### 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 сотрудников на предприятии»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Лавров С.В./

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**ПРОВЕРИЛ**

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

(подпись) (Ф.И.О.)

Воркута

2023

Оглавление

[МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ 1](#_Toc133346480)

[КУРСОВОЙ ПРОЕКТ 1](#_Toc133346481)

[ВВЕДЕНИЕ 3](#_Toc133346482)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133346483)

[1.1 Платформа .NET 4](#_Toc133346484)

[1.2 Язык программирования С# 6](#_Toc133346485)

[1.3 Windows Presentation Foundation (WPF) 9](#_Toc133346486)

[1.4 СУБД SQL server 12](#_Toc133346487)

[1.5 Microsoft SQL Server Management Studio 13](#_Toc133346488)

[1.6 Entity Framework 15](#_Toc133346489)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 16](#_Toc133346490)

[2.1 Разработка диаграммы ERD 16](#_Toc133346491)

[2.2 Разработка базы данных 17](#_Toc133346492)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 20](#_Toc133346493)

[3.1 Разработка интерфейса информационной системы 20](#_Toc133346494)

[3.2 Программирование информационной системы 24](#_Toc133346495)

[ЗАКЛЮЧЕНИЕ 28](#_Toc133346496)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 29](#_Toc133346497)

# ВВЕДЕНИЕ

Каждая организация, независимо от ее размеров, имеет отдел кадров, который, в свою очередь, может быть представлен одним, двумя или большим количеством специалистов. Отдел кадров является тем звеном, которое соединяет рядовых сотрудников и вышестоящих руководителей. Правильная организация работы отдела кадров является важным моментом в деятельности всего предприятия, так как зависимость всех работников от вовремя выполненного перевода, полученного приказа об увольнении или принятии на работу, правильно посчитанного стажа очень велика. Причем эта зависимость является как материальной, так и моральной. Отдел кадров, являясь точкой, с которой все начинается, вынужден работать с большим количеством людей. Это, в свою очередь приводит к тому, что происходит документооборот, являющийся источником интенсивных информационных потоков.

У специалистов хранится огромное число приказов, личных дел, анкет и других разного рода документов, которые являются необходимыми при такой работе. Возможность выполнения и хранения всех этих документов в электронном виде является важной и актуальной задачей. Все документы являются результатами выполнения каких – либо кадровых операций (или же исходными данными для них). Таким образом, у специалиста появиться возможность выполнения всех кадровых операций в электронном виде, что эффективнее, быстрее и удобнее, чем в бумажном.

**Объект:** система «Учёт сотрудников на предприятии».

**Предмет:** автоматизация бизнес процеса учёт сотрудников на предприятии.

**Цель:** разработка системы «Учёт сотрудников на предприятии».

**Задачи:**

* выбрать инструментарий;
* спроектировать базу данных;
* разработать прототип бизнес процеса «Учёт сотрудников на предприятии».

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# Платформа .NET

Microsoft .NET Framework – программная технология, предназначенная для создания распределенных программ и веб-приложений, и основанная на кроссплатформенных взаимодействиях реализуемых между различными существующими архитектурами программирования. Считается, что платформа .NET Framework является ответом компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) на набравшую к тому времени большую популярность платформу [Java](https://ru.wikipedia.org/wiki/Java_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0)) компании [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) (ныне принадлежит [Oracle](https://ru.wikipedia.org/wiki/Oracle)).

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

Платформа .NET решает многие проблемы, которые досаждали программистам в прошлом. К их числу относятся проблемы, связанные с развертыванием приложений, управлением версиями, утечкой памяти, а также проблемы безопасности. Платформа .NET позволяет разрабатывать мощные, независимые от языка программирования, настольные приложения и масштабируемые (расширяемые) Web-службы, основанные на открытых стандартах Интернет и построенные на базе новой мощной полнофункциональной библиотеки классов .NET Framework. Разработчикам программного обеспечения технология .NET предоставляет новую программную платформу и великолепные инструментальные средства разработки, в которых основную роль играет язык XML (eXtensible Markup Language – расширяемый язык разметки).

Несомненно, на развитие технологии .NET оказали влияние другие технологические достижения, в частности расширяемый язык разметки XML, платформа Java™, а также модель компонентных объектов Microsoft (Component Object Model – COM).

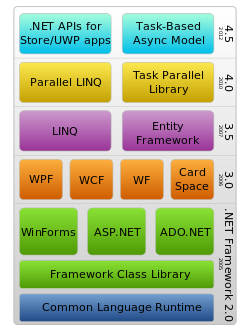
[](https://commons.wikimedia.org/wiki/File:DotNet.svg?uselang=ru)

Рисунок 1.1 (Технологии .NET Framework)

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

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

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

Среды разработки, поддерживающие .NET:

* [Microsoft Visual Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) (C#, Visual Basic .NET, Managed C++, F#)
* [SharpDevelop](https://ru.wikipedia.org/wiki/SharpDevelop)
* [MonoDevelop](https://ru.wikipedia.org/wiki/MonoDevelop)
* [Embarcadero RAD Studio](https://ru.wikipedia.org/wiki/Embarcadero_RAD_Studio) (Delphi for .NET); ранее [Borland Developer Studio](https://ru.wikipedia.org/wiki/Borland_Developer_Studio) (Delphi for .NET, C#)
* [A#](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))
* [Zonnon](https://ru.wikipedia.org/wiki/Zonnon)
* [PascalABC.NET](https://ru.wikipedia.org/wiki/PascalABC.NET)
* [JetBrains Rider](https://ru.wikipedia.org/wiki/JetBrains_Rider)

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

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

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

* [C#](https://ru.wikipedia.org/wiki/C_Sharp)
* [Visual Basic .NET](https://ru.wikipedia.org/wiki/Visual_Basic_.NET)
* [JScript .NET](https://ru.wikipedia.org/wiki/JScript_.NET)
* [C++/CLI](https://ru.wikipedia.org/wiki/C%2B%2B/CLI) — новая версия [Managed C++](https://ru.wikipedia.org/wiki/Managed_C_Plus_Plus)
* [F#](https://ru.wikipedia.org/wiki/F_Sharp) — член семейства языков программирования [ML](https://ru.wikipedia.org/wiki/ML), включён в VS2010/VS2012/ VS2015/VS2017/VS2019/VS2022
* [J#](https://ru.wikipedia.org/wiki/Visual_J_Sharp) — последний раз был включён в VS200

# Язык программирования С#

C# — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в.NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript. 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.

Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения. Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда-выражения поддерживают приемы функционального программирования. Синтаксис LINQ создает общий шаблон для работы с данными из любого источника. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем. В C# имеется Единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

C# подчеркивает Управление версиями, чтобы обеспечить совместимость программ и библиотек с течением времени. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

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.

Обеспечение взаимодействия между языками является ключевой особенностью .NET. Код IL, созданный компилятором C#, соответствует спецификации общих типов (CTS). Код IL, созданный из кода на C#, может взаимодействовать с кодом, созданным из версий .NET для языков F#, Visual Basic, C++. Существует более 20 других языков, совместимых с CTS. Одна сборка может содержать несколько модулей, написанных на разных языках .NET, и все типы могут ссылаться друг на друга, как если бы они были написаны на одном языке.

В дополнение к службам времени выполнения .NET также включает расширенные библиотеки. Эти библиотеки поддерживают множество различных рабочих нагрузок. Они упорядочены по пространствам имен, которые предоставляют разные полезные возможности: от операций файлового ввода и вывода до управления строками и синтаксического анализа XML, от платформ веб-приложений до элементов управления Windows Forms. Обычно приложение C# активно используют библиотеку классов .NET для решения типовых задач.

Для первого знакомства с языком программирования традиционно используется программа "Hello, World". Вот ее пример на C#:

C#

using System;

class Hello

{

static void Main()

{

Console.WriteLine("Hello, World");

}

}

Программа "Hello, World" начинается с директивы using, которая ссылается на пространство имен System. Пространства имен позволяют иерархически упорядочивать программы и библиотеки C#. Пространства имен содержат типы и другие пространства имен. Например, пространство имен System содержит несколько типов (в том числе используемый в нашей программе класс Console) и несколько других пространств имен, таких как IO и Collections. Директива using, которая ссылается на пространство имен, позволяет использовать типы из этого пространства имен без указания полного имени. Благодаря директиве using в коде программы можно использовать сокращенное имя Console.WriteLine вместо полного варианта System.Console.WriteLine.

Класс Hello, объявленный в программе "Hello, World", имеет только один член — это метод с именем Main. Метод Main объявлен с модификатором static. Методы экземпляра могут ссылаться на конкретный экземпляр объекта, используя ключевое слово this, а статические методы работают без ссылки на конкретный объект. По стандартному соглашению точкой входа программы C# является статический метод с именем Main.

Выходные данные программы создаются в методе WriteLine класса Console из пространства имен System. Этот класс предоставляется библиотеками стандартных классов, ссылки на которые компилятор по умолчанию добавляет автоматически.

# Windows Presentation Foundation (WPF)

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

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

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.

* Создание экземпляров классов
* Настройка свойств
* Вызов методов
* 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. Такое разделение внешнего вида и поведения имеет ряд преимуществ.

* Затраты на разработку и обслуживание снижаются, так как разметка, определяющая внешний вид, не связана тесно с кодом, обуславливающим поведение.
* Повышается эффективность разработки, так как дизайнеры, занимающиеся внешним видом приложения, могут работать параллельно с разработчиками, реализующими поведение приложения.
* Глобализация и локализация приложений WPF упрощена.

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.

В приведенном ниже примере XAML используется для определения внешнего вида окна, содержащего одну кнопку.

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

Title="Window with Button"

Width="250" Height="100">

<!-- Add button to window -->

<Button Name="button">Click Me!</Button>

</Window>

Этот код XAML определяет окно и кнопку с помощью элементов Window и Button. Каждый элемент настраивается с помощью атрибутов, например атрибута Window элемента Title , определяющего текст заголовка окна. Во время выполнения WPF преобразует элементы и атрибуты, определенные в разметке, в экземпляры классов WPF. Например, элемент Window преобразуется в экземпляр класса Window , свойство Title которого является значением атрибута Title.

Результат XAML кода:



Рисунок 1.2 Пример кнопки

using System.Windows;

namespace SDKSample

{

public partial class AWindow : Window

{

public AWindow()

{

// InitializeComponent call is required to merge the UI

// that is defined in markup with this class, including

// setting properties and registering event handlers

InitializeComponent();

}

void button\_Click(object sender, RoutedEventArgs e)

{

// Show message box when button is clicked.

MessageBox.Show("Hello, Windows Presentation Foundation!");

}

Метод InitializeComponent вызывается из конструктора класса кода программной части для слияния пользовательского интерфейса, определенного в разметке, с классом кода программной части. (InitializeComponent создается при построении приложения, поэтому реализовывать его вручную не нужно.) Сочетание x:Class и InitializeComponent гарантирует правильную инициализацию реализации при создании.

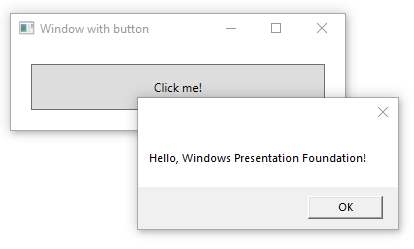


Рисунок 1.3 Пример Кнопки с всплывающим окном

# СУБД SQL server

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

Программное обеспечение MySQL - это ПО с открытым кодом.

ПО с открытым кодом означает, что применять и модифицировать его может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно. При этом каждый пользователь может изучить исходный код и изменить его в соответствии со своими потребностями. Использование программного обеспечения MySQL регламентируется лицензией GPL (GNU General Public License),  в которой указано, что можно и чего нельзя делать с этим программным обеспечением в различных ситуациях. Если работа в рамках GPL вас не устраивает или планируется встраивание MySQL-кода в коммерческое приложение, есть возможность купить коммерческую лицензированную версию у компании MySQL AB.

