**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 «Подбор, найм и сопровождение трудовых ресурсов»

Выполнил студент гр. ИСП-20 /\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_Иконников А.О\_\_/

(подпись) (Ф.И.О.)

**ОЦЕНКА: \_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**ПРОВЕРИЛ**

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

(подпись) (Ф.И.О.)

Воркута

2023

Оглавление

[Введение 3](#_Toc133136048)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133136049)

[1.1 Платформа .NET 4](#_Toc133136050)

[1.2 Язык программирования С# 6](#_Toc133136051)

[1.3 Windows Presentation Foundation (WPF) 7](#_Toc133136052)

[1.4 СУБД SQL server 10](#_Toc133136053)

[1.5 Microsoft SQL Server Management Studio 11](#_Toc133136054)

[1.6 Entity Framework 12](#_Toc133136055)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 14](#_Toc133136056)

[2.1 Разработка диаграммы ERD 14](#_Toc133136057)

[2.2 Разработка базы данных 16](#_Toc133136058)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 21](#_Toc133136059)

[3.1 Разработка интерфейса информационной системы 21](#_Toc133136060)

[3.2 Программирование информационной системы 35](#_Toc133136061)

[ЗАКЛЮЧЕНИЕ 44](#_Toc133136062)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 45](#_Toc133136063)

ВВЕДЕНИЕ

Актуальность выбранной темы в том, что в любой фирме, организации необходимы процессы учета. Например, учет о уволенных работниках, учет о поступивших работниках и т.д. Учет работников включает в себя не только процессы создание графика отпусков и заработной платы, но и подбор, найм и сопровождение трудовых ресурсов.

Система, автоматизирующие управление персоналом организации, должны включать в себя функции подбора, найма и сопровождения трудовых ресурсов. Данная система гораздо упростит и сократит время работы для отдела кадров предприятия. Это поможет создавать отчеты гораздо быстрее и уменьшить количество ошибок в отчете.

Данная система подойдет для работников отдела кадров, так как они работают с персоналом. Они занимаются как подбором и наймом, так и сопровождением трудовых ресурсов.

**Объект:** информационная система «Подбор, найм и сопровождение трудовых ресурсов».

**Предмет:** автоматизация бизнес процесса подбора трудовых ресурсов.

**Цель работы** – разработать автоматизированную информационную систему «Подбор, найм и сопровождение трудовых ресурсов».

**Задачи:**

* выбор инструментария,
* спроектировать базу данных,
* разработать информационную систему.

ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

* 1. Платформа .NET

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для различных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду. В настоящее время .NET Framework развивается в виде .NET.

Считается, что платформа .NET Framework является ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java компании Sun Microsystems (ныне принадлежит Oracle).

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

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

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

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL) (ранее назывался Microsoft Intermediate Language, MSIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), 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 «на лету» (just in time) преобразует промежуточный байт-код 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, избавляя разработчика от части работы.

Архитектура .NET Framework описана и опубликована в спецификации Common Language Infrastructure (CLI), разработанной Microsoft и утверждённой ISO и ECMA. В CLI описаны типы данных .NET, формат метаданных о структуре программы, система исполнения байт-кода и многое другое.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке Framework Class Library (FCL). В FCL входят классы Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation и другие. Ядро FCL называется Base Class Library (BCL).

Среды разработки, поддерживающие .NET:

• Microsoft Visual Studio (C#, Visual Basic .NET, Managed C++, F#)

• SharpDevelop

• MonoDevelop

• Embarcadero RAD Studio (Delphi for .NET); ранее Borland Developer Studio (Delphi for .NET, C#)

• A#

• Zonnon

• PascalABC.NET

• JetBrains Rider

Приложения .NET также можно разрабатывать в текстовом редакторе, просто вызывая компилятор из командной строки.

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:

• C#

• Visual Basic .NET

• JScript .NET

• C++/CLI — новая версия Managed C++

• F# — член семейства языков программирования ML, включён в VS2010/VS2012/ VS2015/VS2017/VS2019/VS2022

* 1. Язык программирования С#

C# (произносится как "си шарп") — 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. Здесь представлен обзор основных компонентов языка C# 8 и более ранних версий. Если вы хотите изучить язык с помощью интерактивных примеров, рекомендуем поработать с вводными руководствами по C#.

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

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.

1.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

WPF предустановлена в Windows Vista (.NET Framework 3.0), Windows 7 (.NET Framework 3.5 SP1), Windows 8 (.NET Framework 4.0 и 4.5), Windows 8.1 (.NET Framework 4.5.1) и Windows 10 (.NET Framework 4.7). С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в браузере 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 (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, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

XAML представляет собой язык декларативного описания интерфейса, основанный на 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 второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента, например, Button (кнопка).

Для работы с WPF требуется любой .NET-совместимый язык. В этот список входит множество языков: C#, F#, VB.NET, C++, Ruby, Python, Delphi (Prism), Lua и многие другие. Для полноценной работы может быть использована как Visual Studio, так и Expression Blend. Первая ориентирована на программирование, а вторая — на дизайн и позволяет делать многие вещи, не прибегая к ручному редактированию 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). Привязка данных представлена классом Binding, который в свою очередь унаследован от MarkupExtension, что позволяет использовать привязки не только в коде, но и в разметке:

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

• MultiBinding — позволяет создавать множественные привязки, указывая несколько элементов

• TemplateBinding — используется в шаблонах для связывания свойства элемента внутри шаблона со свойством элемента, к которому применен шаблон

• PriorityBinding — 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

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 элементов управления

Позволяют менять графическое оформление элементов и представлены классом ControlTemplate. В отличие от стилей, можно менять не только графическое представление элемента, но и его структуру. При этом шаблон элемента управления задается через свойство Template.

В отличие от шаблонов элементов управления, задаются для определенного контекста данных (который в блочных элементах управления задается через свойство DataContext, а в списковых через ItemsSource). Сам шаблон данных представлен классом DataTemplate. 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.

