УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40-01-03 «Информационные системы и технологии»

Специализация «Издательско-полиграфический комплекс»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА**

по дисциплине «Объектно-ориентированное программирование»

Тема «Разработка программного средства для управления персоналом предприятия»

Исполнитель

|  |  |  |
| --- | --- | --- |
| cтудентка 2 курса группы 2    Руководитель |  | \_\_\_\_\_\_\_\_\_ Е.А.Гладкая  подпись, дата |
| ассистент |  | \_\_\_\_\_\_\_\_\_ М.В. Гладкий  подпись, дата |
| Курсовой проект защищен с оценкой |  | \_\_\_\_\_\_\_\_\_ |
| Руководитель \_\_\_\_\_\_\_ |  | М.В.Гладкий |

подпись

Минск 2016

**Содержание**

[Введение 2](#_Toc452404674)

[1.Постановка задачи 3](#_Toc452404675)

[1.1 Алгоритмы решения 3](#_Toc452404676)

[1.2 Обзор прототипов 5](#_Toc452404677)

[2.Разработка архитектуры проекта 8](#_Toc452404678)

[2.1 Обобщенная структура 8](#_Toc452404679)

[2.1 Диаграммы UML, взаимосвязь всех компонентов 9](#_Toc452404680)

[3.Разработка функциональной модели и модели данных ПС 10](#_Toc452404681)

[3.1 Модель базы данных 10](#_Toc452404682)

[3.2 Выполняемые функции 12](#_Toc452404683)

[4. Руководство пользователя 16](#_Toc452404684)

[5.Тестирование 23](#_Toc452404685)

[Заключение 25](#_Toc452404686)

[Список используемых источников 26](#_Toc452404687)

[Приложение А 27](#_Toc452404688)

# Введение

Наиболее широко информационные системы и технологии используются в производственной, управленческой и финансовой деятельности.

В общем, можно сказать, что информационные технологии незаменимы практически во всех сферах деятельности человека. Соответственно, данный вопрос, несомненно, актуален в настоящее время, как для каждого отдельного человека, так и для общества.

На протяжении долгих лет работодатели имели информацию о своих сотрудниках, которая хранилась в печатном виде. С развитием информационных технологий это стало неактуально и небезопасно.

Отдел кадров выполняет очень важную роль – учитывает и контролирует весь персонал предприятия. Его виды и структура деятельности могут немного отличаться, это зависит и от направления работы, и от размера предприятия.   
Персонал занимается формированием штата организации. Кадровый отдел должен обеспечивать организацию соответствующими специалистами. Но мало только принять человека на работу. Увольнение, найм, перемещение сотрудника должны быть оформлены документально, в соответствии с законодательством страны. И все это сопровождается «круговоротом» информации. Отдел в идеале должен обеспечивать не только правильный документооборот, но и успешность ее дел при помощи грамотно выстроенной кадровой политики.

Для надежного и удобного структурирования информации работники отдела использую различные программные средства. В настоящее время их большое изобилие. В основном они выполняют одинаковые операции. Исходя из вышесказанного можно сделать вывод, что данная тема актуальна и требует постоянного усовершенствования.

# 1.Постановка задачи

Задачей данной курсовой работы является разработка программного приложения для управления персоналом. Его область применения – отдел кадров – совокупность специализированных подразделений в структуре предприятия (с занятыми в них должностными лицами – руководителями, специалистами, техническим персоналом), призванных управлять персоналом предприятия.

Основной функционал приложения:

— добавление сотрудников в базу данных;

— добавление сведений о сотрудниках;

— поиск сотрудников;

— удаление сотрудников из базы данных;

— запись сведений о сотрудниках в Excel-файл.

## **1.1 Алгоритмы решения**

В разработке приложения были использованы нижеперечисленные технологии:

— ADO.NET Entity Framework;

— Microsoft SQL Server;

— LINQ.

Также были использованы паттерны «Repository» и «Unit of Work».

*Entity Framework (EF)* – это объектно-реляционный модуль сопоставления, позволяющий разработчикам .NET работать с реляционными данными с помощью объектов, специализированных для доменов. Это устраняет необходимость в написании с большей части кода для доступа к данным, который обычно требуется разработчикам.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Также хочется выделить Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами БД. Она состоит из трех уровней: концептуальный, уровень хранилища и уровень сопоставления.

EF предполагает три возможных способа взаимодействия с БД:

— database first;

— model first;

— code first.

В процессе выполнения курсового проекта была использована первая модель — Database first (EF создает набор классов, которые отражают модель конкретной БД).

*Microsoft SQL Server* — система управления реляционными БД. Основной используемый язык запросов — Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

*Language Integrated Query (LINQ)* — проект Microsoft по добавлению синтаксиса языка запросов, напоминающего SQL, в языки программирования платформы.

*Паттерн «Repository»* — позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. Репозиторий добавляет программе гибкость при работе с разными типами подключений.

Система со сложной моделью области определения может быть упрощена при помощи дополнительного уровня, например, Data Mapper, который бы изолировал объекты от кода доступа к БД. В таких системах может быть полезным добавление ещё одного слоя абстракции поверх слоя распределения данных (Data Mapper), в котором бы был собран код создания запросов. Это становится ещё более важным, когда в области определения множество классов или при сложных, тяжелых запросах. В таких случаях добавление этого уровня особенно помогает сократить дублирование кода запросов.

Паттерн Repository инкапсулирует объекты, представленные в хранилище данных и операции, производимые над ними, предоставляя более объектно-ориентированное представление реальных данных. Repository также преследует цель достижения полного разделения и односторонней зависимости между уровнями области определения и распределения данных.

Паттерн «Unit of Work» — обслуживает набор объектов, изменяемых в бизнес-транзакции (бизнес-действии) и управляет записью изменений и разрешением проблем конкуренции данных. Когда необходимо писать и читать из БД, важно следить за тем, что вы изменили и, если не изменили – не записывать данные в БД. Также необходимо вставлять данные о новых объектах и удалять данные о старых.

Можно записывать в БД каждое изменение объекта, но это приведёт к большому количеству мелких запросов к БД, что закончится замедлением работы приложения. Более того, это требует держать открытую транзакцию всё время работы приложения, что непрактично, если приложение обрабатывает несколько запросов одновременно. Ситуация ещё хуже, если необходимо следить за чтением из и БД, чтобы избежать не консистентного чтения.

Реализация паттерна Unit of Work следит за всеми действиями приложения, которые могут изменить БД в рамках одного бизнес-действия. Когда бизнес-действие завершается, Unit of Work выявляет все изменения и вносит их в БД.

## **1.2 Обзор прототипов**

*Отдел кадров Плюс 2016*

Данное приложение используется в отделе кадров организаций различных отраслей, с различными численностью сотрудников. Программа позволяет:

— оперативно формировать кадровые документы;

— осуществлять быстрый поиск информации;

— вести календарные напоминания;

— быстро формировать отчеты;

— работу одновременно нескольких специалистов отдела кадров.

Преимущества:

— персональный менеджер;

— простота и удобство;

— не требует дополнительного ПО.

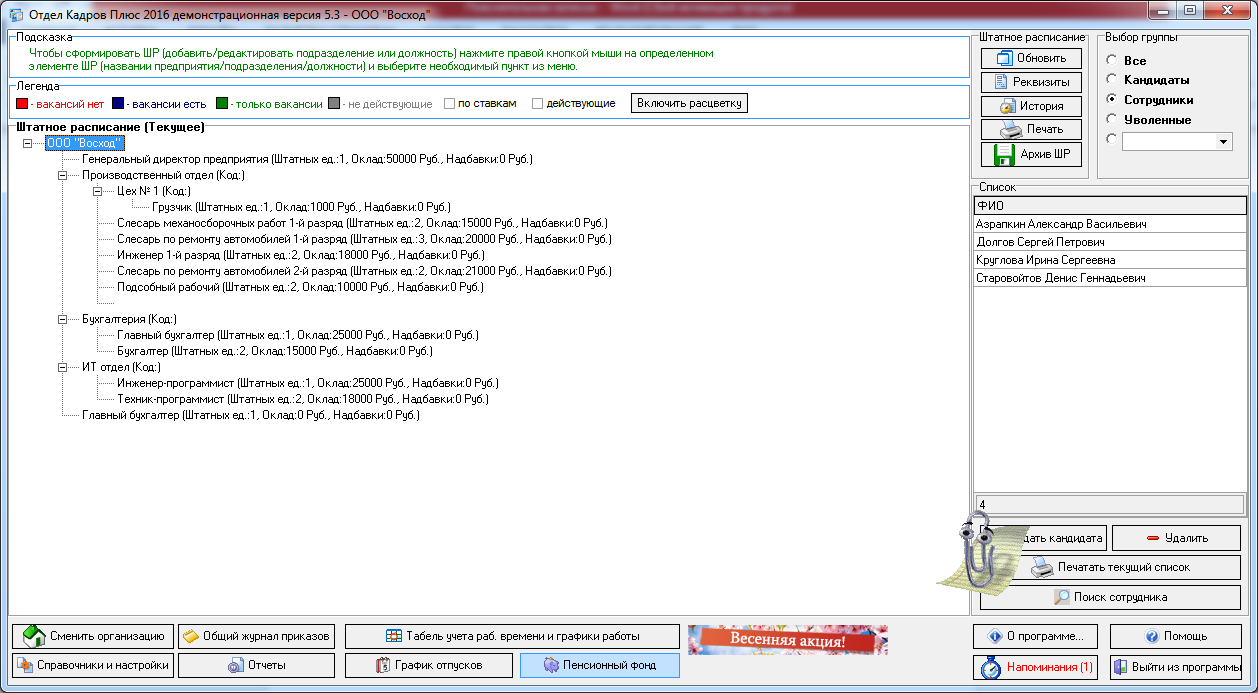


Рис.1.1 – Главная форма приложения «Отдел кадров Плюс 2016»

Весомым недостатком программы является то, что она платная, а это не всегда удобно пользователю.

*Кадровый учет*

Кадровый учет – это бесплатная программа для отделов кадров.

