**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

«Учёт сдельной оплаты труда»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Шамсулин А.Ф./

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**ПРОВЕРИЛ**

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

(подпись) (Ф.И.О.)

Воркута

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc133416280)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133416281)

[1.1 Платформа .NET 4](#_Toc133416282)

[1.2 Язык программирования С# 4](#_Toc133416283)

[1.3 Windows Presentation Foundation (WPF) 7](#_Toc133416284)

[1.4СУБД SQL server 10](#_Toc133416285)

[1.1Microsoft SQL Server Management Studio 10](#_Toc133416286)

[1.2Entity Framework 12](#_Toc133416287)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 14](#_Toc133416288)

[2.1 Разработка диаграммы ERD 14](#_Toc133416289)

[2.2 Разработка базы данных 15](#_Toc133416290)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 17](#_Toc133416291)

[3.1 Разработка интерфейса информационной системы 17](#_Toc133416292)

[3.2 Программирование информационной системы 23](#_Toc133416293)

[ЗАКЛЮЧЕНИЕ 28](#_Toc133416294)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 29](#_Toc133416295)

# ВВЕДЕНИЕ

Учёт сдельной оплаты труда на любос предприятии по праву занимает одно из центральных мест в системе учёта. Заработная плата – основной источник дохода рабочих и служащих, с её помощью осуществляется контроль за мерой труда и потребления, она используется как важнейший предмет в области экономики. Заработная плата – это вознаграждение за труд в зависимости от квалификации работника, сложности, количества, качества и условий выполнения работы. Для работодателя же труд выступает в качестве одного из элементов издержек производства.

Актуальность данной работы заключается в необходимости проектирования, разработки и внедрения информационной системы учёта и контроля рабочего времени сотрудников торговой организации, способной оказывать качественную поддержку для успешного ведения управленческой деятельности в частности и коммерческой деятельности в целом. И в отслеживании и учёте всех сотрудников на предприятии.

Проблематика данной работы заключается в отсутствии качественных недорогих инструментов и программных продуктов по учёту и контролю рабочего времени сотрудников и самих сотрудников наряду с повышением нагрузки на персонал торговых организаций.

**Объект** - информационная система «Учёт сдельной оплаты труда».

**Предмет** – автоматизация бизнес-процесса учёта оплаты труда.

**Цель** – разработка информационной системы «Расчёт сдельной оплаты труда».

Для достижения данной цели необходимо решить следующие **задачи**:

* выбрать инструментарий к информационной системе «Учёт сдельной оплаты труда»;
* спроектировать базу данных к информационной системе «Учёт сдельной оплаты труда»;
* разработать информационную систему «Учёт сдельной оплаты труда».

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# 1.1 Платформа .NET

Платформа .NET Framework – это специальная технология, предназначенная для выполнения различных веб-служб и создания приложений на ОС Windows.

Платформа .NET Framework состоит из общеязыковой среды выполнения (среды CLR или Common Language Runtime) и библиотеки классов .NET Framework. Основа платформы — это среда CLR. Её можно считать агентом, управляющим кодом во время выполнения и предоставляющим основные службы, такие как управление памятью, потоками и удаленное взаимодействие.

При разработке платформы .NET Framework учитывались цели:

* Обеспечение среды выполнения кода. Минимизация конфликтов при развертывании программного обеспечения и управлениями версиями;
* Обеспечение объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода;
* Разработка взаимодействия на основе промышленных стандартов, что позволяет интегрировать код платформы .NET Framework с любым другим кодом;
* Обеспечение среды выполнения кода, которая бы исключала проблемы с производительностью сред выполнения сценариев или интерпретируемого кода;
* Предоставление единиц принципов разработки для разных типов приложения, таких как приложения Windows и веб-приложения;

Предоставление среды выполнения кода, гарантирующей безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним разработчиком.

Одной из основных идей Microsoft .NET является совместимость программных частей, написанных на разных языках. Например, служба, написанная на [C++](https://it-black.ru/category/rubriki/samouchitel-po-%D1%81/) 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.

# 1.2 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" \o "Область видимости).

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" \o "Переменная типа) 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.

# 1.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) — аналог WinForms, система для построения клиентских приложений [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows) с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework) (начиная с версии [3.0](https://ru.wikipedia.org/wiki/.NET_Framework_3.0" \o ".NET Framework 3.0)), использующая язык [XAML](https://ru.wikipedia.org/wiki/XAML" \o "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" \o "Браузер) 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" \o "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" \o "DirectX), в отличие от [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms" \o "Windows Forms), где используется [GDI](https://ru.wikipedia.org/wiki/GDI" \o "GDI)/[GDI+](https://ru.wikipedia.org/wiki/GDI%2B" \o "GDI+). Производительность [WPF](https://ru.wikipedia.org/wiki/WPF" \o "WPF) выше, чем у [GDI+](https://ru.wikipedia.org/wiki/GDI%2B" \o "GDI+) за счёт использования аппаратного ускорения графики через [DirectX](https://ru.wikipedia.org/wiki/DirectX" \o "DirectX).