1.4 СУБД SQL server

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

Особенности функционирования SQL SERVER:

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

• высокоскоростной доступ к данным, обеспечиваемый надежной клиент-серверной архитектурой СУБД;

• простота работы и администрирования, обусловленные понятной структурой языка программирования SQL;

• безопасность хранения информации в БД - благодаря возможности шифрования данных и резервного копирования.

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

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

Виды SQL-server:

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

MS SQL server - многопользовательский программный продукт, разработанный компанией Microsoft, обладающий высокой производительностью и отказоустойчивостью, тесно интегрированный с ОС Windows. Этот сервер поддерживает удаленные подключения, работает с многими популярными типами данных, дает возможность создавать триггеры и хранимые данные, имеет практичные и удобные утилиты для настройки;

Oracle Database server - СУБД, предназначенная для создания, консолидации и управления базами данных в облачной среде. Используя этот сервер, можно как автоматизировать обычные бизнес-операции, так и выполнять динамический многомерный анализ данных (OLAP), проводить операции с документами xml-формата и управлять разделенной и локальной информацией;

IBM DB2 - семейство СУБД для работы с реляционными базами данных, признанное самым производительным, имеющим высокие технические показатели и возможности масштабирования. SQL-серверы этой группы характеризуются мультиплатформенностью, способностью к мгновенному созданию резервных копий и восстановлению БД, реорганизации таблиц в онлайн-режиме, разбиению баз данных, определению пользователями новых типов данных;

MySQL - СУБД, разработанная и поддерживаемая компанией Oracle. В основном она используется локальными или удаленными клиентами, позволяя им работать с таблицами разных типов, поддерживающих полнотекстовый поиск или выполняющих транзакции на уровне отдельных записей;

PostgreSQL - СУБД с открытым исходным кодом, работающая с объектно-реляционными (поддерживающими пользовательские объекты) базами данных. Также PostgreSQL предназначена для создания, хранения и извлечения сложных структур данных. Она поддерживает самые различные типы данных (среди них - числовые, текстовые, булевы, денежные, бинарные данные, сетевые адреса, xml и другие).

1.5 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) — 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.

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 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.

SSMS работает только в Windows (AMD или Intel). Если вам требуется средство, которое работает на платформах, отличных от Windows, рассмотрите Azure Data Studio. Azure Data Studio — это кроссплатформенное средство для macOS, Linux и Windows. Дополнительные сведения см. в разделе Azure Data 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.

1.6 Entity Framework

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

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

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.

Entity Framework включает поставщик данных EntityClient. Этот поставщик управляет соединениями, преобразует запросы сущностей в запросы, зависящие от источника данных, и возвращает модуль чтения данных, который Entity Framework использует для материализации данных сущности в объекты.

ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

2.1 Разработка диаграммы ERD

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

Базовыми понятиями ER-модели данных (ER — Entity— Relati-onship) 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. Структурой данных называют совокупность правил и ограничений, которые отражают связи, существующие между отдельными частями (элементами) данных.

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 сущности.

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. Экземпляр атрибута — определенная характеристика конкретного экземпляра сущности.

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

Первичный ключ — это атрибут или совокупность атрибутов и (или) связей, предназначенная для уникальной идентификации каждого экземпляра сущности (совокупность признаков, позволяющих идентифицировать объект). Ключевые атрибуты помещают в начало списка и помечают символом «#».

Описательные атрибуты могут быть обязательными или необязательными.

Обязательные атрибуты для каждой сущности всегда имеют конкретное значение, необязательные могут быть не определены. Обязательные и необязательные описательные атрибуты помечают символами «\*» и «°» соответственно.

Связь — это отношение одной сущности к другой или к самой себе. Каждая связь может иметь одну из двух модальностей связей. 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.

Связь может иметь разную модальность с разных концов. Каждая сущность может обладать любым количеством связей с другими сущностями модели. Различают три типа отношений «один-к-одному»; «один-ко-многим»; «многие-ко-многим».

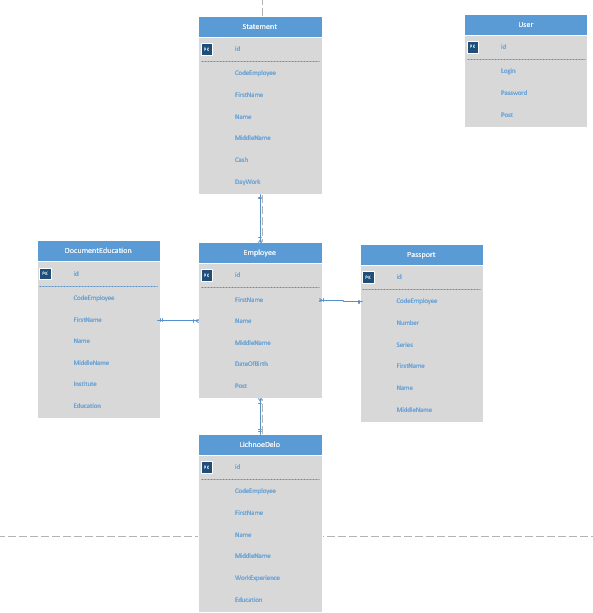


Рисунок 2.1. ER (Entity-Relationship Diagram).

Основными сущностями для решения указанной задачи являются: Пользователь (), Работник (), Личное дело (), Паспорт (), Данные об образование (), Ведомость (). Связь на диаграмме проходит следующим образом, от id работника тянется связь к другим сущностям, которые заполняются благодаря сущности работника. Отношение между ними относится к типу «один-ко-многим». Также есть сущность, которая не имеет связи, так как она отвечает за отдельную базу, базу пользователей.