В каких случаях следует отдавать предпочтение СУБД MySQL?

MySQL является очень быстрым, надежным и легким в использовании. Если вам требуются именно эти качества, попробуйте поработать с данным сервером. MySQL обладает также рядом удобных возможностей, разработанных в тесном контакте с пользователями. Сравнительные характеристики MySQL и других средств управления базами данных приведены на нашей странице тестов производительности. Первоначально сервер MySQL разрабатывался для управления большими базами данных с целью обеспечить более высокую скорость работы по сравнению с существующими на тот момент аналогами. И вот уже в течение нескольких лет данный сервер успешно используется в условиях промышленной эксплуатации с высокими требованиями. Несмотря на то что MySQL постоянно совершенствуется, он уже сегодня обеспечивает широкий спектр полезных функций. Благодаря своей доступности, скорости и безопасности MySQL очень хорошо подходит для доступа к базам данных по Internet.

Технические возможности СУБД MySQL

 ПО MySQL  является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API). Мы также поставляем сервер MySQL в виде многопоточной библиотеки, которую можно подключить к пользовательскому приложению и получить компактный, более быстрый и легкий в управлении продукт.

Доступно также большое количество программного обеспечения MySQL,

разработанного сторонними разработчиками. Вполне возможно, что СУБД MySQL уже поддерживается вашим любимым приложением или языком.

# Microsoft SQL Server Management Studio

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.

Также есть SQL Server Management Studio Express для Express версии сервера, которая является бесплатной. Однако в ней нет поддержки ряда компонентов (Analysis Services, Integration Services, Notification Services, Reporting Services) и SQL Server 2005 Mobile Edition.

Начиная с версии 16.5.3 пакет SSMS выделен в отдельный обновляемый продукт, доступный для скачивания на сайте Microsoft. Текущая доступная версия SSMS 18.4 (15.0.18206.0) (general availability) поддерживает MS SQL server начиная с версии 2008 по 2019.

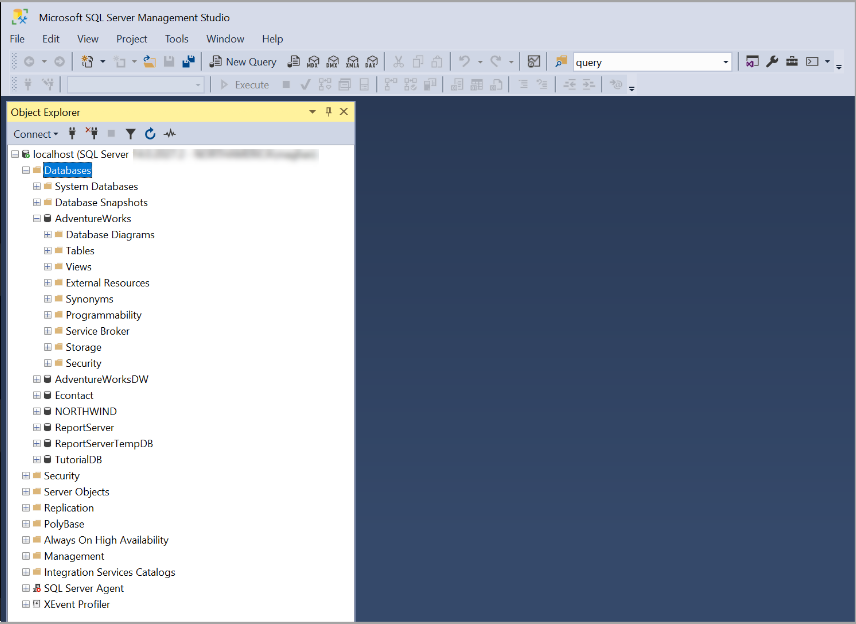


Рисунок 1.2 SQL Server Management Studio

Компоненты среды SQL Server Management Studio:

**Обозреватель объектов.**

Обозреватель объектов используется для просмотра всех объектов и управления ими в одном экземпляре SQL Server (или более).

**Обозреватель шаблонов.**

Template Explorer используется для создания файлов со стандартным текстом, которые можно использовать для ускорения разработки запросов и скриптов, и управления ими.

**Обозреватель решений.**

Устаревший обозреватель решений используется для создания проектов, применяемых для управления такими элементами администрирования, как скрипты и запросы.

**Visual Database Tools.**

Средства визуального проектирования, входящие в Среда Management Studio, используются для создания запросов, таблиц и баз данных схем.

**Редакторы запросов и текста.**

Редакторы языков Среда Management Studio используются для интерактивного написания и отладки запросов и скриптов.

# Entity Framework

Entity Framework представляет ORM-технологию (object-relational mapping - отображения данных на реальные объекты) от компании Microsoft для доступа к данным. Entity Framework Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными как с объектами классом независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Как технология доступа к данным Entity Framework Core работает поверх платформы .NET и поэтому может использоваться на различных платформах стека .NET. Это и стандартные платформы типа Windows Forms, консольные приложения, WPF, UWP и ASP.NET Core. При этом кроссплатформенная природа EF Core позволяет задействовать ее не только на ОС Windows, но и на Linux и Mac OS X.

Поскольку Entity Framework Core работает на основе платформы .NET, то он развивается вместе с данной платформой. Текущая версия EF Core - 7.0 была выпущена в ноябре 2022 года вместе с .NET 7. И технология продолжает развиваться.

Entity Framework Core поддерживает множество различных систем баз данных. Таким образом, мы можем через EF Core работать с любой СУБД, если для нее имеется нужный провайдер. По умолчанию на данный момент Microsoft предоставляет ряд встроенных провайдеров: для работы с MS SQL Server, для SQLite, для PostgreSQL. Также имеются провайдеры от сторонних поставщиков, например, для MySQL.