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

[XAML](https://ru.wikipedia.org/wiki/XAML" \o "XAML) представляет собой язык декларативного описания интерфейса, основанный на [XML](https://ru.wikipedia.org/wiki/XML" \o "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" \o "Элемент управления) 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)" \o "Кнопка (элемент интерфейса программ)) (кнопка).

Для работы с WPF требуется любой [.NET](https://ru.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework)-совместимый язык. В этот список входит множество языков: [C#](https://ru.wikipedia.org/wiki/C_Sharp" \o "C Sharp), [F#](https://ru.wikipedia.org/wiki/F_Sharp" \o "F Sharp), [VB.NET](https://ru.wikipedia.org/wiki/VB.NET" \o "VB.NET), [C++](https://ru.wikipedia.org/wiki/Visual_C%2B%2B" \o "Visual C++), [Ruby](https://ru.wikipedia.org/wiki/IronRuby" \o "IronRuby), [Python](https://ru.wikipedia.org/wiki/IronPython" \o "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)" \o "Oxygene (язык программирования)), [Lua](https://ru.wikipedia.org/w/index.php?title=Lua.NET&action=edit&redlink=1" \o "Lua.NET (страница отсутствует)) и многие другие. Для полноценной работы может быть использована как [Visual Studio](https://ru.wikipedia.org/wiki/Visual_Studio" \o "Visual Studio), так и [Expression Blend](https://ru.wikipedia.org/wiki/Expression_Blend" \o "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" \o "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" \o "Реляционная СУБД), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL" \o "Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase" \o "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" \o "Американский национальный институт стандартов)/[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" \o "Международная организация по стандартизации) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL" \o "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)— 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.

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».

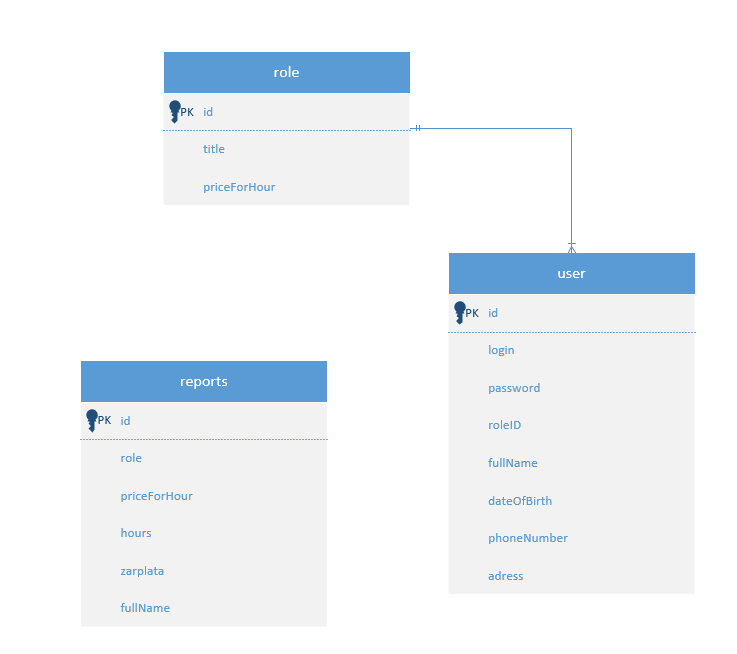


Рис. 2.1. ER Диаграмма

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

Проанализируем атрибуты сущностей. reports может иметь ФИО, должность, ставку в час, часы, зарплату; user может иметь логин, пароль, айди должности, ФИО, дата рождения, номер телефона, адрес. У каждой сущности есть ID.

# 2.2 Разработка базы данных

Для подключения базы данных в обозревателе объектов нужно подключиться к экземпляру компонента SQL Server Database Engine. Правой кнопкой мыши нажать на узел базы данных и выбрать команду создать базу данных. В поле новая базы данных ввести имя базы данных.

Для создания базы данных, приняв все значения по умолчанию, необходимо нажать на кнопку ОК, иначе необходимо сделать следующее:

Поменять имя владельца, нажав ( … ) и выбрать другого владельца.

Изменить параметры сортировки базы данных, выбрав страницу параметров и указав из списка желаемые параметры сортировки.

Изменить модель восстановления, выбрав страницу параметров и модель восстановления из списка.

Изменить параметры базы данных, выбрав страницу параметров и изменив параметры базы данных.

Добавить новую файловую группу, перейдя на страницу файловых групп. Нажать кнопку добавить и ввести значения для файловой группы.

Добавить расширенное свойство в базу данных, выбрав страницу расширенных свойств. В столбце имени ввести имя расширенного свойства. В столбце значений ввести текст расширенного свойства.

Чтобы создать базу данных, нажмите кнопку ОК.