Проанализируем атрибуты сущностей. Работник: id, FirstName, Name, MiddleName, DateOfBirth, Post. Паспорт: id, FirstName, Name, MiddleName, Number, Series. Документ об образование: id, FirstName, Name, MiddleName, Institute, Education. Личное дело: id, FirstName, Name, MiddleName, WorkExprerience, Education. Ведомость: id, FirstName, Name, MiddleName, Cash, WorkDay. Пользователь: id, Login, Password, Post.

2.2 Разработка базы данных

SQL Server Management Studio — это набор административных средств для управления компонентами, относящимися к SQL Server. Эта интегрированная среда позволяет пользователям выполнять разнообразные задачи, например резервное копирование данных, редактирование запросов и автоматизацию общих функций в одном интерфейсе.

Среда SQL Server Management Studio включает в себя следующие средства:

Редактор кода — богатый возможностями редактор скриптов для написания и редактирования скриптов. Среда SQL Server Management Studio предоставляет четыре версии редактора кода: редактор запросов ядра СУБД для скриптов Transact-SQL, редактор запросов многомерных выражений, редактор запросов расширения интеллектуального анализа данных и редактор запросов XML/A.

Обозреватель объектов для размещения, изменения, создания скрипта или выполнения объектов, принадлежащих экземплярам SQL Server.

Обозреватель шаблонов для размещения и написания сценариев шаблонов.

Обозреватель решений для организации и хранения связанных скриптов как части проекта.

Окно свойств для отображения текущих свойств выбранных объектов.

Среда SQL Server Management Studio обеспечивает эффективность рабочих процессов, предоставляя:

Отключенный доступ. Можно писать и изменять скрипты, не соединяясь с экземпляром SQL Server.

Создание сценариев из любого диалогового окна. Можно создать скрипт из любого диалогового окна, а также читать, изменять, сохранять и многократно использовать скрипты после создания.

Немодальные диалоговые окна. При обращении к диалоговому окну интерфейса можно просмотреть другие ресурсы в среде SQL Server Management Studio, не закрывая диалоговое окно.

Разработка базы данных начинается с ранее разработанной ERD диаграммой. Откроем Microsoft SQL Server Management Studio и подключаемся к нашему серверу.

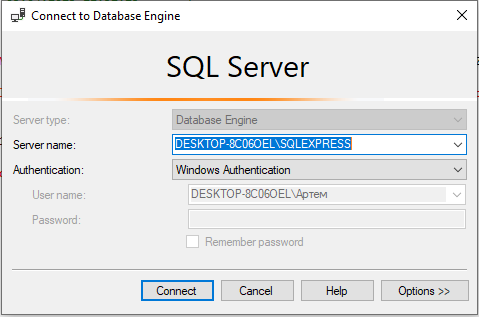


Рисунок 2.2 Окно подключения к локальному серверу

После чего у нас станет доступна таблица Object Explorer в которой будут показаны файлы и базы данных на локальном сервере. Создадим базу данных, нажимаем ПКМ по папке «Databases» и выбираем «New Database».

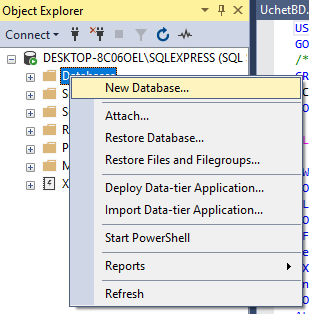


Рисунок 2.3 Создание базы данных

Далее нам выводиться окно, в котором мы даём название для базы данных и

н нажимаем на кнопку OK. В Object Explorer появиться созданная база данных, развернём её, нажмём ПКМ по папке Database Diagrams и нажмём на New Database Diagram.

После чего нам выведется поле для создания диаграммы базы данных, на котором мы можем создавать таблицы с название столбцов и их типом данных.

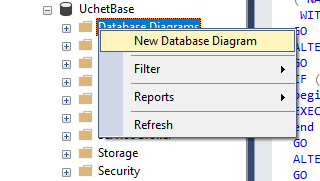


Рисунок 2.4 Обозреватель базы данных

Для создания таблицы данных необходимо нажать ПКМ в любом месте диаграммы,

выбираем «New Table» и даём название для нашей таблицы. Для информационной системы необходимо создать шесть таблиц: Пользователь, Работник, Паспорт, Ведомость, Документ об образование, Личное дело. После чего заполним таблицы необходимыми полями и типами данных из ERD таблицы.



Рисунок 2.5 Все таблицы

Далее создаём связи как в ERD сущностях и после сохраняем таблицы нажав сочетание клавиш «Ctrl + S» и в высветившимся окне пишем название нашей диаграммы.

Созданные нами таблицы можно увидеть в Object Explorer кликнув на название базы данных и перейдя в папку Tables. Зажав уникальный ключ, тянем к полу в нужной таблице и создаётся связь.

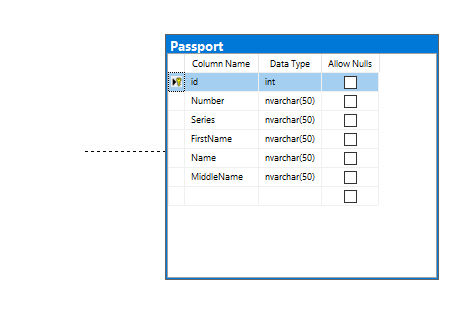


Рисунок 2.6 Создание связи

После создания связей наша таблица полностью готова.



Рисунок 2.7 База данных.

В данной работе используется шесть таблиц базы данных. Employee содержит уникальный ключ, ФИО, Дату рождения, Должность. Другие таблицы так же содержат в себе ФИО и свои уникальные ключи с другими атрибутами. Таблица User не имеет связей с другими таблицами так как она отвечает за пользовательскую часть.

ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Разработка интерфейса информационной системы

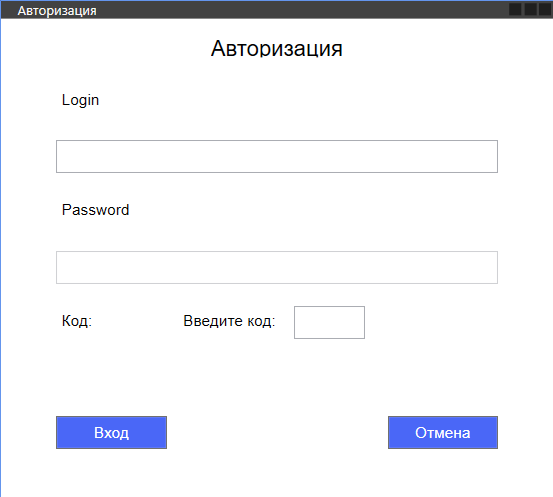


Рисунок 3.1 Форма Авторизации

На рисунке 3.1 показана форма авторизации, в котором имеется три TextBox(Login, Password, Введите код), TextBlock(Код), Две кнопки(Вход, Отмена). При правильном вводе логина, откроется окно ввода пароля. При правильном вводе пароля появится код подтверждения, который надо подтвердить.

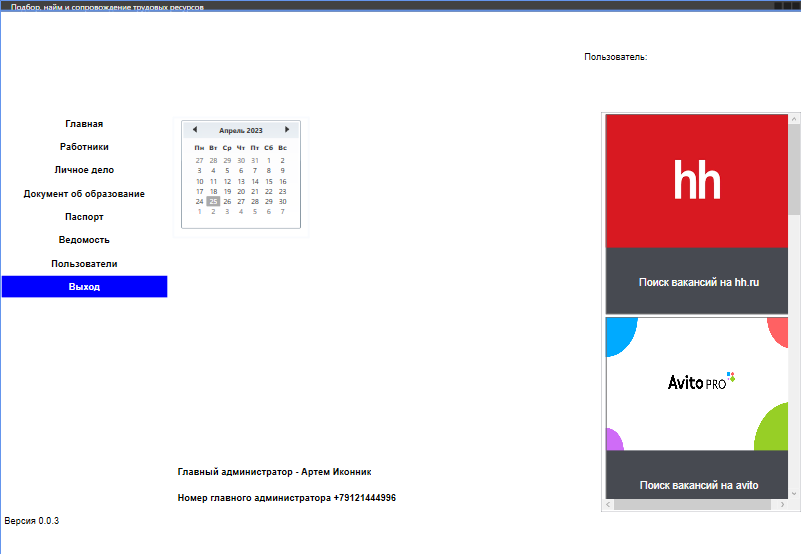


Рисунок 3.2 Форма главного окна

На рисунке 3.2 показано главное окно, в котором есть календарь для проверки даты, четыре кнопки: Поиск вакансий на hh.ru, Поиск вакансий на avito, Погода, Новости. Так же имеется Имя Главного администратора и его номер.

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="250"></ColumnDefinition>

<ColumnDefinition Width="950"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="150"></RowDefinition>

<RowDefinition Height="600"></RowDefinition>

<RowDefinition Height="34"></RowDefinition>

</Grid.RowDefinitions>

<Grid Grid.Row="1" Grid.Column="0">

<StackPanel>

<Button Click="Btn\_Glav" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Главная" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_Empl" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Работники" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_LD" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Личное дело" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_DE" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Документ об образование" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_Pass" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Паспорт" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_State" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Ведомость" BorderBrush="Transparent" Background="Transparent"></Button>

<Button x:Name="Btn\_User1" Click="Btn\_User" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Пользователи" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_Exit" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Выход" Foreground="White" BorderBrush="Transparent" Background="#0000ff"></Button>

</StackPanel>

</Grid>

<Calendar HorizontalAlignment="Left" Grid.Row="1" Grid.Column="1" Width="200" Margin="10,10,0,413"/>

<ListView HorizontalAlignment="Right" Width="300" Grid.Row="1" Grid.Column="1">

<Grid Width="280" Height="300" Background="#474A51">

<Image Source="/Image/hh.jpg" Width="280" Height="200" VerticalAlignment="Top" Stretch="Fill"></Image>

<Label FontSize="16" FontFamily="Arial" Content="Поиск вакансий на hh.ru" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,238,0,0" Foreground="White"></Label>

<Button Click="Btn\_hh" Background="Transparent"></Button>

</Grid>

<Grid Width="280" Height="300" Background="#474A51">

<Image Source="/Image/avito.png" Width="280" Height="200" VerticalAlignment="Top" Stretch="Fill"></Image>

<Label FontSize="16" FontFamily="Arial" Content="Поиск вакансий на avito" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,238,0,0" Foreground="White"></Label>

<Button Click="Btn\_avito" Background="Transparent"></Button>

</Grid>

<Grid Width="280" Height="300" Background="#474A51">

<Image Source="/Image/pogoda.jpg" Width="280" Height="200" VerticalAlignment="Top" Stretch="Fill"></Image>

<Label FontSize="16" FontFamily="Arial" Content="Погода" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,238,0,0" Foreground="White"></Label>

<Button Click="Btn\_pogoda" Background="Transparent"></Button>

</Grid>

<Grid Width="280" Height="300" Background="#474A51">

<Image Source="/Image/novosti.jpg" Width="280" Height="200" VerticalAlignment="Top" Stretch="Fill"></Image>

<Label FontSize="16" FontFamily="Arial" Content="Новости" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,238,0,0" Foreground="White"></Label>

<Button Click="Btn\_novosti" Background="Transparent"></Button>

</Grid>

</ListView>

<Label Width="300" Height="34" Grid.Row="1" Grid.Column="1" FontWeight="Bold" Content="Главный администратор - Артем Иконников" FontSize="14" FontFamily="Arial" Margin="10,527,640,39"></Label>

<Label Height="34" Grid.Row="1" Grid.Column="1" FontWeight="Bold" Content="Номер главного администратора +79121444996" FontSize="14" FontFamily="Arial" Margin="10,566,572,0"/>

