**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский политехнический техникум»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода

информационных систем

Разработка информационной системы

«Автосалон»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Жуков А.Н./

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc133332330)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133332331)

[1.1 Платформа .NET 4](#_Toc133332332)

[1.2 Язык программирования С# 6](#_Toc133332333)

[1.3 Windows Presentation Foundation (WPF) 8](#_Toc133332334)

[1.4 СУБД SQL server 11](#_Toc133332335)

[1.5 Microsoft SQL Server Management Studio 11](#_Toc133332336)

[1.6 Entity Framework 13](#_Toc133332337)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 15](#_Toc133332338)

[2.1 Разработка диаграммы ERD 15](#_Toc133332339)

[2.2 Разработка базы данных 16](#_Toc133332340)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 18](#_Toc133332341)

[3.1 Разработка интерфейса информационной системы 18](#_Toc133332342)

[3.2 Программирование информационной системы 23](#_Toc133332343)

[ЗАКЛЮЧЕНИЕ 30](#_Toc133332344)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 31](#_Toc133332345)

# ВВЕДЕНИЕ

Одной из тенденций современного автобизнеса является создание и развитие сети автосалонов. Очень важным в таких организациях становится вопрос консолидации данных из филиалов и структурных подразделений сети в единую базу данных. Для руководства компании крайне важно иметь объективную информацию о ситуации в филиалах в режиме реального времени. В тоже время для работников автосалонов нужна информация не только о своем складе, но и обо всех имеющихся в наличии автомобилях на других складах компании.

Следовательно, создание распределенной информационной системы. Развитие автобизнеса во всем мире на современном этапе зачастую представляется практически невозможным без внедрения систем автоматизации.

**Объект** - информационная система «Автосалон»

**Предмет** – автоматизация бизнес-процесса учёта автомобилей системы «Автосалон»

**Цель** - создание автоматизированной информационной системы «Автосалон»

**Задачи:**

* выбрать инструментарий к информационной системе «Автосалон»;
* спроектировать базу данных к информационной системе «Автосалон»;
* разработать информационную систему «Автосалон»;

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# Платформа .NET

.NET Framework — программная платформа, выпущенная компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в [2002 году](https://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4). Основой платформы является общеязыковая среда исполнения [Common Language Runtime (CLR)](https://ru.wikipedia.org/wiki/Common_Language_Runtime), которая подходит для различных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду. В настоящее время .NET Framework развивается в виде [.NET](https://ru.wikipedia.org/wiki/.NET).

Считается, что платформа .NET Framework является ответом компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) на набравшую к тому времени большую популярность платформу [Java](https://ru.wikipedia.org/wiki/Java_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0)) компании [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) (ныне принадлежит [Oracle](https://ru.wikipedia.org/wiki/Oracle)).

Хотя .NET Framework является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства [Windows](https://ru.wikipedia.org/wiki/Windows), существуют независимые проекты (прежде всего это [Mono](https://ru.wikipedia.org/wiki/Mono) и [Portable.NET](https://ru.wikipedia.org/wiki/Portable.NET)), позволяющие запускать программы .NET Framework на некоторых других операционных системах.

Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчика за счёт предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах.

Вторым принципом стала ориентация на системы, работающие под управлением семейства операционных систем Microsoft Windows.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код [Common Intermediate Language (CIL)](https://ru.wikipedia.org/wiki/Common_Intermediate_Language) (ранее назывался Microsoft Intermediate Language, MSIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной [Common Language Runtime (CLR)](https://ru.wikipedia.org/wiki/Common_Language_Runtime), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR встроенный в неё [JIT-компилятор](https://ru.wikipedia.org/wiki/JIT) «на лету» (just in time) преобразует промежуточный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и [системе исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), избавляя разработчика от части работы.

Архитектура .NET Framework описана и опубликована в спецификации [Common Language Infrastructure (CLI)](https://ru.wikipedia.org/wiki/Common_Language_Infrastructure), разработанной Microsoft и утверждённой [ISO](https://ru.wikipedia.org/wiki/ISO) и [ECMA](https://ru.wikipedia.org/wiki/ECMA). В CLI описаны типы данных .NET, формат метаданных о структуре программы, система исполнения [байт-кода](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) и многое другое.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке [Framework Class Library (FCL)](https://ru.wikipedia.org/wiki/Framework_Class_Library). В FCL входят классы [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), [ADO.NET](https://ru.wikipedia.org/wiki/ADO.NET), [ASP.NET](https://ru.wikipedia.org/wiki/ASP.NET), [Language Integrated Query](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [Windows Presentation Foundation](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Communication Foundation](https://ru.wikipedia.org/wiki/Windows_Communication_Foundation) и другие. Ядро FCL называется [Base Class Library (BCL)](https://ru.wikipedia.org/wiki/Base_Class_Library).

Среды разработки, поддерживающие .NET:

[Microsoft Visual Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) (C#, Visual Basic .NET, Managed C++, F#)

[SharpDevelop](https://ru.wikipedia.org/wiki/SharpDevelop)

[MonoDevelop](https://ru.wikipedia.org/wiki/MonoDevelop)

[Embarcadero RAD Studio](https://ru.wikipedia.org/wiki/Embarcadero_RAD_Studio) (Delphi for .NET); ранее [Borland Developer Studio](https://ru.wikipedia.org/wiki/Borland_Developer_Studio) (Delphi for .NET, C#)

[A#](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))

[Zonnon](https://ru.wikipedia.org/wiki/Zonnon)

[PascalABC.NET](https://ru.wikipedia.org/wiki/PascalABC.NET)

[JetBrains Rider](https://ru.wikipedia.org/wiki/JetBrains_Rider)

Приложения .NET также можно разрабатывать в текстовом редакторе, просто вызывая [компилятор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) из [командной строки](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0).

Одной из основных идей Microsoft .NET является совместимость программных частей, написанных на разных языках. Например, служба, написанная на C++ для Microsoft .NET, может обратиться к методу класса из библиотеки, написанной на Delphi; на C# можно написать класс, наследованный от класса, написанного на Visual Basic .NET, а исключение, созданное методом, написанным на C#, может быть перехвачено и обработано в Delphi. Каждая библиотека (сборка) в .NET имеет сведения о своей версии, что позволяет устранить возможные конфликты между разными версиями сборок.

Языки, поставляемые вместе с [Microsoft Visual Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio):

[C#](https://ru.wikipedia.org/wiki/C_Sharp)

[Visual Basic .NET](https://ru.wikipedia.org/wiki/Visual_Basic_.NET)

[JScript .NET](https://ru.wikipedia.org/wiki/JScript_.NET)

[C++/CLI](https://ru.wikipedia.org/wiki/C%2B%2B/CLI) — новая версия [Managed C++](https://ru.wikipedia.org/wiki/Managed_C_Plus_Plus)

[F#](https://ru.wikipedia.org/wiki/F_Sharp) — член семейства языков программирования [ML](https://ru.wikipedia.org/wiki/ML), включён в VS2010/VS2012/ VS2015/VS2017/VS2019/VS2022

[J#](https://ru.wikipedia.org/wiki/Visual_J_Sharp) — последний раз был включён в VS2005

# Язык программирования С#

C# — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

С#‎ разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов С#‎, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет С#‎, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#‎, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др.

**Версия 8.0**

Модификатор readonly. Был создан для обозначения члена, который не изменит состояние.

