить туда модель нашей БД. Для этого нажимаем правой кнопкой мыши по папке Models → Добавить

В поиске вводим «ADO» и нажимаем «Модель ADO.NET EDM» (Рисунок 18).

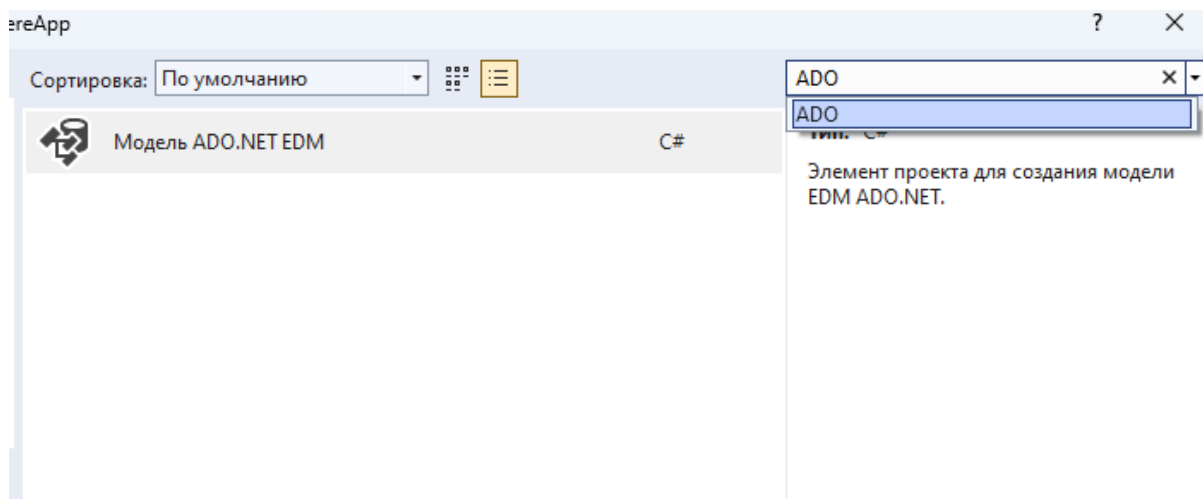


Рисунок 18. Добавление модели БД.

Следуя всем инструкциям, вводим название нашей БД и добавляем в папку Models (Рисунок 19).

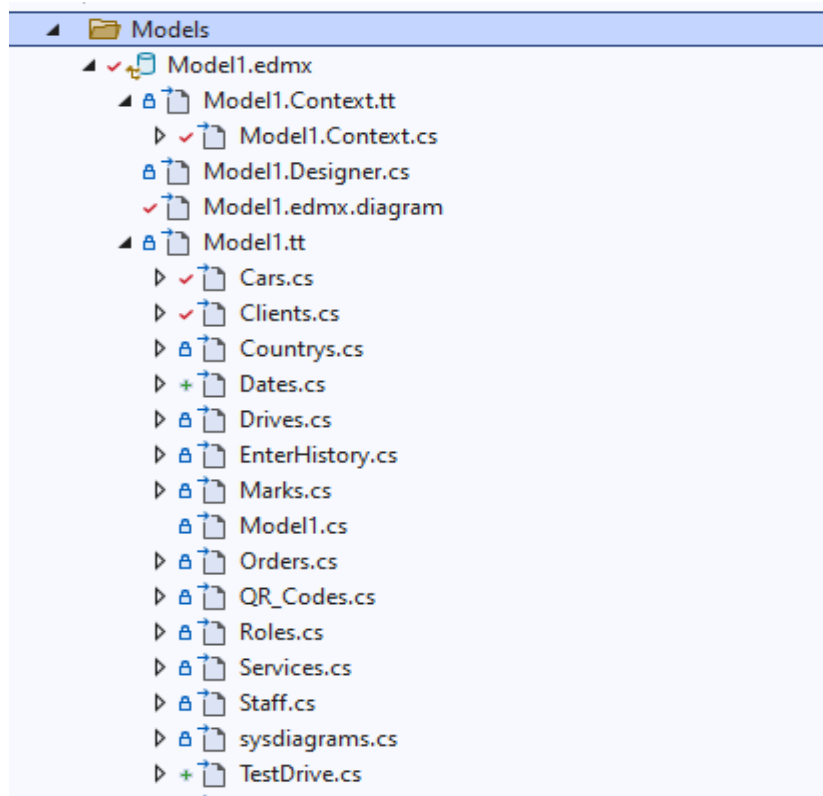


Рисунок 19. Подключенная модель

Теперь, написанный нами ранее код подключения, а именно класс DbConnect будет работать исправно и на страницах далее, нам нужно будет обращаться именно к этому классу для подключения к БД.

Следующим шагом в подключении БД является редактирование файла Model1.Context.cs. Переходим к странице кода и дописываем следующий код:

```
public partial class AutoSphereEntities : DbContext
{
    private static AutoSphereEntities _context;
    public AutoSphereEntities()
        : base("name=AutoSphereEntities")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        throw new UnintentionalCodeFirstException();
    }
    public static AutoSphereEntities GetContext()
    {
        if (_context == null)
            _context = new AutoSphereEntities();
        return _context;
    }

    public virtual DbSet<Cars> Cars { get; set; }
}
```

```

public virtual DbSet<Clients> Clients { get; set; }
public virtual DbSet<Country> Country { get; set; }
public virtual DbSet<Drives> Drives { get; set; }
public virtual DbSet<EnterHistory> EnterHistory { get; set; }
public virtual DbSet<Marks> Marks { get; set; }
public virtual DbSet<Orders> Orders { get; set; }
public virtual DbSet<QR_Codes> QR_Codes { get; set; }
public virtual DbSet<Roles> Roles { get; set; }
public virtual DbSet<Services> Services { get; set; }
public virtual DbSet<Staff> Staff { get; set; }
public virtual DbSet<sysdiagrams> sysdiagrams { get; set; }
public virtual DbSet<Users> Users { get; set; }
public virtual DbSet<Zones> Zones { get; set; }
public virtual DbSet<TestDrive> TestDrives { get; set; }
public virtual DbSet<Dates> Dates { get; set; }
public static Users currentuser = null;
}

```

Этот код определяет класс `AutoSphereEntities`, который наследуется от класса `DbContext` в Entity Framework. Этот класс представляет контекст базы данных и содержит свойства `DbSet` для каждой таблицы в базе данных, а также статический метод `GetContext()`, который возвращает экземпляр контекста базы данных. Конструктор класса инициализирует базовый класс `DbContext` и задает имя подключения к базе данных. Метод `OnModelCreating` используется для настройки модели базы данных. Строка `throw new UnintentionalCodeFirstException();` генерирует исключение, если модель базы данных была создана неправильно.

Теперь наша БД полностью подключена к проекту.

Код приложения

Страница авторизации

Страница авторизации в WPF приложении представляет собой графический интерфейс, который позволяет пользователю ввести свои учетные данные (например, логин и пароль) для входа в приложение. Она обычно содержит следующие элементы:

1. Поля ввода: текстовые поля для ввода имени пользователя (логина) и пароля.

2. Кнопка "Войти": пользователь может нажать на эту кнопку, чтобы отправить данные авторизации на сервер и войти в приложение.
3. Чек бокс «Показать пароль»

Страница авторизации может быть разработана в соответствии с корпоративным стилем приложения или в соответствии с общепринятыми стандартами дизайна пользовательского интерфейса. Она должна быть интуитивно понятной, легко доступной и удобной для использования. Также важно обеспечить безопасность передачи данных пользователя (например, через протокол HTTPS) и защитить приложение от возможных атак на авторизацию (например, через использование капчи или двухфакторной аутентификации).

Начнём с XAML разметки страницы авторизации. Создаём новую страницу и прописываем следующий код:

```
<Grid>
    <StackPanel Width="300">
        <Image Source="Resources\loginImage.png" Height="150" Width="234"
Margin="35" ></Image>
        <Label FontSize="20" Content="Авторизация" HorizontalAlignment="Cen-
ter" FontFamily="Impact"></Label>
        <Label HorizontalAlignment="Center" Content="Логин:" FontFamily="Im-
pact"></Label>
        <TextBox Name="LoginTB" HorizontalAlignment="Center" Width="200"
></TextBox>
        <Label FontFamily="Impact" Content="Пароль" HorizontalAlignment="Cen-
ter" ></Label>
        <TextBox Name="TxbPassword" Width="200" Visibility="Collapsed"
></TextBox>

        <PasswordBox Width="200" Name="Password"></PasswordBox>
        <CheckBox Name="CheckBoxPass" FontFamily="Impact" Unchecked="Check-
Box_Unchecked" HorizontalAlignment="Center" Content="Показать пароль" Margin="5"
Checked="CheckBox_Checked"></CheckBox>
        <Canvas Name="Noises" Visibility="Collapsed"></Canvas>
        <StackPanel Visibility="Collapsed" Width="200" Height="150" Orienta-
tion="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Bottom"
Name="SymbolsGen">

            </StackPanel>
        <TextBox Width="200" Name="GetCapcha" Visibility="Collapsed"></Text-
Box>
        <Button Name="UpdateCapcha" Visibility="Collapsed" Content="Обновить
капчу" Width="200" Margin="5" Click="Button_Click_1"></Button>
        <Button Name="ConfirmCapcha" Width="200" Height="20" Margin="5" Con-
tent="Подтвердить капчу" Visibility="Collapsed" ></Button>
```

```

        <Button FontFamily="Impact" Content="Войти" Width="200" Click="Button_Click" ></Button>
    </StackPanel>
</Grid>

```

Этот код определяет графический интерфейс для страницы авторизации в WPF-приложении.

Код определяет элемент Grid, который содержит StackPanel, который в свою очередь содержит несколько элементов управления, таких как изображение (Image), заголовки (Label), текстовые поля (TextBox), поле для ввода пароля (PasswordBox), флажок (CheckBox) и кнопки (Button).

Некоторые элементы скрыты (свойство Visibility="Collapsed"), такие как текстовое поле для ввода пароля (TextBox Name="TxbPassword") и элементы для генерации капчи.

Код также содержит обработчики событий для кнопок, которые будут выполняться при нажатии пользователем на них. Например, обработчик события Button_Click будет вызываться при нажатии на кнопку "Войти", и обработчик события Button_Click_1 будет вызываться при нажатии на кнопку "Обновить капчу" (Рисунок 20).

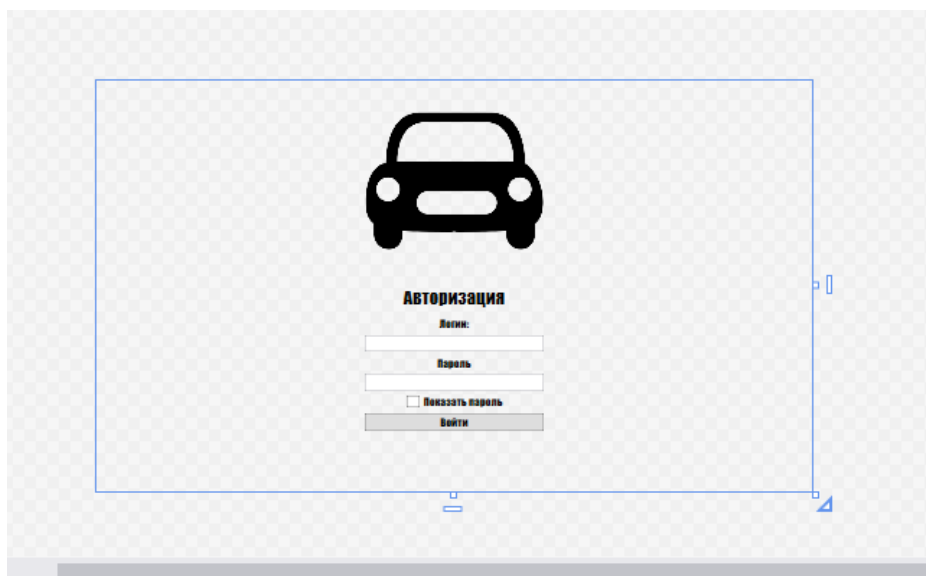


Рисунок 20. Страница авторизации

Теперь перейдём к коду работы страницы. Нам нужно подключить БД, сверить введённый пароль, прописать логику «показать пароль», логику кнопки «Войти», а так же попытки входа, после которых будет выскакивать просьба пройти капчу.