<Label Width="250" Height="34" Grid.Row="2" Grid.Column="0" Content="Версия 0.0.3" FontSize="14" FontFamily="Arial"></Label>

<TextBlock FontSize="14" FontFamily="Arial" x:Name="tb\_User" Grid.Row="0" Grid.Column="1" Width="200" Height="30" HorizontalAlignment="Right" Margin="0 0 20 0"></TextBlock>

<TextBlock FontSize="14" FontFamily="Arial" Grid.Row="0" Grid.Column="1" Width="100" Height="30" HorizontalAlignment="Left" Margin="625,60,0,60">Пользователь:</TextBlock>

</Grid>

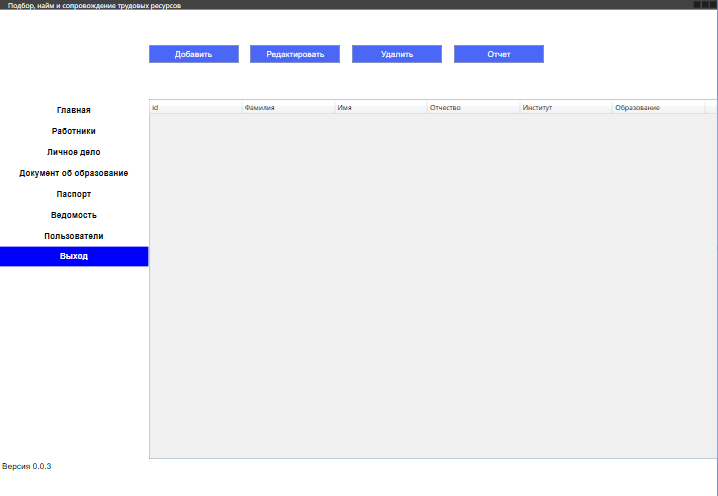


Рисунок 3.3.1 Форма окна документов об образование

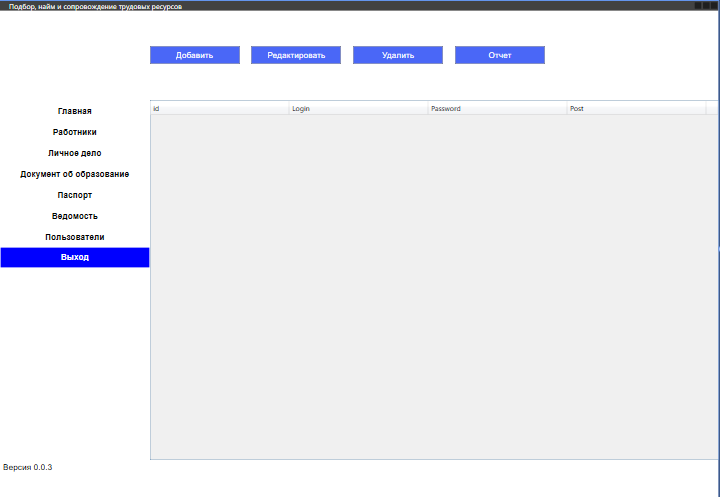


Рисунок 3.3.2 Форма окна пользователей

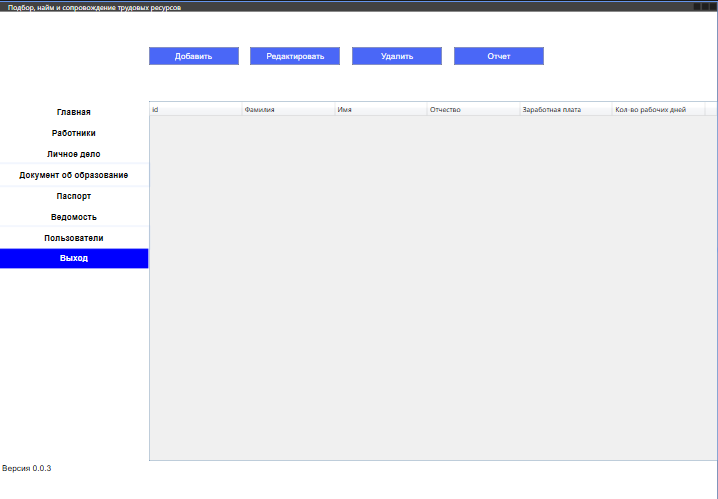


Рисунок 3.3.3 Форма окна ведомости

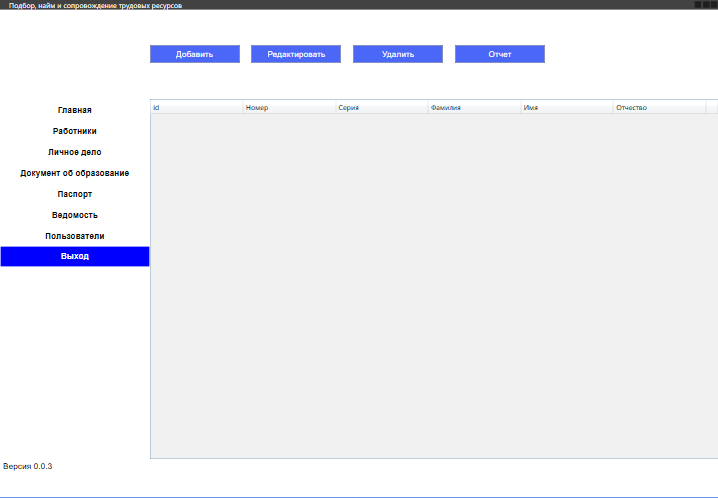


Рисунок 3.3.4 Форма окна данных паспорта

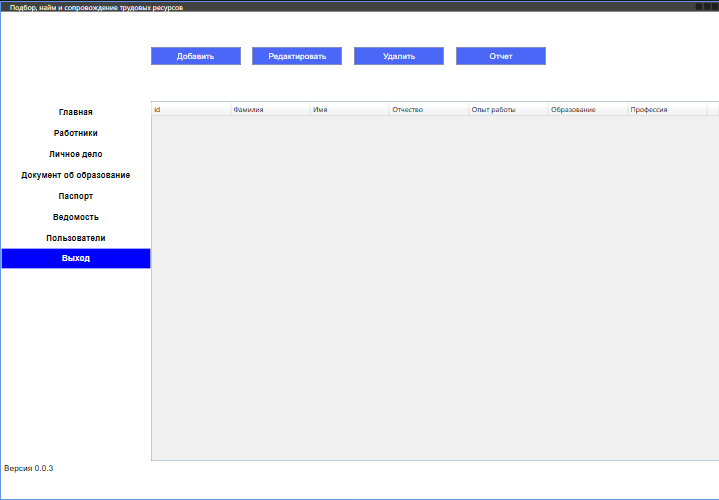


Рисунок 3.3.5 Форма окна личных дел

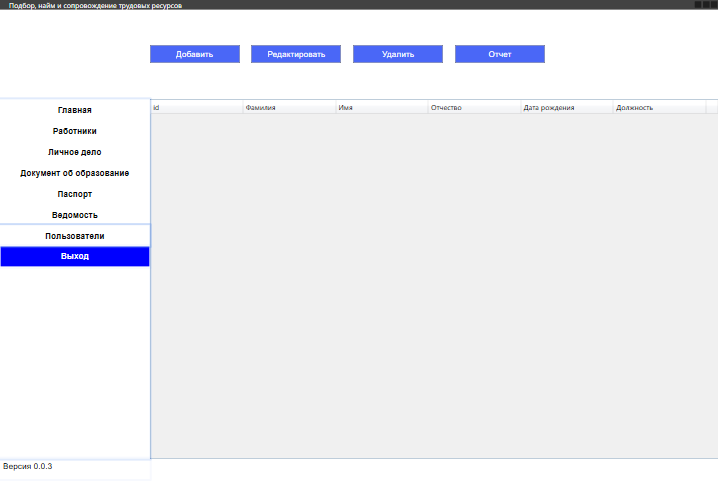


Рисунок 3.3.6 Форма окна работников

Выше перечисленные окна таблиц баз данных “ трудовых ресурсов” имеют идентичный код, так как они имеют кнопки добавления, удаления, редактирования, отчета и DataGrid в котором в зависимости от сущности, присваивается значение колонки.

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="250"></ColumnDefinition>

<ColumnDefinition Width="950"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="150"></RowDefinition>

<RowDefinition Height="600"></RowDefinition>

<RowDefinition Height="34"></RowDefinition>

</Grid.RowDefinitions>

<Grid Grid.Row="1" Grid.Column="0">

<StackPanel>

<Button Click="Btn\_Glav" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Главная" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_Empl" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Работники" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_LD" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Личное дело" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_DE" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Документ об образование" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_Pass" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Паспорт" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_State" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Ведомость" BorderBrush="Transparent" Background="Transparent"></Button>

<Button x:Name="Btn\_User1" Click="Btn\_User" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Пользователи" BorderBrush="Transparent" Background="Transparent"></Button>