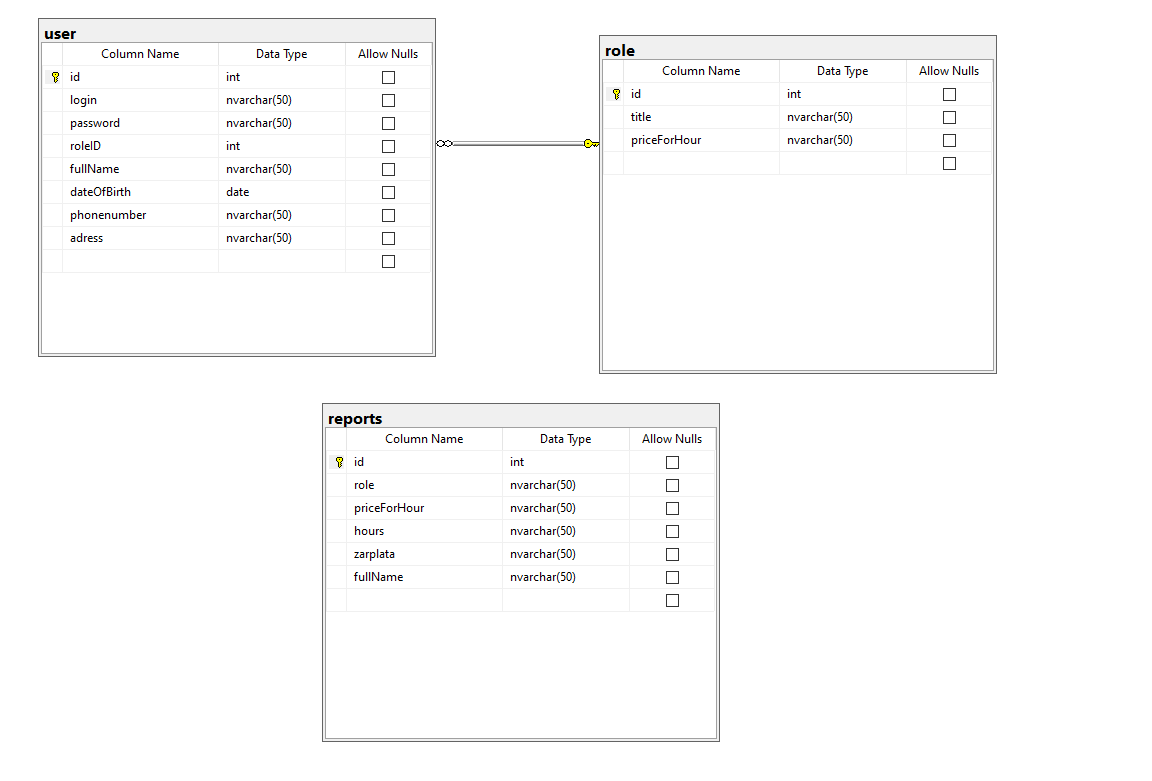


Рис. 2.2. База данных приложения

В базе данных имеется 3 таблицы, есть user и role – они отвечают за авторизацию в приложении, reports – для сохранения информации по отчётам зарплат.

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка интерфейса информационной системы

На рисунке 3.1 окно авторизации в программу. Предлагается авторизация через ввод логина, после чего проходит проверка на соответствие с базой данных, если логина нет в базе данных, ввод пароля будет недоступен. После успешного ввода логина и пароля производится проверка пользователя кодом подтверждения.

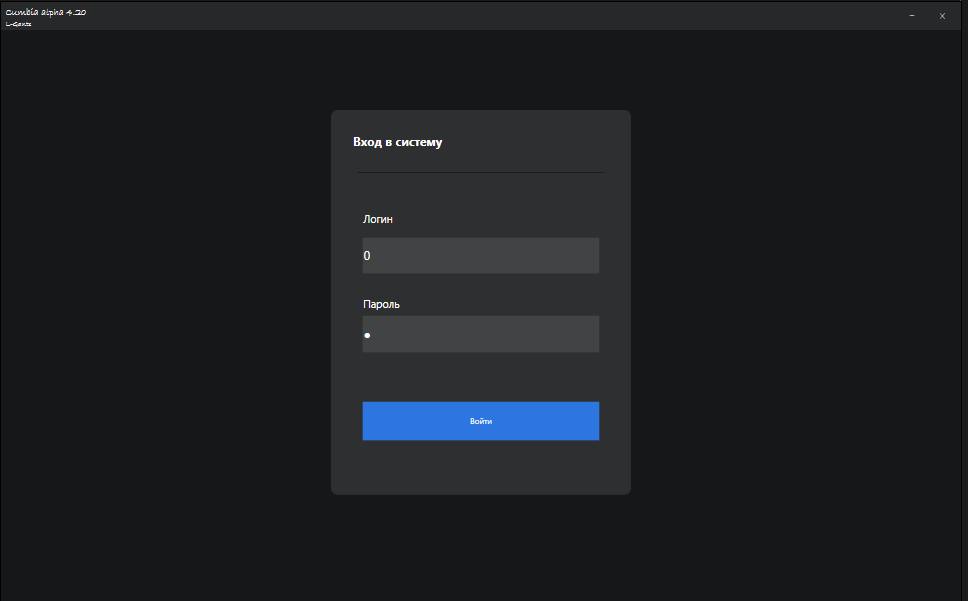


Рис. 3.1 Окно входа в систему

<Grid Background="#FF151718">

Скрытый Грид для окна кода подтверждения

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

<Image Height="NaN" Margin="223,310,977,318" Source="images/Rectangle 1.png" Stretch="Fill" Width="NaN" OpacityMask="#FF2D2F30"/>