Код страницы:

```
public partial class LogINPage : Page
{
    private int attempts = 0;
    Random _random = new Random();
    private TextBlock myTextBlock;

    public LogINPage()
    {
        InitializeComponent();

        Classes.DbConnect.modeldb = new Models.AutoSphereEntities();
    }

    private void CheckBox_Checked(object sender, RoutedEventArgs e)
    {
        if (string.IsNullOrEmpty>Password.Password))
        {
            MessageBox.Show("Для начала, введите пароль!", "Уведомление");

            CheckBoxPass.IsChecked = false;
        }
        else
        {
            TxbPassword.Visibility = Visibility.Visible;
            Password.Visibility = Visibility.Collapsed;
            TxbPassword.Text = Password.Password;
        }
    }

    private void CheckBox_UnChecked(object sender, RoutedEventArgs e)
    {
        TxbPassword.Visibility = Visibility.Collapsed;
        Password.Visibility = Visibility.Visible;
        TxbPassword.Text = Password.Password;
    }

    private void LogIN()
    {
        try
        {
            var userObj = Classes.DbConnect.modeldb.Users.FirstOrDefault(x=>
x.Login == LoginTB.Text && x.Password == Password.Password);

            if (userObj != null)
```



```

        {
            Models.AutoSphereEntities.currentuser = userObj;
            MessageBox.Show("Здравствуйте " + userObj.Roles.RoleName + ",
" + userObj.Login, "Warning", MessageBoxButton.OK, MessageBoxImage.Information);

            switch (userObj.RoleID)
            {
                case 1:
                    Manager.MainFrame.Navigate(new AdminPage());
                    break;
                case 2:
                    Manager.MainFrame.Navigate(new ManagerPage());
                    break;
                case 3:
                    var clientdata = Classes.DbConnect.modeldb.Clients.FirstOrDefault(c => c.ClientID == userObj.ClientID);
                    if (clientdata != null)
                    {
                        User user = new User
                        {
                            FirstName = clientdata.FirstName,
                            MiddleName = clientdata.MiddleName,
                            LastName = clientdata.LastName,
                            Email = clientdata.Email,
                            PhoneNumber = clientdata.PhoneNumber,
                            PassNumber = clientdata.PassNumber,
                            ClientID = clientdata.ClientID
                        };
                        Manager.MainFrame.Navigate(new ClientPage(user));
                    }
                    else
                    {
                        MessageBox.Show("Данные клиента не найдены!",
"Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);
                    }

                    break;

                default:
                    MessageBox.Show("Данные не обнаружены!",
"Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);
                    break;
            }
        }
        else
        {
            MessageBox.Show("Не верный логин или пароль", "Уведомление");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message.ToString(), "Критическая
работа приложения",
            MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}

private void Button_Click(object sender, RoutedEventArgs e)

```

```

    {
        var userObj = Classes.DbConnect.modeldb.Users.FirstOrDefault(x =>
x.Login == LoginTB.Text && x.Password == Password.Password);

        if (userObj == null)
        {
            MessageBox.Show("Неверный логин или пароль", "Ошибка при
авторизации",
                MessageBoxButton.OK, MessageBoxImage.Error);
            attempts++;
            CheckAttempts();
        }
        else
        {
            LogIN();
        }
    }
    private void CheckAttempts()
    {
        if (attempts == 2)
        {
            MessageBox.Show("Слишком много неудачных попыток! Подтвердите,
что вы человек", "Не удастся войти!", MessageBoxButton.OK,
MessageBoxImage.Warning);
            Noises.Visibility = Visibility.Visible;
            SymbolsGen.Visibility = Visibility.Visible;
            GetCapcha.Visibility = Visibility.Visible;
            UpdateCapcha.Visibility = Visibility.Visible;
            ConfirmCapcha.Visibility = Visibility.Visible;
            LoginTB.Visibility = Visibility.Collapsed;
            Password.Visibility = Visibility.Collapsed;
            GenerateNoisesForCapcha(30);
            GenerateSymbols(3);
            if (GetCapcha.Text != Symbols)
            {
            }

        }
        else
        {
            if (attempts == 3)
            {
                MessageBox.Show("Возможность входа заблокирована", "Слишком
много неудачных попыток", MessageBoxButton.OK, MessageBoxImage.Warning);
                Blocked.Visibility = Visibility.Visible;

                // Блокируем элементы интерфейса
                foreach (UIElement element in UIInt.Children)
                {
                    if (element is Control control && control.Name != "Exit-
Button" && control.Name != "SupportButton")
                    {
                        control.IsEnabled = false;
                    }
                }

                // Запускаем таймер на 10 секунд
                double seconds = 10;
                DispatcherTimer timer = new DispatcherTimer();
                timer.Interval = TimeSpan.FromSeconds(1);
            }
        }
    }

```

```

timer.Tick += (sender, args) =>
{
    seconds--;
    TimerTextBlock.Text = $"Попробуйте снова через {seconds}
сек.";
    if (seconds == 0)
    {
        timer.Stop();
        Blocked.Visibility = Visibility.Collapsed;
        TimerTextBlock.Visibility = Visibility.Collapsed;
        // Разблокируем элементы интерфейса
        foreach (UIElement element in UIInt.Children)
        {
            if (element is Control control && control.Name !=
"ExitButton" && control.Name != "SupportButton")
            {
                control.IsEnabled = true;
                attempts = 0;
            }
        }
    }
};
timer.Start();
TimerTextBlock.Visibility = Visibility.Visible;

}
}
}
private void GenerateNoisesForCapcha(int volumeNoise)
{
    Noises.Visibility = Visibility.Visible;
    for (int i = 0; i < volumeNoise; i++)
    {
        Ellipse ellipse = new Ellipse
        {
            Fill = new SolidColorBrush(Color.FromArgb((byte)_ran-
dom.Next(100, 200),
                (byte)_random.Next(0, 256),
                (byte)_random.Next(0, 256),
                (byte)_random.Next(0, 256)));
        };
        ellipse.Height = ellipse.Width = _random.Next(20, 60);

        Canvas.SetLeft(ellipse, _random.Next(0, 290));
        Canvas.SetTop(ellipse, _random.Next(0, 100));

        Noises.Children.Add(ellipse);
    }
}
public string Symbols = "";
private void GenerateSymbols(int count)
{
    string alphabet = "ABCDEFGHJKLMN123456789";
    for (int i = 0; i < count; i++)

```

```

        {
            string symbol = alphabet.ElementAt(_random.Next(0, alphabet.Length)).ToString();
            TextBlock lbl = new TextBlock();

            lbl.Text = symbol;
            lbl.FontSize = _random.Next(20, 40);
            lbl.RenderTransform = new RotateTransform(_random.Next(-45, 45));
            lbl.Margin = new Thickness(20, 20, 20, 20);

            Noises.Visibility = Visibility.Visible;
            SymbolsGen.Children.Add(lbl);
            Symbols = Symbols + symbol;
            myTextBlock = lbl;
        }
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        Symbols = "";
        SymbolsGen.Children.Clear();
        Noises.Children.Clear();

        GenerateSymbols(3);
        GenerateNoisesForCapcha(25);
    }
}

```

Класс LogINPage является частичным классом Page. Внутри класса есть приватные переменные attempts и _random типа int и Random соответственно, а также приватная переменная типа TextBlock myTextBlock. Конструктор класса LogINPage инициализирует компоненты.

Метод CheckBox_Checked проверяет, что поле Password.Password не пустое, и если да, то выводит сообщение о том, что необходимо ввести пароль, а затем снимает выбор с CheckBoxPass. Если же Password.Password не пустое, то метод отображает TxbPassword и скрывает Password, а также присваивает TxbPassword значение Password.Password.

Метод CheckBox_UnChecked скрывает TxbPassword и отображает Password, а также присваивает TxbPassword значение Password.Password.

Метод LogIN проверяет, есть ли пользователь в базе данных с таким же логином и паролем, как введены пользователем в поле LoginTV и Password. Если есть, то происходит вход в систему, отображается окно с приветствием и перенаправляет пользователя на нужную страницу в зависимости от его роли (AdminPage, ManagerPage, ClientPage). Если же пользователь не найден в базе данных, то выводится сообщение о неправильном логине или пароле. В случае ошибки в процессе выполнения метода LogIN, выводится сообщение с описанием ошибки.

Метод Button_Click проверяет, есть ли пользователь в базе данных с таким же логином и паролем, как введены пользователем в поле LoginTV и Password. Если пользователя нет, то выводится сообщение о неправильном логине или пароле, переменная attemps увеличивается на 1 и вызывается метод CheckAttemps. Если пользователь найден, то происходит вход в систему и вызывается метод LogIN.

Метод CheckAttemps проверяет значение переменной attemps. Если attemps равна 2, то отображаются элементы интерфейса для подтверждения, что пользователь не является ботом, генерируются шумы и символы и отображаются на экране. Если значение поля GetCapcha не равно значению переменной Symbols, то пользователь не сможет продолжить работу с системой. Если значение переменной attemps равно 3, то отображается окно с сообщением о том, что возможность входа заблокирована, блокируются элементы интерфейса (кроме ExitButton и SupportButton), запускается таймер на 10 секунд, после чего элементы интерфейса разблокируются. Если значение переменной attemps не равно 2 или 3, то метод ничего не делает.

Страница авторизации и функция показа пароля (Рисунки 21,22).



Авторизация

Логин:

Пароль

☐ Показать пароль

Войти

Рисунок 21. Страница авторизации



Авторизация

Логин:

Пароль



Показать пароль

Войти

Рисунок 22. Показать пароль

Код показа пароля представлен ниже:

```
private void CheckBox_Checked(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrEmpty>Password.Password))
    {
        MessageBox.Show("Для начала, введите пароль!", "Уведомление");

        CheckBoxPass.IsChecked = false;
    }
    else
    {
        TxbPassword.Visibility = Visibility.Visible;
        Password.Visibility = Visibility.Collapsed;
        TxbPassword.Text = Password.Password;
    }
}
```

Этот код обрабатывает событие Checked (отмечен) для элемента управления CheckBox и выполняет следующие действия:

Проверяет, является ли пароль пустым или равным null с помощью метода `string.IsNullOrEmpty()`.

Если пароль пуст, то выводит сообщение "Для начала, введите пароль!" с помощью метода `MessageBox.Show()` и снимает отметку с элемента `CheckBox`, чтобы снова его отметить, пользователь должен сначала ввести пароль.

Если пароль не пуст, то скрывает поле ввода пароля (`Password`) и отображает введенный пользователем пароль, который был введен в `PasswordBox`

Вывод данных в DataGrid

`DataGrid` - это элемент управления, который отображает данные в виде таблицы в WPF (Windows Presentation Foundation). `DataGrid` позволяет пользователям взаимодействовать с данными, редактировать их, сортировать и фильтровать. Это очень удобно для отображения большого объема данных в удобочитаемом формате, а также для выполнения операций с этими данными.