Возможности программы:

— ведение баз сотрудников по нескольким организациям;

— контроль вакансий;

— поиск по различной информации;

— автоматическое заполнение унифицированных форм;

— добавление в базу фотографий сотрудников;

— автоматическое составление журналов;

— изменение шаблонов и создание собственных;

— расчет отпусков;

— поиск по базе данных сотрудников;

— экспорт документов в форматы файлов Word и Excel.

Недостатки:

Данная программа устарела, вместе с ней устарел интерфейс. Также изменились ее возможности, что делает программу не удобной и убыточной для пользователя (взаимодействие с другими программами стало невозможных в результате их обновления и совершенствования)

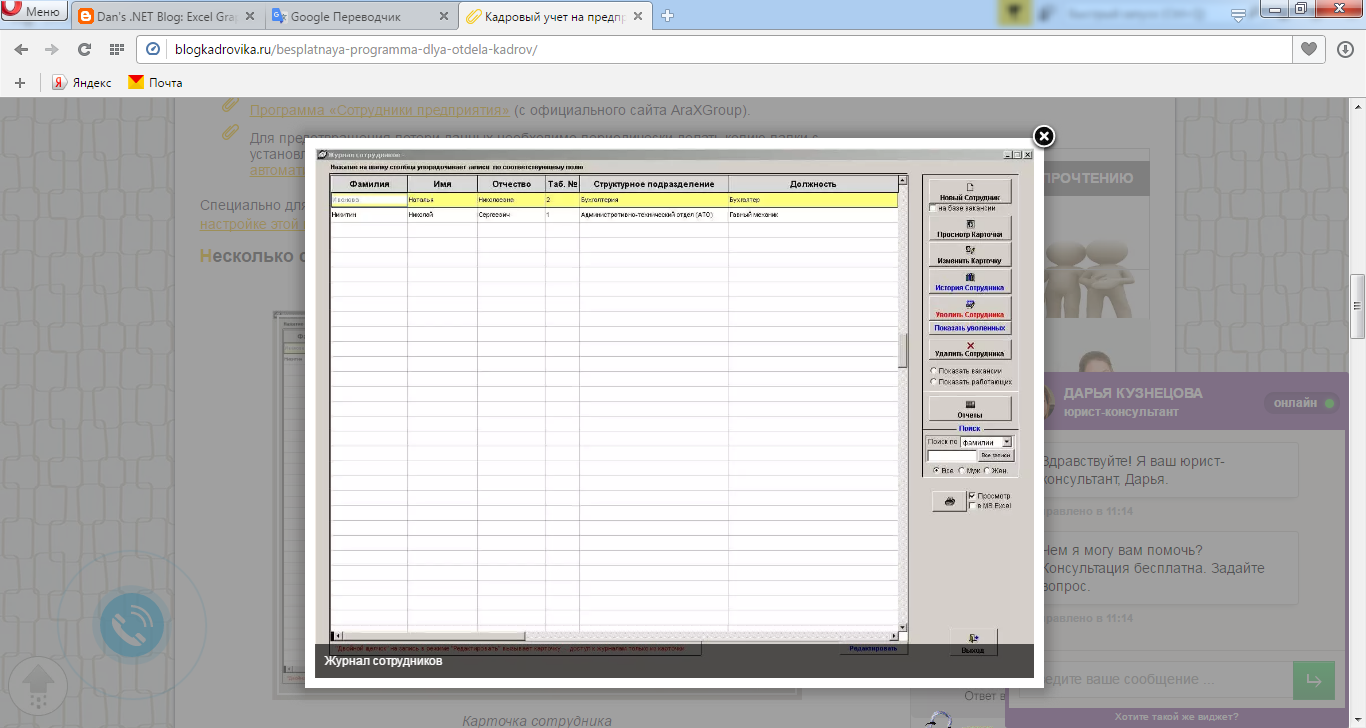
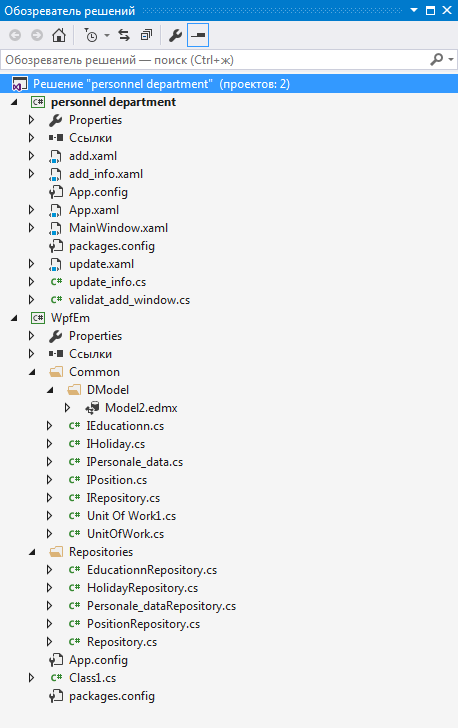


Рис. 1.2 – Журнал сотрудников «Кадровый учет»

# 2.Разработка архитектуры проекта

## **2.1 Обобщенная структура**

Структура проекта представлена на рисунке 2.1



2.1– Структура проекта

— Properties **–** свойства проекта, содержит информацию о сборке, используемых ресурсах и настройках;

— References **–** перечень сборок, используемых в проекте;

— App.config **–** файл с параметрами проекта;

— Model2.edmx **–** файл, в котором представлены классы для работы с таблицами базы данных;

Repositories – папка, содержащая классы для работы с данными каждой таблицы:

— EducationnRepository.cs – класс для работы с таблицей БД education;

— HolidayRepository.cs – класс для работы с таблицей БД holiday;

— PositionRepository.cs – класс для работы с таблицей БД position;

— Personale\_dataRepository.cs – класс для работы с таблицей БД personale\_data;

— Repository.cs – класс, в котором определяются дополнительные методы для работы с каждой сущностью нашей БД.

Common — папка, содержащая интерфейсы для классов, которые содержаться в папке Repositories:

— IEducationn.cs – интерфейс для класса EducationnRepository.cs;

— IHoliday.cs – интерфейс для класса HolidayRepository.cs;

— IPosition.cs – интерфейс для класса PositionRepository.cs;

— IRepositiry – интерфейс для класса Repository.cs;

— Unit Of Work1.cs – интерфейс для класса с помощью которого реализуется метод Dispose;

— IPersonale\_data.cs – интерфейс для класса Personale\_dataRepository.cs;

— Update\_info.cs – класс, содержащий статические поля, которые применяются для редактирования существующего объекта.

— validat\_add\_window.cs – класс для валидации данных;

— add – форма для «Добавления сотрудников в базу данных»;

— MainWindow – форма «Главное меню»;

— update – форма для «Редактирования»;

## **Диаграммы UML, взаимосвязь всех компонентов**

*Диаграмма в UML* — это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации. Основная цель диаграмм — визуализация разрабатываемой системы с разных точек зрения.

Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена на рисунке 2.3



Рис 2.3 – Диаграмма классов

# 3.Разработка функциональной модели и модели данных ПС

## **Модель базы данных**

Для реализации поставленной задачи была создана база данных p\_d. Для её создания использовалось система управления реляционными базами данных Microsoft SQL Server 2008.

База данных состоит из 4 таблиц, перечисленные ниже.

Диаграмма на рисунке 3.1 иллюстрирует структуру таблицы «position»,которая содержит информацию о занимаемой должности сотрудника:

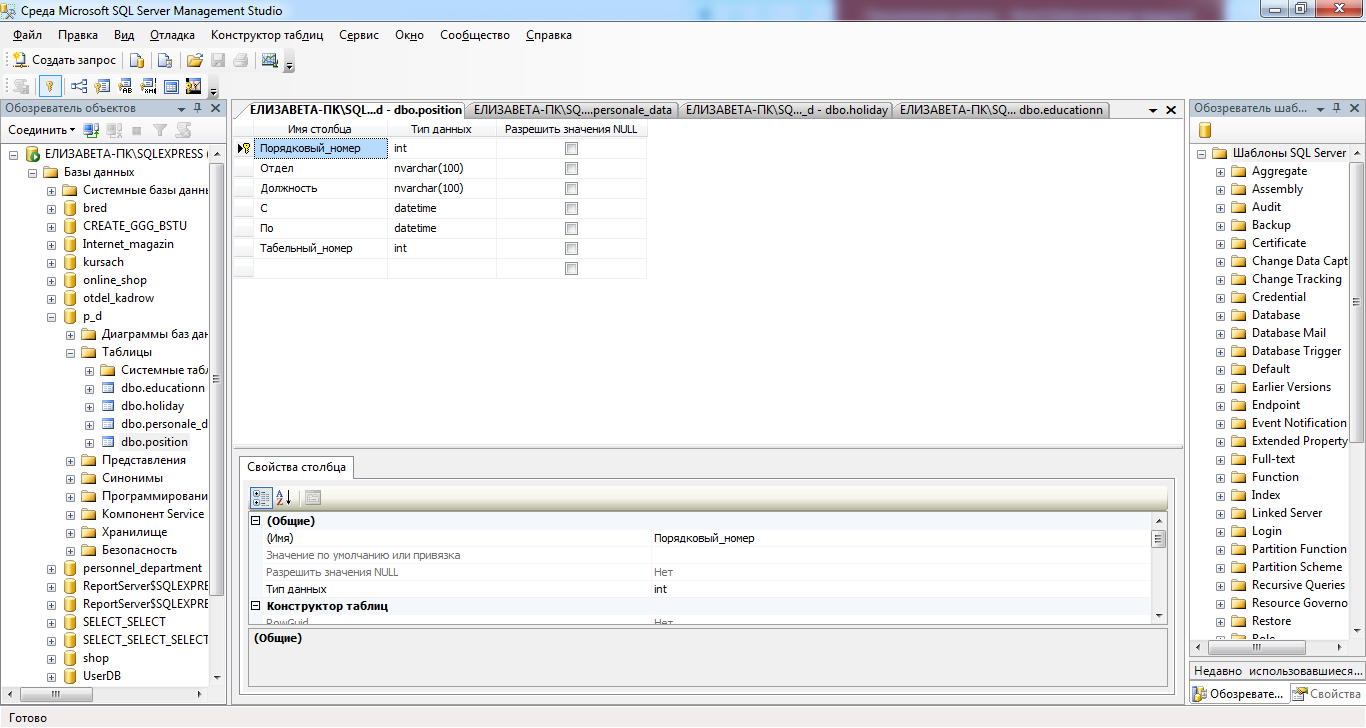


Рисунок 3.1 – Структура таблицы «position»