<TextBlock HorizontalAlignment="Left" Margin="279,418,0,0" TextWrapping="Wrap" VerticalAlignment="Top" FontFamily="Arial" FontSize="14" Foreground="White"><Run Text="Код подт"/><Run Text="верждения"/></TextBlock>

<TextBlock HorizontalAlignment="Left" Margin="246,339,0,0" TextWrapping="Wrap" VerticalAlignment="Top" FontFamily="Arial" FontSize="14" Foreground="White"><Run Language="ru-ru" Text="Введите код для авторизации"/></TextBlock>

<TextBox MaxLength="4" Name="VVOD" HorizontalAlignment="Left" Margin="278,370,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="130" Background="#FF414344" Foreground="White" VerticalContentAlignment="Center" Height="34" FontFamily="Arial" FontSize="14" BorderBrush="{x:Null}"/>

<Button Content="Далее" HorizontalAlignment="Left" Margin="325,495,0,0" VerticalAlignment="Top" Background="#FF116ACC" Foreground="White" Height="35" Width="60" Click="Next\_Btn"/>

<TextBox HorizontalContentAlignment="Center" IsEnabled="False" Name="CODE" HorizontalAlignment="Left" Margin="259,439,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="168" Foreground="White" VerticalContentAlignment="Center" Height="54" FontSize="20" FontFamily="Arial" BorderBrush="#FF2D2F30" Background="{x:Null}"/>

<Button Content="" HorizontalAlignment="Left" Margin="290,498,0,0" VerticalAlignment="Top" Height="29" Width="26" BorderBrush="{x:Null}" Foreground="{x:Null}" Click="Button\_Click\_3">

<Button.Background>

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

</Button.Background>

</Button>

</Grid>

Авторизация

<Image Height="NaN" Margin="495,162,495,161" Source="images/Rectangle 1.png" Stretch="Fill" Width="NaN" OpacityMask="Black"/>

<Image Margin="0,0,0,858" Source="images/Title bar.png" Stretch="Fill" Name="TitleBar"/>

<TextBox Name="TxbLogin" VerticalContentAlignment="Center" HorizontalAlignment="Center" Margin="0,352,0,0" TextWrapping="Wrap" Text="0" VerticalAlignment="Top" Width="358" Height="56" Background="#FF414344" Foreground="White" SelectionBrush="{x:Null}" BorderBrush="{x:Null}" FontSize="18"/>

<Button Name="Auth\_Btn" Content="Войти" HorizontalAlignment="Center" Margin="0,598,0,0" VerticalAlignment="Top" Height="60" Width="358" BorderBrush="{x:Null}" Background="#FF2D76E2" RenderTransformOrigin="0.5,0.5" Foreground="White" Click="Button\_Click"/>

<Button Name="Auth\_Btn\_2" Visibility="Hidden" Content="Войти" HorizontalAlignment="Center" Margin="0,598,0,0" VerticalAlignment="Top" Height="60" Width="358" BorderBrush="{x:Null}" Background="#FF2D76E2" RenderTransformOrigin="0.5,0.5" Foreground="White" Click="Button\_Click\_22"/>

<PasswordBox Name="TxbPass" VerticalContentAlignment="Center" HorizontalAlignment="Center" Margin="0,470,0,0" Password="0" VerticalAlignment="Top" Width="358" Height="56" Background="#FF414344" Foreground="White" SelectionBrush="{x:Null}" BorderBrush="{x:Null}" FontSize="18"/>

<TextBlock HorizontalAlignment="Left" Margin="543,314,0,0" TextWrapping="Wrap" Text="Логин" VerticalAlignment="Top" Foreground="White" Height="24" Width="48" FontSize="16"/>

<TextBlock HorizontalAlignment="Left" Margin="543,442,0,0" TextWrapping="Wrap" Text="Пароль" VerticalAlignment="Top" Foreground="White" Height="24" Width="61" FontSize="16"/>

<TextBlock HorizontalAlignment="Left" Margin="528,197,0,0" TextWrapping="Wrap" Text="Вход в систему" VerticalAlignment="Top" Foreground="White" Height="24" Width="160" FontSize="18" FontWeight="Bold"/>

<Image Margin="535,255,535,644" Source="images/Line 1.png" Stretch="Fill"/>

<Button HorizontalAlignment="Left" Margin="1393,6,0,0" VerticalAlignment="Top" Width="35" Height="30" BorderBrush="{x:Null}" Click="Button\_Click\_1">

<Button.Background>

<ImageBrush ImageSource="images/Close.png"/>

</Button.Background>

</Button>

<Button HorizontalAlignment="Left" Margin="1349,6,0,0" VerticalAlignment="Top" Width="35" Height="30" BorderBrush="{x:Null}" Click="Button\_Click\_2">