`DataGrid` в WPF поддерживает связывание данных, множественный выбор, сортировку, фильтрацию, редактирование и многие другие функции. Он также предоставляет множество событий, которые можно использовать для обработки действий пользователя.

XAML Код страницы с выводом данных в `DataGrid` представлен ниже:

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition Height="50"/>
        <RowDefinition Height="50"/>
    </Grid.RowDefinitions>
    <StackPanel Orientation="Horizontal" Grid.Row="1">
        <Button Background="AntiqueWhite" Content="Клиенты" FontFamily="Times
New Roman" Margin="5" Click="Button_Click_2"></Button>
        <Button Background="AntiqueWhite" Content="Заказы" FontFamily="Times
New Roman" Margin="5" Click="Button_Click"></Button>
        <Button Background="AntiqueWhite" Content="Машины" FontFamily="Times
New Roman" Margin="5" Click="Button_Click_3"></Button>
        <Button Background="AntiqueWhite" Content="История входа" FontFam-
ily="Times New Roman" Margin="5" Click="Button_Click_1"></Button>
```



```

        <Button Content="Сотрудники" FontFamily="Times New Roman" Margin="5" Background="Green" BorderBrush="Red"></Button>
        <Button Content="Список машин" FontFamily="Times New Roman" Margin="5" Background="AntiqueWhite" Click="Button_Click_4"></Button>
        <Button Background="AntiqueWhite" Content="Сложить стоимость всех автомобилей" FontFamily="Times New Roman" Margin="5" Click="Button_Click_4" Width="57"></Button>
        <Button Width="100" Click="Button_Click_5" Background="AntiqueWhite" Content="Создать таблицу" Margin="5"></Button>

    </StackPanel>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Right" Width="auto" Grid.Row="1">
        <Button Name="BtnEdit" Content="Добавить данные" Click="BtnRedData_Click" FontFamily="Times New Roman" HorizontalAlignment="Right" Margin="5" Background="AntiqueWhite" />
        <Button Name="BtnDel" Content="Удалить запись" Click="BtnDel_Click" FontFamily="Times New Roman" HorizontalAlignment="Right" Margin="5" Background="AntiqueWhite" />
    </StackPanel>

    <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="0 0 0 25" Grid.RowSpan="2">
        <Button Margin="3" Width="20" Name="BtnFirstPage" Click="BtnFirstPage_Click" Content="&lt;&lt;" FontFamily="Impact" Background="AntiqueWhite" />
        <Button Margin="3" Width="20" Name="BtnPreviousPage" Click="BtnPreviousPage_Click" Content="&lt;" FontFamily="Impact" Background="AntiqueWhite" />
        <Label Name="LblPages" VerticalAlignment="Center" FontSize="16" FontFamily="Impact" Height="28" >2/5</Label>
        <Button Margin="3" Width="20" Name="BtnNextPage" Click="BtnNextPage_Click" Content="&gt;" FontFamily="Impact" Background="AntiqueWhite" />
        <Button Margin="3" Width="20" Name="BtnLastPage" Click="BtnLastPage_Click" Content="&gt;&gt;" FontFamily="Impact" Background="AntiqueWhite" />
        <Button Content="123123" Margin="3" Click="Button_Click_1" ></Button>
    </StackPanel>

    <DataGrid Name="StaffView" Visibility="Visible" IsReadOnly="True" AutoGenerateColumns="False" SelectionChanged="StaffView_SelectionChanged">
        <DataGrid.Columns>
            <DataGridTextColumn Header="Номер сотрудника" Binding="{Binding StaffID}"></DataGridTextColumn>
            <DataGridTextColumn Header="Имя" Binding="{Binding FirstName}"></DataGridTextColumn>
            <DataGridTextColumn Header="Фамилия" Binding="{Binding MiddleName}"></DataGridTextColumn>
            <DataGridTextColumn Header="Отчество" Binding="{Binding LastName}"></DataGridTextColumn>
            <DataGridTextColumn Header="Дата рождения" Binding="{Binding DateOfBirth}"></DataGridTextColumn>
            <DataGridTextColumn Header="Номер роли" Binding="{Binding RoleID}"></DataGridTextColumn>
            <DataGridTextColumn Header="Номер зоны" Binding="{Binding ZoneID}"></DataGridTextColumn>
            <DataGridTextColumn Header="Зарплата" Binding="{Binding Salary, StringFormat='{0:N0} руб.}'"></DataGridTextColumn>
        </DataGrid.Columns>
    </DataGrid>

```

```

        <DataGridTemplateColumn>
            <DataGridTemplateColumn.CellTemplate>
                <DataTemplate>
                    <Button Name="BtnRedData" Content="Редактировать
даные" Click="BtnRedData_Click" FontFamily="Times New Roman" HorizontalAlign-
ment="Right" Margin="5" Background="AntiqueWhite" />
                </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
        </DataGridTemplateColumn>
    </DataGrid.Columns>
</DataGrid>

</Grid>

```

Этот код представляет собой описание разметки пользовательского интерфейса на языке разметки XAML для приложения WPF.

Он содержит элемент Grid, который в свою очередь содержит три строки (строки определяются через элементы RowDefinition), а также несколько элементов StackPanel и DataGrid.

В первой строке находится StackPanel, в котором располагается несколько кнопок для переключения между разными разделами приложения (клиенты, заказы, машины и т.д.). Каждая кнопка имеет свой обработчик событий, который вызывается при щелчке на кнопке.

Во второй строке находится StackPanel, который содержит две кнопки: "Удалить данные" и "Редактировать данные". Эти кнопки также имеют обработчики событий, которые вызываются при щелчке на них.

В третьей строке находится StackPanel, который содержит несколько кнопок для навигации по страницам данных. Также в этой строке есть элемент Label, который отображает текущую страницу и общее количество страниц. В этой строке также есть кнопка, которая имеет обработчик событий, вызываемый при щелчке на кнопке.

В конце кода находится элемент DataGrid, который отображает данные о сотрудниках в таблице. Каждый столбец таблицы определяется через элемент DataGridTextColumn. Последний столбец таблицы определяется через элемент DataGridTemplateColumn и содержит кнопку

"Редактировать данные" для каждой строки таблицы. Каждая кнопка имеет свой обработчик событий, который вызывается при щелчке на кнопке.

Теперь перейдём к коду страницы:

```
int _currentPage = 1, _countInPage = 10, _maxPages;
public StaffPage()
{
    InitializeComponent();
    StaffView.ItemsSource = AutoSphereEntities.GetContext().Staff.ToList();

    StaffView.Visibility = Visibility.Visible;
}

private void BtnLastPage_Click(object sender, RoutedEventArgs e)
{
    _currentPage = _maxPages;
    RefreshData();
}
private void RefreshData()
{
    var data = AutoSphereEntities.GetContext().Staff.ToList();

    // List<Models.Ingredient> listIngredients = _context.Ingredient.ToList();

    _maxPages = (int)Math.Ceiling(data.Count * 1.0 / _countInPage);
    data = data.Skip((_currentPage - 1) *
_countInPage).Take(_countInPage).ToList();

    LblPages.Content = $"{_currentPage}/{_maxPages}";

    StaffView.ItemsSource = data;

    ManageButtonsEnable();
}
private void NavigateToSelectedPage(object sender, RoutedEventArgs e)
{
    Button btn = sender as Button;
    string pageStr = btn.Content.ToString();
    int page = int.Parse(pageStr);
    _currentPage = page;
    RefreshData();
}
private void ManageButtonsEnable()
{
    BtnLastPage.IsEnabled = BtnNextPage.IsEnabled = true;
    BtnFirstPage.IsEnabled = BtnPreviousPage.IsEnabled = true;

    if (_currentPage == 1)
    {
        BtnFirstPage.IsEnabled = BtnPreviousPage.IsEnabled = false;
    }

    if (_currentPage == _maxPages)
    {
        BtnLastPage.IsEnabled = BtnNextPage.IsEnabled = false;
    }
}
```

```

private void BtnNextPage_Click(object sender, RoutedEventArgs e)
{
    _currentPage++;
    RefreshData();
}

private void Button_Click_2(object sender, RoutedEventArgs e)
{
}

private void Button_Click(object sender, RoutedEventArgs e)
{
}

private void Button_Click_3(object sender, RoutedEventArgs e)
{
}

private void Button_Click_1(object sender, RoutedEventArgs e)
{
}

private void Button_Click_4(object sender, RoutedEventArgs e)
{
}

private void Button_Click_5(object sender, RoutedEventArgs e)
{
    Microsoft.Office.Interop.Word.Application wordApp = new Microsoft.Of-
fice.Interop.Word.Application();
    wordApp.Visible = true;

    Microsoft.Office.Interop.Word.Document wordDoc = wordApp.Docu-
ments.Add();
    wordDoc.Activate();

    // Название таблицы
    Microsoft.Office.Interop.Word.Paragraph para = wordDoc.Content.Para-
graphs.Add();
    para.Range.Text = "Список сотрудников";
    para.Range.Font.Bold = 1;
    para.Format.SpaceAfter = 10;
    para.Range.InsertParagraphAfter();

    // Таблица с данными
    Microsoft.Office.Interop.Word.Table table = wordDoc.Ta-
bles.Add(wordDoc.Bookmarks.get_Item(@"\endofdoc").Range, StaffView.Items.Count +
1, 8);
    table.Cell(1, 1).Range.Text = "Номер сотрудника";
    table.Cell(1, 2).Range.Text = "Имя";
    table.Cell(1, 3).Range.Text = "Фамилия";
    table.Cell(1, 4).Range.Text = "Отчество";
    table.Cell(1, 5).Range.Text = "Дата рождения";
    table.Cell(1, 6).Range.Text = "Номер роли";
    table.Cell(1, 7).Range.Text = "Номер зоны";
    table.Cell(1, 8).Range.Text = "Зарплата";
}

```

```

int row = 2;
foreach (var item in StaffView.Items)
{
    Staff staff = item as Staff;
    table.Cell(row, 1).Range.Text = staff.StaffID.ToString();
    table.Cell(row, 2).Range.Text = staff.FirstName;
    table.Cell(row, 3).Range.Text = staff.MiddleName;
    table.Cell(row, 4).Range.Text = staff.LastName;
    table.Cell(row, 5).Range.Text = staff.DateOfBirth.ToString();
    table.Cell(row, 6).Range.Text = staff.RoleID.ToString();
    table.Cell(row, 7).Range.Text = staff.ZoneID.ToString();
    table.Cell(row, 8).Range.Text = staff.Salary.ToString();

    row++;
}

// Добавляем границы к таблице
Microsoft.Office.Interop.Word.Borders borders = table.Borders;
borders.InsideLineStyle = Microsoft.Office.Interop.Word.WdLin-
eStyle.wdLineStyleSingle;
borders.OutsideLineStyle = Microsoft.Office.Interop.Word.WdLin-
eStyle.wdLineStyleSingle;
borders.InsideColor = Microsoft.Office.Interop.Word.WdColor.wdColor-
Black;
borders.OutsideColor = Microsoft.Office.Interop.Word.WdColor.wdColor-
Black;
}

private void BtnDel_Click(object sender, RoutedEventArgs e)
{
    var staffForRemoving = StaffView.SelectedI-
tems.Cast<Staff>().ToList();

    if (MessageBox.Show($"Вы собираетесь удалить {staffForRemov-
ing.Count()} записей", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Ques-
tion) == MessageBoxResult.Yes)
    {
        try
        {
            AutoSphereEntities.GetContext().Staff.RemoveRange(staffForRe-
moving);