Для хранения личных данных создана таблица personale\_data представленная на рисунке 3.2:

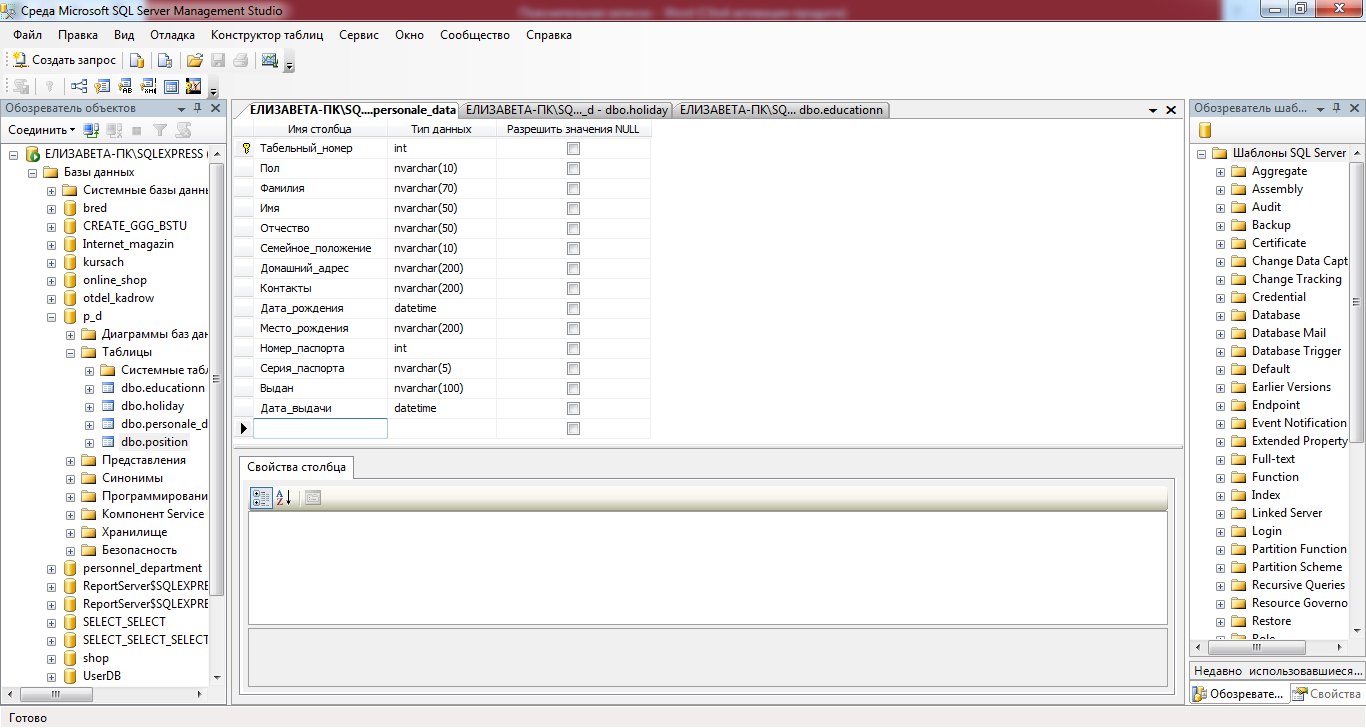


Рисунок 3.2 – Структура таблицы «personale\_data»

Сведения об образовании сотрудника можно посмотреть в таблице Education (рисунок 3.3):

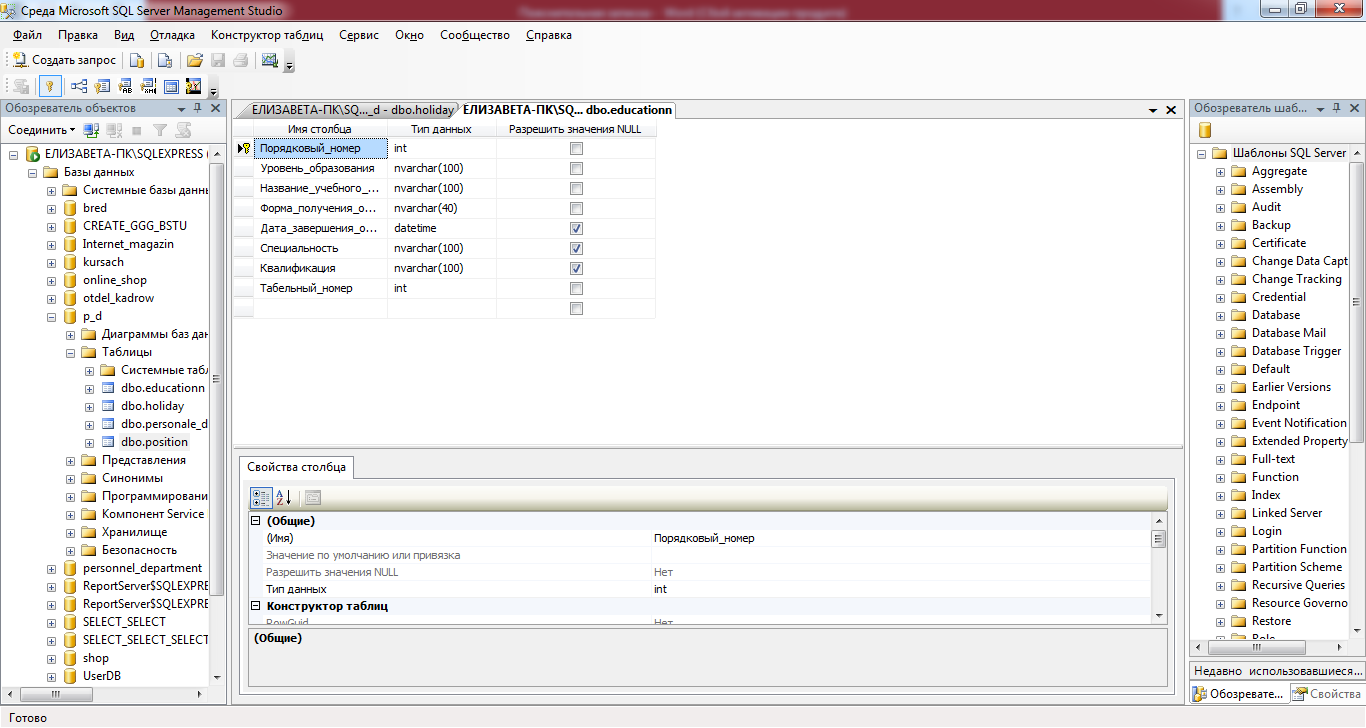


Рисунок 3.3 – Структура таблицы «Education»

Таблица holyday, представленная на рисунке 3.4, демонстрирует сведения об отпусках сотрудника:

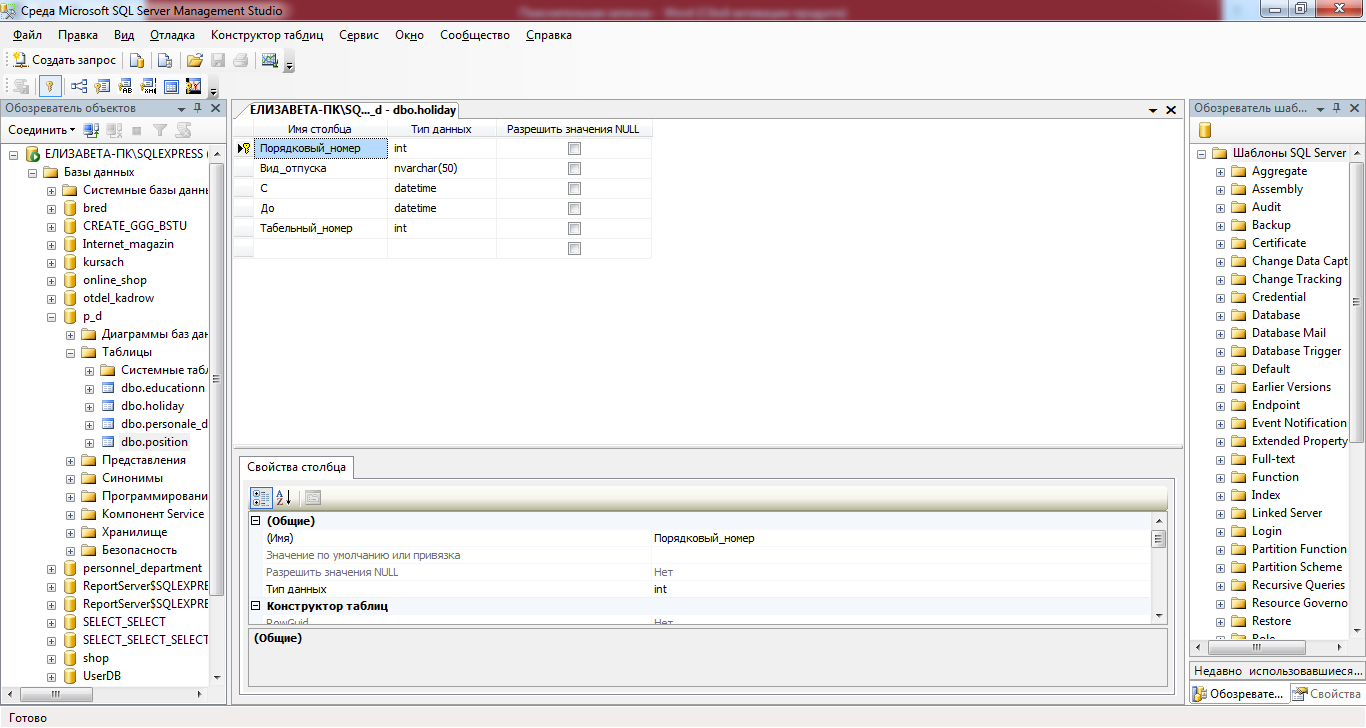


Рисунок 3.4 – Структура таблицы «holyday»

Взаимосвязь таблиц базы данных представлена на рисунке 3.5:

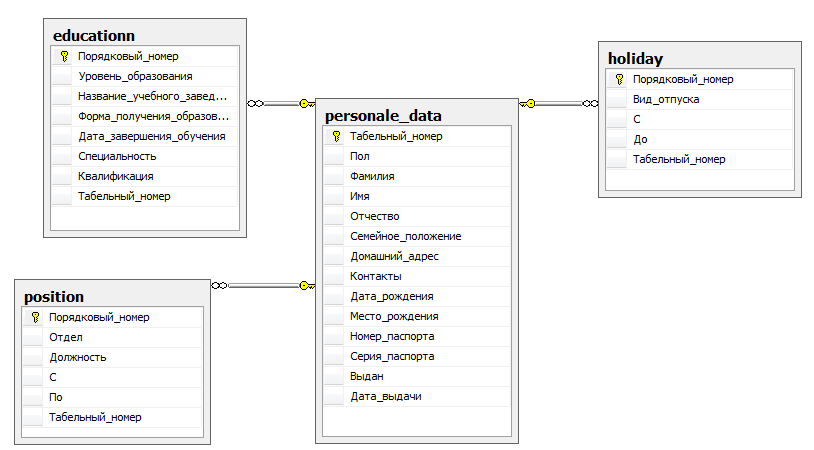


Рисунок 3.5 – Взаимосвязь таблиц базы данных

## **3.2 Выполняемые функции**

В проекте использовались паттерны, реализацию которых мы рассмотрим.