<Button.Background>

<ImageBrush ImageSource="images/Minimize.png"/>

</Button.Background>

</Button>

<TextBlock HorizontalAlignment="Left" Margin="7,7,0,0" TextWrapping="Wrap" Text="Cumbia alpha 4.20" VerticalAlignment="Top" Foreground="#FFFCFCFC" FontFamily="Bradley Hand ITC" FontSize="15"/>

<TextBlock HorizontalAlignment="Left" Margin="7,27,0,0" TextWrapping="Wrap" Text="L-Gante" VerticalAlignment="Top" Foreground="#FFFCFCFC" FontFamily="Bradley Hand ITC" FontSize="11"/>

</Grid>

</Window>

На рисунке 3.2 представлено окно главной страницы. В этом окне пользователь может ознакомиться с новостями компании и перейти на другие окна.

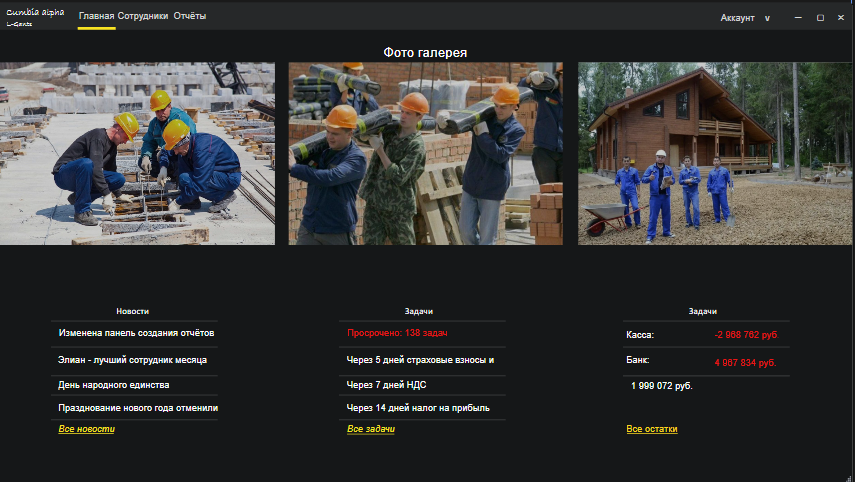


Рис. 3.2 Окно продаж

Разметка страницы, выплывающее окно для выхода из аккаунта и фотогалерея.

<Grid>

<Grid Background="#FF151718">

<Rectangle Fill="#FF262829" Margin="0,-4,0,0" Height="41" VerticalAlignment="Top"/>

<Grid HorizontalAlignment="Left" Height="34" VerticalAlignment="Top" Width="130">

<TextBlock HorizontalAlignment="Left" TextWrapping="Wrap" Text="Cumbia alpha 4.20" VerticalAlignment="Top" Foreground="#FFFCFCFC" FontFamily="Bradley Hand ITC" FontSize="15" Height="19" Width="120" Margin="11,3,0,0"/>

<TextBlock HorizontalAlignment="Left" Margin="11,22,0,0" TextWrapping="Wrap" Text="L-Gante" VerticalAlignment="Top" Foreground="#FFFCFCFC" FontFamily="Bradley Hand ITC" FontSize="11" Height="14" Width="39"/>

</Grid>

<Label Content="Фото галерея" HorizontalAlignment="Center" Margin="0,54,0,0" Foreground="White" FontFamily="Arial" FontSize="20" Width="136" Height="32" VerticalAlignment="Top"/>

<Grid Height="720" VerticalAlignment="Top">

<Grid Margin="0,17,0,0" Height="210" VerticalAlignment="Top" HorizontalAlignment="Right" Width="316" Name="aye" Visibility="Hidden">

<Image Source="images/Light.png"/>

<Grid Margin="45,41,214,108">

<Image Name="ava" HorizontalAlignment="Center" VerticalAlignment="Center" Source="images/Image1.png"/>

</Grid>

<Grid/>

<Grid>

<Label Content="Статус" HorizontalAlignment="Left" Margin="41,110,0,0" VerticalAlignment="Top" Foreground="White" FontSize="14"/>

<Label Content="Elian Pidor" HorizontalAlignment="Left" Margin="133,48,0,0" VerticalAlignment="Top" Foreground="White" FontSize="14" Height="32" Width="139"/>

<Label Content="Администратор" HorizontalAlignment="Left" Margin="158,110,0,0" VerticalAlignment="Top" Foreground="White" FontSize="14"/>

<Button HorizontalAlignment="Center" Margin="0,144,0,0" VerticalAlignment="Top" Width="232" Height="27" BorderBrush="{x:Null}" Click="Button\_Click\_Exit" Content="Выйти из аккаунта">