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2.1 Разработка диаграммы ERD

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

Каждая сущность должна иметь:

■ уникальное имя;

■ один или несколько атрибутов, которые либо принадлежат сущности, либо наследуются через связь;

■ один или несколько атрибутов, которые однозначно идентифицируют каждый экземпляр сущьности

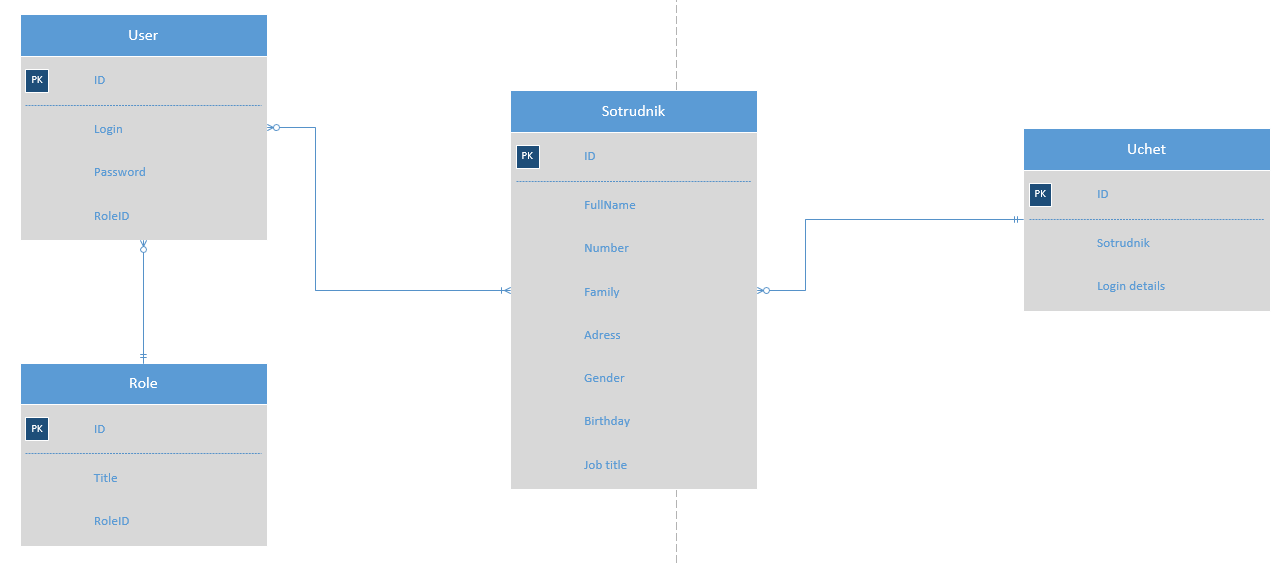
****Основными сущностями являются: Пользователь, роль, сотрудник и учёт (ИС). У каждой сущности есть атрибуты. У сущности «Пользователь» есть логин, пароль и роль. У сущности «роль» имеется заголовок и роль. У Сущности «Сотрудник» есть ФИО, номер телефона, семейный статус, адрес проживания, пол, дата рождения и должность. У сущности «Учёт» имеются все данные о сотруднике (регистрационные данные и вся ифнормация). Сущности Пользователь и Роль регистрируют сотрудника, сущность Сотрудники хранит в себе информацию о сотруднике, а в сущности Учёт хранятся все данные.

Рисунок 2.1 ER диаграмма

# 2.2 Разработка базы данных

Что бы создать базу данных, нужно запустить “Microsoft SQL Server Management Studio” и выполнить подключение к серверу.

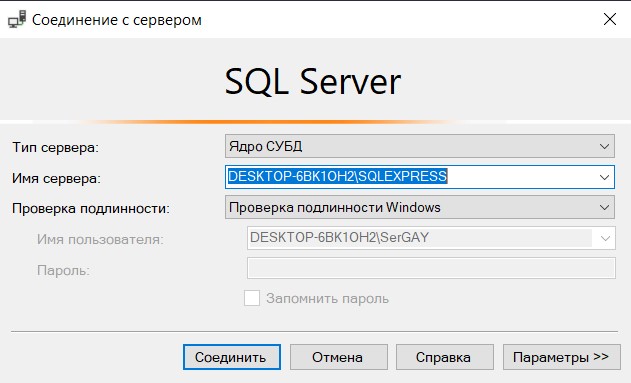


Рисунок 2.2 Подключение к серверу

Следующим шагом будет нажатие “Обозреватель объектов” на панели в левой части экрана, нажимаем правой кнопкой мыши по “Базы данных” и выбираем “Создать базу данных”.

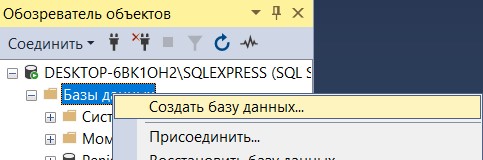
****

Рисунок 2.3 Создание БД

В обозревателе объектов появилась созданная нами база данных, открываем её и нажимаем на “Диаграммы баз данных” и выбираем “Создать диаграмму базы данных”.

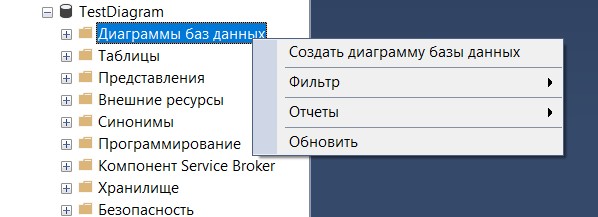


Рисунок 2.4 Создание диаграммы

Для того что бы создавать таблицы нашей базы данных, нужно нажать правой кнопкой мыши по пустому месту нашей диаграммы и выбрать пункт “Создать таблицу” и вводим название нашей таблицы.