Паттерн «Repository» используется для изоляции логики хранения данных. Это промежуточное звено между базой данных и классами программы. Репозиторий добавляет программе гибкость при работе с разными типами подключений. Первоначально он представлен в виде интерфейсов.

Сразу создается интерфейс «IRepository.cs», который наследуется всеми остальными интерфейсами.

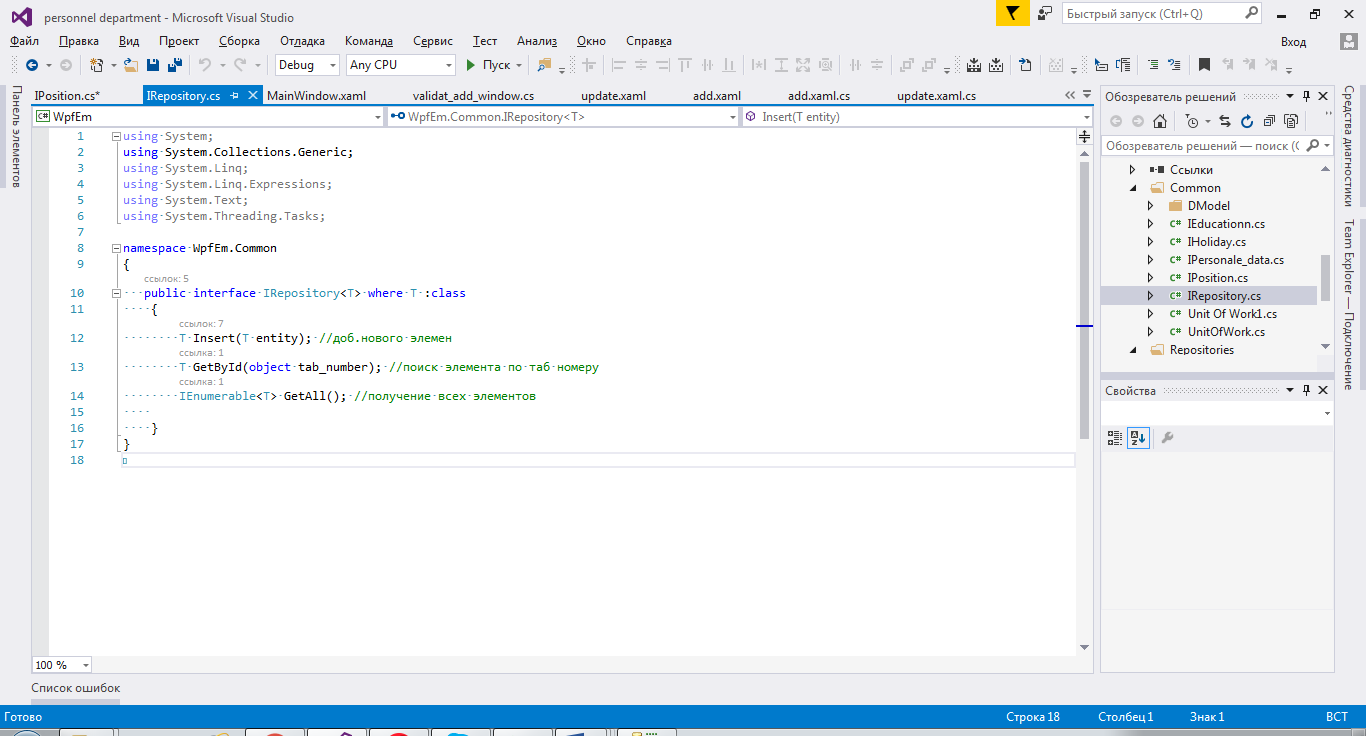


Рис 3.1 – Интерфейс «IRepository»

Затем создаются интерфейсы для каждой таблицы. В нашем случае их 4.

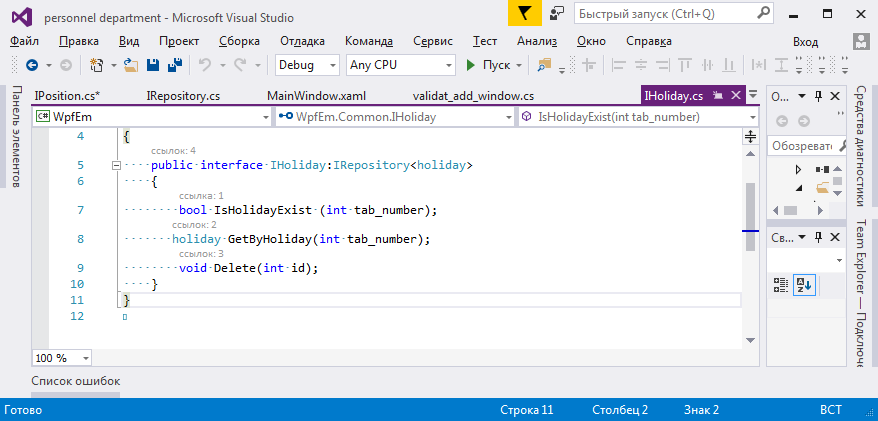


Рис 3.2 – Интерфейс «IHoliday»

На основании этого интерфейса создается класс «HolidayRepository», который содержит логику работы с данными, которые находятся в таблице holiday.

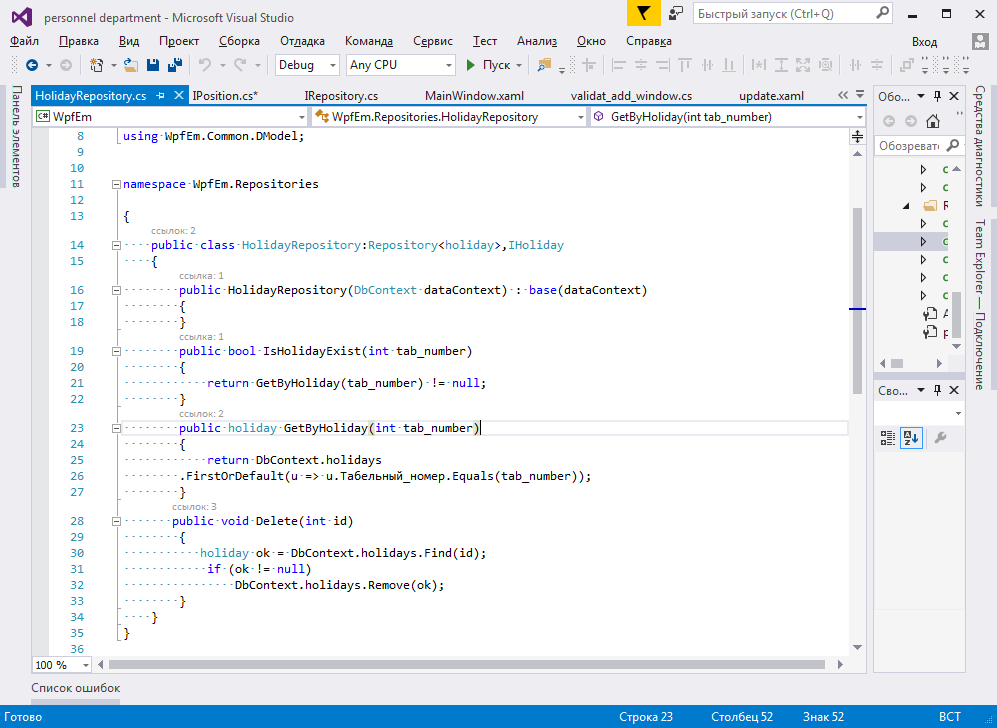


Рис 3.2 – Класс «HolidayRepository»

Аналогичные интерфейсы и классы создаются для остальных 3 таблиц.

В каждой таблице существует функция поиска объекта таблицы по табельному номеру.