Методы интерфейсов по умолчанию. Теперь при создании метода интерфейса можно объявить его реализацию по умолчанию, которую можно переопределить в классе, который реализует этот интерфейс.

Сопоставление шаблонов. Возможность позволяет работать с шаблонами в зависимости от формата в связанных, но различных типах данных.

Рекурсивные шаблоны. Является выражением шаблона, которое применяется к результатам другого выражения шаблона.

Выражения switch позволяют сократить количество case и break, а также фигурных скобок.

Шаблоны свойств. Позволяет сопоставлять свойства исследуемого объекта с помощью { variable : value } => ... .

Шаблоны кортежей. Используется, если нужно работать с несколькими наборами входных данных. (value1, value2,..) => ...

Объявление using. Это объявление переменной, которому предшествует ключевое слово using. Оно сообщает компилятору, что объявляемая переменная должна быть удалена в конце [области видимости](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C_%D0%B2%D0%B8%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8).

Статический локальный метод. Теперь можно убедиться в том, что метод не охватывает какие-либо переменные из области видимости с помощью добавления к нему модификатора static.

Удаляемые ссылочные структуры. Ссылочные структуры не могут реализовать IDisposable (как и любые другие интерфейсы). Поэтому чтобы удалить ref struct, необходим доступный void Dispose().

Типы значений, допускающие значение null. Теперь, чтобы указать, что [переменная типа](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B0) значений допускает значение null, необходимо поставить к имени типа ?

Асинхронные потоки. Это, во-первых, интерфейс IAsyncEnumerable<T>. А во-вторых конструкция foreach с await.

Асинхронные высвобождаемые типы. Начиная с C# 8.0 язык поддерживает асинхронные освобождаемые типы, реализующие интерфейс System.IAsyncDisposable. Операнд выражения using может реализовывать IDisposable или IAsyncDisposable. В случае IAsyncDisposable компилятор создает код для await, возвращенного Task из IAsyncDisposable.DisposeAsync.

Индексы и диапазоны. Диапазоны и индексы обеспечивают лаконичный синтаксис для доступа к отдельным элементам или диапазонам в последовательности. Нововведение включает в себя операторы ^ и .. , а также System.Index и System.Range

Оператор присваивания объединения с null. Оператор ??= можно использовать для присваивания значения правого операнда левому операнду только в том случае, если левый операнд принимает значение null.

Неуправляемые сконструированные типы. Начиная с C# 8.0, сконструированный тип значения является неуправляемым, если он содержит поля исключительно неуправляемых типов (например, универсальный тип <T>).

Выражение stackalloc во вложенных выражениях. Теперь если результат выражения stackalloc имеет тип System.Span<T> или System.ReadOnlySpan<T>, то его можно использовать в других выражениях.

# Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) — аналог WinForms, система для построения клиентских приложений [Windows](https://ru.wikipedia.org/wiki/Windows) с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) (начиная с версии [3.0](https://ru.wikipedia.org/wiki/.NET_Framework_3.0)), использующая язык [XAML](https://ru.wikipedia.org/wiki/XAML).

WPF предустановлена в [Windows Vista](https://ru.wikipedia.org/wiki/Windows_Vista) ([.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) 3.0), [Windows 7](https://ru.wikipedia.org/wiki/Windows_7) ([.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) 3.5 SP1), [Windows 8](https://ru.wikipedia.org/wiki/Windows_8) (.NET Framework 4.0 и 4.5), [Windows 8.1](https://ru.wikipedia.org/wiki/Windows_8.1) (.NET Framework 4.5.1) и [Windows 10](https://ru.wikipedia.org/wiki/Windows_10) (.NET Framework 4.7). С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в [браузере](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) приложений.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык [XAML](https://ru.wikipedia.org/wiki/XAML) (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является [DirectX](https://ru.wikipedia.org/wiki/DirectX), в отличие от [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), где используется [GDI](https://ru.wikipedia.org/wiki/GDI)/[GDI+](https://ru.wikipedia.org/wiki/GDI%2B). Производительность [WPF](https://ru.wikipedia.org/wiki/WPF) выше, чем у [GDI+](https://ru.wikipedia.org/wiki/GDI%2B) за счёт использования аппаратного ускорения графики через [DirectX](https://ru.wikipedia.org/wiki/DirectX).

Также существует урезанная версия [CLR](https://ru.wikipedia.org/wiki/CLR), называющаяся [WPF/E](https://ru.wikipedia.org/wiki/WPF/E), она же известна как [Silverlight](https://ru.wikipedia.org/wiki/Silverlight).

[XAML](https://ru.wikipedia.org/wiki/XAML) представляет собой язык декларативного описания интерфейса, основанный на [XML](https://ru.wikipedia.org/wiki/XML). Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на [элементы управления](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента, например, [Button](https://ru.wikipedia.org/wiki/%D0%9A%D0%BD%D0%BE%D0%BF%D0%BA%D0%B0_(%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC)) (кнопка).

Для работы с WPF требуется любой [.NET](https://ru.wikipedia.org/wiki/.NET_Framework)-совместимый язык. В этот список входит множество языков: [C#](https://ru.wikipedia.org/wiki/C_Sharp), [F#](https://ru.wikipedia.org/wiki/F_Sharp), [VB.NET](https://ru.wikipedia.org/wiki/VB.NET), [C++](https://ru.wikipedia.org/wiki/Visual_C%2B%2B), [Ruby](https://ru.wikipedia.org/wiki/IronRuby), [Python](https://ru.wikipedia.org/wiki/IronPython), [Delphi (Prism)](https://ru.wikipedia.org/wiki/Oxygene_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [Lua](https://ru.wikipedia.org/w/index.php?title=Lua.NET&action=edit&redlink=1) и многие другие. Для полноценной работы может быть использована как [Visual Studio](https://ru.wikipedia.org/wiki/Visual_Studio), так и [Expression Blend](https://ru.wikipedia.org/wiki/Expression_Blend). Первая ориентирована на программирование, а вторая — на дизайн и позволяет делать многие вещи, не прибегая к ручному редактированию [XAML](https://ru.wikipedia.org/wiki/XAML). Примеры этому — анимация, стилизация, состояния, создание элементов управления и так далее.

WPF предоставляет широкий спектр возможностей по созданию интерактивных настольных приложений:

**Привязка данных**

Это гибкий механизм, который позволяет через расширения разметки XAML связывать различные данные (от значений свойств элементов управления до общедоступных свойств, реализующих поля базы данных через [Entity Framework](https://ru.wikipedia.org/wiki/Entity_Framework)). Привязка данных представлена классом [Binding](http://msdn.microsoft.com/ru-ru/library/system.windows.data.binding.aspx), который в свою очередь унаследован от [MarkupExtension](http://msdn.microsoft.com/ru-ru/library/system.windows.markup.markupextension.aspx), что позволяет использовать привязки не только в коде, но и в разметке:

Помимо основного класса Binding в WPF реализовано еще несколько механизмов привязок:

[MultiBinding](http://msdn.microsoft.com/ru-ru/library/system.windows.data.multibinding.aspx) — позволяет создавать множественные привязки, указывая несколько элементов

[TemplateBinding](http://msdn.microsoft.com/ru-ru/library/system.windows.templatebindingextension.aspx) — используется в шаблонах для связывания свойства элемента внутри шаблона со свойством элемента, к которому применен шаблон

[PriorityBinding](http://msdn.microsoft.com/ru-ru/library/system.windows.data.prioritybinding.aspx) — ранжирует список привязок и выбирает из них свойство (согласно приоритету) к которому будет применена привязка. Если привязка, имеющая наивысший приоритет успешно возвращает значение, то нет необходимости обрабатывать другие привязки в списке.