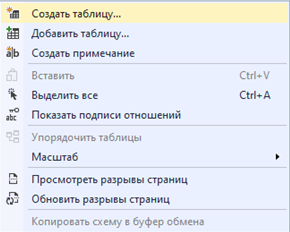
****

Рисунок 2.5 Создание таблиц

Таблица создалась, в которую мы можем писать атрибуты и уникальные идентификаторы. Вводим атрибуты и присваиваем им типы данных.

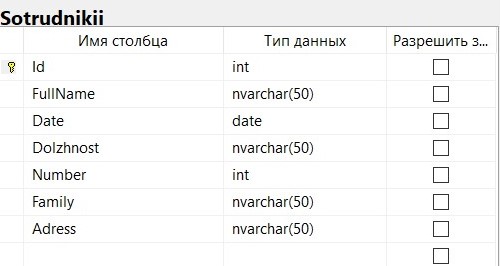


Рисунок 2.6 Типы данных и атрибуты

После создания и заполнения таблиц, соединяем их связями. Для этого зажимаем ключик и ведём к нужному полю в другой таблице.

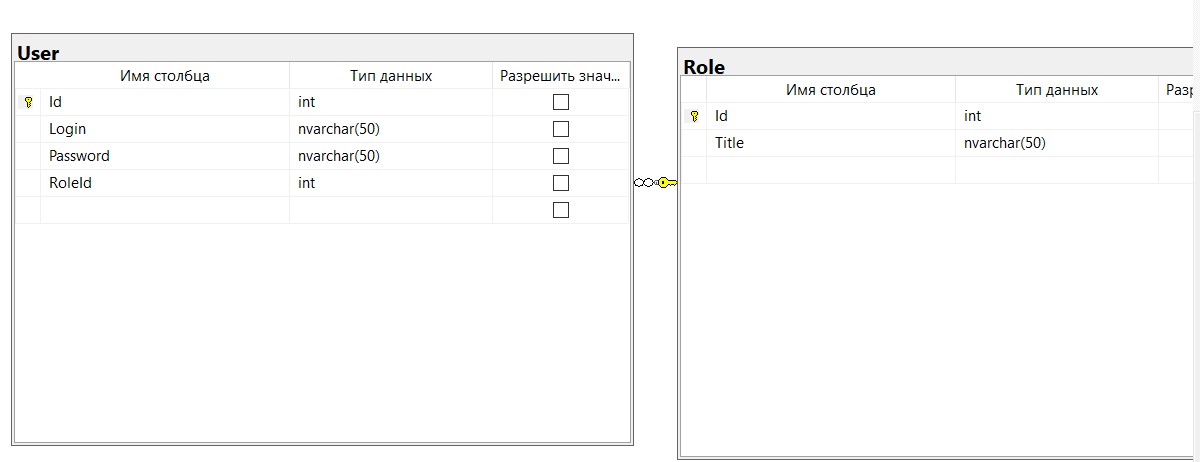


Рисунок 2.7 Связь

Последним шагом будет сохранить диаграмму и выбрать во вкладке “таблицы” табличку и нажать 200 rows.

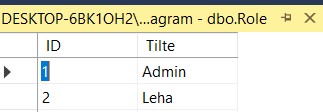


Рисунок 2.8 Заполнение таблиц

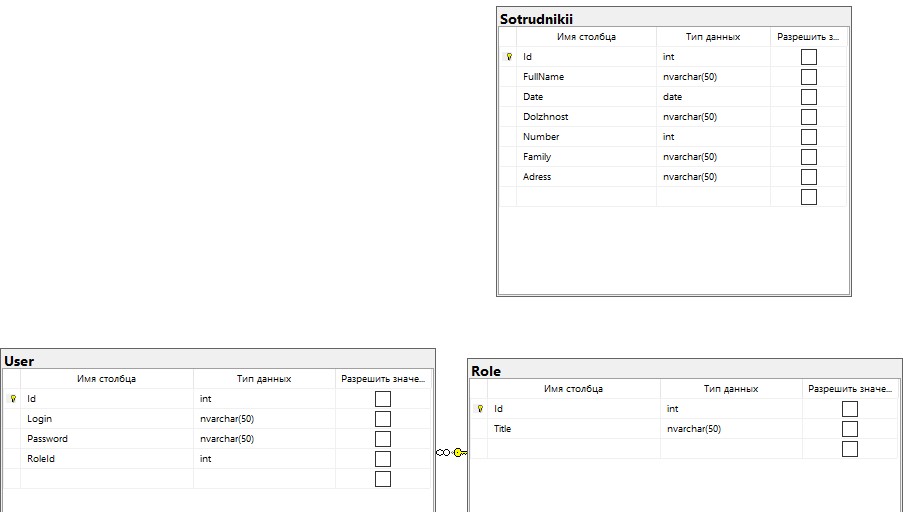
Финальная база данных ИС выглядит следующим образом:

Рисунок 2.9 Финальная БД

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка интерфейса информационной системы

Ниже представлена авторизация с проверкой логина, после которой идёт проверка пароля и в завершении всплывает капча с рандомно генерируемыми числами с возможностью обновить эти числа и таймером в 10 секунд, после чего происходит переход на следующее окно. Так же в авторизации происходит разграничение прав доступа (администратор или пользователь) путём проверки логина и пароля.

Алгоритм авторизации:

* сотрудник вводит логин и пароль;
* при вводе пароля сотрудником и нажатии кнопки Войти на служебный телефон отправляется СМС с единоразовым кодом доступа;
* сотрудник вводит код и далее получает доступ к необходимому функционалу.