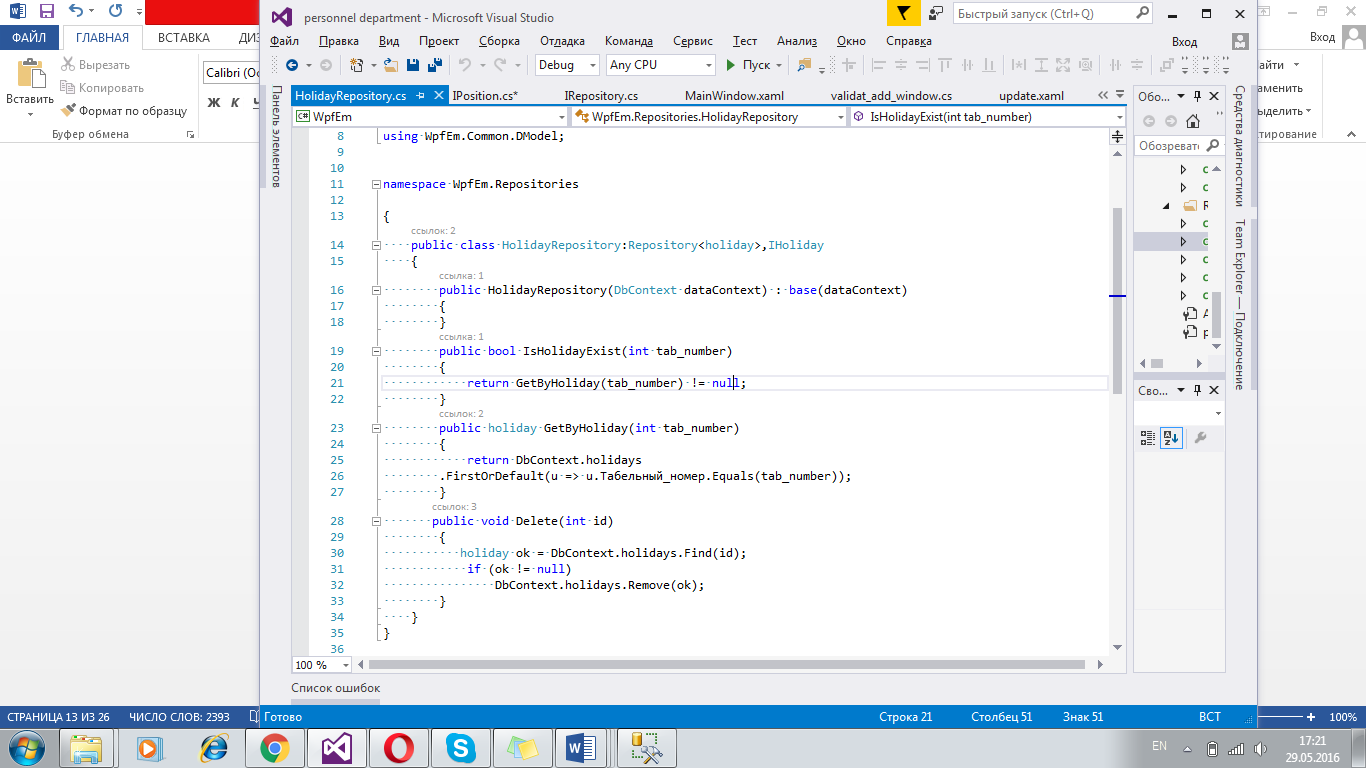


Рис 3.3 – Поиск объекта БД по табельному номеру

А также функция удаления.

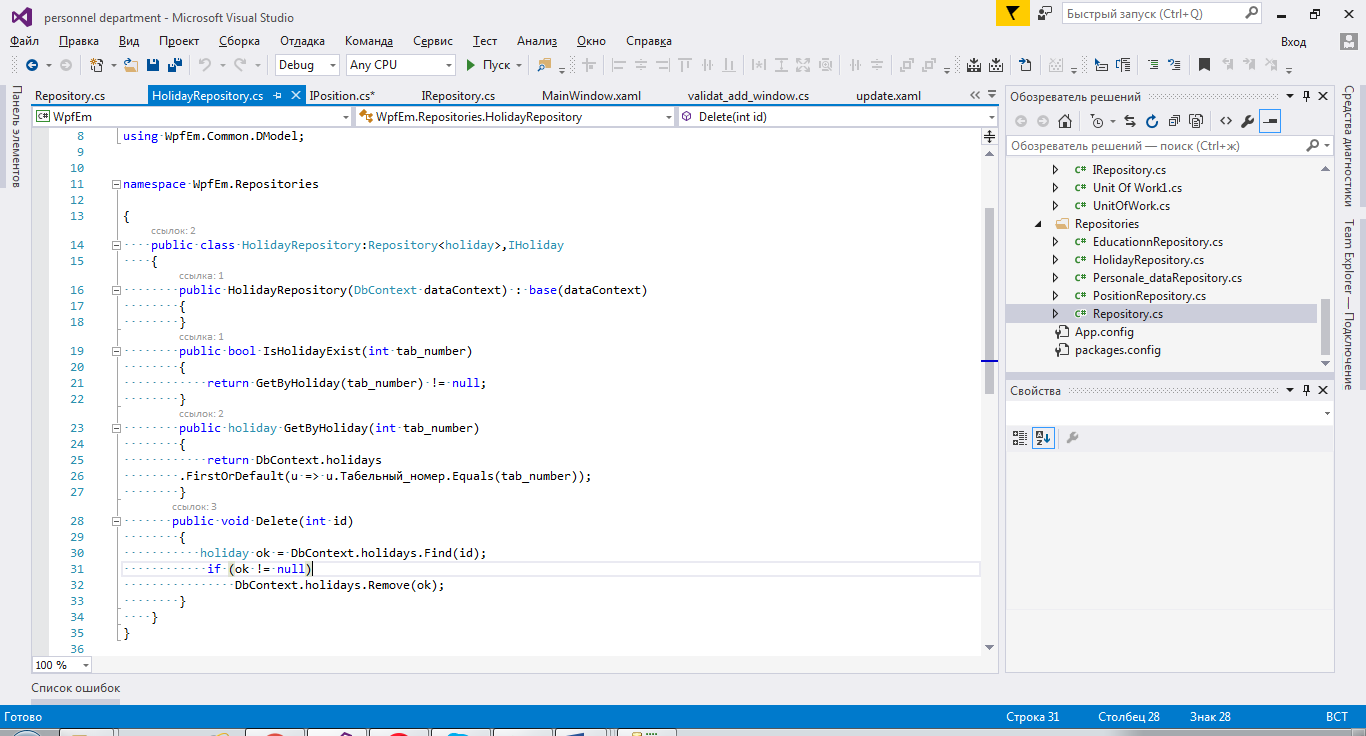


Рис 3.4 – функция удаления объекта из БД (предварительный поиск по табельному номеру).

В свою очередь паттерн «Unit of Work» содержит логику подключения к БД.

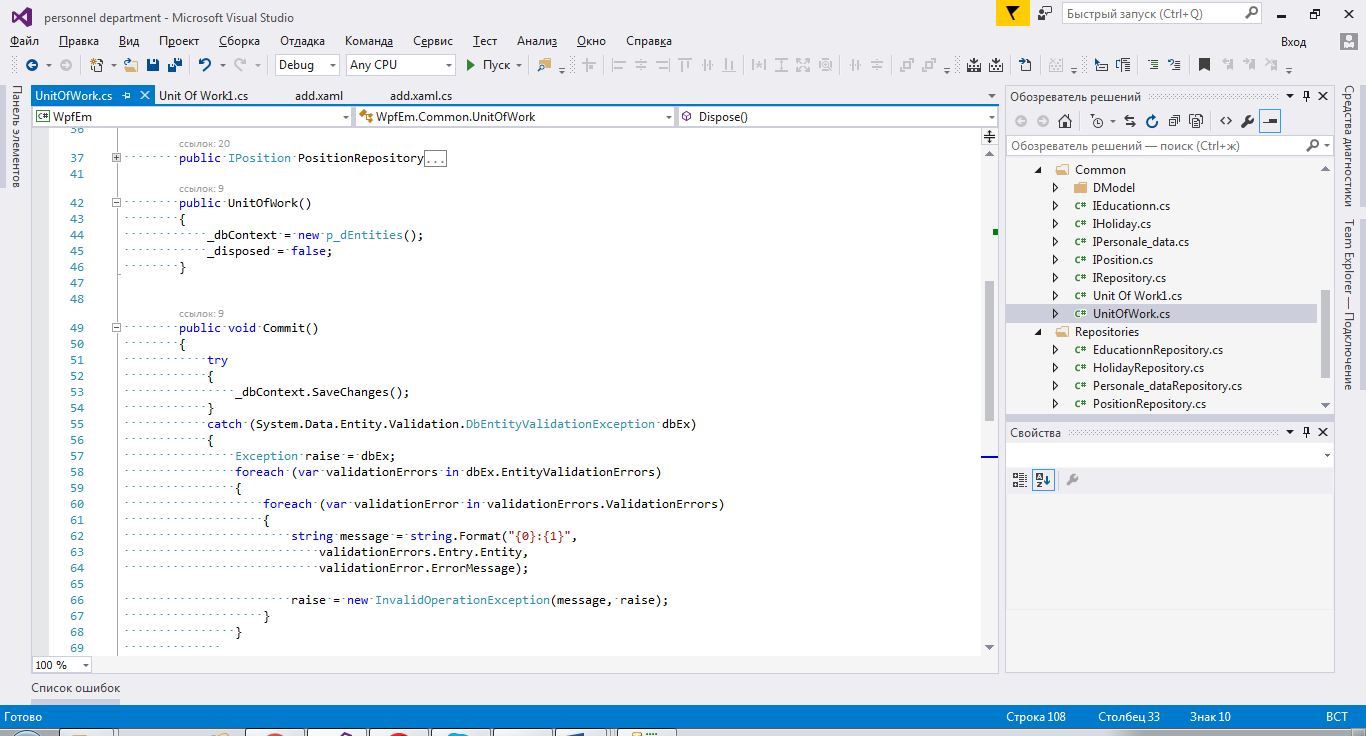


Рис 3.5 – Методы класса «Unit of Work».

Взаимодействие все этих классов происходит в классе «Repository».

