**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский политехнический техникум»**

# КУРСОВОЙ ПРОЕКТ

По дисциплине МДК.05.02 Разработка кода

информационных систем

ТЕМА: РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ «АВТОТРАНСПОРТНОЕ ПРЕДПРИЯТИЕ»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Погожев А.С /

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**ПРОВЕРИЛ**

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

(подпись) (Ф.И.О.)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc133389942)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133389943)

[1.1 Платформа .NET 4](#_Toc133389944)

[1.2 Язык программирования C# 5](#_Toc133389945)

[1.3 Windows Presentation Foundation 6](#_Toc133389946)

[1.4 СУБД SQL Server 8](#_Toc133389947)

[1.5 Microsoft SQL Server Management Studio 10](#_Toc133389948)

[1.6 Entity Framework 11](#_Toc133389949)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 14](#_Toc133389950)

[2.1 Разработка диаграммы ERD 14](#_Toc133389951)

[2.2 Разработка базы данных 16](#_Toc133389952)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 22](#_Toc133389953)

[3.1 Разработка интерфейса информационной системы 22](#_Toc133389954)

[3.2 Программирование информационной системы 34](#_Toc133389955)

[ЗАКЛЮЧЕНИЕ 41](#_Toc133389956)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 42](#_Toc133389957)

# ВВЕДЕНИЕ

**Автотранспортное предприятие (АТП)** - организация, осуществляющая перевозки автомобильным транспортом, а также хранение, техническое обслуживание и ремонт подвижного состава. АТП решает вопросы, связанные с доставкой, т.е. выбор вида транспорта, методы организации перевозок, тип транспортных средств, участие в организации погрузочно-разгрузочных работ, использование современных технологий в организации размещения, учета товаров и запасов на складах и терминалах.

Грузоперевозки всегда были и будут актуальны так как перевозки на большие расстояния и с большим объёмом грузов требуют специальный транспорт, способный перевозить большие объёмы грузов. Так же грузоперевозки являются очень важной частью экономики страны, без которых многие товары, привезённые из других стран, не доставлялись бы в дальние уголки нашей страны. Ранее все операции записывались в путевые журналы и журналы учёта заказов. Автоматизированная информационная система необходима для ускорения получения, обработки и выполнения заказов.

**Объект** – информационная система “Автотранспортное предприятие”.

**Предмет** – автоматизация бизнес-процессов автотранспортного предприятия.

**Цель** – разработать информационную систему автотранспортного предприятия.

**Задачи:**

* выбрать инструментарий;
* спроектировать базу данных;
* разработать информационную систему.

При написании курсового проекта использовались методы: теоретический (анализ документации), эмпирический метод (изучение сред разработки).

Практическая значимость заключается в том, что данный приложение можно использовать для внедрения в информационную систему автотранспортного предприятия.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

## Платформа .NET

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows. При разработке платформы .NET Framework учитывались следующие цели.

Платформа .NET Framework состоит из общеязыковой среды выполнения (среды CLR) и библиотеки классов .NET Framework. Основой платформы .NET Framework является среда CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками и удаленное взаимодействие. При этом средой накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность. Фактически основной задачей среды выполнения является управление кодом. Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом. Библиотека классов является комплексной объектно-ориентированной коллекцией повторно используемых типов, которые применяются для разработки приложений — начиная с обычных приложений, запускаемых из командной строки, и приложений с графическим интерфейсом (GUI) и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как веб-формы и веб-службы XML.

Платформа .NET Framework может размещаться неуправляемыми компонентами, которые загружают среду CLR в собственные процессы и запускают выполнение управляемого кода, создавая таким образом программную среду, позволяющую использовать средства как управляемого, так и неуправляемого выполнения. Платформа .NET Framework не только предоставляет несколько базовых сред выполнения, но также поддерживает разработку базовых сред выполнения независимыми производителями.

Веб-приложения – это одна из распространенных групп приложений, которые пишутся под .Net. Особенность заключается в том, что такие приложения работают через браузер и требуют стабильного интернет-соединения. Сюда могут входить простые одностраничные сайты, но чаще всего это крупномасштабные проекты, требующие больших усилий.

Проще говоря, платформа.NET – это программное обеспечение, которое может выполнять следующие задачи.

Транслировать код языка программирования .NET в инструкции, которые может обрабатывать вычислительное устройство.

Предоставлять утилиты для эффективной разработки программного обеспечения. С их помощью можно, например, определить текущее время или напечатать текст на экране.

Определять набор типов данных для хранения на компьютере такой информации, как текст, числа и даты.

Семейство продуктов Visual Studio предоставляет отличные возможности для разработки .NET в Windows, Linux и macOS. В Visual Studio Marketplace есть тысячи расширений редактора от Microsoft и других компаний. Если вы предпочитаете использовать другой редактор, для многих популярных редакторов существуют инструменты командной строки и подключаемые модули .NET.

## 1.2 Язык программирования C#

C# — объектно-ориентированный, **ориентированный на компоненты** язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# — **объектно-ориентированный**язык. Вы определяете типы и их поведение.

Программы C# выполняются в .NET, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (CLR) и набор библиотек классов. Среда CLR — это реализация общеязыковой инфраструктуры языка (CLI), являющейся международным стандартом, от корпорации Майкрософт. CLI является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

Исходный код, написанный на языке C# компилируется в промежуточный язык (IL), который соответствует спецификациям CLI. Код на языке IL и ресурсы, в том числе растровые изображения и строки, сохраняются в сборке, обычно с расширением .dll. Сборка содержит манифест с информацией о типах, версии, языке и региональных параметрах для этой сборки.

При выполнении программы C# сборка загружается в среду CLR. Среда CLR выполняет JIT-компиляцию из кода на языке IL в инструкции машинного языка. Среда CLR также выполняет другие операции, например, автоматическую сборку мусора, обработку исключений и управление ресурсами. Код, выполняемый средой CLR, иногда называют "управляемым кодом". "Неуправляемый код" компилируется на машинный язык, предназначенный для конкретной платформы.

С# популярен за счет своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции.

Популярность языка – еще одно значимое преимущество. Большое количество поклонников C# способствуют его развитию. Также это благоприятно влияет на рост числа вакансий, связанных с разработкой на языке Microsoft.

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.

## 1.3 Windows Presentation Foundation

WPF существует в виде подмножества типов .NET, которые по большей части находятся в пространстве имен System.Windows. Если ранее вы создавали приложения в .NET с помощью таких платформ, как ASP.NET и Windows Forms, то должны быть знакомы с фундаментальными принципами программирования WPF. Вы выполняете следующие действия:

1. Создание экземпляров классов
2. Настройка свойств
3. Вызов методов
4. Обработка событий

WPF включает в себя дополнительные конструкции программирования, которые расширяют возможности свойств и событий: свойства зависимостей и перенаправленные события.

WPF позволяет разрабатывать приложения, используя как разметку, так и код программной части, что привычно для разработчиков на ASP.NET. Разметка XAML обычно используется для определения внешнего вида приложения, а управляемые языки программирования (код программной части) — для реализации его поведения.

XAML — это язык разметки на основе XML, который служит для определения внешнего вида приложения в декларативной форме. Обычно он используется для определения окон, страниц и пользовательских элементов управления, а также их заполнения элементами управления, фигурами и графическими элементами.

Элементы управления чаще всего используются для определения ввода данных пользователем и реагирования на него. Система ввода WPF использует как прямые, так и перенаправленные события для поддержки ввода текста, управления фокусом и определения положения указателя мыши.

Приложения часто предъявляют сложные требования к вводу. WPF предоставляет систему команд, которая отделяет действия по вводу данных пользователем от кода, реагирующего на эти действия. Система команд позволяет нескольким источникам вызывать одну и ту же логику команд. Например, рассмотрим распространенные операции редактирования, используемые различными приложениями: Копировать, Вырезать и Вставить. Эти операции можно вызывать путем выполнения различных действий пользователя, если они реализованы с помощью команд.