<Button Click="Btn\_Exit" FontWeight="Bold" FontFamily="Arial" FontSize="14" Height="35" Width="250" Content="Выход" Foreground="White" BorderBrush="Transparent" Background="#0000ff"></Button>

</StackPanel>

</Grid>

<Grid Grid.Row="0" Grid.Column="1">

<StackPanel Orientation="Horizontal">

<Button Click="Btn\_Add" Margin="0 0 20 0" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Добавить" Foreground="White" Background="#4A67F7"></Button>

<Button Click="Btn\_Edit" Margin="0 0 20 0" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Редактировать" Foreground="White" Background="#4A67F7"></Button>

<Button Click="Btn\_Del" Margin="0 0 20 0" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Удалить" Foreground="White" Background="#4A67F7"></Button>

<Button Click="Btn\_Otchet" Margin="0 0 20 0" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Отчет" Foreground="White" Background="#4A67F7"></Button>

</StackPanel>

</Grid>

<DataGrid AutoGenerateColumns="False" x:Name="EmployeeGrid" Grid.Row="1" Grid.Column="1" IsReadOnly="True">

<DataGrid.Columns>

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

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

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

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

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

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

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Редактировать" Name="BtnEdit" Click="Btn\_Edit"></Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Label Width="250" Height="34" Grid.Row="2" Grid.Column="0" Content="Версия 0.0.3" FontSize="14" FontFamily="Arial"></Label>

</Grid>

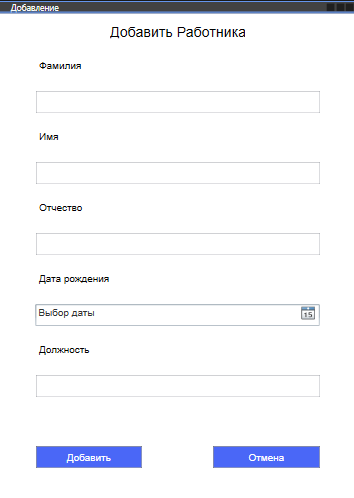


Рисунок 3.4 Форма добавления работников

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"></ColumnDefinition>

<ColumnDefinition Width="400"></ColumnDefinition>

<ColumnDefinition Width="50"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="34"></RowDefinition>

</Grid.RowDefinitions>

<Label Width="200" Height="30" FontFamily="Arial" FontSize="20" Content="Добавить Работника" Grid.Row="0" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Фамилия" Grid.Row="1" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Имя" Grid.Row="3" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Отчество" Grid.Row="5" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Дата рождения" Grid.Row="7" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Должность" Grid.Row="9" Grid.Column="1"></Label>

<TextBox x:Name="txb\_FirstName" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="2" Grid.Column="1"></TextBox>