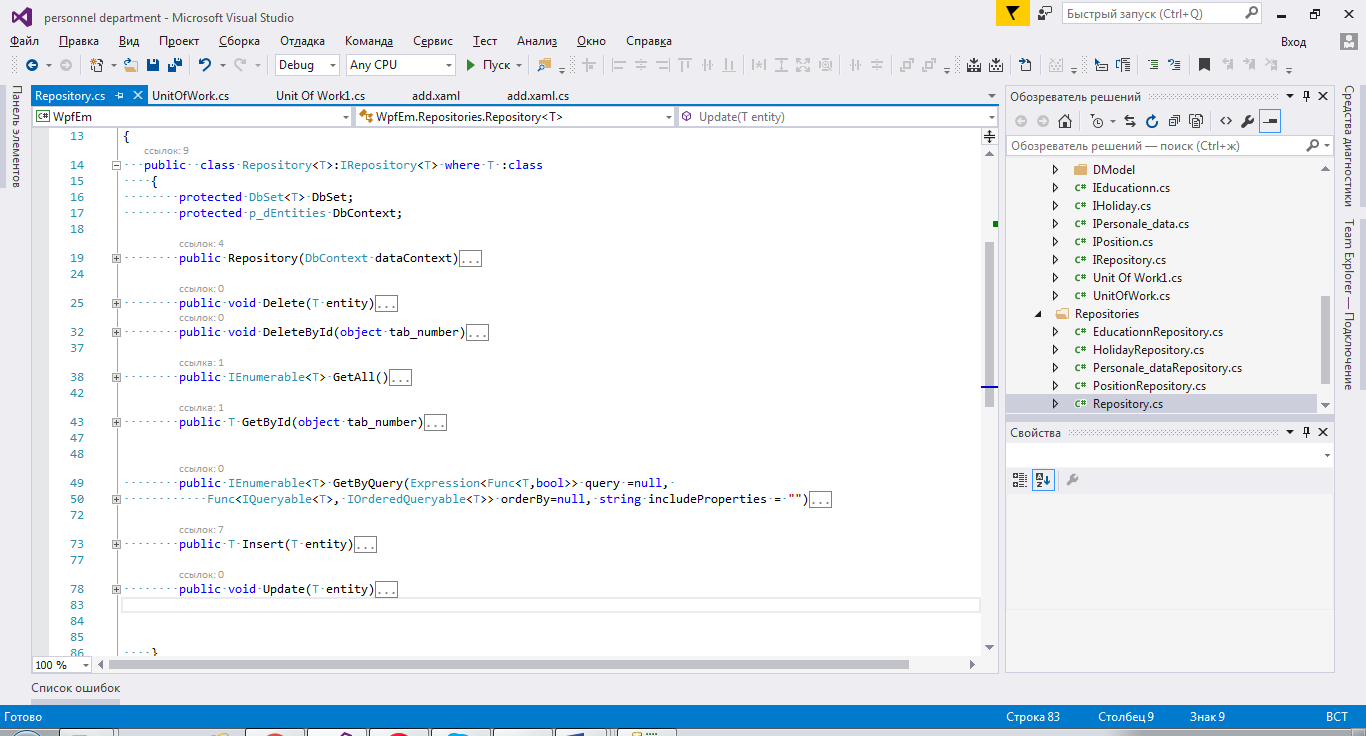


Рис 3.6 – Класс«Repository»

Также происходит запись данных в Excel-файл. Для этого была использована библиотека Epplus.

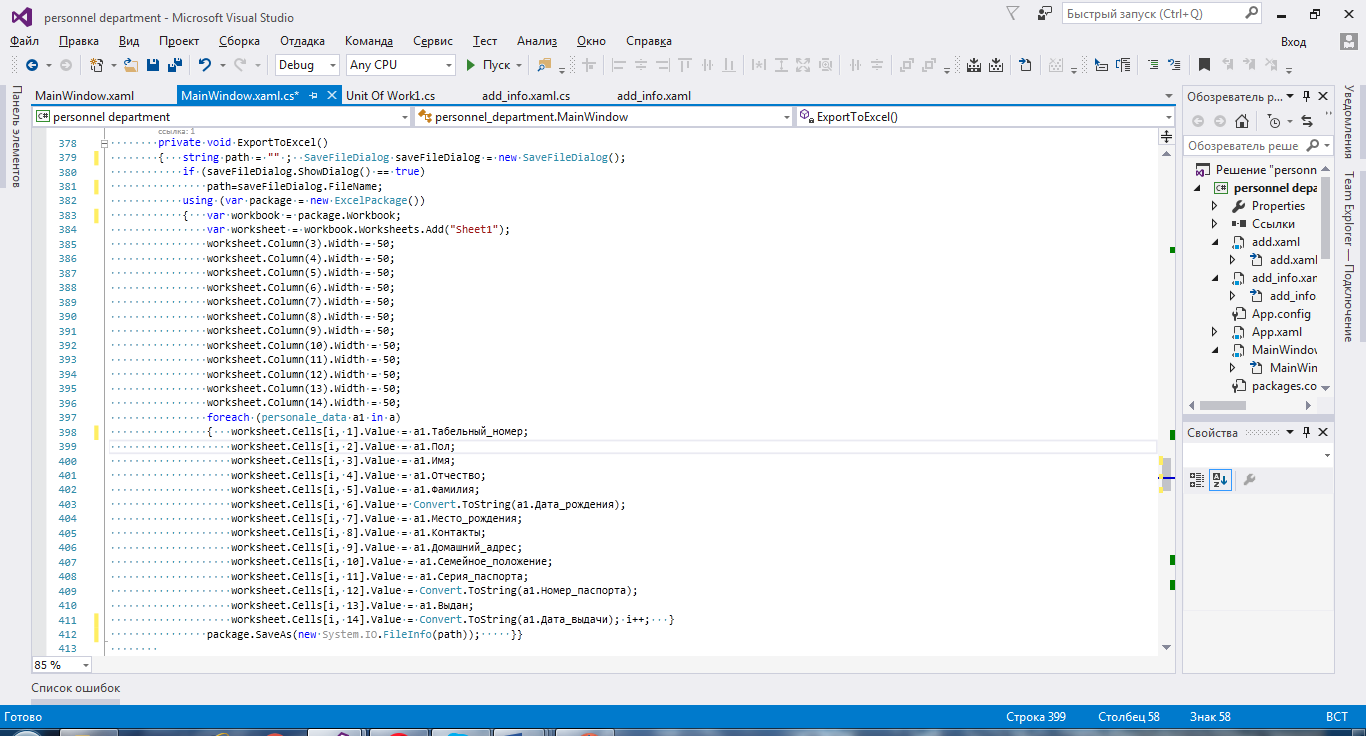


Рис 3.7 – Функция записи данных в Excel-файл.

# 4. Руководство пользователя

При запуске приложения открывается главная страница, на которой отображается весь функционал приложения

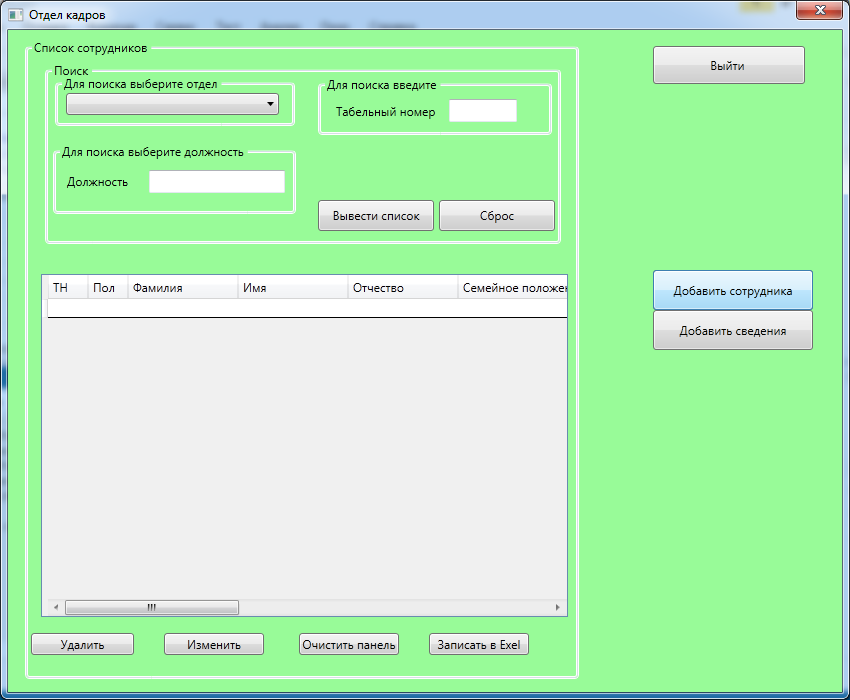


Рисунок 4.1 – Главное окно приложения

Главная форма подразделена на 3 зоны:

— поиск сотрудника;

— работа с данными на панели;

— добавление информации.

После заполнении полей из зоны поиска и нажатии кнопки «Вывести список», происходит поиск сотрудника, результаты которого отображаются на панели.

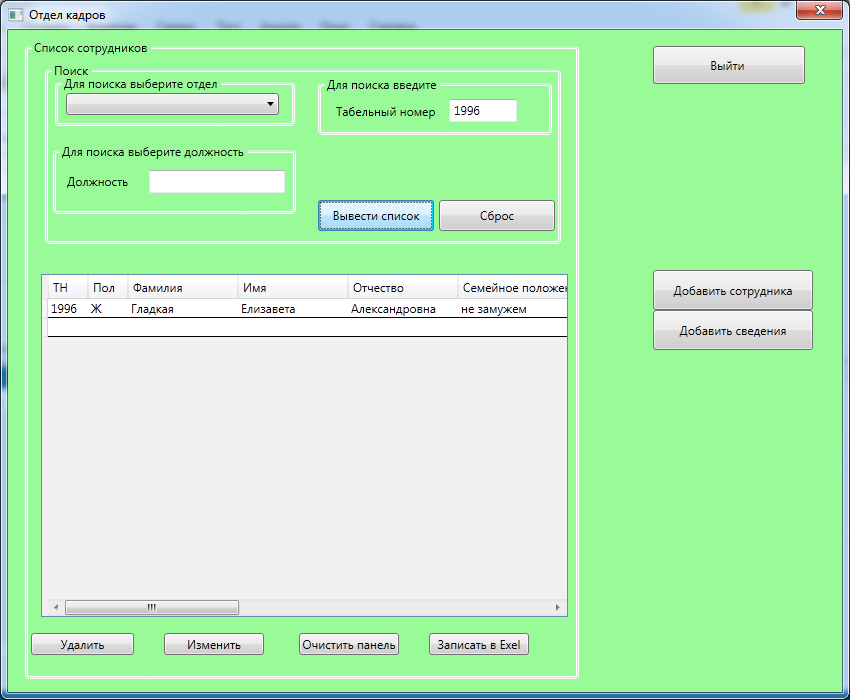


Рисунок 4. 2 – Результаты поиска

Кнопки «Удалить», «Изменить» работаю тогда, когда в панели выделена информация о сотруднике.

При дальнейшем нажатии на кнопку «Изменить» появляется окно с полями, куда вводится информация.