<Button.Background>

<ImageBrush ImageSource="images/Lightsad.png"/>

</Button.Background>

</Button>

</Grid>

</Grid>

<Button HorizontalAlignment="Right" Width="119" BorderBrush="{x:Null}" Click="Button\_Click\_Account" Content="Аккаунт v" Foreground="#FFD9D9D9" Background="{x:Null}" FontSize="14" FontFamily="Arial" Margin="0,0,101,0" Height="37" VerticalAlignment="Top"/>

<Button BorderBrush="{x:Null}" Click="Button\_Click\_1\_1" Content="×" Background="{x:Null}" Foreground="#FFD9D9D9" FontFamily="Arial" FontSize="20" HorizontalAlignment="Right" Width="34" Height="37" VerticalAlignment="Top"/>

<Button HorizontalAlignment="Right" Width="30" BorderBrush="{x:Null}" Click="Button\_Click\_2\_1" Content="−" Background="{x:Null}" Foreground="#FFD9D9D9" FontSize="20" FontFamily="Arial" FontWeight="Bold" Margin="0,0,66,0" Height="37" VerticalAlignment="Top"/>

<Button Content="▢" HorizontalAlignment="Right" Click="Button\_Click" Width="27" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="#FFD9D9D9" FontFamily="Arial" FontSize="14" Margin="0,0,34,0" Height="37" VerticalAlignment="Top"/>

<Button HorizontalAlignment="Left" Margin="118,-4,0,0" VerticalAlignment="Top" Width="57" Height="41" BorderBrush="{x:Null}" Click="Button\_Click\_Main" Content="Главная" Foreground="#FFD9D9D9" Background="{x:Null}" FontSize="14" FontFamily="Arial"/>

<Rectangle HorizontalAlignment="Left" Height="4" Margin="118,33,0,0" VerticalAlignment="Top" Width="57" Fill="#FFF9E423"/>

<Button HorizontalAlignment="Left" Margin="257,-4,0,0" VerticalAlignment="Top" Width="59" Height="41" BorderBrush="{x:Null}" Click="Button\_Click\_Reports" Content="Отчёты" Foreground="#FFD9D9D9" Background="{x:Null}" FontSize="14" FontFamily="Arial"/>

<Button Visibility="Hidden" HorizontalAlignment="Left" Margin="316,-4,0,0" VerticalAlignment="Top" Width="70" Height="41" BorderBrush="{x:Null}" Click="Button\_Click\_Payments" Content="Выплаты" Foreground="#FFD9D9D9" Background="{x:Null}" FontSize="14" FontFamily="Arial"/>

<Button HorizontalAlignment="Left" Margin="175,-4,0,0" VerticalAlignment="Top" Width="82" Height="41" BorderBrush="{x:Null}" Click="Button\_Click\_Staff" Content="Сотрудники" Foreground="#FFD9D9D9" Background="{x:Null}" FontSize="14" FontFamily="Arial"/>

</Grid>

На рисунке 3.3 представлено окно со списком сотрудников, которые есть в базе данных. В данном окне можно добавить, удалить, отредактировать данные по пользователю.

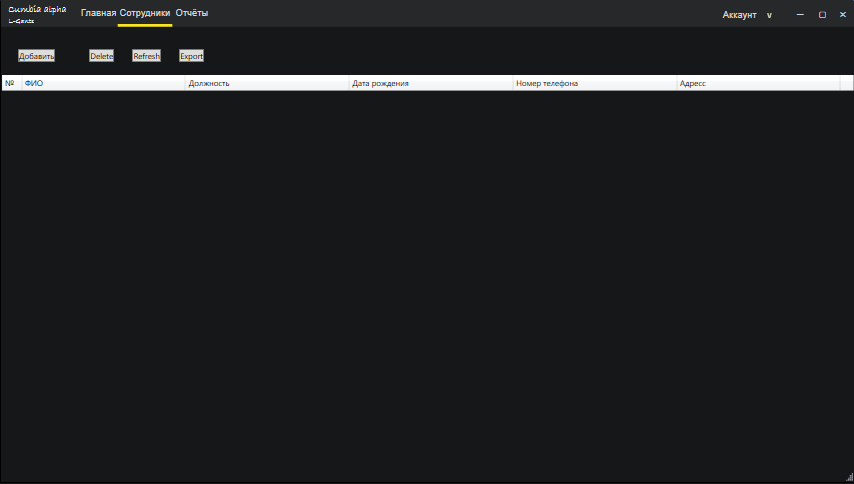


Рис. 3.3 Окно списка сотрудников

DataGrid для вывода данных о сотрудниках в виде таблицы.

<DataGrid x:Name="Users\_DataGrid" BorderBrush="{x:Null}" IsReadOnly="True" AutoGenerateColumns="False" Margin="0,108,0,0" Background="#FF151718">

<DataGrid.Columns>

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

<DataGridTextColumn Header="ФИО" Width="\*" Binding="{Binding fullName}"/>

<DataGridTextColumn Header="Должность" Width="\*" Binding="{Binding role.title}"/>

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

<DataGridTextColumn Header="Номер телефона" Width="\*" Binding="{Binding phonenumber}"/>

<DataGridTextColumn Header="Адресс" Width="\*" Binding="{Binding adress}"/>

<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 окно с отчётами, в котором отображаются данные из базы данных в DataGrid. Присутствует возможность добавления, редактирования, удаления данных из таблицы и также экспорт данных в отдельный документ.