<TextBox x:Name="txb\_Name" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="4" Grid.Column="1"></TextBox>

<TextBox x:Name="txb\_MiddleName" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="6" Grid.Column="1"></TextBox>

<DatePicker x:Name="txb\_DateOfBirth" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="8" Grid.Column="1"></DatePicker>

<TextBox x:Name="txb\_Post" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="10" Grid.Column="1"></TextBox>

<Button Click="Btn\_Save" HorizontalAlignment="Left" Grid.Row="12" Grid.Column="1" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Добавить" Foreground="White" Background="#4A67F7"></Button>

<Button Click="Btn\_Cel" HorizontalAlignment="Right" Grid.Row="12" Grid.Column="1" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Отмена" Foreground="White" Background="#4A67F7"></Button>

</Grid>

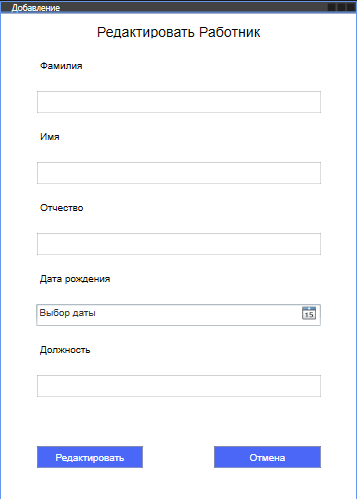


Рисунок 3.5 Форма редактирование работников

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"></ColumnDefinition>

<ColumnDefinition Width="400"></ColumnDefinition>

<ColumnDefinition Width="50"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

<RowDefinition Height="34"></RowDefinition>

</Grid.RowDefinitions>

<Label Width="240" Height="30" FontFamily="Arial" FontSize="20" Content="Редактировать Работника" Grid.Row="0" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Фамилия" Grid.Row="1" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Имя" Grid.Row="3" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Отчество" Grid.Row="5" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Дата рождения" Grid.Row="7" Grid.Column="1"></Label>

<Label HorizontalAlignment="Left" Width="200" Height="30" FontFamily="Arial" FontSize="14" Content="Должность" Grid.Row="9" Grid.Column="1"></Label>

<TextBox Text="{Binding FirstName}" x:Name="txb\_FirstName" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="2" Grid.Column="1"></TextBox>

<TextBox Text="{Binding Name}" x:Name="txb\_Name" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="4" Grid.Column="1"></TextBox>

<TextBox Text="{Binding MiddleName}" x:Name="txb\_MiddleName" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="6" Grid.Column="1"></TextBox>

<DatePicker SelectedDate="{Binding DateOfBirth}" x:Name="txb\_DateOfBirth" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="8" Grid.Column="1"></DatePicker>

<TextBox Text="{Binding Post}" x:Name="txb\_Post" Width="400" Height="30" FontFamily="Arial" FontSize="14" Grid.Row="10" Grid.Column="1"></TextBox>

<Button Click="Btn\_Edit" HorizontalAlignment="Left" Grid.Row="12" Grid.Column="1" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Редактировать" Foreground="White" Background="#4A67F7"></Button>

<Button Click="Btn\_Can" HorizontalAlignment="Right" Grid.Row="12" Grid.Column="1" Width="150" Height="30" FontFamily="Arial" FontSize="14" Content="Отмена" Foreground="White" Background="#4A67F7"></Button>

</Grid>

3.2 Программирование информационной системы

Код окна Авторизации:

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

Объявление глобального класса, для видимости атрибутов

//Обьявляем глобальные переменные

public static class Globals

{

public static int UserRole;

public static string Login;

public static User userinfo { get; set; }

}

Создание кнопки проверки на логин

//Делаем проверку на логин

private void Btn\_Next(object sender, RoutedEventArgs e)

{

var CurrentUser = AppData.db.User.FirstOrDefault(u => u.Login == txt\_Login.Text);

if (CurrentUser != null)

{

Globals.UserRole = CurrentUser.id;

Globals.userinfo = CurrentUser;

txt\_Password.IsEnabled = true;

Btn\_next.Visibility = Visibility.Hidden;

Btn\_next1.Visibility = Visibility.Visible;

}

else

{

MessageBox.Show("Пользователь не найден!");

}

}

Создание кнопки проверки пароля

//Делаем проверку на пароль

private async void Btn\_Next1(object sender, RoutedEventArgs e)

{

var CurrentUser1 = AppData.db.User.FirstOrDefault(u => u.Password == txt\_Password.Password);

if (CurrentUser1 != null)

{

Globals.UserRole = CurrentUser1.id;

Globals.userinfo = CurrentUser1;

Btn\_next1.Visibility = Visibility.Hidden;

Btn\_next2.Visibility = Visibility.Visible;

Создание Кода Подтверждения с таймером 10 секунд

//Делаем появление кода подтверждения

while (true)

{

Random x = new Random();

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

txt\_Code.Text = a.ToString();

await Task.Delay(10000);

}

}

else

{

MessageBox.Show("Пароль не верен!");

}

}

Создание проверки кода подтверждения

//Делаем Проверку Кода подтверждения

private void Btn\_Next2(object sender, RoutedEventArgs e)

{

if (txt\_Code.Text == txt\_CodePod.Text)

{

MessageBox.Show("Вы вошли!");

WindowGlav windowGlav = new WindowGlav();

windowGlav.tb\_User.Text = txt\_Login.Text;

windowGlav.Show();

this.Close();

}

else

{

MessageBox.Show("Вы не вошли!");

}

}

Создание кнопки закрытия программы

//Кнопка закрытия программы

private void Btn\_Cancel(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

}

Код окон (Работники, Пользователи, Паспорт, Личное дело, Ведомость, Документ об образование):

public partial class WindowDE : Window

{

public WindowDE()

{

//Проверка на админа и пользователя

InitializeComponent();

if (MainWindow.Globals.UserRole == 1)