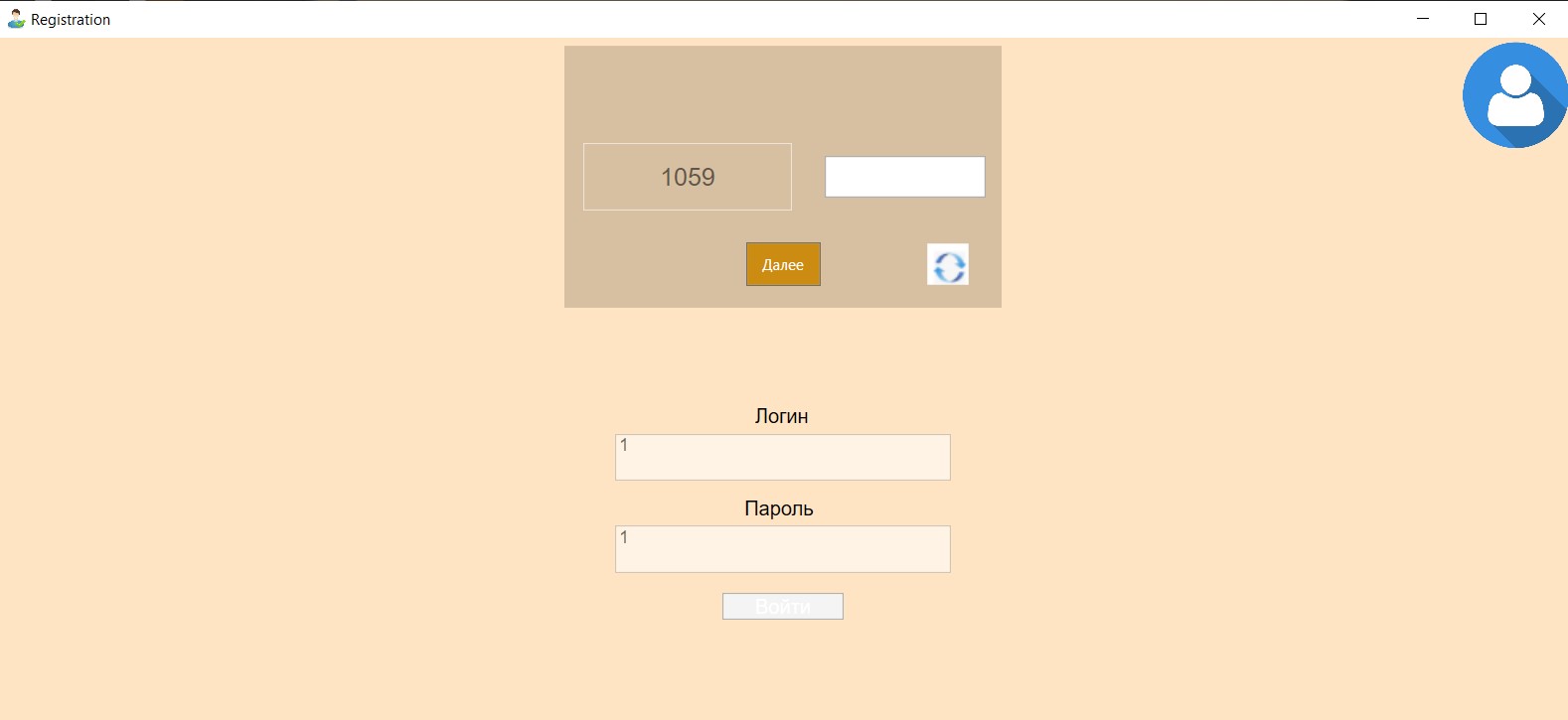


Рисунок 3.1 Авторизация с проверкой логина, пароля и капчи

Данная часть кода отвечает за Введение логина и пароля, в случае правильно введённых данных, появится капча с рандомными числами для входа. При этом если входные данные введены не верно, то капча не появится и высветится ошибка. Основная разметка <Grid> для капчи, и <button> для перехода на следующие окно.

<Grid Background="Bisque">

<Grid.RowDefinitions>

<RowDefinition Height="21\*"/>

<RowDefinition Height="35\*"/>

<RowDefinition Height="8\*"/>

</Grid.RowDefinitions>

<Grid Visibility="Hidden" x:Name="kapt" >//Делаем капчу невидимой если не пройдут проверки логина и пароля

<Rectangle HorizontalAlignment="Center" Height="211" VerticalAlignment="Center" Width="352" Fill="#FFD7BFA2"/>

<TextBlock HorizontalAlignment="Center" Margin="0,72,0,0" Grid.Row="1" TextWrapping="Wrap" Text="Логин" VerticalAlignment="Top" FontSize="16" FontFamily="Arial" Height="18" Width="45"/>

<TextBlock HorizontalAlignment="Center" Margin="0,146,0,0" Grid.Row="1" TextWrapping="Wrap" Text="Пароль" VerticalAlignment="Top" FontFamily="Arial" FontSize="16" Height="18" Width="62"/>

<TextBox x:Name="TxbLogin" HorizontalAlignment="Center" Margin="0,95,0,0" Grid.Row="1" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="120" Height="38" FontFamily="Arial" FontSize="14" />

<TextBox x:Name="TxbPassword" HorizontalAlignment="Center" Grid.Row="1" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="120" Height="38" Margin="0,169,0,0" FontFamily="Arial" FontSize="14"/>

<Button Name="Autr" Content="Войти" HorizontalAlignment="Center" Margin="0,212,0,0" Grid.Row="1" VerticalAlignment="Top" Click="BtnSignIn\_Click" Width="98" FontSize="16" FontFamily="Arial" Height="22" Background="#FFCC8B11" Foreground="White"/>

<Button Visibility="Hidden" x:Name="Autr\_2" Content="Войти" HorizontalAlignment="Center" Margin="0,223,0,0" Grid.Row="1" VerticalAlignment="Top" Click="BtnSignIn\_Click\_2" Width="98" FontSize="16" FontFamily="Arial" Height="22" Background="#FFCC8B11" Foreground="White"/>