**Стили**

Позволяют создавать стилевое оформление элементов и, как правило, используются только в разметке:

Если стиль задается в ресурсах (например в словаре ресурсов), то можно использовать атрибут x:Key для указания уникального ключа. Затем в элементе управления, к которому необходимо применить стиль, нужно использовать расширение разметки StaticResource для связывания с этим стилем. Если использовать этот прием, то стили не будут нагромождать разметку.

**Шаблоны данных**

В отличие от шаблонов элементов управления, задаются для определенного контекста данных (который в блочных элементах управления задается через свойство [DataContext](http://msdn.microsoft.com/en-us/library/system.windows.frameworkelement.datacontext.aspx), а в списковых через [ItemsSource](http://msdn.microsoft.com/ru-ru/library/system.windows.controls.itemscontrol.itemssource.aspx)). Сам шаблон данных представлен классом [DataTemplate](http://msdn.microsoft.com/ru-ru/library/system.windows.datatemplate.aspx). Для обозначения типа данных, к которому необходимо применить шаблон, используется свойство DataType.

**Ресурсы**

Система ресурсов позволяет объединять шаблоны, стили, кисти, анимацию и многие другие интерактивные элементы, что существенно упрощает работу с ними. Ресурсы задаются в свойстве Resources класса FrameworkElement, от которого унаследованы все элементы управления, панели компоновки и даже класс Application. Это позволяет создавать многоуровневую систему ресурсов:

ресурсы внутри объекта — действительны только для этого объекта

ресурсы внутри панели компоновки (например Grid) — позволяет задать границу контекста ресурсов на уровне этой панели

ресурсы внутри окна Window — если в приложении используется несколько окон, то ресурсы одного окна не будут доступны ресурсам другого окна

ресурсы приложения — доступны повсеместно (как правило задаются в отдельном словаре ресурсов)

**Графика**

WPF представляет обширный, масштабируемый и гибкий набор графических возможностей:

Графика, не зависящая от разрешения и устройства. Основной единицей измерения в графической системе WPF является аппаратно-независимый пиксель, который составляет 1/96 часть дюйма независимо от фактического разрешения экрана.

Дополнительная поддержка графики и анимации. WPF упрощает программирование графики за счет автоматического управления анимацией. Разработчик не должен заниматься обработкой сцен анимации, циклами отрисовки и билинейной интерполяцией

Аппаратное ускорение. Графическая система WPF использует преимущества графического оборудования, чтобы уменьшить использование ЦП.

**Двухмерная графика**

WPF предоставляет библиотеку общих двухмерных фигур, нарисованных с помощью векторов, таких, как прямоугольники и эллипсы, а также графические пути. И в своей функциональности фигуры реализуют многие возможности, которые доступны обычным элементам управления.

Двухмерная графика в WPF включает визуальные эффекты, такие как градиенты, точечные рисунки, чертежи, рисунки с видео, поворот, масштабирование и наклон.

**Трехмерная графика**

WPF также включает возможности трехмерной отрисовки, интегрированные с двухмерной графикой, что позволяет создавать более яркий и интересный пользовательский интерфейс.

# СУБД SQL server

Microsoft SQL Server — [система управления реляционными базами данных (РСУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

В зависимости от задачи, которую выполняет команда T-SQL, он может принадлежать к одному из следующих типов:

DDL (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

В частности, к этому типу мы можем отнести следующие команды:

* CREATE: создает объекты базы данных (саму базу даных, таблицы, индексы и т.д.)
* ALTER: изменяет объекты базы данных
* DROP: удаляет объекты базы данных
* TRUNCATE: удаляет все данные из таблиц

DCL (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным. В частности, это следующие команды:

* GRANT: предоставляет права для доступа к данным
* REVOKE: отзывает права на доступ к данным

# Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа, настройки, администрирования и разработки всех компонентов SQL Server, Базы данных SQL Azure, Управляемого экземпляра SQL Azure, SQL Server на виртуальной машине Azure и Azure Synapse Analytics. Среда SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе обширную группу графических инструментов с рядом многофункциональных редакторов скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех профессиональных уровней.

Среда SQL Server Management Studio предназначена для доступа к службам Службы Analysis Services, Integration Services и Reporting Services, а также для их настройки, администрирования и управления ими. Хотя все три технологии бизнес-аналитики полагаются на среду SQL Server Management Studio, административные задачи, связанные с каждой из этих технологий, несколько отличаются.

Среда SQL Server Management Studio позволяет управлять объектами Службы Analysis Services, например выполнять их резервное копирование и обработку.

Среда Management Studio позволяет создавать проекты скриптов служб Службы Analysis Services, в которых выполняются разработка и сохранение скриптов с использованием многомерных выражений (MDX), расширений интеллектуального анализа данных (DMX) и XML для аналитики (XMLA). Проекты скриптов служб Службы Analysis Services используются для выполнения задач управления или повторного создания баз данных, кубов и других объектов в экземплярах служб Службы Analysis Services. Например, можно разработать скрипт XMLA в проекте скрипта служб Службы Analysis Services, который создает объекты непосредственно в существующем экземпляре служб Службы Analysis Services. Проекты скриптов служб Службы Analysis Services могут быть сохранены в составе решения и интегрироваться с контролем исходного кода.

**Управление решениями Integration Services с помощью SQL Server Management Studio**

Среда SQL Server Management Studio позволяет использовать службу Integration Services для управления пакетами и наблюдения за выполняющимися пакетами. В среде Среда Management Studio можно организовать пакеты в папки, выполнять, импортировать и экспортировать пакеты, переносить пакеты служб DTS и обновлять пакеты служб Integration Services.

**Управление проектами Reporting Services с помощью SQL Server Management Studio**

Среда SQL Server Management Studio позволяет включать компоненты служб Reporting Services, администрировать серверы и базы данных, управлять ролями и заданиями.

Она реализует функции управления общими расписаниями (в папке «Общие расписания») и базами данных сервера отчетов (ReportServer, ReportServerTempdb). Можно также создать роль RSExecRole в системной базе данных Master, когда база данных сервера отчетов перемещается в новое или другое ядро СУБД SQL Server.

# Entity Framework

Entity Framework — это решение для работы с базами данных, которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Также код с использованием EF пишется гораздо быстрее.

Например, работая с базами данных напрямую, разработчик должен беспокоиться о подключении, подготовке SQL и параметров, отправке запросов и транзакций. На Entity Framework всё это делается автоматически — программист же работает непосредственно с сущностями и только говорит EF, что нужно сохранить изменения.

В этой статье будут разобраны основы применения Entity Framework, для понимания которых нужно владеть ADO.NET — пользоваться базами данных, писать SQL-запросы и работать с подключениями.

Entity Framework позволяет значительно сократить код работы с базами данных. При этом он предоставляет большие возможности. Например, можно использовать:

* foreign keys;
* связи one-to-one, one-to-many и many-to-many;
* параметризацию запросов;
* хранимые процедуры.

Однако стоит учитывать, что EF выступает прослойкой между приложением и базой данных, поэтому может ухудшаться производительность. Для небольших проектов это допустимо, но если программа должна работать под большой нагрузкой, то лучше использовать чистый ADO.NET.