{

Btn\_User1.Visibility = Visibility.Visible;

}

else

{

Btn\_User1.Visibility = Visibility.Collapsed;

}

}

//Подключение элементов грида от таблицы бд

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

DEGrid.ItemsSource = AppData.db.DocumentEducation.ToList();

}

//Ниже перечисленные кнопки для переключения межжду окнами

private void Btn\_Glav(object sender, RoutedEventArgs e)

{

WindowGlav windowGlav = new WindowGlav();

windowGlav.Show();

this.Close();

}

private void Btn\_LD(object sender, RoutedEventArgs e)

{

WindowLD windowLD = new WindowLD();

windowLD.Show();

this.Close();

}

private void Btn\_DE(object sender, RoutedEventArgs e)

{

}

private void Btn\_Pass(object sender, RoutedEventArgs e)

{

WindowPassport windowPassport = new WindowPassport();

windowPassport.Show();

this.Close();

}

private void Btn\_State(object sender, RoutedEventArgs e)

{

WindowStatement windowStatement = new WindowStatement();

windowStatement.Show();

this.Close();

}

private void Btn\_User(object sender, RoutedEventArgs e)

{

WindowUser windowUser = new WindowUser();

windowUser.Show();

this.Close();

}

private void Btn\_Exit(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

private void Btn\_Empl(object sender, RoutedEventArgs e)

{

WindowEmployee windowEmployee = new WindowEmployee();

windowEmployee.Show();

this.Close();

}

//Конец окон переключения между окнами

//Окно Добавления элемента в бд

private void Btn\_Add(object sender, RoutedEventArgs e)

{

AddDE addDE = new AddDE();

addDE.Show();

}

//Окно редактирование элемента в бд

private void Btn\_Edit(object sender, RoutedEventArgs e)

{

EditDE editDE = new EditDE(DEGrid.SelectedItem as DocumentEducation);

editDE.Show();

}

//Окно удаления элемента в бд

private void Btn\_Del(object sender, RoutedEventArgs e)

{

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

{

var CurrentUser = DEGrid.SelectedItem as DocumentEducation;

AppData.db.DocumentEducation.Remove(CurrentUser);

AppData.db.SaveChanges();

DEGrid.ItemsSource = AppData.db.DocumentEducation.ToList();

MessageBox.Show("Удалено");

}

}

Код окна добавления:

public partial class AddDE : Window

{

public AddDE()

{

InitializeComponent();

}

//Сохранение данных в бд

private void Btn\_Save(object sender, RoutedEventArgs e)

{

DocumentEducation de = new DocumentEducation();

de.FirstName = txb\_FirstName.Text;

de.Name = txb\_FirstName.Text;

de.MiddleName = txb\_FirstName.Text;

de.Institute = txb\_Institute.Text;

de.Education = txb\_Education.Text;

AppData.db.DocumentEducation.Add(de);

AppData.db.SaveChanges();

MessageBox.Show("Пользователь был добавлен в базу!");

}

//Кнопка отмены

private void Btn\_Can(object sender, RoutedEventArgs e)

{

this.Close();

}

}

Код окна редактирования;

public partial class EditDE : Window

{

//Объявление переменной для бд

private DocumentEducation \_de = new DocumentEducation();

public EditDE(DocumentEducation selectedDocumentEducation)

{

InitializeComponent();

if (selectedDocumentEducation != null)

\_de = selectedDocumentEducation;

DataContext = \_de;

}

//Кнопка сохранения

private void Brn\_Save(object sender, RoutedEventArgs e)

{

//Создание предупреждение с пустым полем

StringBuilder erros = new StringBuilder();

if (string.IsNullOrWhiteSpace(\_de.FirstName))

erros.AppendLine("Укажите фамилию");

if (string.IsNullOrWhiteSpace(\_de.Name))

erros.AppendLine("Укажите Имя");

if (string.IsNullOrWhiteSpace(\_de.MiddleName))

erros.AppendLine("Укажите фамилию");

if (string.IsNullOrWhiteSpace(\_de.Institute))

erros.AppendLine("Укажите институт");

if (string.IsNullOrWhiteSpace(\_de.Education))

erros.AppendLine("Укажите образование");

if (erros.Length > 0)

{

MessageBox.Show(erros.ToString());

}

if (\_de.id == 0)

{

UchetBaseEntities3.GetContext().DocumentEducation.Add(\_de);

}

//Сохранение таблицы и бд

try

{

UchetBaseEntities3.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена");

}

catch (Exception ex)

{

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

}

}

//Кнопка отмены

private void Brn\_Can(object sender, RoutedEventArgs e)

{

this.Close();

}

ЗАКЛЮЧЕНИЕ

При выполнении данной работы, а именно конструирования автоматизированной системы «Подбор, найм и сопровождение трудовых ресурсов», 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 сотрудников отдела кадров. С её помощью можно быстро создавать отчёты.

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

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

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Грин Дженнифер. Head First: изучение C#/ Грин Джениффер, Стиллмен Эндрю – Санкт-Питербург: 2022.
2. Мэтью Мак-Дональд. Windows Presentation Foundation: изучение С# WPF/ Мэтью Мак-Дональд – Санкт-Питербург: 2016.
3. Stack overflow exception. Интерфейсы // C# - Киберфорум – Режим доступа: [https://www.cyberforum.ru/csharp](https://www.cyberforum.ru/csharp-beginners/thread2225109.html?ysclid=lgw4535h24160143326)
4. Newest 'c#' Questions // Stack Overflow. – Режим доступа: [https://stackoverflow.com/questions](https://stackoverflow.com/questions/tagged/c%23)
5. Документация по C#. Начало работы, руководства, справочные материалы. // Microsoft Learn. – Режим доступа:  <https://learn.microsoft.com/ru-ru/dotnet/csharp/>
6. 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/)
7. 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>