<Image Margin="1180,0,0,131" Source="/znachok.png"/>

</Grid>

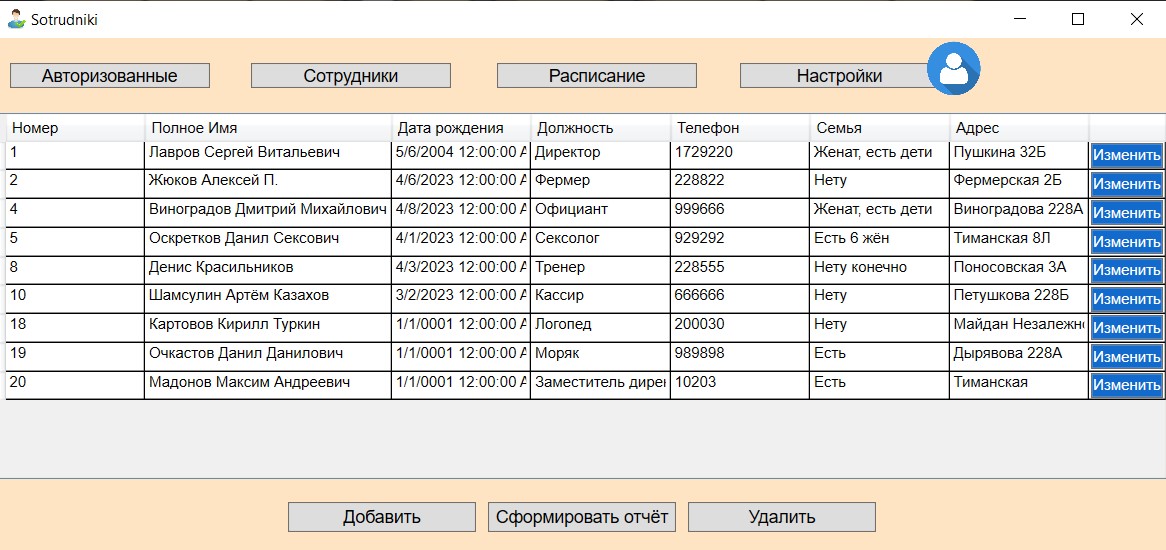
Следующие окно Сотрудники, предназначено для хранения информации о сотрудниках, со всеми их данными (Номер, полное имя, дата рождения, должность, телефон, семья и адрес). Так же есть возможность добавить нового сотрудника, удалить данные о сотруднике и измененить данные. Так же возможно формирование отчётов в Word документ.

Рисунок 3.2 Сотрудники

<DataGrid FontFamily="Arial"

Grid.Row="1"

AutoGenerateColumns="False"

x:Name="UsersGrid2" IsReadOnly="True">

<DataGrid.Columns>

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

<DataGridTextColumn Header="Полное Имя" Width="Auto" Binding="{Binding FullName}"/>

<DataGridTextColumn Header="Дата рождения" Width="\*" Binding="{Binding Date}"/>

<DataGridTextColumn Header="Должность" Width="\*" Binding="{Binding Dolzhnost}"/>

<DataGridTextColumn Header="Телефон" Width="\*" Binding="{Binding Number}"/>

<DataGridTextColumn Header="Семья" Width="\*" Binding="{Binding Family}"/>

<DataGridTextColumn Header="Адрес" 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>

<StackPanel Grid.Row="2"

HorizontalAlignment="Center"

Orientation="Horizontal"

VerticalAlignment="Center">

<Button Height="24"

Click="Add2\_Btn\_Click"

Content="Добавить"

Width="150" Margin="5" FontFamily="Arial" FontSize="14"/>

<Button Height="24"

Click="otchet\_Click"

Content="Сформировать отчёт"

Width="150" Margin="5" FontFamily="Arial" FontSize="14"/>

<Button x:Name="Btn\_Del"

Height="24"

Click="RemoveBtn\_Click"

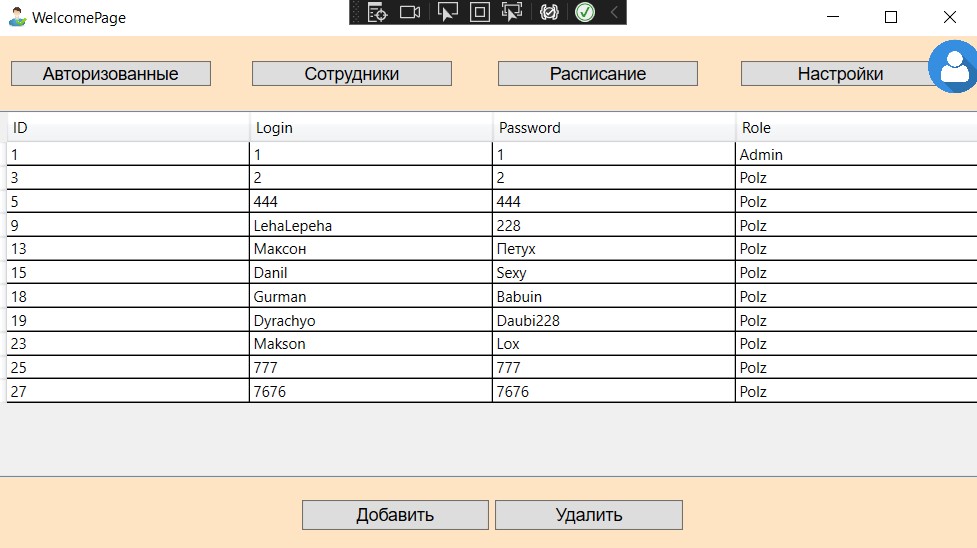
Content="Удалить"

Width="150" FontSize="14" FontFamily="Arial"/>

</StackPanel>

В разметке XAML прописаны кнопки добавления, изменения, удаления и формирования отчётов. Размечен дата грид для связи с бд с помощью {Binding}. Так же в <DataGrid> прописаны кнопки изменения.

Окно авторизованных сотрудников представлено ниже на рисунке 3.3, в нём храняться данные для входа всех сотрудников предприятия. Так же это окно доступно только администратору, потому что только администратор может регистрировать нового пользователя (сотрудника).