**Предоставление жизненного цикла моделям**

Многолетним и общим подходом к разработке является подход, при котором построение приложения или службы представляет собой его разделение на три части: модель домена, логическую модель и физическую модель. Модель домена определяет сущности и связи в моделируемой системе. Логическая модель для реляционной базы данных обеспечивает нормализацию сущностей и связей в целях создания таблиц с ограничениями внешнего ключа. В физической модели учитываются возможности конкретной системы обработки данных путем определения, зависящих от ядра базы данных подробных сведений о хранении данных, которые касаются секционирования и индексирование.

Физическая модель совершенствуется администраторами базы данных в целях повышения производительности, но программисты, которые разрабатывают код приложения, в основном вынуждены ограничиваться работой с логической моделью, подготавливая SQL-запросы и вызывая хранимые процедуры. Модели домена в основном используются как инструмент для представления и обмена мнениями о требованиях к приложению, поэтому чаще всего служат в качестве практически не изменяющихся схем, которые рассматриваются и обсуждаются на ранних стадиях проекта, после чего выходят из сферы внимания. Во многих коллективах разработчиков принято пропускать этап создания концептуальной модели и начинать с определения таблиц, столбцов и ключей в реляционной базе данных.

Entity Framework дает жизнь модели, позволяя разработчикам запрашивать сущности и связи в модели предметной области (называемой концептуальной моделью в Entity Framework), в то же время полагаться на Entity Framework преобразования этих операций в команды, относящиеся к источнику данных. Это позволяет отказаться от применения в приложениях жестко заданных зависимостей от конкретного источника данных.

При работе в режиме Code First концептуальная модель сопоставлена с режимом хранения в коде. Entity Framework может вычислять концептуальную модель на основе типов объектов и дополнительных конфигураций, которые вы определяете. Метаданные сопоставления формируются во время выполнения на основе сочетания определений типов домена и дополнительной информации о конфигурации, которая указана в коде. Entity Framework создает базу данных по мере необходимости на основе метаданных. Дополнительные сведения см. в разделе Создание модели.

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2.1 Разработка диаграммы ERD

Данная диаграмма — (ER-модель данных) обеспечивает стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Диаграммы «сущность— связь» в отличие от функциональных диаграмм определяют спецификации структур данных программного обеспечения.

Базовыми понятиями ER-модели данных (ER — Entity— Relationship) являются сущность, атрибут и связь. Сущность — это класс однотипных реальных или абстрактных объектов (людей, событий, состояний, предметов и т.п.), информация о которых имеет существенное значение для рассматриваемой предметной области.

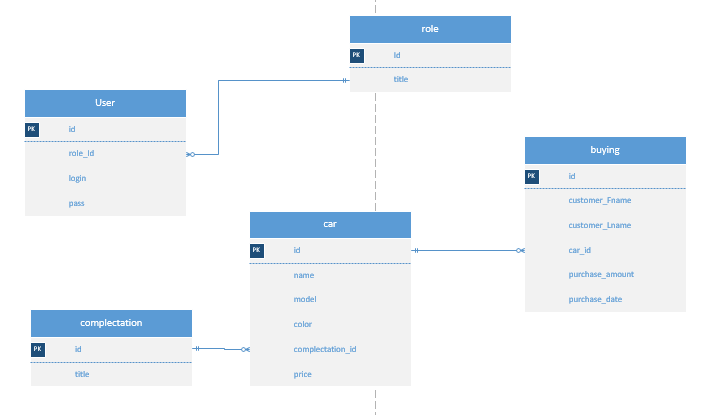
Атрибут — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут, таким образом, представляет собой некоторый тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов.

Атрибуты делятся на ключевые, т. е. входящие в состав уникального идентификатора ключа, и описательные — прочие.

Первичный ключ — это атрибут или совокупность атрибутов и связей, предназначенная для уникальной идентификации каждого экземпляра сущности.

Описательные атрибуты могут быть обязательными или необязательными. Обязательные атрибуты для каждой сущности всегда имеют конкретное значение, необязательные могут быть не определены.

Связь — это отношение одной сущности к другой или к самой себе. Если любой экземпляр одной сущности связан хотя бы с одним экземпляром другой сущности, то связь является обязательной. Необязательная связь представляет собой условное отношение между сущностями. Каждая сущность может обладать любым количеством связей с другими сущностями модели. Различают три типа отношений «один-к-одному»; «один-ко-многим»; «многие-ко-многим».

Рис. 2.1 ER Диаграмма

Основными сущностями для решения указанной задачи являются: user, buying. Сразу возникает очевидная связь между сущностями user и role для авторизации.

Проанализируем атрибуты сущностей. buiing может иметь имя и фамилию покупателя, информация о машине, сумма покупки и дата покупки, у каждой сущности есть ID. Каждый новый отчёт имеет номер.

# 2.2 Разработка базы данных

Создание базы данных.

В обозревателе объектовподключитесь к экземпляру компонента Компонент SQL Server Database Engine и разверните его.

Щелкните правой кнопкой мыши узел Базы данных и выберите команду Создать базу данных.

В поле Новая база данныхвведите имя базы данных.

Чтобы создать базу данных, приняв все значения по умолчанию, нажмите кнопку ОК; в противном случае продолжайте выполнять указанные ниже дополнительные действия.

Чтобы изменить имя владельца, нажмите ( … ) и выберите другого владельца.

Чтобы изменить значения первичных данных по умолчанию и файлы журнала транзакций, выберите соответствующую ячейку в сетке Файлы базы данных и введите новое значение. Дополнительные сведения см. в статье AДобавление файлов данных или журналов в базу данных.

Чтобы изменить параметры сортировки базы данных, выберите страницу Параметры и выберите из списка желаемые параметры сортировки.

Чтобы изменить модель восстановления, выберите страницу Параметры и модель восстановления из списка.

Чтобы изменить параметры базы данных, выберите страницу Параметры и измените параметры базы данных. Описание каждого параметра см. в разделе Параметры ALTER DATABASE SET (Transact-SQL).

Чтобы добавить новую файловую группу, перейдите на страницу Файловые группы. Нажмите кнопку Добавить и введите значения для файловой группы.

Чтобы добавить расширенное свойство в базу данных, выберите страницу Расширенные свойства.

В столбце Имя введите имя расширенного свойства.

В столбце Значение введите текст расширенного свойства. Например, введите одно или несколько предложений, которые описывают базу данных.

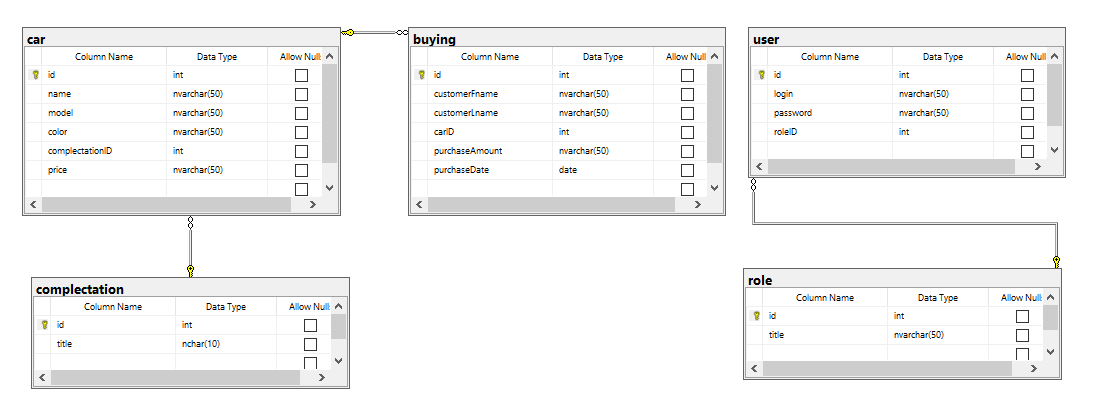
Чтобы создать базу данных, нажмите кнопку ОК.