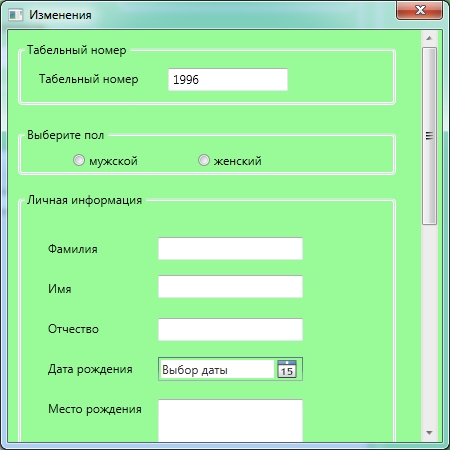


Рис 4.3 – Изменение базовой информации сотрудника.

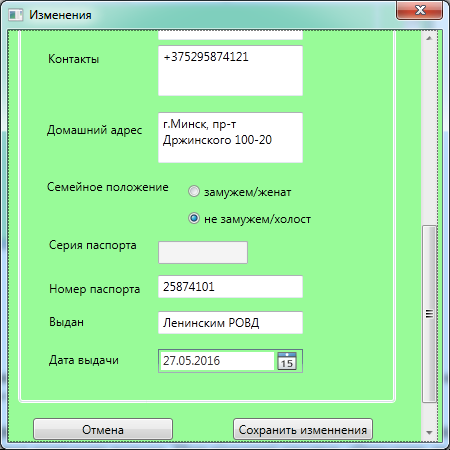


Рис 4.4 – Изменение базовой информации сотрудника.

При нажатии кнопки «Удалить» происходит удаление сотрудника из БД.

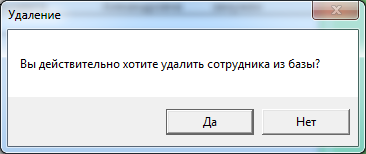


Рис 4.5 – Удаление сотрудника.

При нажатии на кнопку «Записать в Excel» данные с панели записываются в Excel-файл.

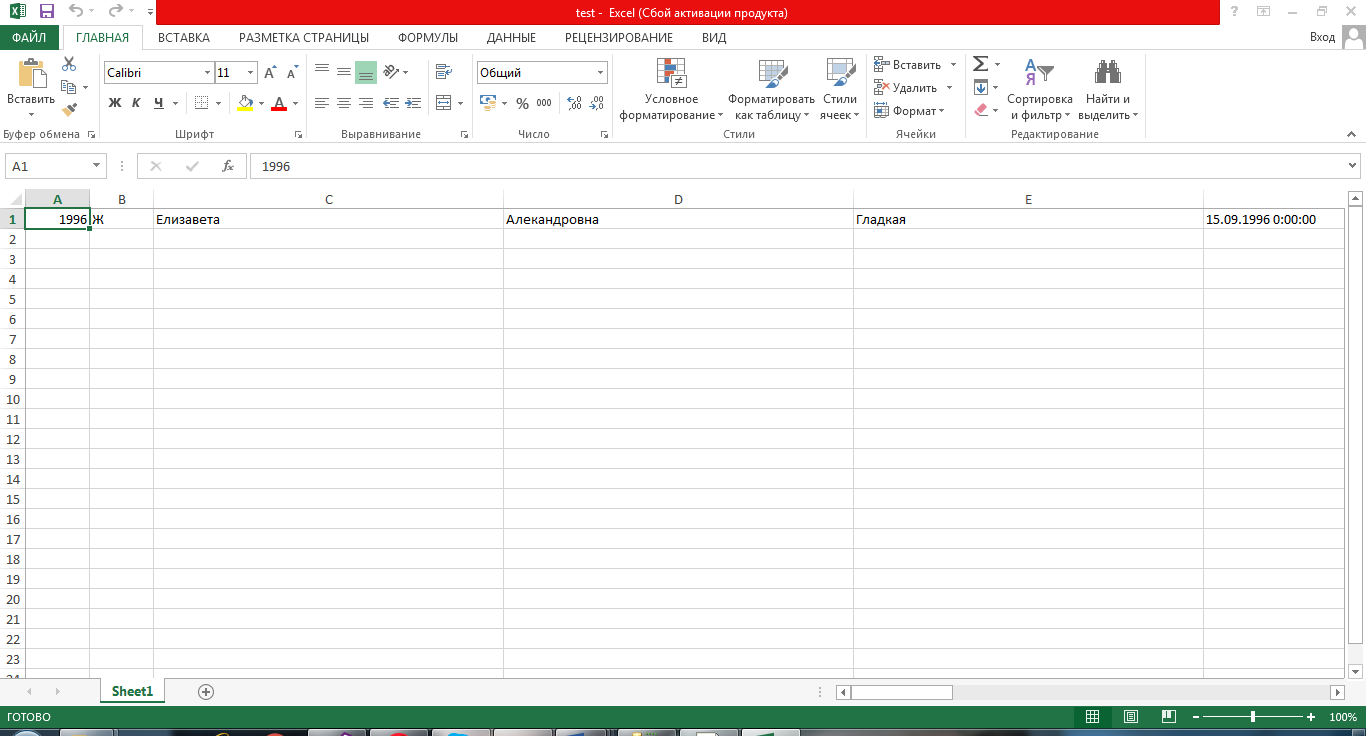


Рис 4.6 – Записанные данные в Excel файле.

При нажатии на кнопку «Сброс» происходит очистка полей, которые определяют критерии поиска. При нажатии кнопки «Очистить панель» панель, содержащая результаты поиска очищается. При нажатии кнопки «Добавить сотрудника» открывается окно с информационными поля, куда заносится базовая информация о новых сотрудниках. Оно представлено на рисунках 4.7 – 4.8

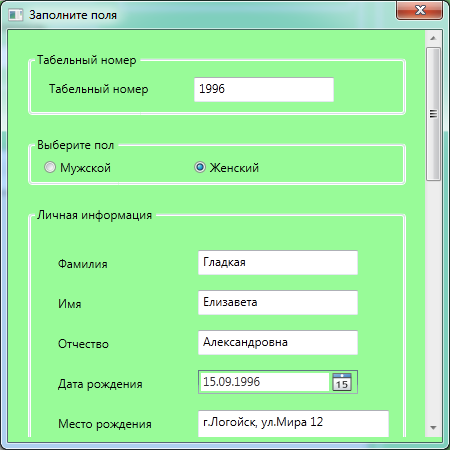


Рис.4.7 – Окно «Добавление сотрудника»

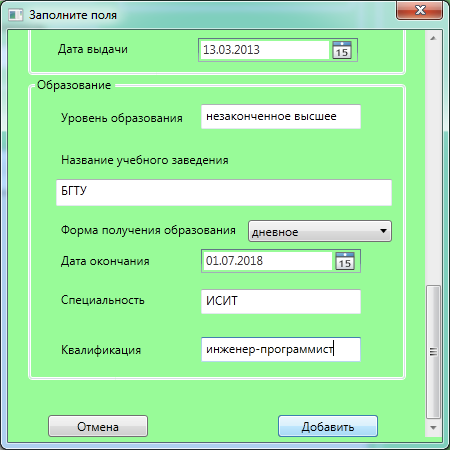


Рис.4.8 – Окно «Добавление сотрудника»

При нажатии на кнопку «Добавить сведения» открывается новое окно, где пользователь должен ввести табельный номер сотрудника, для которого будет вводить информацию.

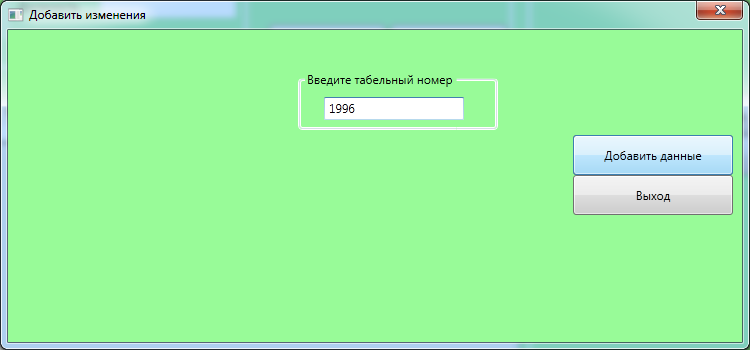


Рис.4.9 – Окно «Добавить изменения»

Происходит проверка БД на наличие сотрудника с заданным табельным номером. Если такого сотрудника нет, то выводится сообщение.

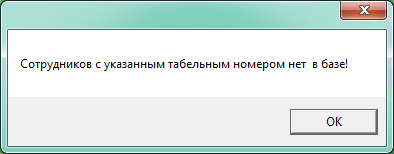


Рис 4.10 – Сообщение, что сотрудника нет в БД

. Если все-таки сотрудник, с указанным табельным номером есть в БД, то открывается окно со списком разделов, в которые пользователь может вводить информацию.

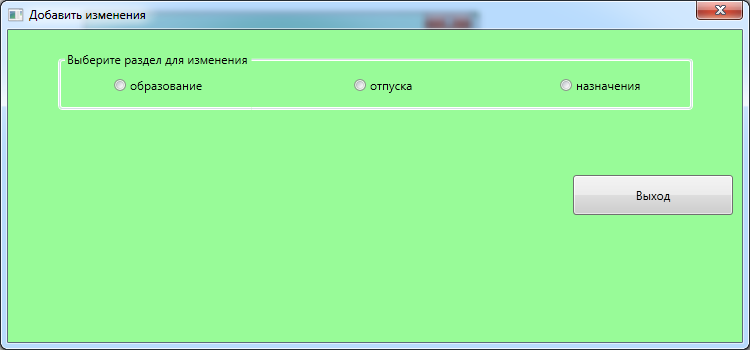


Рис 4.11 – Окно для выбора раздела.

При выборе раздела «Образование» появится форма с полями для заполнения информации об образовании.