Рисунок 3.3 Авторизованные пользователи

<RowDefinition Height="60"/>

</Grid.RowDefinitions>

<DataGrid Grid.Row="1"

AutoGenerateColumns="False"

x:Name="UsersGrid" IsReadOnly="True">

<DataGrid.Columns>

<DataGridTextColumn Header="ID" Width="\*" Binding="{Binding Id}"/>

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

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

<DataGridTextColumn Header="Role" Width="\*" Binding="{Binding Role.Title}"/>

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Row="2"

HorizontalAlignment="Center"

Orientation="Horizontal"

VerticalAlignment="Center">

<Button Height="24"

Click="Add\_Btn\_Click"

Content="Добавить"

Width="150" Margin="5" HorizontalAlignment="Left" FontFamily="Arial" FontSize="14"/>

<Button x:Name="Btn\_Del"

Height="24"

Click="RemoveBtn\_Click"

Content="Удалить"

Width="150" HorizontalAlignment="Right" FontFamily="Arial" FontSize="14"/>

</StackPanel>

</Grid>

XAML разметка отвечает за работу кнопок и вывод данных из бд, путём создания DataGrid и связи каждой колонки с колонкой в бд чрез {Binding}. Так же убрано добавление пустой строки по умолчанию и возможность вписывать в таблице.

# 3.2 Программирование информационной системы

В данной части кода происходит объявление глобальных переменных для разграничения прав доступа (Окно авторизации).

public static class Globals //Глобальные переменные для разграничения

{

public static int UserRole;

public static User userinfo { get; set; }

}

В этой части кода происходит проверка логина с проверкой роли (администратор или пользователь). В случае ошибки ввода логина всплывает ошибка ("Такого пользователя не существует!").

private void BtnSignIn\_Click(object sender, RoutedEventArgs e) //Разграничение с проверкой логина

{

var CurrentUser = AppData.db.User.FirstOrDefault(u => u.Login == TxbLogin.Text);

if (CurrentUser != null)

{

Globals.UserRole = CurrentUser.RoleId;

Globals.userinfo = CurrentUser;

TxbPassword.IsEnabled = true;

Autr.Visibility = Visibility.Hidden;

Autr\_2.Visibility = Visibility.Visible;

}

else

{

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

}

}

В этой части кода происходит финальная проверка на правильность ввода пароля и генерация рандомного числа для капчи.

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

{

var CurrentUser1 = AppData.db.User.FirstOrDefault(u => u.Login == TxbLogin.Text && u.Password == TxbPassword.Text); //Разграничение с проверкой логина и пароля

if (CurrentUser1 != null)

{

Globals.UserRole = CurrentUser1.RoleId;

Globals.userinfo = CurrentUser1;

if (kapt.Visibility == Visibility.Hidden)

kapt.Visibility = Visibility.Visible;

TxbLogin.IsEnabled = false;

TxbPassword.IsEnabled = false;

Autr\_2.IsEnabled = false;

while (true)

{

Random x = new Random();

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

TXB1.Text = a.ToString();

await Task.Delay(10000);

}

}

else

{

MessageBox.Show("Пароль не верен!");

}

В окне Сотрудники происходит проверка if для разграничения, в случе если авторизованный сотрудник админ, кнопка для перехода в окно авторизованные становится видимой.

public Sotrudniki() //Связь датагрида с бд и разграничение

{

InitializeComponent();

UsersGrid2.ItemsSource = PenisEntities.GetContext().Sotrudnikii.ToList();

if (MainWindow.Globals.UserRole == 1)

{

skrit.Visibility = Visibility.Visible;

}

else

{

skrit.Visibility = Visibility.Collapsed;

}

}

В этой части кода окна сотрудники происходит удаление данных для входа сотрудника с уведомлением “Вы уверены?”.

private void RemoveBtn\_Click(object sender, RoutedEventArgs e) //Кнопка удаления с сохранением изменений в бд

{

if (MessageBox.Show("Вы уверены?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

var CurrentSotrudnikii = UsersGrid2.SelectedItem as Sotrudnikii;

AppData.db.Sotrudnikii.Remove(CurrentSotrudnikii);

AppData.db.SaveChanges();

UsersGrid2.ItemsSource = AppData.db.Sotrudnikii.ToList();

MessageBox.Show("Всё удалено");

}

}

В последней части кода происходит формирование отчётов с разметкой и оформлением в Word документе

private void otchet\_Click(object sender, RoutedEventArgs e)//Формирование отчётов, с разметкой в ворде

{

var allUsers = PenisEntities.GetContext().User.ToList();

var allRequest = PenisEntities.GetContext().Sotrudnikii.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 = "Данные о сотрудниках";

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 работника";

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

cellRange.Text = "Адрес";

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

cellRange.Text = "Дата рождения";

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

cellRange.Text = "Должность";

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

cellRange.Text = "ФИО";

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

cellRange.Text = "Номер телефона";

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];

if (currentCategory.FullName != null)

{

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.Adress);

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

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

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

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

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

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

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

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

}

}

В окне Авторизованные происходит переход на окно добавления сотрудника при нажатии кнопки “добавть”.

private void Add\_Btn\_Click(object sender, RoutedEventArgs e)//Переход на другое окно

{

NewUser newuser = new NewUser();

newuser.Show();

this.Close();

}

В этой части кода так же проводится проверка удаления с уведомлением “Вы уверены?”

private void RemoveBtn\_Click(object sender, RoutedEventArgs e)//Кнопка удаления, с предупреждением

{

if (MessageBox.Show("Вы уверены?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

var CurrentUser = UsersGrid.SelectedItem as User;

AppData.db.User.Remove(CurrentUser);

AppData.db.SaveChanges();

UsersGrid.ItemsSource = AppData.db.User.ToList();

MessageBox.Show("Всё удалено");

}

}

# ЗАКЛЮЧЕНИЕ

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

Экономический эффект от внедрения автоматизированной информационной системы «Учёт сотрудников на предприятии» ожидается за счет сокращения затрат времени на выполняемые администратором, исключения ошибок при формировании отчетов, увеличения времени на анализ и добавления сотрудников. 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:

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

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

Интернет-ресурсы

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