Рис. 2.2 База данных приложения

Здесь 5 таблиц, есть user и role – они отвечают за авторизацию в приложении, car – это информация о машинах, complectation – вспомогательный для информации о комплектации автомобиля, buying – информация о покупках.

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка интерфейса информационной системы

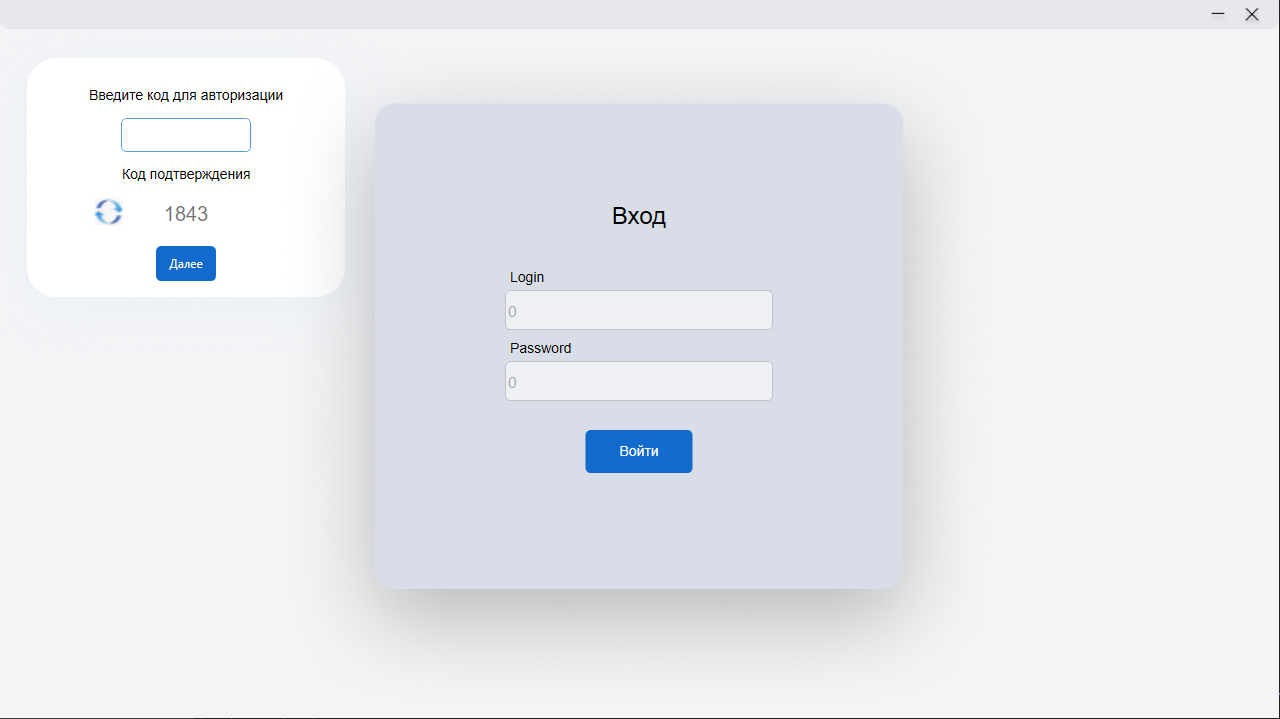
На рисунке 3.1 представлено окно входа в систему и ввода кода авторизации. На этом окне пока вы не ввели логин который есть в базе данных поле с паролем не откроется, полсе этого появляется окошко с вводом кода который генерируется рандомно каждые 10 секунд.

Рис. 3.1 Окно входа в систему

<Grid Height="720" Width="1280" VerticalAlignment="Top" HorizontalAlignment="Center" Background="#FFF3F4F6">

Скрытый Grid с окном ввода кода.

<Grid Visibility="Hidden" x:Name="Auth\_Win\_1">

<Image Margin="-32,10,874,351" Source="/car\_rect.png" Stretch="Fill"/>

<TextBlock HorizontalAlignment="Left" Margin="123,168,0,0" TextWrapping="Wrap" VerticalAlignment="Top" FontFamily="Arial" FontSize="14"><Run Text="Код подт"/><Run Text="верждения"/></TextBlock>

<TextBlock HorizontalAlignment="Left" Margin="90,89,0,0" TextWrapping="Wrap" VerticalAlignment="Top" FontFamily="Arial" FontSize="14"><Run Language="ru-ru" Text="Введите код для авторизации"/></TextBlock>

<TextBox MaxLength="4" Name="TXB2" HorizontalAlignment="Left" Margin="122,120,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="130" Background="White" Foreground="#FF7B7B7B" VerticalContentAlignment="Center" Height="34" FontFamily="Arial" FontSize="14"/>

<Button Content="Далее" HorizontalAlignment="Left" Margin="157,248,0,0" VerticalAlignment="Top" Background="#FF116ACC" Foreground="White" Height="35" Width="60" Click="Next\_Btn"/>

<TextBox HorizontalContentAlignment="Center" IsEnabled="False" Name="TXB1" HorizontalAlignment="Left" Margin="103,189,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="168" Background="White" Foreground="Black" VerticalContentAlignment="Center" Height="54" FontSize="20" FontFamily="Arial" BorderBrush="White"/>

<Button Click="Reboot\_Btn\_Click" HorizontalAlignment="Left" Margin="90,194,0,0" VerticalAlignment="Top" Height="34" Width="36">

<Button.Background>

<ImageBrush ImageSource="/reboot.png"/>

</Button.Background>

</Button>

</Grid>

Основной код для окна авторизации.

<Image Margin="0,0,0,689" Source="/closepanel.png" Stretch="Fill"/>

<Button Content="" HorizontalAlignment="Left" Margin="1242,10,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Width="22" Height="21" Click="Button\_Click">

<Button.Background>

<ImageBrush ImageSource="/close\_button.png"/>

</Button.Background>

</Button>

<Image Margin="276,36,276,0" Source="/authrest.png" Stretch="Fill"/>

<TextBlock HorizontalAlignment="Center" Margin="0,204,0,0" TextWrapping="Wrap" Text="Вход" VerticalAlignment="Top" FontSize="24" FontFamily="Arial"/>

<Button Name="Auth\_Btn" Content="Войти" HorizontalAlignment="Center" Margin="0,432,0,0" VerticalAlignment="Top" Height="43" Width="107" BorderBrush="{x:Null}" Foreground="White" Background="#FF116ACC" Click="Auth\_Btn\_Click" FontSize="14" FontFamily="Arial"/>

<TextBox VerticalContentAlignment="Center" HorizontalAlignment="Center" Margin="0,363,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="268" Height="40" FontSize="16" Background="White" Foreground="#FF7B7B7B" Name="TxbPass"/>

<Button Content="" HorizontalAlignment="Left" Margin="1208,10,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Width="22" Height="20" Click="Button\_Click\_2">

<Button.Background>

<ImageBrush ImageSource="/rollup\_button.png" Stretch="Uniform"/>

</Button.Background>

</Button>

<Label Content="Login" HorizontalAlignment="Left" Margin="506,266,0,0" VerticalAlignment="Top" FontFamily="Arial" FontSize="14"/>