Ниже перечислены встроенные элементы управления WPF:

* **Кнопки**: Button и RepeatButton.
* **Вывод данных**: DataGrid, ListViewи TreeView.
* **Вывод и выбор дат**: Calendar и DatePicker.
* **Диалоговые окна**: OpenFileDialog, PrintDialog и SaveFileDialog.
* **Рукописный ввод**: InkCanvas и InkPresenter.
* **Документы**: DocumentViewer, FlowDocumentPageViewer, FlowDocumentReader, FlowDocumentScrollViewer и StickyNoteControl.
* **Ввод**: TextBox, RichTextBox и PasswordBox.
* **Макет**: Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window и WrapPanel.
* **Мультимедиа**: Image, MediaElement и SoundPlayerAction.
* **Меню**: ContextMenu, Menu и ToolBar.
* **Навигация**: Frame, Hyperlink, Page, NavigationWindowи TabControl.
* **Выбор**: CheckBox, ComboBox, ListBox, RadioButton и Slider.
* **Информирование пользователя**: AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock и ToolTip.

При создании пользовательского интерфейса вы компонуете элементы управления, настраивая их расположение и размер. Основным требованием любого макета является адаптация к изменениям размеров окна и параметров экрана. Платформа WPF избавляет вас от необходимости писать код для адаптации макета к таким условиям, предоставляя первоклассную расширяемую систему макета.

Ключевым элементом системы макета является относительное позиционирование, которое упрощает адаптацию к меняющимся характеристикам окна и экрана. Система макета также управляет взаимодействием между элементами управления для определения макета. Взаимодействие протекает в два этапа: сначала элемент управления сообщает родительскому объекту о требуемом расположении и размере, а затем родительский объект сообщает родительскому элементу, какое пространство он может занять.

Система макета доступна дочерним элементам управления посредством базовых классов WPF. Для стандартных макетов, таких как сетка, наложение и закрепление, в WPF имеется несколько элементов управления макетом:

* Canvas: дочерние элементы управления предоставляют собственный макет.
* DockPanel: дочерние элементы управления выравниваются по краям панели.
* Grid: дочерние элементы управления упорядочиваются по строкам и столбцам.
* StackPanel: дочерние элементы управления располагаются с наложением по вертикали или по горизонтали.
* VirtualizingStackPanel: дочерние элементы управления виртуализируются и располагаются в линию по горизонтали или по вертикали.
* WrapPanel: дочерние элементы управления размещаются слева направо и помещаются в следующую строку, если в текущей строке недостаточно места.

## 1.4 СУБД SQL Server

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

СУБД SQL server используются для создания, размещения, хранения и управления реляционными (табличными) базами данных на специальных серверах или в облаке. Они работают через настольные приложения и web-сайты. К основным преимуществам их функционирования относятся:

* высокоскоростной доступ к данным, обеспечиваемый надежной клиент-серверной архитектурой СУБД;
* простота работы и администрирования, обусловленные понятной структурой языка программирования SQL;
* безопасность хранения информации в БД - благодаря возможности шифрования данных и резервного копирования.

Специфика работы сервера базы данных SQL server заключается в транзакционной обработке данных. Это означает, что по каждому запросу от СУБД обрабатывается и сохраняется небольшое количество информации.

Применение SQL server позволяет автоматизировать решение различных бизнес-задач, поддерживать проведение аналитики данных в режиме онлайн, отслеживать направление ресурсов СУБД, управлять транзакциями (операциями по обработке данных).

Для реализации функций СУБД на сегодняшний день чаще всего используются следующие SQL-серверы:

* MS SQL server - многопользовательский программный продукт, разработанный компанией Microsoft, обладающий высокой производительностью и отказоустойчивостью, тесно интегрированный с ОС Windows. Этот сервер поддерживает удаленные подключения, работает с многими популярными типами данных, дает возможность создавать триггеры и хранимые данные, имеет практичные и удобные утилиты для настройки;
* Oracle Database server - СУБД, предназначенная для создания, консолидации и управления базами данных в облачной среде. Используя этот сервер, можно как автоматизировать обычные бизнес-операции, так и выполнять динамический многомерный анализ данных (OLAP), проводить операции с документами xml-формата и управлять разделенной и локальной информацией;
* IBM DB2 - семейство СУБД для работы с реляционными базами данных, признанное самым производительным, имеющим высокие технические показатели и возможности масштабирования. SQL-серверы этой группы характеризуются мультиплатформенностью, способностью к мгновенному созданию резервных копий и восстановлению БД, реорганизации таблиц в онлайн-режиме, разбиению баз данных, определению пользователями новых типов данных;
* MySQL - СУБД, разработанная и поддерживаемая компанией Oracle. В основном она используется локальными или удаленными клиентами, позволяя им работать с таблицами разных типов, поддерживающих полнотекстовый поиск или выполняющих транзакции на уровне отдельных записей;
* PostgreSQL - СУБД с открытым исходным кодом, работающая с объектно-реляционными (поддерживающими пользовательские объекты) базами данных. Также PostgreSQL предназначена для создания, хранения и извлечения сложных структур данных. Она поддерживает самые различные типы данных (среди них - числовые, текстовые, булевы, денежные, бинарные данные, сетевые адреса, xml и другие).

## 1.5 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.

SQL Server Management Studio (SSMS) — утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

## 1.6 Entity Framework

Entity Framework — это набор технологий в ADO.NET, которые поддерживают разработку программных приложений, ориентированных на данные. Архитекторам и разработчикам приложений, ориентированных на обработку данных, приходится учитывать необходимость достижения двух совершенно различных целей. Они должны моделировать сущности, связи и логику решаемых бизнес-задач, а также работать с ядрами СУБД, используемыми для сохранения и получения данных. Данные могут распределяться по нескольким системам хранения данных, в каждой из которых применяются свои протоколы, но даже в приложениях, работающих с одной системой хранения данных, необходимо поддерживать баланс между требованиями системы хранения данных и требованиями написания эффективного и удобного для обслуживания кода приложения.

Платформа Entity Framework позволяет работать с данными в форме специфических для домена объектов и свойств (например, с клиентами и их адресами) без необходимости учитывать формат базовых таблиц и столбцов базы данных, где хранятся эти данные. Entity Framework дает разработчикам возможность работать с данными на более высоком уровне абстракции, создавать и сопровождать приложения, ориентированные на работу с данными, одновременно с этим сокращая объем кода по сравнению с традиционными приложениями. поскольку Entity Framework является компонентом платформа .NET Framework, Entity Framework приложения могут работать на любом компьютере, на котором установлена платформа .NET Framework с пакетом обновления 1 (SP1) версии 3,5.

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.

При работе со средствами работы с моделью EDM концептуальная модель, модель хранения и сопоставление между ними выражены в схемах на основе XML и определены в файлах с именами с соответствующими расширениями.

* Язык CSDL определяет концептуальную модель. Язык CSDL — это реализация EDMEntity Framework. Расширение файла - CSDL.
* Язык SSDL определяет модель хранения данных, которая также называется логической моделью. Расширение файла - SSDL.
* Язык MSL определяет сопоставление модели хранения и концептуальной модели. Расширение файла - MSL.

Модель хранения и сопоставления при необходимости могут быть изменены без изменения концептуальной модели, классов данных и кода приложения. Модели хранения зависят от поставщика, поэтому можно работать с согласованной концептуальной моделью через различные источники данных.

Entity Framework использует эти файлы модели и сопоставления для создания, чтения, обновления и удаления операций с сущностями и связями в концептуальной модели в эквивалентных операциях в источнике данных. Entity Framework даже поддерживает сопоставление сущностей в концептуальной модели с хранимыми процедурами в источнике данных.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

## 2.1 Разработка диаграммы ERD

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи — глаголов.