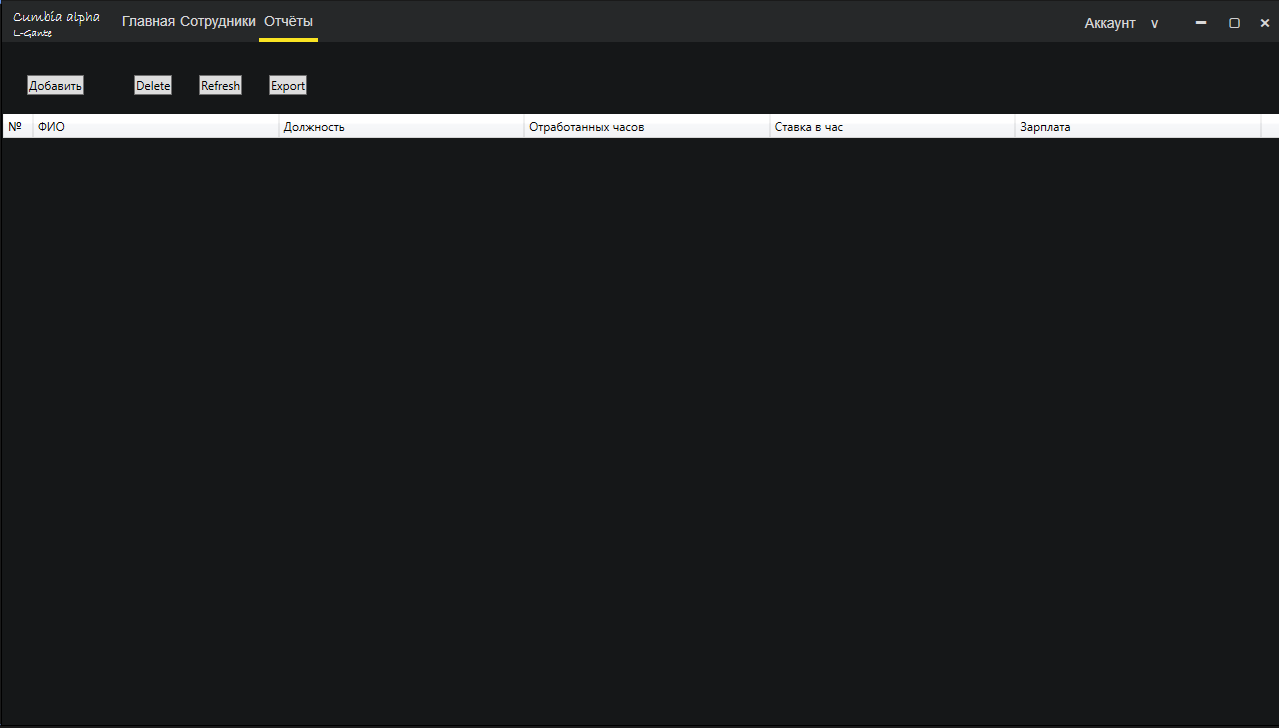


Рис. 3.4. Окно добавления данных

На рисунке 3.5 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.

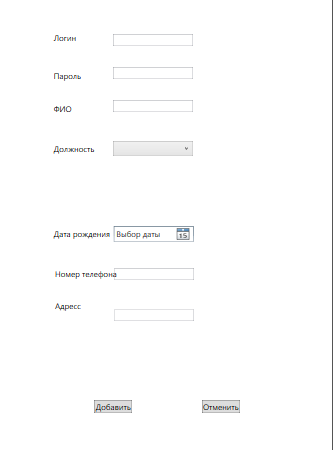


Рис. 3.5. Окно добавления данных

# 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 Button\_Click(object sender, RoutedEventArgs e)

{

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

if (CurrentUser != null)

{

TxbPass.IsEnabled = true;

Auth\_Btn.Visibility = Visibility.Hidden;

Auth\_Btn\_2.Visibility = Visibility.Visible;

TxbLogin.IsEnabled = false;

TxbPass.Focus();

}

else

{

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 Button\_Click\_22(object sender, RoutedEventArgs e)

{

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

if (CurrentUser1 != null)

{

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

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

TxbLogin.IsEnabled = false;

TxbPass.IsEnabled = false;

Auth\_Btn\_2.IsEnabled = false;

VVOD.Focus();

Globals.UserRoles = CurrentUser1.roleID;

Globals.userinfo = CurrentUser1;

while (true)

{

Random x = new Random();

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

CODE.Text = a.ToString();

await Task.Delay(10000);

}

}

else

{

MessageBox.Show("Пароль не верен!");

TxbPass.Clear();

TxbPass.IsEnabled = false;

TxbLogin.IsEnabled = true;

TxbLogin.Clear();

Auth\_Btn\_2.Visibility = Visibility.Hidden;

Auth\_Btn.Visibility = Visibility.Visible;

}

}

Код окна с отчётами.