<Label Content="Password" HorizontalAlignment="Left" Margin="506,337,0,0" VerticalAlignment="Top" FontFamily="Arial" FontSize="14"/>

</Grid>

</Window>

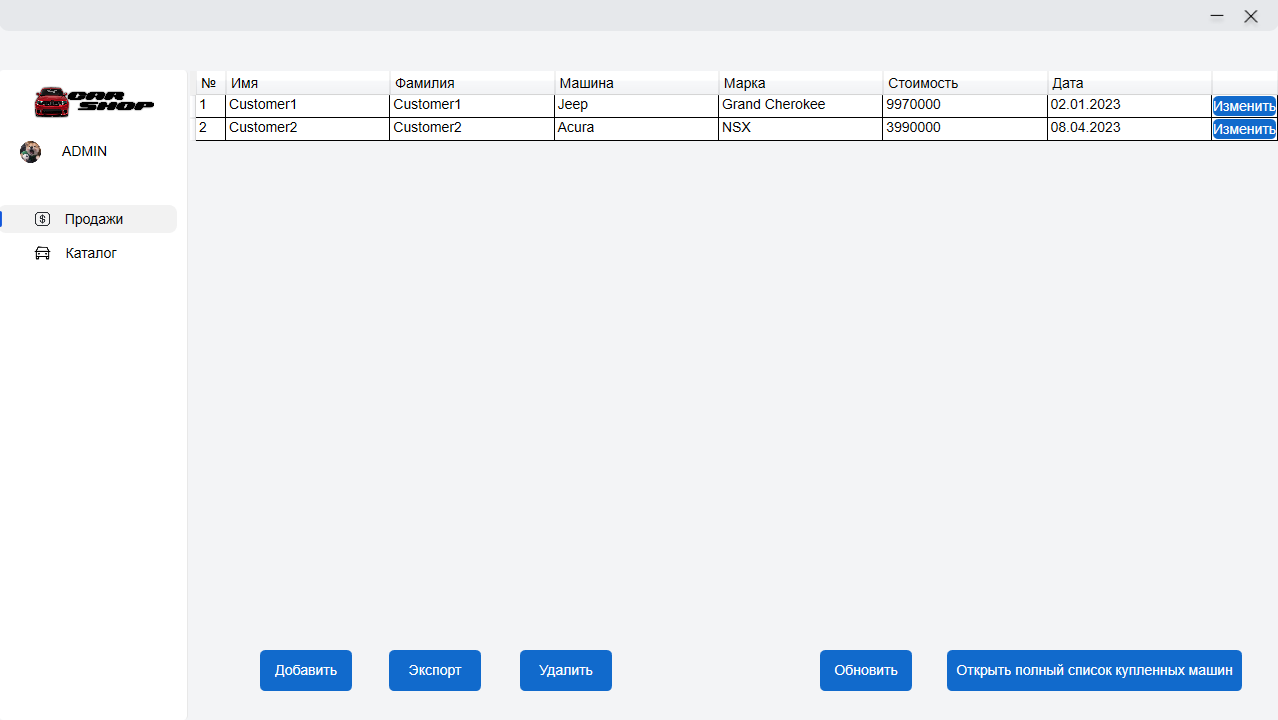
На рисунке 3.2 предаставлено окно с выводом покупок. В этом окне пользователь может добавить инфрмацию о совершённых покупках, отредактировать существующие данные, администратор может удалить данные из таблицы.

Рис. 3.2 Окно продаж

<Grid Height="720" Width="1280" VerticalAlignment="Center" HorizontalAlignment="Center" Background="#FFF3F4F6">

Разметка окна.

<Grid.RowDefinitions>

<RowDefinition Height="7\*"/>

<RowDefinition Height="65\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="19\*"/>

<ColumnDefinition Width="109\*"/>

</Grid.ColumnDefinitions>

Таблица DataGrid для вывода данных из базы данных.

<DataGrid x:Name="Sells\_DataGrid" Grid.Column="1" Grid.Row="1" BorderBrush="{x:Null}" IsReadOnly="True" AutoGenerateColumns="False" Margin="0,0,0,98" Background="#FFF3F4F6" FontFamily="Arial" FontSize="14">

<DataGrid.Columns>

<DataGridTextColumn Header="№" Width="30" Binding="{Binding id}"/>

<DataGridTextColumn Header="Имя" Width="\*" Binding="{Binding customerFname}"/>

<DataGridTextColumn Header="Фамилия" Width="\*" Binding="{Binding customerLname}"/>

<DataGridTextColumn Header="Машина" Width="\*" Binding="{Binding car.name}"/>

<DataGridTextColumn Header="Марка" Width="\*" Binding="{Binding car.model}"/>

<DataGridTextColumn Header="Стоимость" Width="\*" Binding="{Binding purchaseAmount}"/>

<DataGridTextColumn Header="Дата" Width="\*" Binding="{Binding purchaseDate, StringFormat=\{0:dd.MM.yyyy\}}"/>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Click="Edit\_Btn\_Click" Content="Изменить" VerticalAlignment="Top" Height="20" Background="#FF116ACC" Foreground="White"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Window>

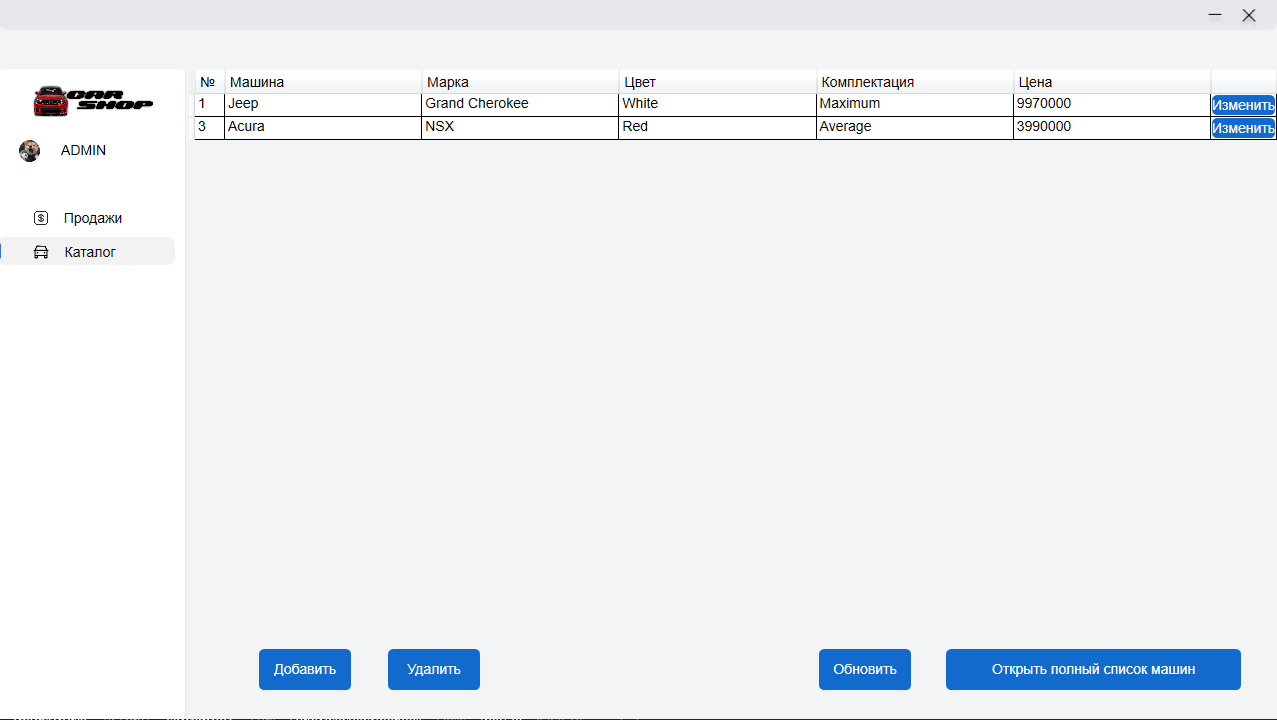
На рисунке 3.3 представлено окно со списком машин, которые есть в базе данных. Здесь пользователь может добавить и отредактировать данные, администратор может удалить данные из таблицы.