Под понятием «сущности» подразумеваются объекты или понятия, несущие важную информацию. С точки зрения грамматики, они, как правило, обозначаются существительными, например, «товар», «клиент», «заведение» или «промоакция». Ниже представлены три наиболее распространенных типа сущностей, используемых в ER-диаграммах.

Связи используются в схемах «сущность-связь» для обозначения взаимодействия между двумя сущностями. Грамматически связи, как правило, выражаются глаголами, например, «назначить», «закрепить», «отследить», и несут полезную информацию, которую невозможно получить, опираясь только на типы сущностей.

ERD-атрибуты характеризуют сущности, позволяя пользователям лучше разобраться в устройстве базы данных. Атрибуты содержат информацию о сущностях, выделенных в концептуальной ER-диаграмме.



Рисунок 2.1. ERD-Диаграмма

На данной диаграмме представлены сущности Customer, Order, Client Manager и Car Manager, которые взаимодействуют между собой. В сущностях имеются различные атрибуты: Id, FirstName, SecondName, MiddleName, Phone, Password, Name, DateOfOrder, Weight и DateOfBirthday. Между сущностями проставлены отношения: Многий к многим, а также Один ко Многим.

## 2.2 Разработка базы данных

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа, настройки, администрирования, администрирования и разработки всех компонентов SQL Server, базы данных Azure SQL, Управляемый экземпляр 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.

Создание базы данных в SQL Server Management Studio:

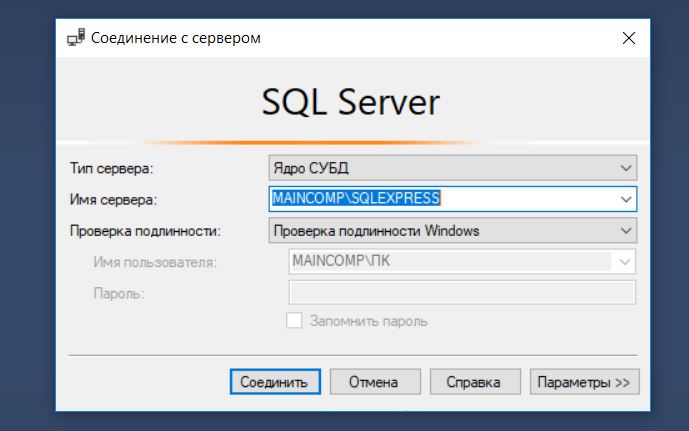
1. Запустить среду SQL Server Management Studio и подключиться к серверу.

Рисунок 2.2. Окно входа в БД

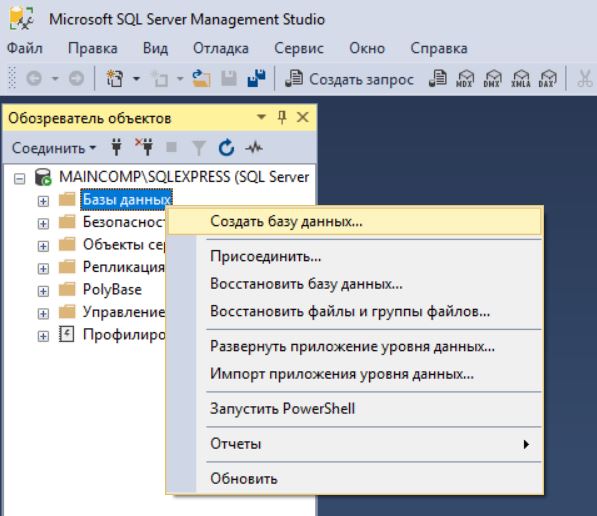
1. В обозревателе объектов щелкнуть ПКМ по контейнеру «База данных» и создать базу данных

Рисунок 2.3. Создание БД через обозреватель объектов

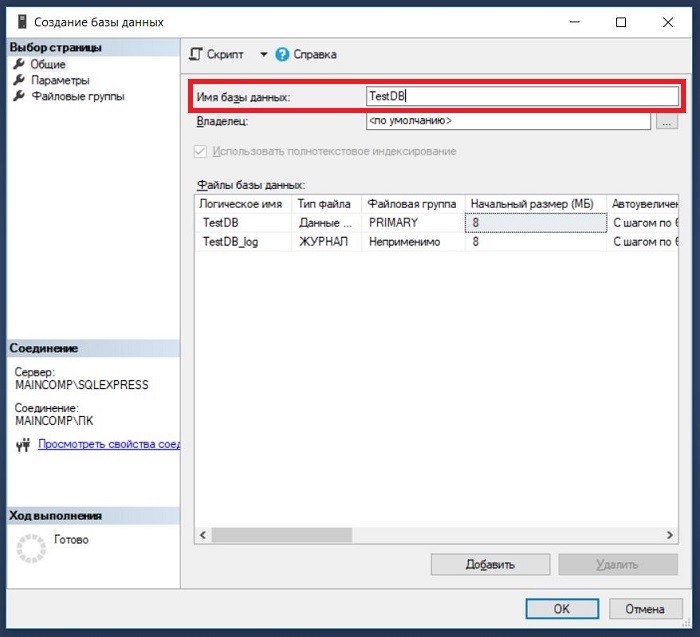
1. Открывается окно «Создание базы данных», где необходимо написать название базы данных т.к. остальные параметры необходимо настраивать уже после создания базы данных.

Рисунок 2.4. Создание БД

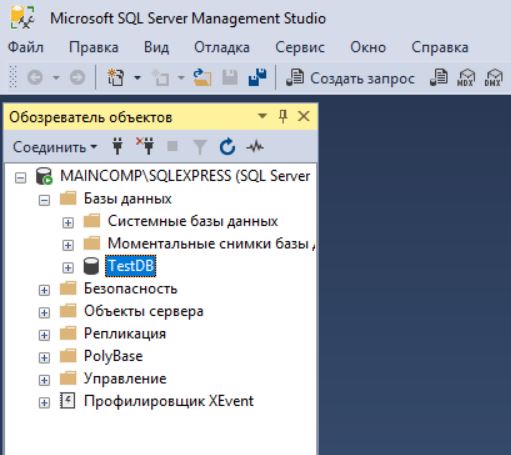
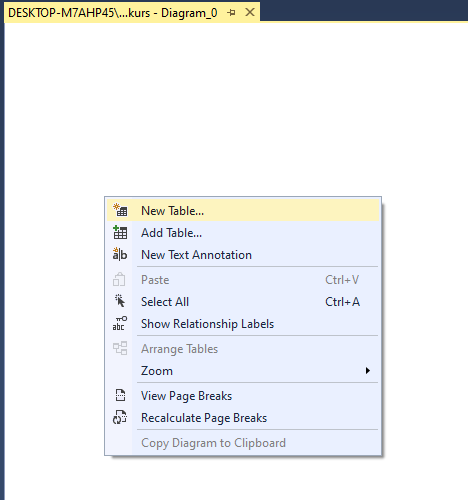
1. База данных создана и видна в обозревателе объектов