            AutoSphereEntities.GetContext().SaveChanges();
            MessageBox.Show("Данные удалены");

            RefreshData();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

private void BtnFirstPage_Click(object sender, RoutedEventArgs e)
{
    _currentPage = 1;
    RefreshData();
}

private void StaffView_SelectionChanged(object sender, Selection-
ChangedEventArgs e)
{

```

```

    }

    private void BtnPreviousPage_Click(object sender, RoutedEventArgs e)
    {
        _currentPage--;
        RefreshData();
    }
    private void BtnRedData_Click(object sender, RoutedEventArgs e)
    {
        Manager.MainFrame.Navigate(new StaffRedOrAdd((sender as Button).DataContext as Staff));
    }
}

```

Код содержит несколько методов обработки событий, которые связаны с кнопками на форме. Некоторые из этих методов не реализованы и не делают ничего. Однако есть методы BtnDel_Click, BtnRedData_Click, NavigateToSelectedPage, ManageButtonsEnable, BtnLastPage_Click, BtnNextPage_Click, BtnPreviousPage_Click и BtnFirstPage_Click, которые выполняют некоторые действия при нажатии на соответствующие кнопки на форме.

Метод RefreshData используется для обновления данных на форме, включая список клиентов, показанный на странице. Этот метод вызывается в нескольких других методах, которые обновляют данные на форме.

Также на форме есть несколько кнопок для навигации по страницам списка клиентов. Методы BtnLastPage_Click, BtnNextPage_Click, BtnPreviousPage_Click и BtnFirstPage_Click обрабатывают нажатие этих кнопок и перемещают пользователя по страницам списка клиентов.

Наконец, есть несколько методов, которые используются для отображения формы для редактирования информации о клиентах. Эти методы включают BtnRedData_Click и RedDataClients.

Запускаем приложение и переходим на страницу StaffPage (Рисунок 23)

MainWindow

АВТО Сфера 15 лет безупречной работы

Номер те

Номер сотрудника	Имя	Фамилия	Отчество	Дата рождения	Номер роли	Номер зоны	Зарплата	
1	Иван	Комаров	Николаевич	7/17/1997 12:00:00 AM	2	1	30,000 Р Руб.	Редактировать данные
2	Даниил	Козлов	Алексеевич	3/29/1994 12:00:00 AM	2	1	450,000 Р Руб.	Редактировать данные
3	Валентина	Морозова	Викторовна	12/6/1992 12:00:00 AM	2	1	50,000 Р Руб.	Редактировать данные
4	Дарья	Лебедева	Николаевна	9/28/1989 12:00:00 AM	1	2	55,000 Р Руб.	Редактировать данные
5	Татьяна	Савицкая	Сергеевна	8/2/1987 12:00:00 AM	2	2	70,000 Р Руб.	Редактировать данные
6	Юлия	Макарова	Викторовна	5/11/1991 12:00:00 AM	2	2	35,000 Р Руб.	Редактировать данные
7	Татьяна	Новикова	Олеговна	2/1/1988 12:00:00 AM	2	1	40,000 Р Руб.	Редактировать данные
8	Егор	Зайцев	Дмитриевич	11/19/1996 12:00:00 AM	2	3	65,000 Р Руб.	Редактировать данные
9	Виктор	Иванов	Олегович	6/7/1998 12:00:00 AM	2	1	25,000 Р Руб.	Редактировать данные
10	Денис	Соколов	Александрович	3/24/1993 12:00:00 AM	2	1	30,000 Р Руб.	Редактировать данные
11	Никита	Лебедев	Викторович	12/10/1984 12:00:00 AM	2	3	33,000 Р Руб.	Редактировать данные
12	Сергей	Сергеев	Романович	8/8/1986 12:00:00 AM	2	3	25,000 Р Руб.	Редактировать данные
13	Антон	Баженов	Валентинович	1/14/1983 12:00:00 AM	1	4	66,000 Р Руб.	Редактировать данные
14	Дмитрий	Смирнов	Юрьевич	9/3/1985 12:00:00 AM	1	4	15,000 Р Руб.	Редактировать данные

Клиенты Заказы Машины История входа **Сотрудники** Список машин Сложить с Создать таблицу

<< < 2/5 > >> 123123

Рисунок 23. Вывод данных о сотрудниках

Поскольку мы находимся на странице сотрудников, кнопку «Сотрудники» горит зелёным цветом, чтобы обозначить на какой именно странице мы находимся. Проверим сортировку пользователей по их зарплате (Рисунок 24).

Номер сотрудника	Имя	Фамилия	Отчество	Дата рождения	Номер роли	Номер зоны	Зарплата	
14	Дмитрий	Смирнов	Юрьевич	9/3/1985 12:00:00 AM	1	4	15,000 Р Руб.	Редактировать данные
9	Виктор	Иванов	Олегович	6/7/1998 12:00:00 AM	2	1	25,000 Р Руб.	Редактировать данные
12	Сергей	Сергеев	Романович	8/8/1986 12:00:00 AM	2	3	25,000 Р Руб.	Редактировать данные
1	Иван	Комаров	Николаевич	7/17/1997 12:00:00 AM	2	1	30,000 Р Руб.	Редактировать данные
10	Денис	Соколов	Александрович	3/24/1993 12:00:00 AM	2	1	30,000 Р Руб.	Редактировать данные
11	Никита	Лебедев	Викторович	12/10/1984 12:00:00 AM	2	3	33,000 Р Руб.	Редактировать данные
6	Юлия	Макарова	Викторовна	5/11/1991 12:00:00 AM	2	2	35,000 Р Руб.	Редактировать данные
7	Татьяна	Новикова	Олеговна	2/1/1988 12:00:00 AM	2	1	40,000 Р Руб.	Редактировать данные
3	Валентина	Морозова	Викторовна	12/6/1992 12:00:00 AM	2	1	50,000 Р Руб.	Редактировать данные
4	Дарья	Лебедева	Николаевна	9/28/1989 12:00:00 AM	1	2	55,000 Р Руб.	Редактировать данные
8	Егор	Зайцев	Дмитриевич	11/19/1996 12:00:00 AM	2	3	65,000 Р Руб.	Редактировать данные
13	Антон	Баженов	Валентинович	1/14/1983 12:00:00 AM	1	4	66,000 Р Руб.	Редактировать данные
5	Татьяна	Савицкая	Сергеевна	8/2/1987 12:00:00 AM	2	2	70,000 Р Руб.	Редактировать данные
2	Даниил	Козлов	Алексеевич	3/29/1994 12:00:00 AM	2	1	450,000 Р Руб.	Редактировать данные

Рисунок 24. Сортировка по зарплате.

На странице так же добавлена кнопка «Создать таблицу», по нажатию на которую будет происходить генерация таблицы в ворд, код представлен ниже:

```
Microsoft.Office.Interop.Word.Application wordApp = new Microsoft.Office.In-
terop.Word.Application();
wordApp.Visible = true;

Microsoft.Office.Interop.Word.Document wordDoc = wordApp.Docu-
ments.Add();
wordDoc.Activate();

// Название таблицы
Microsoft.Office.Interop.Word.Paragraph para = wordDoc.Content.Para-
graphs.Add();
para.Range.Text = "Список сотрудников";
para.Range.Font.Bold = 1;
para.Format.SpaceAfter = 10;
para.Range.InsertParagraphAfter();

// Таблица с данными
Microsoft.Office.Interop.Word.Table table = wordDoc.Ta-
bles.Add(wordDoc.Bookmarks.get_Item("\\endofdoc").Range, StaffView.Items.Count +
1, 8);

table.Cell(1, 1).Range.Text = "Номер сотрудника";
table.Cell(1, 2).Range.Text = "Имя";
table.Cell(1, 3).Range.Text = "Фамилия";
table.Cell(1, 4).Range.Text = "Отчество";
table.Cell(1, 5).Range.Text = "Дата рождения";
table.Cell(1, 6).Range.Text = "Номер роли";
table.Cell(1, 7).Range.Text = "Номер зоны";
table.Cell(1, 8).Range.Text = "Зарплата";

int row = 2;
```



```

foreach (var item in StaffView.Items)
{
    Staff staff = item as Staff;
    table.Cell(row, 1).Range.Text = staff.StaffID.ToString();
    table.Cell(row, 2).Range.Text = staff.FirstName;
    table.Cell(row, 3).Range.Text = staff.MiddleName;
    table.Cell(row, 4).Range.Text = staff.LastName;
    table.Cell(row, 5).Range.Text = staff.DateOfBirth.ToString();
    table.Cell(row, 6).Range.Text = staff.RoleID.ToString();
    table.Cell(row, 7).Range.Text = staff.ZoneID.ToString();
    table.Cell(row, 8).Range.Text = staff.Salary.ToString();

    row++;
}

// Добавляем границы к таблице
Microsoft.Office.Interop.Word.Borders borders = table.Borders;
borders.InsideLineStyle = Microsoft.Office.Interop.Word.WdLin-
eStyle.wdLineStyleSingle;
borders.OutsideLineStyle = Microsoft.Office.Interop.Word.WdLin-
eStyle.wdLineStyleSingle;
borders.InsideColor = Microsoft.Office.Interop.Word.WdColor.wdColor-
Black;
borders.OutsideColor = Microsoft.Office.In-
terop.Word.WdColor.wdColorBlack;

```

В этом коде происходит создание нового Word документа, в начале страницы документа прописывается название таблицы, программа рисует строки таблицы и заполняет их информацией, после чего красит рамки в чёрный цвет и заполняет поля данными, взятыми из DataGridView сотрудников (Рисунок 25).

Список сотрудников

Номер сотрудника	Имя	Фамилия	Отчество	Дата рождения	Номер роли	Номер зоны	Зарплата
1	Иван	Комаров	Николаевич	17.07.1997 0:00:00	2	1	30000,000 0
2	Даниил	Козлов	Алексеевич	29.03.1994 0:00:00	2	1	450000,00 00
3	Валентин а	Морозов а	Виктор на	06.12.1992 0:00:00	2	1	50000,000 0
4	Дарья	Лебедева	Николаевич на	28.09.1989 0:00:00	1	2	55000,000 0
5	Татьяна	Савицкая	Сергеевич а	02.08.1987 0:00:00	2	2	70000,000 0
6	Юлия	Макаров а	Виктор на	11.05.1991 0:00:00	2	2	35000,000 0
7	Татьяна	Новикова	Олеговна	01.02.1988 0:00:00	2	1	40000,000 0
8	Егор	Зайцев	Дмитриевич	19.11.1996 0:00:00	2	3	65000,000 0
9	Виктор	Иванов	Олегович	07.06.1998 0:00:00	2	1	25000,000 0
10	Денис	Соколов	Александрович	24.03.1993 0:00:00	2	1	30000,000 0

Рисунок 25. Таблица сотрудников сооружённая программой

В таблицу вносятся такие данные как:

- Номер сотрудника
- Имя
- Фамилия
- Отчество
- Дата рождения
- Номер роли
- Номер зоны
- Зарплата

Присутствует на странице сотрудников так же и кнопка удаления записей, рассмотрим код:

```
private void BtnDel_Click(object sender, RoutedEventArgs e)
{
    var staffForRemoving = StaffView.SelectedItems.Cast<Staff>().ToList();

    if (MessageBox.Show($"Вы собираетесь удалить {staffForRemoving.Count()} записей", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)
    {
        try
        {
            AutoSphereEntities.GetContext().Staff.RemoveRange(staffForRemoving);

            AutoSphereEntities.GetContext().SaveChanges();
            MessageBox.Show("Данные удалены");

            RefreshData();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

При выборе записи в DataGrid, выбранная строка записывается в переменную staffForRemoving, которая принимает в себя значение выбранных предметов из DataGrid.