Рис. 3.2 Окно каталога

DataGrid для вывода данных машинах из базы данных.

<DataGrid x:Name="Catalog\_DataGrid" IsReadOnly="True" Grid.Column="1" Grid.Row="1" BorderBrush="{x:Null}" AutoGenerateColumns="False" Margin="0,0,0,98" Background="#FFF3F4F6" FontFamily="Arial" FontSize="14">

<DataGrid.Columns>

<DataGridTextColumn Header="№" Width="30" Binding="{Binding id}"/>

<DataGridTextColumn Header="Машина" Width="\*" Binding="{Binding name}"/>

<DataGridTextColumn Header="Марка" Width="\*" Binding="{Binding model}"/>

<DataGridTextColumn Header="Цвет" Width="\*" Binding="{Binding color}"/>

<DataGridTextColumn Header="Комплектация" Width="\*" Binding="{Binding complectation.title}"/>

<DataGridTextColumn Header="Цена" Width="\*" Binding="{Binding price}"/>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Click="Edit\_Btn\_Click" Content="Изменить" VerticalAlignment="Top" Height="20" Background="#FF116ACC" Foreground="White"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

На рисунке 3.4 представлено добавление информации в базу данных. Здесь пользователь вводит нужные данные в TextBox и ComboBox и нажимает на кнопку сохранить. Если пользоватеь не ввёл какие то данные, то высветится ошбка о том что не все данные введены и приложение не закроется.

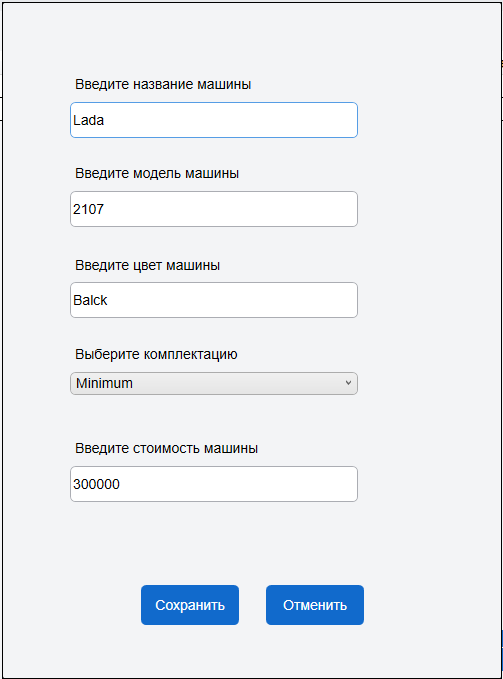


Рис. 3.4 Окно добавления данных

# 3.2 Программирование информационной системы

Код окна авторизации:

Класс проверки логина по нажатию кнопки. Если логин есть в базе данных, то отурывается поле с вводом пароля, а если пользователя нет в базе данных вылезет ошибка.

private void Auth\_Btn\_Click(object sender, RoutedEventArgs e)

{

var CurrentUser = AppData.db.user.FirstOrDefault(u => u.login == TxbLogin.Text); //проверка логина

if (CurrentUser != null) //если правильный логин то открывается поле с паролем

{

TxbPass.IsEnabled = true;

Auth\_Btn.Visibility = Visibility.Hidden;

Auth\_Next\_Btn.Visibility = Visibility.Visible;

TxbPass.Focus();

}

else //если нет то ошибка

{

MessageBox.Show("Такого пользователя не существует!");

}

}

Класс проверки пароля по нажатию кнопки. Если пароль совпадает с данными из таблицы бд, то всплывает окно с вводом кода подтверждения, а если пароль не правильный, то вылезает ошибка и очищаются поля с пролем и логином.

private async void Auth\_Btn\_Click\_2(object sender, RoutedEventArgs e)

{

var CurrentUser1 = AppData.db.user.FirstOrDefault(u => u.login == TxbLogin.Text && u.password == TxbPass.Text); // проверка пароля

if (CurrentUser1 != null)

{

if (Auth\_Win\_1.Visibility == Visibility.Hidden)

Auth\_Win\_1.Visibility = Visibility.Visible;

TxbLogin.IsEnabled = false;

TxbPass.IsEnabled = false;

Auth\_Next\_Btn.IsEnabled = false;

TXB2.Focus();

Globals.UserRoles = CurrentUser1.roleID;

Globals.userinfo = CurrentUser1;

Цикл для генерации кода подтверждения.

while (true) //рандомизация кода и сброс кода каждые 10 секунд

{

Random x = new Random();

int a = x.Next(1000, 9999);

TXB1.Text = a.ToString();

await Task.Delay(10000);

}

}

else //если пароль не верный то ошибка

{

MessageBox.Show("Пароль не верен!");

TxbPass.Clear();

TxbPass.IsEnabled = false;

TxbLogin.Clear();

Auth\_Next\_Btn.Visibility = Visibility.Hidden;

Auth\_Btn.Visibility = Visibility.Visible;

}

}

Класс для кнопки перезагрузки кода подтверждения.

private async void Reboot\_Btn\_Click(object sender, RoutedEventArgs e)

{

while (true)

{

Random x = new Random();

int a = x.Next(1000, 9999);

TXB1.Text = a.ToString();

await Task.Delay(10000);

}

}

private void Next\_Btn(object sender, RoutedEventArgs e)

{

if (TXB2.Text == TXB1.Text) //если код верный то переход на другое окно

{

Window3 ebatb = new Window3();

ebatb.Show();

Close();

}

else //если код не верный то ошибка

{

MessageBox.Show("Код подтверждения не верный!");

TXB2.Clear();

}

}

Реализация разгранечения ролей. В данном куске кода идёт проверка – если вы вошли под администратором то у вас высвечивается в поле с аккаунтом что вы ADMIN, а если вы вошли под обычным пользователем то у вас высветится User и не будет возможности удаления данных.

if (MainWindow.Globals.UserRoles == 1) //разгранечение ролей

{

Del\_Btn.Visibility = Visibility.Visible;

TextBlock.Text = "ADMIN";

}

else

{

Del\_Btn.Visibility = Visibility.Collapsed;

TextBlock.Text = "User";

}

Код окна с покупками.