Рисунок 2.5. БД в обозревателе объектов

Рисунок 2.6. Создание «Сущности» в БД

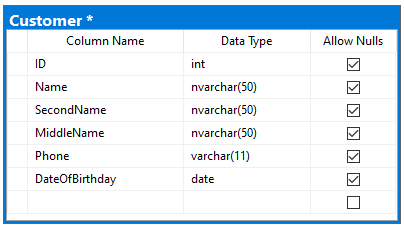
****

Рисунок 2.7. Созданная «Сущность» с атрибутами и типами данных

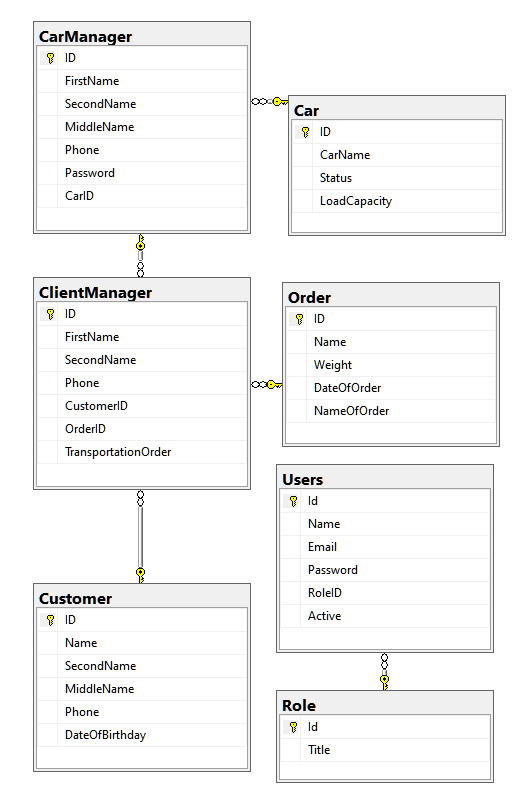


Рисунок 2.8. Созданная диаграмма Базы Данных с внесенными в нее данными

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 3.1 Разработка интерфейса информационной системы

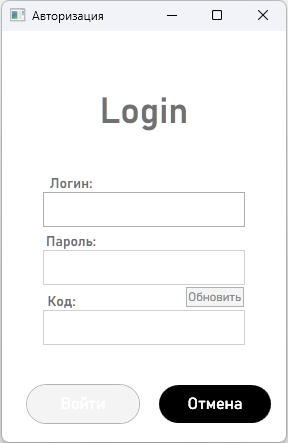


Рисунок 3.1. Скриншот окна авторизации

На рисунке 3.1 окно авторизации, состоящее из окна ввода логина, окна ввода пароля, окна ввода кода, кнопка обновления кода, и две кнопки: Войти и Отмена. Так же на кнопку отмена производится очистка полей ввода.

Код окна авторизации:

<Grid>

<Button x:Name="Vhod" Click="Vhod\_Click" Content="Войти" FontFamily="Bahnschrift SemiLight" FontSize="16" Margin="24,353,0,0" RenderTransformOrigin="1,0" VerticalContentAlignment="Center" BorderBrush="{x:Null}" OpacityMask="Black" Foreground="White" Height="40" VerticalAlignment="Top" HorizontalAlignment="Left" Width="114" Background="Black" IsEnabled="False">

<Button.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform Angle="0"/>

<TranslateTransform/>

</TransformGroup>

</Button.RenderTransform>

<Button.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="20"/>

</Style>

</Button.Resources>

</Button>

<TextBox x:Name="LOGIN" Margin="0,60,0,0" TextWrapping="Wrap" Text="Login" BorderThickness="0,0,0,0" FontFamily="Bahnschrift SemiBold" FontSize="36" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Foreground="Black" VerticalAlignment="Top" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Width="174" MaxWidth="174" MaxHeight="43.2066666666667" IsEnabled="False" VerticalScrollBarVisibility="Disabled"/>

<TextBox x:Name="LogBox" KeyUp="LogBlock\_Up" HorizontalAlignment="Center" Margin="0,161,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="202" Height="35" VerticalContentAlignment="Center"/>

<PasswordBox x:Name="PassBox" KeyUp="PassBlock\_Up" HorizontalAlignment="Center" Margin="0,219,0,0" VerticalAlignment="Top" Width="202" Height="35" VerticalContentAlignment="Center" IsEnabled="False"/>

<TextBox x:Name="LogText" Margin="42,144,188,0" TextWrapping="Wrap" Text="Логин:" BorderThickness="0,0,0,0" FontFamily="Bahnschrift SemiBold" FontSize="14" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Foreground="Black" VerticalAlignment="Top" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" MaxWidth="174" MaxHeight="43.2066666666667" IsEnabled="False" VerticalScrollBarVisibility="Disabled" />

<TextBox x:Name="PassText" Margin="36,202,182,0" TextWrapping="Wrap" Text="Пароль:" BorderThickness="0,0,0,0" FontFamily="Bahnschrift SemiBold" FontSize="14" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Foreground="Black" VerticalAlignment="Top" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" MaxWidth="174" MaxHeight="43.2066666666667" IsEnabled="False" VerticalScrollBarVisibility="Disabled"/>

<Button x:Name="Otmena" Click="Otmena\_Click" Content="Отмена" FontFamily="Bahnschrift SemiLight" FontSize="16" Margin="156,353,0,0" RenderTransformOrigin="1,0" VerticalContentAlignment="Center" BorderBrush="{x:Null}" OpacityMask="Black" Foreground="White" Height="40" VerticalAlignment="Top" HorizontalAlignment="Left" Width="114" Background="Black">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="20"/>

</Style>

</Button.Resources>

<Button.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform Angle="0"/>

<TranslateTransform/>

</TransformGroup>

</Button.RenderTransform>

</Button>

<TextBox x:Name="CodBox" KeyUp="CodeBlock\_Up" HorizontalAlignment="Center" Margin="0,279,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="202" Height="35" VerticalContentAlignment="Center" IsEnabled="False"/>

<TextBox x:Name="cod" Margin="42,262,207,0" TextWrapping="Wrap" Text="Код:" BorderThickness="0,0,0,0" FontFamily="Bahnschrift SemiBold" FontSize="14" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Foreground="Black" VerticalAlignment="Top" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" MaxWidth="174" MaxHeight="43.2066666666667" IsEnabled="False" VerticalScrollBarVisibility="Disabled" />

<TextBlock x:Name="CodeBlock" HorizontalAlignment="Left" Margin="87,262,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="38" FontFamily="Bahnschrift" Foreground="#FF676767" Height="17" FontSize="14"/>

<Button x:Name="Reloadd" Content="Обновить" Click="Reload\_Click" HorizontalAlignment="Left" Margin="184,256,0,0" VerticalAlignment="Top" IsEnabled="False" Height="20" Width="58" Background="#FFDDDDDD" FontFamily="Bahnschrift Light"/>

</Grid>

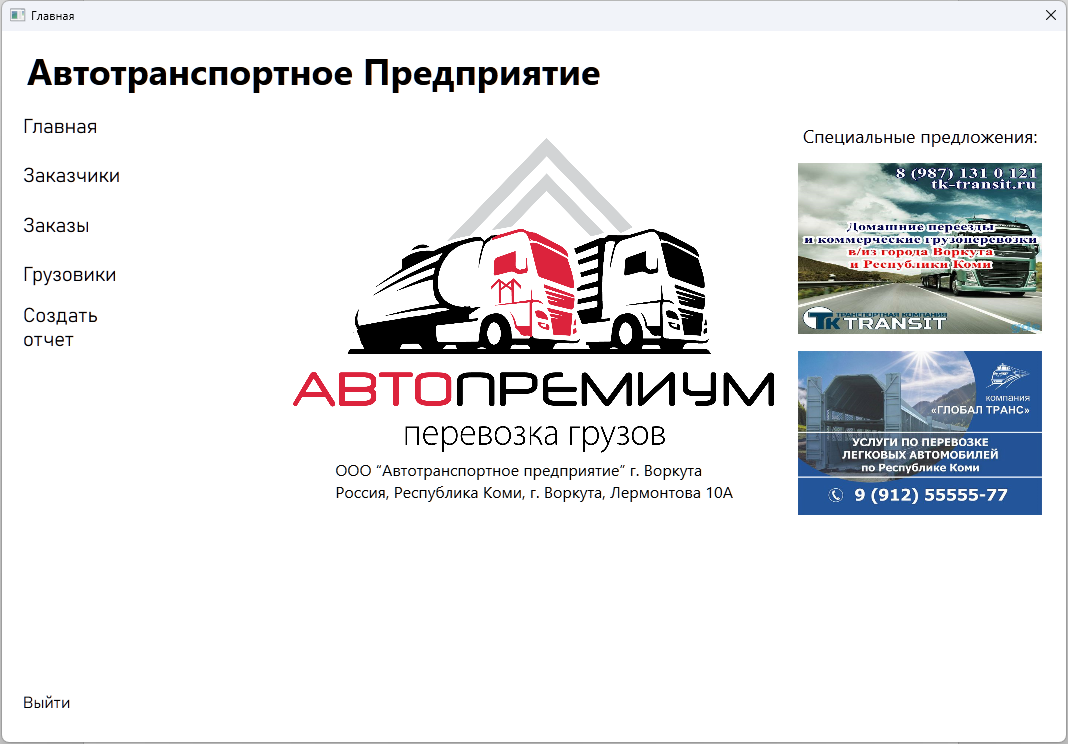


Рисунок 3.2. Скриншот главного окна программы

На рисунке 3.2 показано главное окно программы. В нем отображаются логотип компании, информация о компании, а так же специальные предложения от компаний – партнеров. Слева имеются кнопки перехода по окнам и создания отчета. Имеется функция создания отчета в формате Excel и кнопка выйти, переходящая на окно авторизации. Ниже показан программный код:

<Grid>

<Button x:Name="Glavnaya" Click="Glav\_Click" Content="Главная" HorizontalAlignment="Left" Margin="20,75,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Background="{x:Null}" FontFamily="Bahnschrift Light" FontSize="20" BorderThickness="0,0,0,0" Height="43" Width="80" VerticalContentAlignment="Center" HorizontalContentAlignment="Left" Grid.IsSharedSizeScope="True" />

<Button x:Name="Zakazchik" Click="Zakazchiki\_Click" Content="Заказчики" HorizontalAlignment="Left" Margin="20,124,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Background="{x:Null}" FontFamily="Bahnschrift Light" FontSize="20" BorderThickness="0,0,0,0" Height="43" Width="100" VerticalContentAlignment="Center" HorizontalContentAlignment="Left" />

<Button x:Name="Orders" Click="Orders\_Click" Content="Заказы" HorizontalAlignment="Left" Margin="20,174,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Background="{x:Null}" FontFamily="Bahnschrift Light" FontSize="20" BorderThickness="0,0,0,0" Height="43" Width="67" VerticalContentAlignment="Center" HorizontalContentAlignment="Left" />

<Button x:Name="Cars" Click="Cars\_Click" Content="Грузовики" HorizontalAlignment="Left" Margin="20,223,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Background="{x:Null}" FontFamily="Bahnschrift Light" FontSize="20" BorderThickness="0,0,0,0" Height="43" Width="97" VerticalContentAlignment="Center" HorizontalContentAlignment="Left" />

<Button x:Name="Vihod" Click="Vihod\_Click" Content="Выйти" HorizontalAlignment="Left" Margin="20,660,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Background="{x:Null}" FontFamily="Bahnschrift Light" FontSize="16" BorderThickness="0,0,0,0" Height="25" Width="50" VerticalContentAlignment="Center" HorizontalContentAlignment="Left" />

<Frame x:Name="MainFrame" Content="" Margin="167,75,0,0" NavigationUIVisibility="Hidden" Navigated="MainFrame\_Navigated" ContentRendered="MainFrame\_ContentRendered"/>

<Button x:Name="Sodzat" Click="Sozdat\_Click" Content="Создать&#xD;&#xA;отчет" HorizontalAlignment="Left" Margin="20,271,0,0" VerticalAlignment="Top" BorderBrush="{x:Null}" Background="{x:Null}" FontFamily="Bahnschrift Light" FontSize="20" BorderThickness="0,0,0,0" Height="53" Width="122" VerticalContentAlignment="Center" HorizontalContentAlignment="Left" />

<Label Content="Автотранспортное Предприятие" HorizontalAlignment="Left" Margin="20,10,0,0" VerticalAlignment="Top" FontSize="36" FontWeight="Bold"/>

</Grid>

Код страницы:

<Grid x:Name="Glavnaya" Background="White">

<Label Content="Специальные предложения:" Margin="0,13,24,0" VerticalAlignment="Top" Height="39" FontSize="18" HorizontalAlignment="Right" Width="244"/>

<Image Margin="0,57,24,0" Source="/c72c83575c9b0d09f41462105512efb3.jpg" Stretch="Fill" Height="171" VerticalAlignment="Top" HorizontalAlignment="Right" Width="244"/>

<Image Margin="0,245,24,0" Source="/YykDH5ucIoA.jpg" Stretch="Fill" Height="164" VerticalAlignment="Top" HorizontalAlignment="Right" Width="244"/>

<Image Margin="0,32,292,0" Source="/8628165eb4c6ded244952ee6d1325d66.png" Stretch="Fill" Height="316" VerticalAlignment="Top" HorizontalAlignment="Right" Width="481"/>

<Label Content="ООО “Автотранспортное предприятие” г. Воркута&#xD;&#xA;Россия, Республика Коми, г. Воркута, Лермонтова 10А" HorizontalAlignment="Right" Margin="0,348,328,0" FontSize="16" Height="53" VerticalAlignment="Top"/>

</Grid>

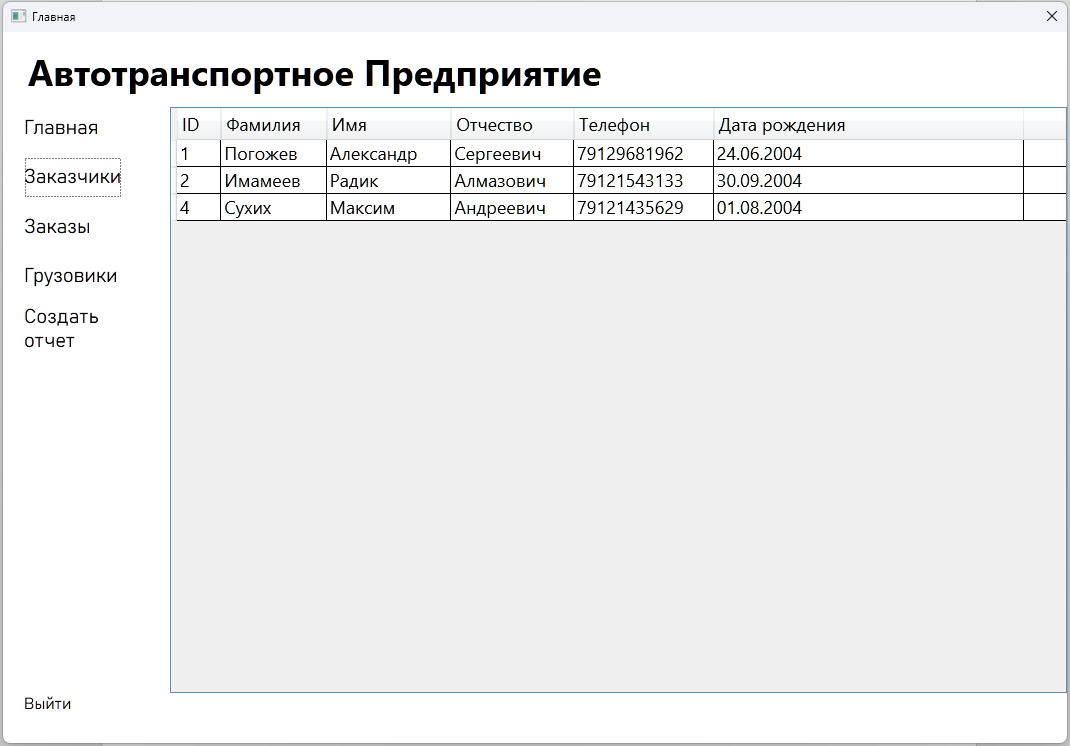


Рисунок 3.3 Страница заказчиков

На рисунке 3.3 находится страница заказчиков, где описывается ID, ФИО, телефон и дата рождения. Слева, как и на главной странице, имеется навигационное меню, используемое для перехода между страницами и кнопка выхода. Программный код:

<Grid x:Name="Zakazhiki" Background="White">

<DataGrid AutoGenerateColumns="False" x:Name="StatementGrid" IsReadOnly="True" FontSize="18" Margin="0,0,0,50">

<DataGrid.Columns>

<DataGridTextColumn Header="ID" Width="auto" Binding="{Binding ID}"></DataGridTextColumn>

<DataGridTextColumn Header="Фамилия" Width="auto" Binding="{Binding SecondName}"></DataGridTextColumn>

<DataGridTextColumn Header="Имя" Width="auto" Binding="{Binding Name}"></DataGridTextColumn>

<DataGridTextColumn Header="Отчество" Width="auto" Binding="{Binding MiddleName}"></DataGridTextColumn>

<DataGridTextColumn Header="Телефон" Width="auto" Binding="{Binding Phone}"></DataGridTextColumn>

<DataGridTextColumn Header="Дата рождения" Width="auto" Binding="{Binding DateOfBirthday, StringFormat={}{0:dd.}{0:MM.}{0:yyyy}}"></DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

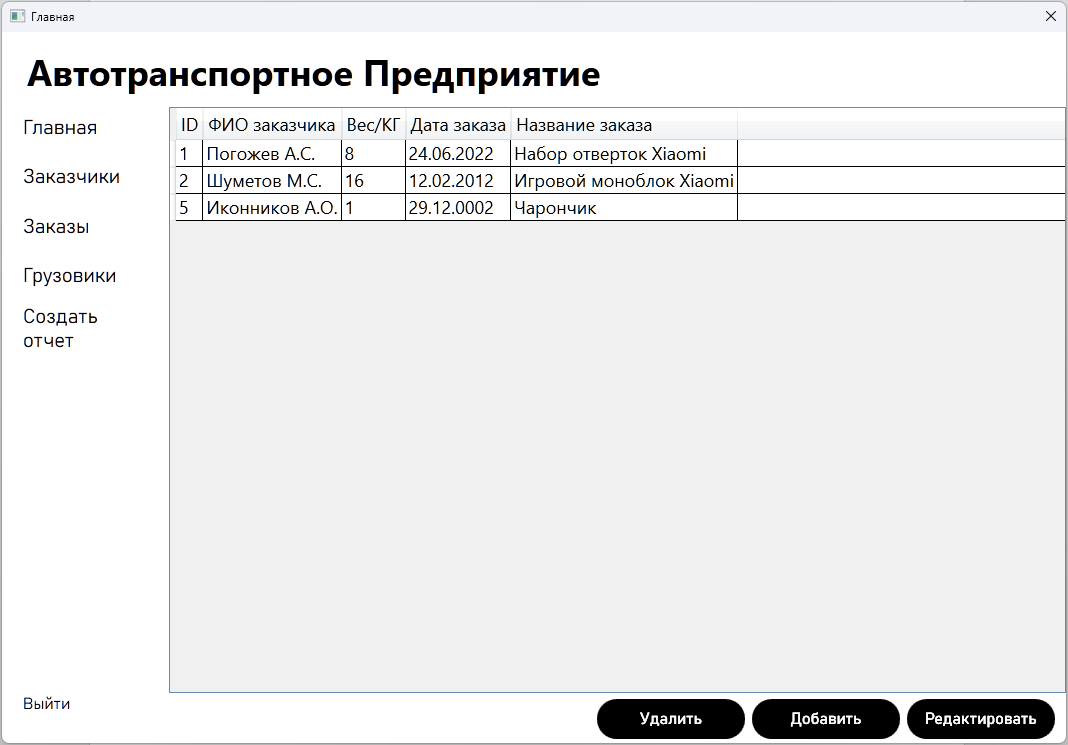


Рисунок 3.4. Страница заказов

На странице заказов (рисунок 3.4) показаны заказчики предприятия, а так же некоторые, необходимые для перевозки данные, такие как ID, ФИО заказчика, вес заказа, дата заказа и наименование заказа. Программный код страницы заказов:

<Grid x:Name="Orders1" Background="White">

<DataGrid AutoGenerateColumns="False" x:Name="StatementGrid" IsReadOnly="True" FontSize="18" Margin="0,0,0,50">

<DataGrid.Columns>

<DataGridTextColumn Header="ID" Width="auto" Binding="{Binding ID}"></DataGridTextColumn>

<DataGridTextColumn Header="ФИО заказчика" Width="auto" Binding="{Binding Name}"></DataGridTextColumn>

<DataGridTextColumn Header="Вес/КГ" Width="auto" Binding="{Binding Weight}"></DataGridTextColumn>

<DataGridTextColumn Header="Дата заказа" Width="auto" Binding="{Binding DateOfOrder, StringFormat={}{0:dd.}{0:MM.}{0:yyyy}}"></DataGridTextColumn>

<DataGridTextColumn Header="Название заказа" Width="auto" Binding="{Binding NameOfOrder}"></DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

<Button x:Name="Redakt" Click="Redakt\_Click" Content="Редактировать" FontFamily="Bahnschrift SemiLight" FontSize="16" Margin="0,0,10,3" RenderTransformOrigin="1,0" VerticalContentAlignment="Center" BorderBrush="{x:Null}" OpacityMask="Black" Foreground="White" Background="Black" Height="42" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="150">

<Button.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform Angle="0"/>

<TranslateTransform/>

</TransformGroup>

</Button.RenderTransform>

<Button.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="20"/>

</Style>

</Button.Resources>

</Button>

<Button x:Name="Add" Click="Add\_Click" Content="Добавить" FontFamily="Bahnschrift SemiLight" FontSize="16" Margin="0,0,165,3" RenderTransformOrigin="1,0" VerticalContentAlignment="Center" BorderBrush="{x:Null}" OpacityMask="Black" Foreground="White" Background="Black" Height="42" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="150">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="20"/>

</Style>

</Button.Resources>

<Button.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform Angle="0"/>

<TranslateTransform/>

</TransformGroup>

</Button.RenderTransform>

</Button>

<Button x:Name="deletett" Click="Deletett\_Click" Content="Удалить" FontFamily="Bahnschrift SemiLight" FontSize="16" Margin="0,0,320,3" RenderTransformOrigin="1,0" VerticalContentAlignment="Center" BorderBrush="{x:Null}" OpacityMask="Black" Foreground="White" Background="Black" Height="42" VerticalAlignment="Bottom" HorizontalAlignment="Right" Width="150">

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="20"/>

</Style>

</Button.Resources>

<Button.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform Angle="0"/>

<TranslateTransform/>

</TransformGroup>

</Button.RenderTransform>

</Button>

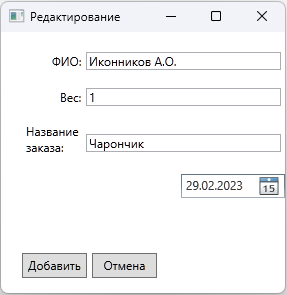
 </Grid>

Рисунок 3.4. Окно редактирования заказа

В окне редактирования заказа (рисунок 3.4) возможно поменять данные заказа или редактировать их. Имеется две кнопки: Добавить, которая отвечает за сохранение изменений в таблице, а так же кнопка Отмена, отвечающая за выход из этого окна и возврат к окну заказов. Программный код представлен ниже:

<Grid>

<Button x:Name="Addd" Content="Добавить" HorizontalAlignment="Left" Margin="21,221,0,0" VerticalAlignment="Top" Width="65" Height="25" Click="Addd\_Click"/>

<Button x:Name="Otmen" Content="Отмена" HorizontalAlignment="Left" Margin="91,221,0,0" VerticalAlignment="Top" Width="65" Height="25" Click="Otmen\_Click"/>