Далее, выскакивает сообщение о предупреждении об удалении определённого количества строк и если пользователь выбрал «Да», происходит попытка удалить данные из таблицы, после чего происходит сохранение изменений и обновление данных в DataGrid

Теперь, рассмотрим код редактирования данных:

```
private Staff _currentStaff = null;
private bool _isNew = false;
public StaffRedOrAdd(Staff currentStaff)
{
    InitializeComponent();

    if (currentStaff != null)
    {
        _currentStaff = currentStaff;
        DataContext = _currentStaff;
    }
    else
    {
        _isNew = true;
        _currentStaff = new Staff();
        DataContext = _currentStaff;
    }
}
```

```

    }
}

private void Button_Click(object sender, RoutedEventArgs e)
{
    if (_currentStaff.StaffID == 0 && !_isNew)
    {
        MessageBox.Show("Выберите запись для редактирования или создайте
        новую.");
        return;
    }

    try
    {
        AutoSphereEntities context = AutoSphereEntities.GetContext();

        if (_isNew)
        {
            context.Staff.Add(_currentStaff);
        }

        context.SaveChanges();
        MessageBox.Show("Сохранено");
        Manager.MainFrame.GoBack();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString());
    }
}

private void RoleIDBox_TextChanged(object sender, TextChangedEventArgs e)
{
}

```

Поскольку страница редактирования так же будет отвечать за страницу добавления нового сотрудника, мы создаём две переменные, первая `Staff _currentStaff` равняется нулю, вторая переменная `bool _isNew` равняется `false`. Первая переменная будет проверять, было ли выбрано поле для редактирования и в случае, если поле выбрано, будет происходить конструктор `if`, в котором в качестве `DataContext` присвоится значение выбранной строки, в противном случае произойдёт операция `else`, которая опознает, что данные для редактирования не выбраны и нужно добавить нового сотрудника.

Далее, происходит проверка при нажатии кнопки «Сохранить», если переменная `isNew` равняется `true` – происходит попытка добавить нового

сотрудника в таблицу. Как только сохранение произойдёт успешно, пользователя об этом уведомит месседж бокс (Рисунок 26).

The screenshot shows a form for adding an employee. The fields are: Имя (Name) with value 'Дмитрий', Фамилия (Surname) with value 'Смирнов', Отчество (Patronymic) with value 'Юрьевичч', and Номер з/п (Salary Number) with value '4'. The Зарплата (Salary) field is set to '15000.0000'. A 'Сохранить' (Save) button is at the bottom. A modal dialog box titled 'Сохранено' (Saved) with an 'OK' button is overlaid on the form.

Рисунок 26. Сохранение изменений, добавлена буква «ч» в отчество.

Далее, произойдёт переход на предыдущую страницу и обновление DataGrid с данными (Рисунок 27).

№	Имя	Фамилия	Отчество	Дата рождения	Возраст	З/п	Зарплата	Действия
11	Никита	Лебедев	Викторович	12/10/1984 12:00:00 AM	2	3	33,000 Р. Руб.	Редактировать данные
12	Сергей	Сергеев	Романович	8/8/1986 12:00:00 AM	2	3	25,000 Р. Руб.	Редактировать данные
13	Антон	Баженов	Валентинович	1/14/1983 12:00:00 AM	1	4	66,000 Р. Руб.	Редактировать данные
14	Дмитрий	Смирнов	Юрьевичч	9/3/1985 12:00:00 AM	1	4	15,000 Р. Руб.	Редактировать данные

[Клиенты](#)
[Заказы](#)
[Машины](#)
[История входа](#)
[Сотрудники](#)
[Список машин](#)
[Сложить с](#)
[Создать таблицу](#)

<< < 2/5 > >> 123123

Рисунок 27. Обновлённый DataGrid

ListView

В WPF (Windows Presentation Foundation) есть ListView - элемент управления, который используется для отображения данных в виде списка. Этот элемент имеет много возможностей для настройки отображения данных, таких как сортировка, группировка, фильтрация и выборка. ListView может отображать данные в разных форматах, включая текст, изображения и элементы управления, и поддерживает свойство привязки данных, что позволяет связывать список с данными и автоматически обновлять его при изменении данных.

ListView также позволяет настраивать внешний вид списка с помощью шаблонов элементов. В нашем WPF приложении мы будем использовать ListView для отображения списка автомобилей, их описания, марок, моделей и цен, а также добавим функции поиска, сортировки и фильтрации.

Порядок действий схож с предыдущими, создаём страницу, создаём разметку, пишем код страницы. Начнём с кода XAML:

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="100"></RowDefinition>
        <RowDefinition Height="*"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100"></ColumnDefinition>
        <ColumnDefinition Width="*"></ColumnDefinition>
        <ColumnDefinition Width="100"></ColumnDefinition>
    </Grid.ColumnDefinitions>

    <StackPanel Orientation="Horizontal" Grid.Column="1">
        <Label Content="Поиск машины" Height="25" FontFamily="Times New Roman" VerticalAlignment="Center"></Label>
        <TextBox Name="SearchList" Width="200" Height="25"
            TextChanged="SearchList_TextChanged"></TextBox>
        <CheckBox Name="ActualCars" Content="В наличии" Cursor="Hand" Unchecked="ActualCars_Unchecked" Height="25" Width="83" VerticalContentAlignment="Center" Margin="5" FontFamily="Times New Roman" Checked="ActualCars_Checked"></CheckBox>
        <CheckBox Name="PodZakaz" Content="Под заказ" Cursor="Hand" Unchecked="PodZakaz_Unchecked" Height="25" Width="83" VerticalContentAlignment="Center" Margin="5" FontFamily="Times New Roman" Checked="PodZakaz_Checked"></CheckBox>
        <Label Content="Выберите модель" Height="25" FontFamily="Times New Roman"></Label>
    </StackPanel>
</Grid>
```

```

        <ComboBox Name="MarksList" Width="200" Height="25" FontFamily="Times
New Roman" Cursor="Hand" Margin="5" SelectionChanged="MarksList_Selection-
Changed"></ComboBox>
        <Button Name="ResetFilters" Height="24" Width="126" Content="Сбросить
фильтры" Click="ResetFilters_Click" Cursor="Hand" ></Button>
        <Button Content="Сформировать список автомобилей" Width="250" Mar-
gin="5" Height="25" Click="Button_Click"></Button>

    </StackPanel>
    <StackPanel Grid.Row="1" Grid.Column="0">
        <Image Source="\Resources\guaranty.png" Grid.Row="0"></Image>
        <Button Content="Корзина" Width="auto" Background="AliceBlue"></But-
ton>

        <DataGrid>
            <DataGrid.Columns>
                <DataGridTextColumn Header="Test"></DataGridTextColumn>
            </DataGrid.Columns>
        </DataGrid>
    </StackPanel>
    <WrapPanel Grid.Row="1" Grid.Column="1" Width="auto">
        <ListView Name="CarsViewPanel" Grid.Column="1" Grid.Row="1"
ScrollViewer.HorizontalScrollBarVisibility="Disabled" VerticalAlignment="Center"
>
            <ListView.ItemsPanel>
                <ItemsPanelTemplate>
                    <WrapPanel Orientation="Horizontal" VerticalAlign-
ment="Center"></WrapPanel>
                </ItemsPanelTemplate>
            </ListView.ItemsPanel>
            <ListView.ItemTemplate>

                <DataTemplate>

                    <Grid>
                        <Grid.RowDefinitions>
                            <RowDefinition Height = "75" ></RowDefinition >
                            <RowDefinition Height = "310" ></RowDefinition >
                            <RowDefinition Height = "auto" ></RowDefinition >
                            <RowDefinition Height = "auto" ></RowDefinition >
                            <RowDefinition Height = "auto" ></RowDefinition >
                        </Grid.RowDefinitions >
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="500"></ColumnDefinition>
                            <ColumnDefinition Width="auto"></ColumnDefini-
tion>

                        </Grid.ColumnDefinitions>
                        <Image Width = "400" Grid.Row = "1" Stretch = "Uni-
formToFill" HorizontalAlignment = "Center" Margin = "5" >
                            <Image.Source>

                                <ImageSource >\Resources\loginImage.png</Im-
ageSource >

                            </Image.Source>
                        </Image >
                        <TextBlock Text = "{Binding CarID}" VerticalAlignment
= "Center" TextAlignment = "Center" Width = "450"
                            TextWrapping = "Wrap" HorizontalAlignment =
"Center" FontSize = "26" Grid.Row = "0" ></TextBlock >

```

```

        <TextBlock Text = "{Binding Mark}" Grid.Row = "2"
HorizontalAlignment = "Center" FontSize = "20" FontWeight = "Bold" ></TextBlock >
        <TextBlock Text = "{Binding Model}" Grid.Row = "3"
FontSize = "14" HorizontalAlignment = "Right" ></TextBlock >
        <TextBlock Text = "{Binding Cost}" Grid.Row = "3"
FontSize = "14" HorizontalAlignment = "Left" ></TextBlock >
        <TextBlock Text="Описание:" Grid.Column="2"
Grid.Row="0" VerticalAlignment="Top"></TextBlock>
        <TextBlock Text="{Binding Actuality}" Grid.Row="3"
FontSize="14" FontWeight="Bold" HorizontalAlignment="Center"></TextBlock>
        <TextBlock Text="adgasdgsdfadgasdgsdfadgasdgsd-
fadgasdfadgasdgsdfadgfadgasdgsd-
fadgasdgsdfadgasdgsdfadsgsdgfadgasdgsdfadgasdgsdasdgsdfadgasdgsdfadsgsdgfadgasdgs
dfadgasdgsddfsgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsdgsd-
fadgasdgsdfadgasdgsdf" Grid.Row="1" Width="300" Height="1000" Grid.Column="2"
TextWrapping="Wrap"></TextBlock>

    </Grid >
    </DataTemplate>
</ListView.ItemTemplate>
</ListView>
</WrapPanel>

</Grid>

```

Этот код является разметкой XAML для создания пользовательского интерфейса. Элемент Grid используется для размещения других элементов интерфейса в строках и столбцах, которые определяются с помощью RowDefinition и ColumnDefinition соответственно.

Первый столбец шириной в 10 пикселей используется для создания промежутка между левой границей окна и основным содержимым интерфейса. Второй столбец шириной в 1700 пикселей содержит StackPanel, который содержит несколько элементов интерфейса, таких как Label, TextBox, CheckBox, Button и ComboBox. Эти элементы используются для поиска и фильтрации информации.

Третий столбец шириной в 200 пикселей содержит StackPanel, который содержит элементы интерфейса для корзины и кнопку заказа тест-драйва.

Строка 0 высотой в 100 пикселей используется для создания пространства для элементов интерфейса в StackPanel из второго столбца.

Строка 1 используется для размещения WrapPanel, который содержит ListView для отображения автомобилей.

WrapPanel позволяет элементам интерфейса перемещаться на новую строку при достижении конца текущей строки. ListView содержит шаблон элемента, который определяется с помощью DataTemplate. DataTemplate содержит несколько элементов TextBlock и Image, которые используются для отображения информации о каждом автомобиле.

Стек Panel в третьем столбце содержит Image и Label, которые отображают гарантию на автомобиль и корзину со списком заказов. Он также содержит DataGrid, который используется для отображения списка заказов в корзине, а также кнопку для заказа тест-драйва. Теперь приступим к рабочему коду, который будет отвечать за вывод информации в ListView:

```
public partial class CarsView : System.Windows.Controls.Page
{
    private List<Cars> allCars;

    public CarsView()
    {
        InitializeComponent();

        // Получаем все марки и привязываем к комбобоксу
        var allMarks = AutoSphereEntities.GetContext().Marks.ToList();
        MarksList.ItemsSource = allMarks;
        MarksList.DisplayMemberPath = "MarkName";
        MarksList.SelectedValue = "MarkName";

        // Получаем все автомобили и запоминаем их
        allCars = AutoSphereEntities.GetContext().Cars.ToList();

        // Заполняем список автомобилей по умолчанию
        UpdateCarsPage();
    }

    private void UpdateCarsPage()
    {
        // Создаем копию всех автомобилей, чтобы фильтровать только ее
        var currentCars = allCars.ToList();

        // Фильтруем список автомобилей по поисковому запросу
        string searchText = SearchList.Text.ToLower();
        currentCars = currentCars.Where(p => p.Model.ToLower().Contains(searchText) || p.Mark.ToLower().Contains(searchText)).ToList();

        // Фильтруем список автомобилей по состоянию "В наличии"
        if (ActualCars.IsChecked == true)
        {
```

```

        currentCars = currentCars.Where(p => p.Actuality == "В
наличии").ToList();
    }

    // Фильтруем список автомобилей по состоянию "Под заказ"
    if (PodZakaz.IsChecked == true)
    {
        currentCars = currentCars.Where(p => p.Actuality == "Под
заказ").ToList();
    }

    // Фильтруем список автомобилей по выбранной марке
    var selectedMark = MarksList.SelectedItem as Marks;
    if (selectedMark != null)
    {
        string selectedBrand = selectedMark.MarkName;
        currentCars = currentCars.Where(p => p.Mark == selected-
Brand).ToList();
    }

    // Обновляем список автомобилей на странице
    CarsViewPanel.ItemsSource = currentCars;
}

private void ActualCars_Unchecked(object sender, RoutedEventArgs e)
{
    UpdateCarsPage();
}

e) private void SearchList_TextChanged(object sender, TextChangedEventArgs
{
    UpdateCarsPage();
}

private void MarksList_SelectionChanged(object sender, Selection-
ChangedEventArgs e)
{
    UpdateCarsPage();
}

private void ActualCars_Checked(object sender, RoutedEventArgs e)
{
    UpdateCarsPage();
}

private void PodZakaz_Unchecked(object sender, RoutedEventArgs e)
{
    UpdateCarsPage();
}

private void PodZakaz_Checked(object sender, RoutedEventArgs e)
{
    UpdateCarsPage();
}

private void ResetFilters_Click(object sender, RoutedEventArgs e)
{
    // Сбрасываем значения фильтров и поискового поля
    SearchList.Text = "";
    MarksList.SelectedIndex = -1;
    ActualCars.IsChecked = false;
    PodZakaz.IsChecked = false;
}

```

```

        // Сбрасываем сортировку
        CollectionViewSource.DefaultView(CarsViewPanel.Items-
Source).SortDescriptions.Clear();

        // Сбрасываем фильтры
        CollectionViewSource.DefaultView(CarsViewPanel.ItemsSource).Filter
= null;

        // Обновляем список автомобилей на странице
        UpdateCarsPage();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        Microsoft.Office.Interop.Word.Application wordApp = new Microsoft.Of-
fice.Interop.Word.Application();
        wordApp.Visible = true;

        Microsoft.Office.Interop.Word.Document wordDoc = wordApp.Docu-
ments.Add();
        wordDoc.Activate();

        // Название таблицы
        Microsoft.Office.Interop.Word.Paragraph para = wordDoc.Content.Para-
graphs.Add();
        para.Range.Text = "Список автомобилей";
        para.Range.Font.Bold = 1;
        para.Format.SpaceAfter = 10;
        para.Range.InsertParagraphAfter();

        // Таблица с данными
        Microsoft.Office.Interop.Word.Table table = wordDoc.Ta-
bles.Add(wordDoc.Bookmarks.get_Item(@"\endofdoc").Range, CarsView-
Panel.Items.Count + 1, 5);
        table.Cell(1, 1).Range.Text = "Марка";
        table.Cell(1, 2).Range.Text = "Модель";
        table.Cell(1, 3).Range.Text = "Цвет";
        table.Cell(1, 4).Range.Text = "Стоимость";
        table.Cell(1, 5).Range.Text = "Статус";

        int row = 2;
        foreach (var item in CarsViewPanel.Items)
        {
            Cars car = item as Cars;
            table.Cell(row, 1).Range.Text = car.Mark;
            table.Cell(row, 2).Range.Text = car.Model;
            table.Cell(row, 3).Range.Text = car.Color;
            table.Cell(row, 4).Range.Text = car.Cost.ToString();
            table.Cell(row, 5).Range.Text = car.Actuality;

            row++;
        }

        // Добавляем границы к таблице
        Microsoft.Office.Interop.Word.Borders borders = table.Borders;
        borders.InsideLineStyle = Microsoft.Office.Interop.Word.WdLin-
eStyle.wdLineStyleSingle;
        borders.OutsideLineStyle = Microsoft.Office.Interop.Word.WdLin-
eStyle.wdLineStyleSingle;
        borders.InsideColor = Microsoft.Office.Interop.Word.WdColor.wdColor-
Black;
    }
}

```

```

        borders.OutsideColor = Microsoft.Office.Interop.Word.WdColor.wdColor-
Black;

        // Сохранение документа
    }
}

```

Этот код представляет пользовательский интерфейс для отображения автомобилей на странице Windows. Страница содержит несколько фильтров, которые пользователь может использовать для поиска автомобилей по различным критериям. Когда пользователь изменяет любой из фильтров, отображение автомобилей на странице обновляется в соответствии с выбранными критериями.

Класс `CarsView` наследует класс `Page` из пространства имен `System.Windows.Controls` и содержит методы и события, которые позволяют отображать и фильтровать список автомобилей на странице.

При создании экземпляра класса `CarsView`, метод `InitializeComponent()` инициализирует компоненты страницы, а затем выполняется инициализация переменных и обновление списка автомобилей на странице с помощью метода `UpdateCarsPage()`.

Метод `UpdateCarsPage()` обновляет список автомобилей на странице в соответствии с выбранными фильтрами. Он создает копию всех автомобилей и затем фильтрует этот список по поисковому запросу, состоянию "В наличии", состоянию "Под заказ" и выбранной марке. В результате получается список автомобилей, который соответствует выбранным критериям, и этот список отображается на странице.

Методы `ActualCars_Unchecked`, `SearchList_TextChanged`, `MarksList_SelectionChanged`, `ActualCars_Checked`, `PodZakaz_Unchecked` и `PodZakaz_Checked` обновляют список автомобилей на странице при изменении соответствующего фильтра.

Метод `ResetFilters_Click` сбрасывает все значения фильтров и поискового поля, а также сбрасывает сортировку и фильтры. Затем он

вызывает метод `UpdateCarsPage()` для обновления списка автомобилей на странице.

Метод `Button_Click` открывает приложение Microsoft Word и создает таблицу со списком автомобилей на странице. Затем он заполняет таблицу данными из списка автомобилей на странице и добавляет границы к таблице.

Запускаем приложение и смотрим результат:

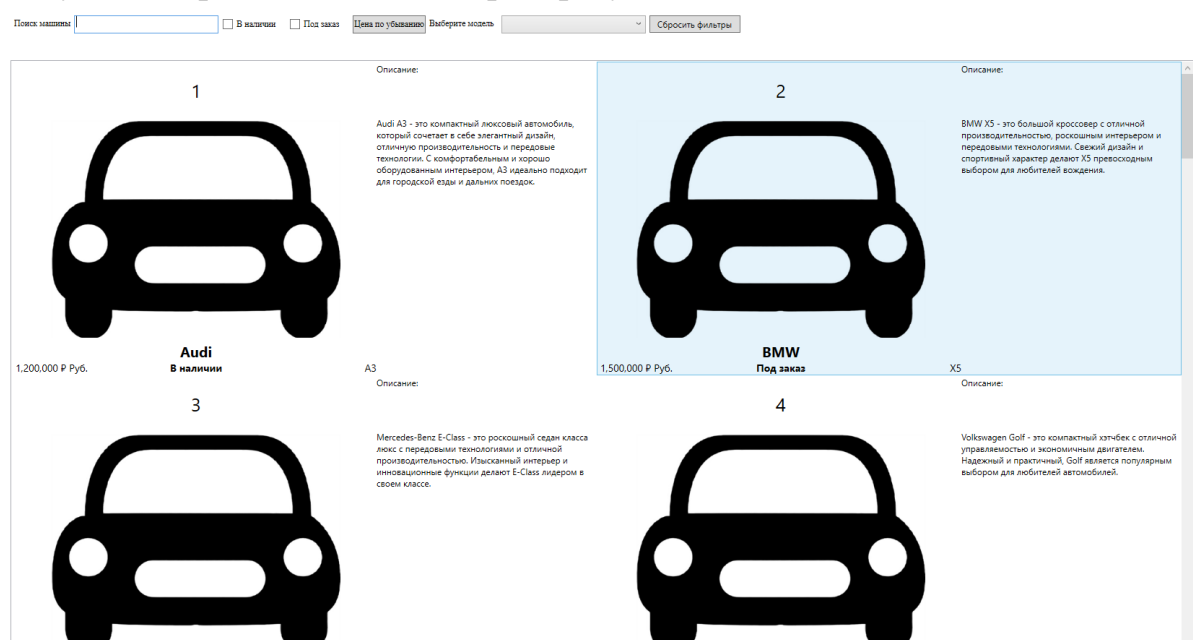


Рисунок 28. ListView всех автомобилей.

Из базы данных было взято описание автомобилей и добавлено в текстовый блок `ListView`, таким образом у каждого автомобиля своё описание.

За поиск автомобиля отвечает следующий код:

```
string searchText = SearchList.Text.ToLower();
currentCars = currentCars.Where(p => p.Model.ToLower().Contains(searchText) || p.Mark.ToLower().Contains(searchText)).ToList();
```

Этот код осуществляет поиск элементов в списке `currentCars` по заданному тексту, который хранится в свойстве `Text` элемента управления `SearchList`.

Первая строка преобразует `searchText` в нижний регистр, чтобы поиск был регистронезависимым.

Вторая строка использует метод Where() для выборки всех элементов списка currentCars, у которых Model или Mark содержат searchText, используя метод ToLower() для приведения Model и Mark к нижнему регистру и метод Contains() для проверки содержания искомого текста.

Затем метод ToList() преобразует результат выборки в список и сохраняет его в переменной currentCars. Таким образом, в currentCars будут храниться только те элементы, у которых Model или Mark содержат искомый текст, независимо от регистра.

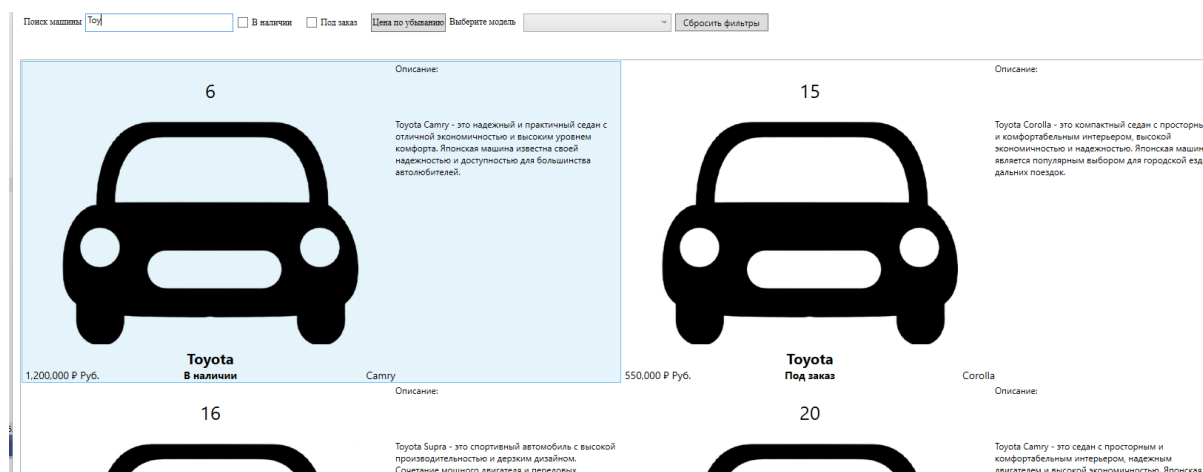


Рисунок 29. Поиск автомобиля Toyota.

С помощью следующего кода была добавлена сортировка автомобилей по цене, от большей к меньшей:

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    // Получаем текущий список автомобилей из ItemsSource
    var currentCars = CarsViewPanel.ItemsSource as List<Cars>;

    // Сортируем список автомобилей в зависимости от флага
    if (!isSortedAscending)
    {
        currentCars = currentCars.OrderBy(p => p.Cost).ToList();
        isSortedAscending = true;
    }
    else
    {
        currentCars = currentCars.OrderByDescending(p =>
p.Cost).ToList();
        isSortedAscending = false;
    }

    // Обновляем список автомобилей на странице
    CarsViewPanel.ItemsSource = currentCars;
}
```

Этот код отвечает за обработку события нажатия на кнопку. При нажатии на кнопку выполняются следующие действия:

1. Получение текущего списка автомобилей из элемента `CarsViewPanel.ItemsSource` в виде списка объектов класса `Cars`.
2. Проверка флага `isSortedAscending`, который указывает, отсортирован ли текущий список по возрастанию цены. Если флаг установлен в `false`, то список сортируется по возрастанию цены и флаг устанавливается в `true`. Если флаг установлен в `true`, то список сортируется по убыванию цены и флаг устанавливается в `false`.
3. Обновление элемента `CarsViewPanel.ItemsSource` новым отсортированным списком автомобилей.

Таким образом, код выполняет сортировку списка объектов класса `Cars` по цене (в порядке возрастания или убывания) и обновляет элемент `CarsViewPanel` для отображения отсортированного списка (Рисунки 30,31).

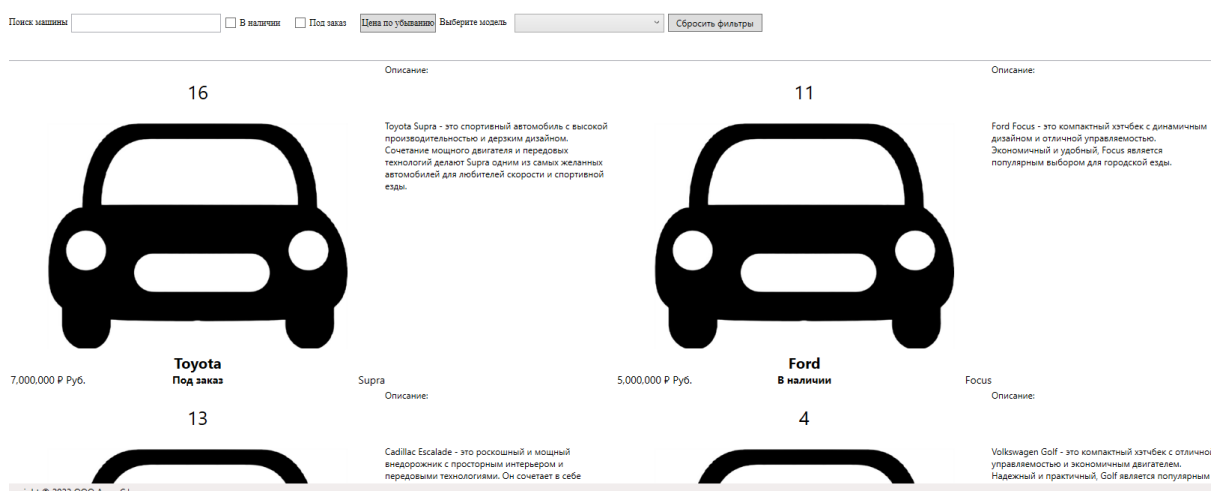


Рисунок 30. Вывод автомобилей с самой высокой ценой

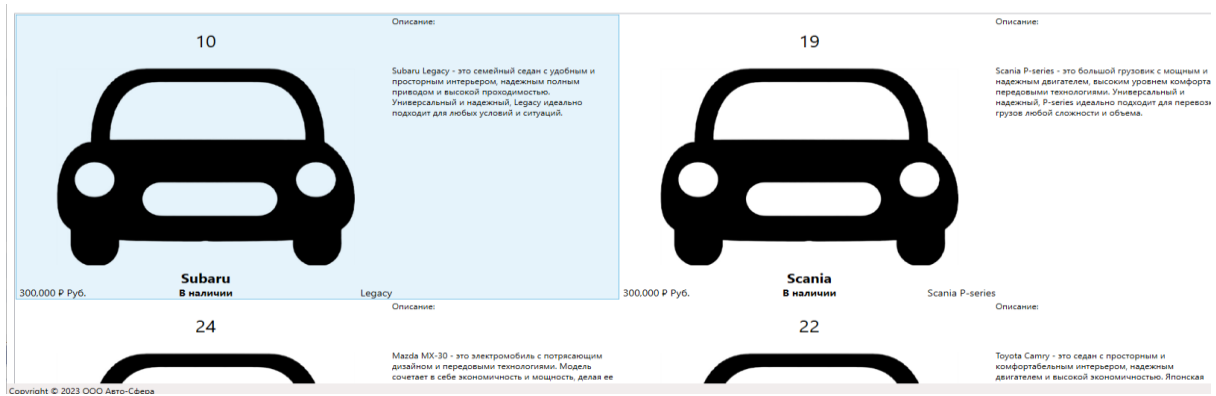


Рисунок 31. Вывод автомобилей сначала с низшей ценой

Так же была добавлена кнопка сброса фильтров, чтобы снять все галочки, очистить поле поиска:

```
private void ResetFilters_Click(object sender, RoutedEventArgs e)
{
    // Сбрасываем значения фильтров и поискового поля
    SearchList.Text = "";
    MarksList.SelectedIndex = -1;
    ActualCars.IsChecked = false;
    PodZakaz.IsChecked = false;

    // Сбрасываем сортировку
    CollectionViewSource.GetDefaultView(CarsViewPanel.ItemsSource)
        .SortDescriptions.Clear();

    // Сбрасываем фильтры
    CollectionViewSource.GetDefaultView(CarsViewPanel.ItemsSource).Filter
        = null;

    ClientDataPanel.Visibility = Visibility.Collapsed;
    // Обновляем список автомобилей на странице
    UpdateCarsPage();
}
```

Этот код отвечает за сброс всех фильтров и сортировок, установленных на странице, и обновляет список автомобилей на странице.

Конкретно, код выполняет следующие действия:

1. Сбрасывает значения всех элементов управления фильтрами на странице: поле поиска, выпадающий список марок, флажки "Актуальные" и "Под заказ".
2. Сбрасывает сортировку списка автомобилей на странице, удаляя все установленные ограничения сортировки.
3. Сбрасывает фильтры списка автомобилей на странице, удаляя все установленные фильтры.
4. Скрывает панель с данными клиента.

Обновляет список автомобилей на странице, вызывая метод UpdateCarsPage() (Рисунки 32,33).

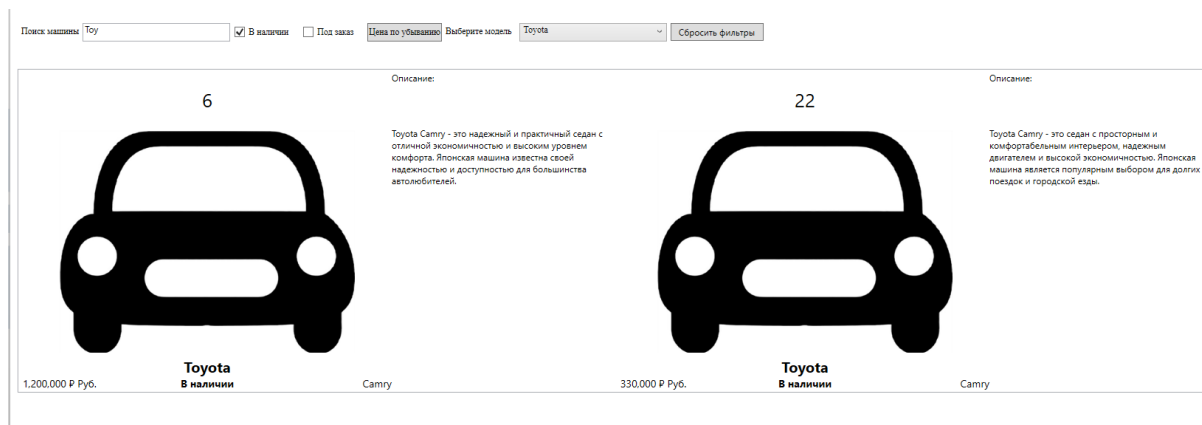


Рисунок 32. Вывод машин с фильтрами

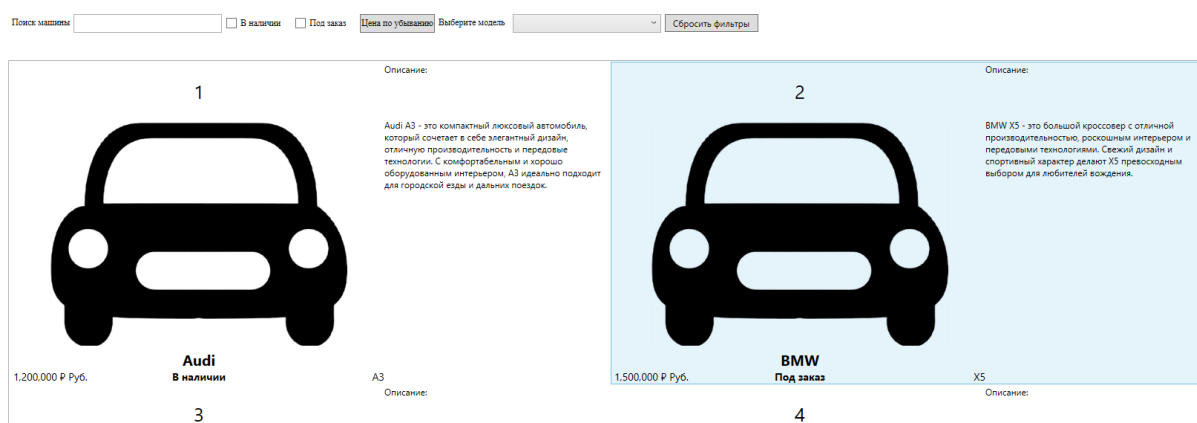


Рисунок 33. Сброшенные фильтры

На странице ListView так же есть ComboBox, в котором представлен список всех марок автомобилей. При выборе определённой марки происходит срабатывание следующего кода:

```
var selectedMark = MarksList.SelectedItem as Marks;
if (selectedMark != null)
{
    string selectedBrand = selectedMark.MarkName;
    currentCars = currentCars.Where(p => p.Mark == selected-
Brand).ToList();
}

// Обновляем список автомобилей на странице
CarsViewPanel.ItemsSource = currentCars;
```

Создаётся новая переменная, которая принимает значение выбранной марки как Марку, далее, происходит проверка, если переменная не равна нулю создаётся переменная, которая принимает значение марки из таблицы

Cars и выводит список автомобилей в соответствии с выборкой (Рисунок 34).