В данном классе идёт переход на главную страницу.

private void Button\_Click\_Main(object sender, RoutedEventArgs e)

{

var Main\_Window = new Main();

Main\_Window.Show();

this.Close();

}

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 = Users\_DataGrid.SelectedItem as user;

AppData.db.user.Remove(CurrentCar);

AppData.db.SaveChanges();

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

Users\_DataGrid.ItemsSource = AppData.db.user.ToList();

}

}

Класс для кнопки обновления таблицы.

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

{

Users\_DataGrid.ItemsSource = AppData.db.user.ToList();

}

Класс для экпорта данных в Word по нажатию кнопки.

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

{

var allRequest = PROGADB1Entities1.GetContext().user.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 пользователях";

userRange.InsertParagraphAfter();

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

Word.Range tableRange = tableParagraph.Range;

Word.Table paymentsTable = document.Tables.Add(tableRange, allRequest.Count() + 1, 6);

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 = "Должность";

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

cellRange.Text = "ФИО";

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

cellRange.Text = "Номер телефона";

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

cellRange.Text = "Адресс проживания";

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

cellRange.Text = "Дата рождения";

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.role.title);

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

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

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

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

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

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

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

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

}

application.Visible = true;

}

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

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

{

var CurrentCar = UsersCmb.SelectedItem as user;

var CurrentRole = postCmb.SelectedItem as role;

var CurrentHrs = priceCmb.SelectedItem as reports;

if (\_currentreports.id >= 0)

PROGADB1Entities1.GetContext().reports.AddOrUpdate(\_currentreports);

try

{

PROGADB1Entities1.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена!");

Close();

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

# ЗАКЛЮЧЕНИЕ

При конструировании автоматизированной информационной системы «Учёт сдельной оплаты труда» была разработана диаграмма Entity Relationship, была спроектирована база данных в приложении Microsoft SQL Server Management Studio. Разработан интерфейс и функционал системы с помощью C#WPF.

Данная система упростит работу с подсчётом заработных плат сотрудников и упростит бумажную волокиту, генерируя отчёты по зарплатам в одно нажатие.

Экономический эффект от внедрения автоматизированной информационной системы «Учёт сдельной оплаты труда» ожидается за счет сокращения времени на подсчёт и заполнение отчётов по каждому сотрудников, за счёт автоматизации этого процесса программой. 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/AsulinShatem

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. WPF4. Подробное руководство Перевод А.Слинкин Главный редактор А. Галунов Зав. редакцией Н. Макарова Редактор Е. Тульсанова Корректоры С. Минин, О. Макарова Верстка Д. Орлова Натан А. WPF 4. Подробное руководство. - Пер. с англ. - СПб.: Символ-Плюс, 2011. - 880 с., ил.
2. Грофф, Джеймс Р., Вайнберг, Пол Н., Оппелъ, Эндрю Дж. Г89 SQL: полное руководство, 3-е изд.: Пер. с англ. - М.: ООО "И.Д. Вильяме", 2015. - 960 с.: ил. - Парал. тит. англ.
3. Долженко, А. И., Глушенко, С. А. Разработка и сопровождение программных систем. Технологии Microsoft.NET для разработки приложений: Лабораторный практикум – Ростов-на-Дону: издательство РГЭУ (РИНХ), 2019. – 133 с.
4. Карпова И.П. Базы данных. Учебное пособие. – Московский государственный институт электроники и математики (Технический университет). – М., 2009.
5. Пахомов Б. И. П12 C# для начинающих. — СПб.: БХВ-Петербург, 2014. — 432 с.: ил.

**Интернет-ресурсы**

1. .NET Framework – Режим доступа: <https://ru.wikipedia.org/wiki/.NET_Framework>
2. C Sharp – Режим доступа:<https://ru.wikipedia.org/wiki/C_Sharp>.
3. Entity Framework: как быстрее написать код для работы с базой данных – Режим доступа: <https://skillbox.ru/media/code/entity_framework/>.
4. Microsoft SQL Server – Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_SQL_Server>.
5. SQL Server Management Studio – Режим доступа: <https://ru.wikipedia.org/wiki/SQL_Server_Management_Studio>.
6. Документация по C# – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/>.
7. Обзор Entity Framework – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/ef/overview>.
8. Обзор платформы .NET – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/framework/get-started/overview>.
9. Обзор функционала SQL Server Management Studio (SSMS) – Режим доступа: <https://info-comp.ru/ssms-feature-overview?ysclid=lenvbogeq8366784796.>
10. Общие сведения о WPF – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8>.
11. Руководство по WPF – Режим доступа: <https://metanit.com/sharp/wpf/?ysclid=lenvi5n1bh823160038>.
12. Руководство по классическим приложениям (WPF .NET) – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0.
13. Что такое MS SQL Server и чем она отличается от других СУБД – Режим доступа: <https://timeweb.com/ru/community/articles/osnovy-raboty-s-mssqlserver>.
14. 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>.