<TextBox x:Name="FIO" HorizontalAlignment="Left" Margin="85,20,0,0" TextWrapping="Wrap" Text="{Binding Name}" VerticalAlignment="Top" Width="195"/>

<TextBox x:Name="Ves" HorizontalAlignment="Left" Margin="85,56,0,0" TextWrapping="Wrap" Text="{Binding Weight}" VerticalAlignment="Top" Width="195"/>

<TextBox x:Name="nazvan" HorizontalAlignment="Left" TextWrapping="Wrap" Text="{Binding NameOfOrder}" VerticalAlignment="Top" Width="195" Margin="85,102,0,0"/>

<DatePicker HorizontalAlignment="Left" Margin="180,142,0,0" VerticalAlignment="Top" SelectedDate="{Binding DateOfOrder}"/>

<Label Content="ФИО:" HorizontalAlignment="Left" Margin="46,16,0,0" VerticalAlignment="Top"/>

<Label Content="Вес:" HorizontalAlignment="Left" Margin="54,52,0,0" VerticalAlignment="Top"/>

<Label Content="Название&#xD;&#xA;заказа:" HorizontalAlignment="Left" Margin="20,86,0,0" VerticalAlignment="Top" Width="67"/>

</Grid>

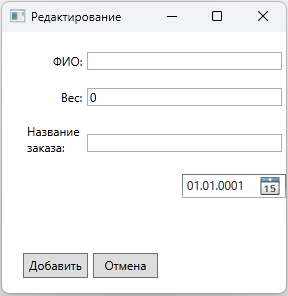


Рисунок 3.5. Окно редактирования

По аналогии с рисунком 3.4 выглядит окно добавления новых заказов. Программный код представлен ниже:

<Grid>

<Button x:Name="Addd" Content="Добавить" HorizontalAlignment="Left" Margin="21,221,0,0" VerticalAlignment="Top" Width="65" Height="25" Click="Addd\_Click"/>

<Button x:Name="Otmen" Content="Отмена" HorizontalAlignment="Left" Margin="91,221,0,0" VerticalAlignment="Top" Width="65" Height="25" Click="Otmen\_Click"/>

<TextBox x:Name="FIO" HorizontalAlignment="Left" Margin="85,20,0,0" TextWrapping="Wrap" Text="{Binding Name}" VerticalAlignment="Top" Width="195"/>

<TextBox x:Name="Ves" HorizontalAlignment="Left" Margin="85,56,0,0" TextWrapping="Wrap" Text="{Binding Weight}" VerticalAlignment="Top" Width="195"/>

<TextBox x:Name="nazvan" HorizontalAlignment="Left" TextWrapping="Wrap" Text="{Binding NameOfOrder}" VerticalAlignment="Top" Width="195" Margin="85,102,0,0"/>

<DatePicker HorizontalAlignment="Left" Margin="180,142,0,0" VerticalAlignment="Top" SelectedDate="{Binding DateOfOrder}"/>

<Label Content="ФИО:" HorizontalAlignment="Left" Margin="46,16,0,0" VerticalAlignment="Top"/>

<Label Content="Вес:" HorizontalAlignment="Left" Margin="54,52,0,0" VerticalAlignment="Top"/>

<Label Content="Название&#xD;&#xA;заказа:" HorizontalAlignment="Left" Margin="20,86,0,0" VerticalAlignment="Top" Width="67"/>

</Grid>

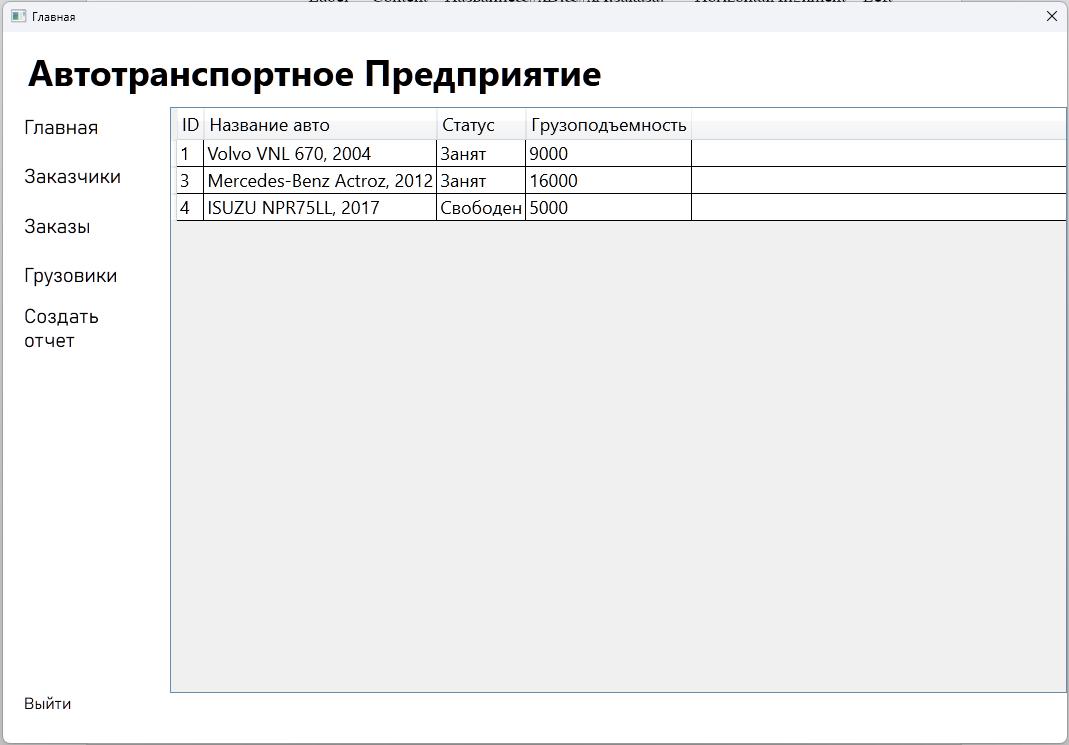


Рисунок 3.6. Страница грузовиков

На данном рисунке представлена страница с машинами, используемых в перевозках. В таблице приведены названия автомобилей, их статус и грузоподъемность. Программный код ниже:

<Grid Background="White">

<DataGrid AutoGenerateColumns="False" x:Name="StatementGrid" IsReadOnly="True" FontSize="18" Margin="0,0,0,50">

<DataGrid.Columns>

<DataGridTextColumn Header="ID" Width="auto" Binding="{Binding ID}"></DataGridTextColumn>

<DataGridTextColumn Header="Название авто" Width="auto" Binding="{Binding CarName}"></DataGridTextColumn>

<DataGridTextColumn Header="Статус" Width="auto" Binding="{Binding Status}"></DataGridTextColumn>

<DataGridTextColumn Header="Грузоподъемность" Width="auto" Binding="{Binding LoadCapacity}"></DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

## 3.2 Программирование информационной системы

Каждый фрейм и окно содержит в себе программный код, используемый для изменения поведения приложения.