Рисунок 34. Выбранная модель «Porsche»

Происходит вывод автомобилей исключительно выбранной марки. На странице администратора была добавлена кнопка «Сформировать список автомобилей», по нажатию на которую происходит срабатывание следующего кода:

```
Microsoft.Office.Interop.Word.Application wordApp = new Microsoft.Office.In-
terop.Word.Application();
wordApp.Visible = true;

Microsoft.Office.Interop.Word.Document wordDoc = wordApp.Docu-
ments.Add();
wordDoc.Activate();

// Название таблицы
Microsoft.Office.Interop.Word.Paragraph para = wordDoc.Content.Para-
graphs.Add();
para.Range.Text = "Список автомобилей";
para.Range.Font.Bold = 1;
para.Format.SpaceAfter = 10;
para.Range.InsertParagraphAfter();

// Таблица с данными
Microsoft.Office.Interop.Word.Table table = wordDoc.Ta-
bles.Add(wordDoc.Bookmarks.get_Item("\\endofdoc").Range, CarsView-
Panel.Items.Count + 1, 5);
table.Cell(1, 1).Range.Text = "Марка";
table.Cell(1, 2).Range.Text = "Модель";
table.Cell(1, 3).Range.Text = "Цвет";
table.Cell(1, 4).Range.Text = "Стоимость";
table.Cell(1, 5).Range.Text = "Статус";
```

```

int row = 2;
foreach (var item in CarsViewPanel.Items)
{
    Cars car = item as Cars;
    table.Cell(row, 1).Range.Text = car.Mark;
    table.Cell(row, 2).Range.Text = car.Model;
    table.Cell(row, 3).Range.Text = car.Color;
    table.Cell(row, 4).Range.Text = car.Cost.ToString();
    table.Cell(row, 5).Range.Text = car.Actuality;

    row++;
}

// Добавляем границы к таблице
Microsoft.Office.Interop.Word.Borders borders = table.Borders;
borders.InsideLineStyle = Microsoft.Office.Interop.Word.WdLineStyle.wdLineStyleSingle;
borders.OutsideLineStyle = Microsoft.Office.Interop.Word.WdLineStyle.wdLineStyleSingle;
borders.InsideColor = Microsoft.Office.Interop.Word.WdColor.wdColorBlack;
borders.OutsideColor = Microsoft.Office.Interop.Word.WdColor.wdColorBlack;

```

Создаётся новый документ, в нём генерируется таблица, выделяются границы каждой ячейки, в каждую ячейку заносятся данные из таблицы Cars для соответствующей ячейки (Рисунок 35).

Список автомобилей

Марка	Модель	Цвет	Стоимость	Статус
Audi	A3	Серебристый-белый	1200000,0000	В наличии
BMW	X5	Чёрный	1500000,0000	Под заказ
Mercedes-Benz	E-Class	Чёрный	600000,0000	В наличии
Volkswagen	Golf	Тёмно-синий	2500000,0000	Под заказ
Porsche	Panamera	Белый	1000000,0000	В наличии
Toyota	Camry	Белый	1200000,0000	В наличии
Honda	Civic	Голубой	1240000,0000	Под заказ
Mazda	Mazda6	Чёрный	500000,0000	В наличии
Nissan	GT-R	Чёрный	450000,0000	Под заказ
Subaru	Legacy	Белый	300000,0000	В наличии
Ford	Focus	Жёлтый	500000,0000	В наличии
Ford	Mustang	Жёлтый	2000000,0000	Под заказ
Cadillac	Escalade	Чёрный	3000000,0000	В наличии
Jeep	Renegade	Коричневый	500000,0000	В наличии
Toyota	Corolla	Серебристый	550000,0000	Под заказ

Рисунок 35. Сгенерированная таблица машин

Корзина

Корзина будет представлять из себя DataGrid, в который будут выводиться заказанные тест драйвы конкретного пользователя. Клиент заказывает тест драйв и в базу данных заносится номер машины, номер клиента, дата и стоимость, которая будет составлять 0.5% от всей стоимости автомобиля.

По нажатию кнопки «Заказать тест драйв» будет происходить выполнение следующего кода:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    var selected = CarsViewPanel.SelectedItem as Cars;
    var selectedDate = DatesComboBox.SelectedItem as Dates;
    DateTime? date = selectedDate?.Date;
    decimal discount = selected.Cost.Value * (decimal)0.005;
    decimal discountedPrice = selected.Cost.Value - discount;

    // Выводим данные в MessageBox с учетом скидки
    string message = "Марка: " + selected.Mark + "\n" +
                    "Модель: " + selected.Model + "\n" +
                    "Год выпуска: " + selected.Year + "\n" +
                    "Цвет: " + selected.Color + "\n" +
                    "Цена: " + discount.ToString("#.##") + " руб. (Цена
теста-драйва составляет 0.5% от общей стоимости автомобиля ";

    MessageBoxResult result = MessageBox.Show(message, "Подтверждение
данных", MessageBoxButton.YesNo, MessageBoxImage.Question);
    var currentUser = AutoSphereEntities.GetContext().Users.FirstOrDefault(u => u.ClientID == user.ClientID);
    if (currentUser == null)
    {
        return;
    }
    var newTestDrive = new TestDrive
    {
        CarID = selected.CarID,
        //Data = selectedDate.Value,
        ClientID = currentUser.ClientID,
        Price = discount,
        Data = date // явное преобразование типа
    };
}
```

Этот код относится к событию клика на кнопке и выполняет несколько действий:

1. Получает выбранный элемент из панели просмотра автомобилей (предполагается, что в этой панели отображаются объекты типа "Cars").

2. Получает выбранную дату из выпадающего списка "DatesComboBox" (предполагается, что в этом списке отображаются объекты типа "Dates").

3. Вычисляет скидку на автомобиль в размере 0,5% от его стоимости.

4. Вычисляет цену тест-драйва автомобиля с учетом скидки.

5. Выводит сообщение о выбранном автомобиле и цене тест-драйва в диалоговом окне "MessageBox".

6. Получает текущего пользователя из базы данных (предполагается, что в базе данных есть таблица "Users" с полями "ClientID" и другими).

7. Если текущий пользователь не найден в базе данных, то выходит из метода.

8. Создает новый объект типа "TestDrive" с заполнением его полей:

- идентификатор выбранного автомобиля ("CarID"),
- идентификатор текущего пользователя ("ClientID"),
- цена тест-драйва с учетом скидки ("Price"),
- выбранная дата тест-драйва ("Data").
- Записывает созданный объект "TestDrive" в базу данных.

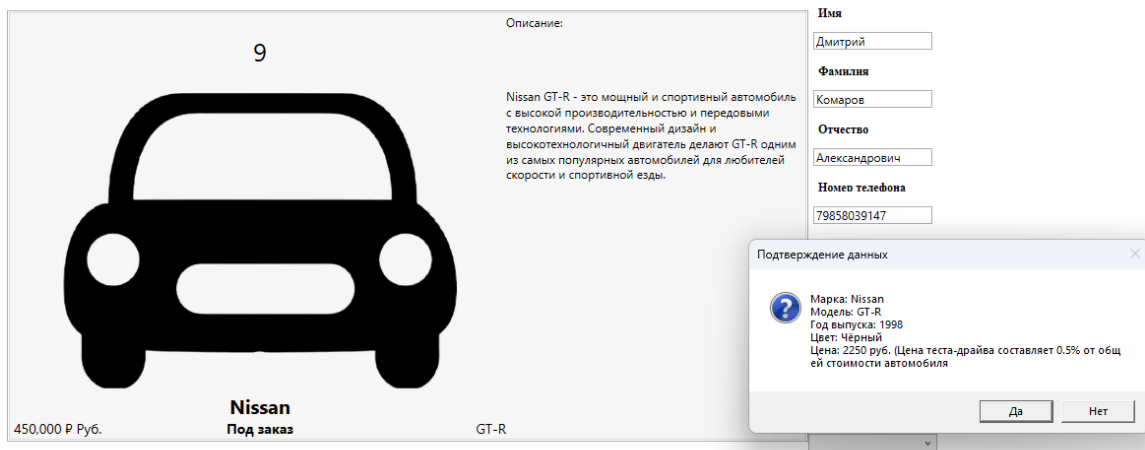



Рисунок 36. Подтверждение данных



Корзина

№	Дата	Цена
2	6/26/2023	7,500 Р.руб.
8		2,500 Р.руб.
8	6/27/2023	2,500 Р.руб.
9		2,250 Р.руб.
9	6/28/2023	2,250 Р.руб.

Заказать тест драйв

Рисунок 37. Корзина клиента

После подтверждения данных, срабатывает следующий код:

```
AutoSphereEntities.GetContext().TestDrives.Add(newTestDrive);
AutoSphereEntities.GetContext().SaveChanges();
Cart.ItemsSource = AutoSphereEntities.GetContext().Test-
Drives.Where(td => td.ClientID ==
```

В DataGridView происходит вывод данных о корзине конкретного клиента, номер машины, цена тест драйва, дата тест драйва.

Заключение

В ходе прохождения производственной (преддипломной) практической подготовки в ООО Авто-Сфера мною было выполнено индивидуальное задание по производственной практике и достигнута цель по приобретению знаний, умений и навыков по ПМ. 01 Разработка модулей программного обеспечения для компьютерных систем, ПМ. 02 Осуществление интеграции программных модулей:

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.

ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования

Мною освоены следующие общие компетенции:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей.

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке.

ОК 11. Планировать предпринимательскую деятельность в профессиональной сфере.

Список литературы

1. "Программирование на C# 7 для профессионалов" А. Троелсен, Ф. Джепикс. - издание: 2018.
2. "WPF 4.5 с примерами на C# 5.0" А. Петзольд. - издание: 2014.
3. "WPF в действии. Пер. с англ." Джон Госсман, Адам Нейро, Джон Стевенс. - издание: 2011.
4. "SQL Server 2017. Руководство для начинающих" Уильям Р. Станек. - издание: 2018.
5. "Программирование баз данных SQL Server 2017. Библиотека программиста" Джейсон Прайс. - издание: 2018.
6. "Microsoft SQL Server 2016. Полное руководство" Брайан Ларсон. - издание: 2016.
7. "Самоучитель SQL Server 2012 Express. Полное руководство для начинающих" Эндрю Брэйнд. - издание: 2013.
8. "Самоучитель Microsoft SQL Server 2014" Кевин Ксир, Питер Лернер, Скотт Шоуп. - издание: 2015.
9. "SQL Server 2017. Практическое руководство для начинающих" Конрад Кинг, Дэниел Киркнесс. - издание: 2018.
10. "Учимся работать с Microsoft SQL Server 2012" Брайан Ларсон. - издание: 2013.