В данном классе идёт переход на страницу с редактированием данных.

private void Edit\_Btn\_Click(object sender, RoutedEventArgs e)

{

AddPage addPage = new AddPage((sender as Button).DataContext as buying);

addPage.Show();

}

В данном классе реализация удаления данных из таблицы. Перед удалением всплывёт окно с подтверждением действия.

private void Del\_Btn\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Вы действительно хотите удалить запись?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

var CurrentCar = Sells\_DataGrid.SelectedItem as buying;

AppData.db.buying.Remove(CurrentCar);

AppData.db.SaveChanges();

MessageBox.Show("Вы успешно удалили звпись!");

Sells\_DataGrid.ItemsSource = AppData.db.buying.ToList();

}

}

Класс открытия таблицы на весь экран.

private void Open\_Full\_Click(object sender, RoutedEventArgs e)

{

SellsFullPage sellsFullPage = new SellsFullPage();

sellsFullPage.Show();

}

Класс для кнопки обновления таблицы.

private void Refresh\_Btn\_Click(object sender, RoutedEventArgs e)

{

Sells\_DataGrid.ItemsSource = AppData.db.buying.ToList();

}

Класс для экпорта данных в Word по нажатию кнопки.

private void Export\_Click(object sender, RoutedEventArgs e)

{

var allRequest = CarShopDBEntities.GetContext().buying.ToList();

var alcar = CarShopDBEntities.GetContext().car.ToList();

var application = new Word.Application();

Word.Document document = application.Documents.Add();

Word.Paragraph userParagraph = document.Paragraphs.Add();

Word.Range userRange = userParagraph.Range;

userRange.Text = "Отчёт о покупках";

userRange.InsertParagraphAfter();

Word.Paragraph tableParagraph = document.Paragraphs.Add();

Word.Range tableRange = tableParagraph.Range;

Word.Table paymentsTable = document.Tables.Add(tableRange, allRequest.Count() + 1, 7);

paymentsTable.Borders.InsideLineStyle = paymentsTable.Borders.OutsideLineStyle

= Word.WdLineStyle.wdLineStyleSingle;

paymentsTable.Range.Cells.VerticalAlignment = Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

Word.Range cellRange;

cellRange = paymentsTable.Cell(1, 1).Range;

cellRange.Text = "Номер по порядку";

cellRange = paymentsTable.Cell(1, 2).Range;

cellRange.Text = "Имя покупателя";

cellRange = paymentsTable.Cell(1, 3).Range;

cellRange.Text = "Фамилия покупателя";

cellRange = paymentsTable.Cell(1, 4).Range;

cellRange.Text = "Марка";

cellRange = paymentsTable.Cell(1, 5).Range;

cellRange.Text = "Модель";

cellRange = paymentsTable.Cell(1, 6).Range;

cellRange.Text = "Сумма покупки";

cellRange = paymentsTable.Cell(1, 7).Range;

cellRange.Text = "Дата покупки";

paymentsTable.Rows[1].Range.Bold = 1;

paymentsTable.Rows[1].Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

for (int i = 0; i < allRequest.Count(); i++)

{

var currentCategory = allRequest[i];

cellRange = paymentsTable.Cell(i + 2, 1).Range;

cellRange.Text = Convert.ToString(currentCategory.id);

cellRange.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

cellRange = paymentsTable.Cell(i + 2, 2).Range;

cellRange.Text = Convert.ToString(currentCategory.customerFname);

cellRange = paymentsTable.Cell(i + 2, 3).Range;

cellRange.Text = Convert.ToString(currentCategory.customerLname);

cellRange = paymentsTable.Cell(i + 2, 4).Range;

cellRange.Text = Convert.ToString(currentCategory.car.name);

cellRange = paymentsTable.Cell(i + 2, 5).Range;

cellRange.Text = Convert.ToString(currentCategory.car.model);

cellRange = paymentsTable.Cell(i + 2, 6).Range;

cellRange.Text = Convert.ToString(currentCategory.purchaseAmount);

cellRange = paymentsTable.Cell(i + 2, 7).Range;

cellRange.Text = currentCategory.purchaseDate.ToString("dd.MM.yyyy");

}

application.Visible = true;

}

public partial class AddPage : Window

{

private buying \_currentbuy = new buying();

public AddPage(buying selectedbuying)

{

InitializeComponent();

if(selectedbuying != null)

{

\_currentbuy = selectedbuying;

}

DataContext = \_currentbuy;

CmbCar.ItemsSource = CarShopDBEntities.GetContext().car.ToList();

}

private void Window\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left)

this.DragMove();

}

В данном классе идёт обработчик пустых полей и если пользователь что то забыл ввсти ему высветится ошибка.

public void Add\_Btn\_Click(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

var CurrentCar = CmbCar.SelectedItem as car;

if (string.IsNullOrWhiteSpace(\_currentbuy.customerFname))

errors.AppendLine("Введите имя покупателя");

if (string.IsNullOrWhiteSpace(\_currentbuy.customerLname))

errors.AppendLine("Введите фамилию покупателя");

if (\_currentbuy.car == null)

errors.AppendLine("Выберете купленную машину");

if (string.IsNullOrWhiteSpace(\_currentbuy.purchaseAmount))

errors.AppendLine("Введите стоимость покупки");

if (\_currentbuy.purchaseDate == null)

errors.AppendLine("Выберите дату");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

В данном цикле производится сохранение введёных данных и если что то пойдёт не так то приложение не закроется и высветится ошибка.

if (\_currentbuy.id >= 0)

CarShopDBEntities.GetContext().buying.AddOrUpdate(\_currentbuy);

try

{

CarShopDBEntities.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена!");

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

# ЗАКЛЮЧЕНИЕ

При выполнении данной работы, а именно конструирования автоматизированной системы «Автосалон», была разработана диаграмма Entity Relationship, была спроектирована база данных в приложении Microsoft SQL Server Management Studio 18, был разработан интерфейс и функционал автоматизированной информационной системы с помощью С# WPF. Данная автоматизированная информационная система была разработана для администратора. С её помощью можно быстро создавать необходимые отчёты.

Экономический эффект от внедрения автоматизированной информационной системы «Автосалон» ожидается за счет сокращения затрат времени на выполняемые администратором, исключения ошибок при формировании отчетов, увеличения времени на анализ. При ручном создании отчетов могут быть допущены ошибки, правильно составленный алгоритм разрабатываемой программы экономит время и исключает ошибки.

Поставленные цели и задачи при проектировании информационной системы были выполнены в полной мере.

Ссылка на репозиторий GitHub: <https://github.com/fankfankov/Kursovoy_Project-C-WPF>

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Грин Дженнифер. Head First: изучаем С#/ Грин Дженнифер, Стиллмен Эндрю – Санкт-Питербург: 2022.
2. Мэтью Мак-Дональд WPF Windows Presentation Foundation в .NET 4.5 с примерами на C#: для профессионалов/ Мэтью Мак-Дональд: Санкт-Питербург, 2016.

Интернет-ресурсы

* + 1. Stack overflow exception. Интерфейсы // C# - Киберфорум – Режим доступа: [https://www.cyberforum.ru/csharp](https://www.cyberforum.ru/csharp-beginners/thread2225109.html?ysclid=lgw4535h24160143326)
    2. Newest 'c#' Questions // Stack Overflow. – Режим доступа: [https://stackoverflow.com/questions](https://stackoverflow.com/questions/tagged/c%23)
    3. Документация по C#. Начало работы, руководства, справочные материалы. // Microsoft Learn. – Режим доступа:  <https://learn.microsoft.com/ru-ru/dotnet/csharp/>
    4. Особые исключения в .NET и как их готовить // Хабр – Режим доступа: [https://habr.com/ru/companies/jugru/articles/426045](https://habr.com/ru/companies/jugru/articles/426045/)
    5. Что такое Windows Presentation Foundation - WPF .NET // Microsoft Learn – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0>