DispatcherTimer timer = new DispatcherTimer();

string code;

public MainWindow()

{

InitializeComponent();

}

Выше представлена часть кода, на которой инициализируется таймер, создается переменная для хранения кода и метод инициализации компонентов окна.

private void gencode()

{

code = null;

Random random = new Random();

string[] massiveCharacters = new string[] { "1", "2", "3", "4", "5", "6", "7", "8", "9", "0" };

for (int i = 0; i < 4; i++)

{

code += massiveCharacters[random.Next(0, massiveCharacters.Length)];

}

CodeBlock.Text = code;

timer.Interval = TimeSpan.FromSeconds(10);

timer.Tick += Timer\_Tick;

timer.Start();

CodBox.IsEnabled = true;

Vhod.IsEnabled = true;

Reloadd.IsEnabled = true;

}

void Timer\_Tick(object sender, EventArgs e)

{

code = null;

MessageBox.Show("Время написания кода вышло. Повторите попытку", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

timer.Stop();

gencode();

}

Выше написана часть кода, используемая для генерации случайного кода и запуск таймера в окне авторизации. С помощью метода Рандом случайным образом генерируется код, состоящий из четырех цифр и выводится в специальное поле для отображения в окне авторизации. Во время выполнения метода, описанного выше, начинается обратный отсчет по истечении которого выводится ошибка и придется вводить код заново.

private void LogBlock\_Up(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

using (var db = new dbkursEntities())

{

var login = dbkursEntities.GetContext().Users.AsNoTracking().FirstOrDefault(l => l.Name == LogBox.Text.Trim());

if (login == null)

{

MessageBox.Show("Неверный логин", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

else

{

PassBox.IsEnabled = true;

LogBox.IsEnabled = false;

PassBox.Focus();

}

}

}

}

Выше написан код, с помощью которого, при написании логина и пароля, достаточно нажать Enter и указатель перейдет с одного поля на другое. Так же представлена функция на проверку правильности введенных данных. Если логин, введенный в поле не существует, то выводится ошибка. По такому же образцу написана проверка на правильность введенного пароля.

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

{

{

timer.Stop();

LogBox.Clear();

PassBox.Clear();

CodBox.Clear();

CodeBlock.Text = "";

LogBox.IsEnabled = true;

PassBox.IsEnabled = false;

CodBox.IsEnabled = false;

Vhod.IsEnabled = false;

Reloadd.IsEnabled = false;

}

}

Выше приведен код, используемый при нажатии кнопки Отмена в окне авторизации. При нажатии на кнопку, содержимое всех полей стирается.

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

{

timer.Stop();

gencode();

CodBox.Focus();

}

Выше представлен код кнопки Обновить. При нажатии на нее, код, получаемый методом рандом или появляется, или обновляется. При этом заново запускается таймер.

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

{

var db = new dbkursEntities();

var auth = db.Users.AsNoTracking().FirstOrDefault(u => u.Name == LogBox.Text && u.Password == PassBox.Password);

if (auth != null & code == CodBox.Text)

{

timer.Stop();

if (auth.Active == true)

{

MessageBox.Show("Доступ запрещён!", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

else

{

Globals.RoleID = auth.RoleID;

Glav NW = new Glav();

NW.Show();

Close();

}

}

else

{

MessageBox.Show("Неверный логин или пароль");

CodBox.Clear();

timer.Stop();

gencode();

}

Выше описан код, используемый кнопкой Войти для дальнейшего доступа к программе. В нем написаны методы, которые обрабатывают нажатие на кнопку войти, кнопку обновить и кнопку отмена. В первых строках кода идет сравнение написанного логина и пароля с логином и паролем в базе данных. Так же при вводе неверного пароля сбрасывается код и таймер.

{

public Zakazchiki()

{

InitializeComponent();

StatementGrid.ItemsSource = dbkursEntities.GetContext().Customer.ToList();

}

}

Выше написан код, используемый на странице заказчики. Этот код получает данные из базы данных и вписывает их в datagrid. Идентичный код используется на странице грузовики.

public Orders()

{

InitializeComponent();

StatementGrid.ItemsSource = dbkursEntities.GetContext().Order.ToList();

}

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

{

if (StatementGrid.SelectedItem != null)

{

Redaktir red = new Redaktir(StatementGrid.SelectedItem as Order);

red.ShowDialog();

dbkursEntities.GetContext().SaveChanges();

}

}

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

{

Redaktir red = new Redaktir(null);

red.ShowDialog();

}

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

{

if (StatementGrid.SelectedItem != null)

{

var Removing = StatementGrid.SelectedItems.Cast<Order>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить следующие {Removing.Count()} элеметнов?", "Внимание",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

dbkursEntities.GetContext().Order.RemoveRange(Removing);

dbkursEntities.GetContext().SaveChanges();

MessageBox.Show("Данные удалены");

StatementGrid.ItemsSource = dbkursEntities.GetContext().Order.ToList();

}

catch (Exception ex)

{

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

}

}

}

else

{

MessageBox.Show("Выберите что нужно удалить", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

В коде выше представлен код, который инициализирует компоненты окна и собирает данные из базы данных. Следующий метод определяет поведение кнопки редактировать. При нажатии кнопки редактировать открывается окно, где можно вносить изменения. Так же при нажатии кнопки добавить, изменения сохраняются. Следующий метод определяет как себя будет вести кнопка добавить. При ее нажатии открывается окно, аналогичное окну редактирования. Ее свойства повторяются. Следующий метод определяет поведение кнопки удалить. При выборе удаляемого объекта и нажатии кнопки удалить, выводится окно, которое спрашивает точно ли вы хотите удалить данные таблицы. При Согласии удаляет данные из таблицы и базы данных. Если объект не выбран и нажмется кнопка удалить, то появится диалоговое окно, которое предупредит, что ничего не выбрано.

# ЗАКЛЮЧЕНИЕ

Необходимо сказать, что в настоящее время не все современные предприятия работают на базе различных автоматизированных систем, внедрение АИС в производственном цикле предприятий не распространено повсеместно. Большинство предприятий не применяют автоматизированные системы управления, поскольку: автоматизация всего комплекса деятельности предприятия недостаточно развита; владельцы предприятий не понимают очевидные преимущества автоматизированного управления; недостаточное количество специалистов в области обеспечения работоспособности систем управления предприятий; стоимость оборудования; большинство сотрудников боятся сокращения в виду того, что машина может заменить человека. Между тем, внедрение АИС на предприятиях позволяет увеличить показатели таких производственных потоков, как:

— составление и контроль исполнения проектов;

— управление загрузкой мощностей предприятия;

— финансовый анализ и бухгалтерский учет;

— управление ресурсами предприятия;

— оптимизация движения различных производственных потоков;

— создание новых изделий, включая техническую документацию;

— оформление заказов и контроль современного исполнения;

— анализ внутренних и внешних изменений, предупреждение о чрезвычайных и внештатных ситуациях и прочее.

Разработана автоматизированная информационная система «Автотранспортного предприятия». Благодаря этой информационной системе упростилось создание, реализация и доставка заказов клиентам. Информационная система “Автотранспортное предприятие” снижает время, затраченное на создание, оформление, заказ, доставку и получение заказа. Автоматизированная информационная система “Автотранспортное предприятие” позволяет использовать меньше ручного труда, заменяя его автоматизированными функциями.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Троелсен Э. Язык программирования C# 7 и платформы .NET и .NET Core 8-е издание: книга/ Э. Троелсен, Ф. Джепикс. СПб. 2018 - 1328 с.
2. Троелсен и Джепикс. Язык программирования C# 7 и платформы .NET и .NET Core
3. Стивен С. Скиена. Алгоритмы. Руководство по разработке
4. Ицик Бен-Ган. Microsoft SQL Server 2012. Основы T-SQL

## Интернет-ресурсы

1. Обзор Entity Framework – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/ef/overview.
2. Обзор платформы .NET – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/framework/get-started/overview.
3. Обзор функционала SQL Server Management Studio (SSMS) – Режим доступа: https://info-comp.ru/ssms-feature-overview?ysclid=lenvbogeq8366784796.
4. Общие сведения о WPF – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8.
5. Руководство по WPF – Режим доступа: https://metanit.com/sharp/wpf/?ysclid=lenvi5n1bh823160038.
6. Руководство по классическим приложениям (WPF .NET) – Режим доступа:
7. Что такое MS SQL Server и чем она отличается от других СУБД – Режим доступа: https://timeweb.com/ru/community/articles/osnovy-raboty-s-mssqlserver.
8. Язык С#: где используют, что пишут, как появился и чем хорош – Режим доступа: https://skillbox.ru/media/code/yazyk-s-gde-ispolzuyut-chto-pishut-kak-poyavilsya-i-chem-khorosh/?ysclid=lenvmtelk1292224169.