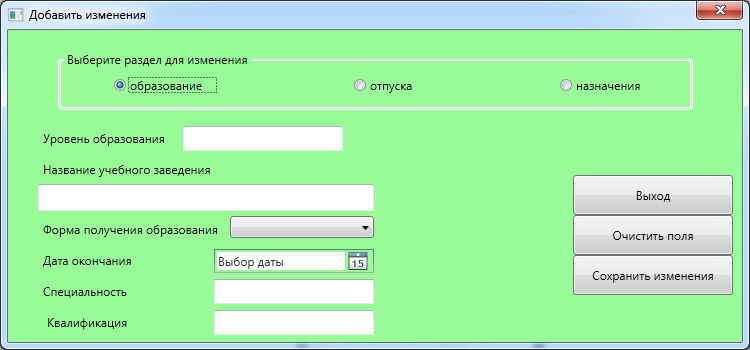


Рис 4.12 – Окно для ввода информации об образовании

При выборе раздела «Отпуска» появится форма с полями для заполнения информации о отпусках.

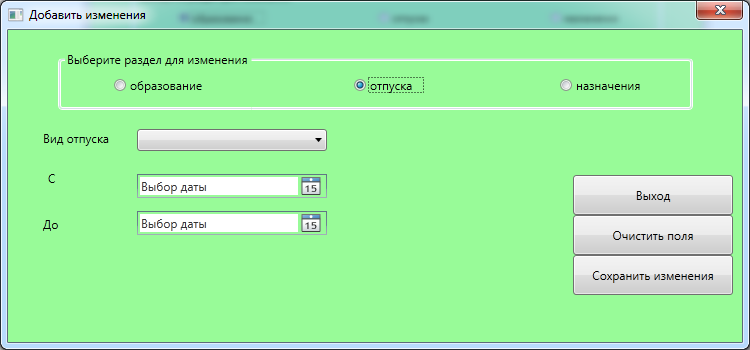


Рис 4.13 – Окно для ввода информации об отпусках

При выборе раздела «Назначения» появится форма с полями для заполнения информации о назначениях.

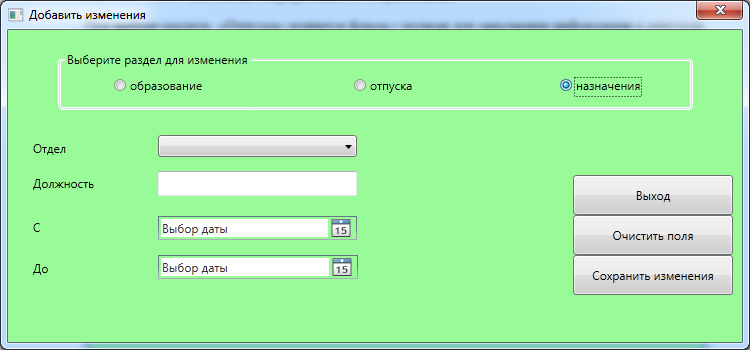


Рис 4.14 – Окно для ввода информации о назначениях.

На предыдущих 3 формах есть кнопки:

«Выход» - выход из этого окна без сохранения действий;

«Очистить поля» - очистка информационных полей соответствующего окна;

«Сохранить изменения» - сохранение введенных изменений в БД.

Для выхода из приложения можно использовать кнопку «Выйти» или крести в правом верхнем углу.

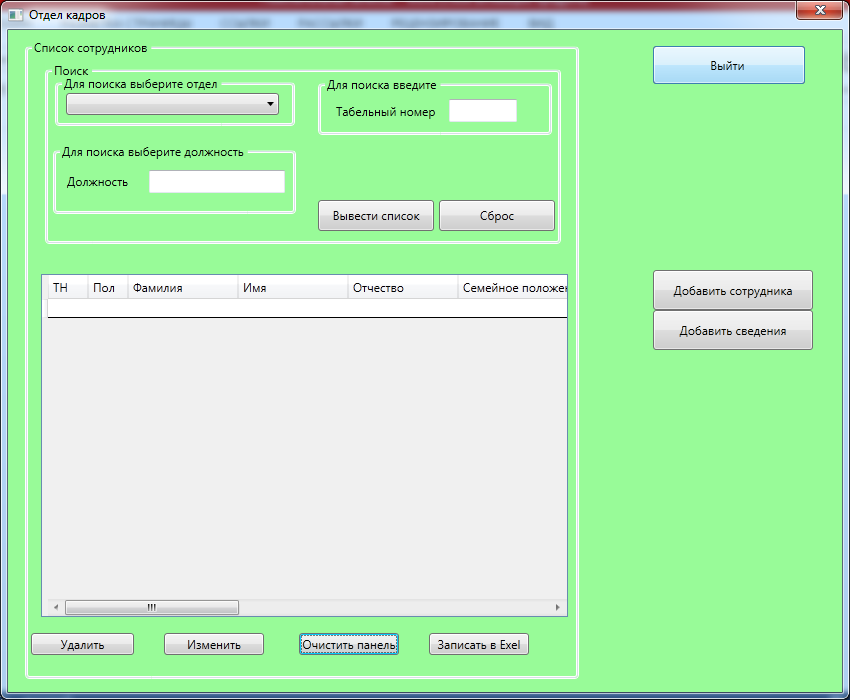


Рис 4.10 – Выход из приложения.

# 5.Тестирование

Тестирование проводится с целью проверки правильности функционирования программы и проверки имеющихся недостатков. Для выполнения тестирования необходимо проработать все возможные варианты действий пользователя, проверить, имеются ли в приложении какие-либо недостатки.

Для тестирования выполним следующие действия:

Запустить приложение. Откроется главное окно. Для поиска сотрудника в БД введем известный или известные критерии для поиска и нажмем на кнопку «Вывести список». Если сотрудник с заданным табельным номером, названием отдела, или названием должности не существует, то выводится окно на рисунке 5.1.

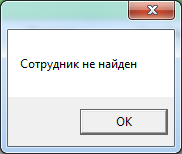


Рис 5.1 - Сообщение, что не сотрудников с указанными параметрами.

Если не выбран не один критерий, то выведется окно с сообщением ошибки

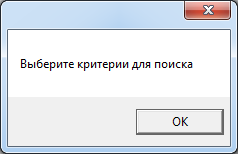


Рис 5.2 - Ошибка, не выбран не один критерий поиска.

При добавлении новых записей о существующих сотрудниках сразу осуществляется проверка на наличие данного сотрудника в БД. Если его нет, то выводится окно ошибки.

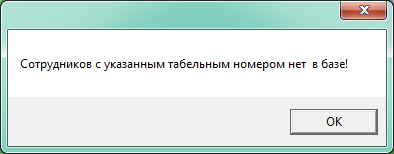


Рис5.3 - Ошибка, указанного сотрудника нет в базе.

При заполнении информационных полей осуществляется проверка на правильность введенных данных. Проверка также осуществляется, если поля не заполнены.

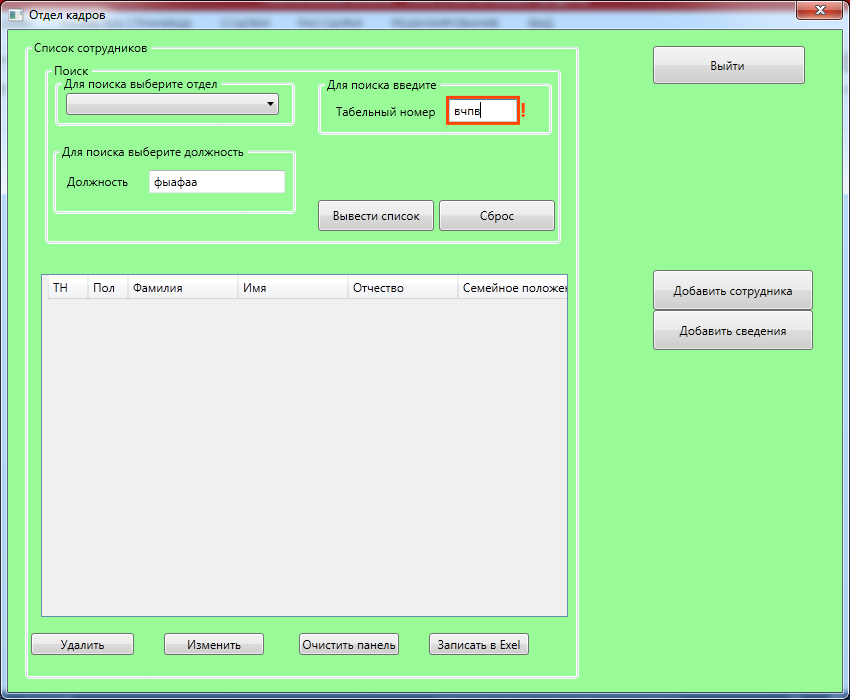


Рис 5.4 - Проверка на правильность ввода табельного номера.

При этом на данное поле стоит ограничение, так как количество цифр в табельном номере не может превышать четырех.

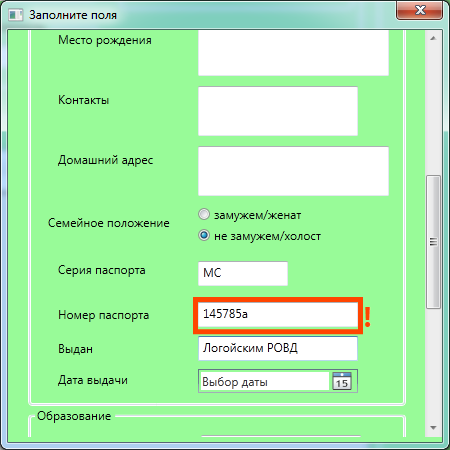


Рис 5.5 – Проверка на правильность ввода сведений о паспорте.

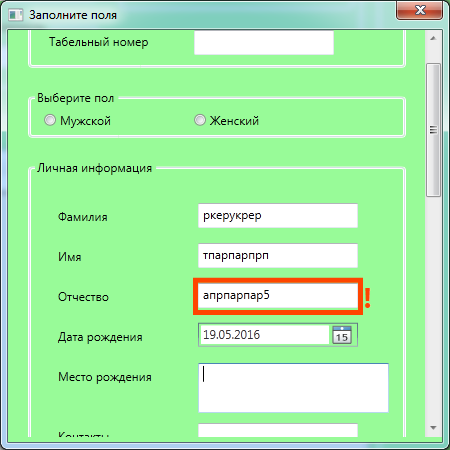


Рис 5.6 – Проверка правильности ввода личной информации.

Тестирование позволяет проверить правильность функционирования приложения, отследить существующие ошибки и проинформировать пользователя о них.

# Заключение

В курсовой работе описана и реализована программа, написанная на языке C#, с использованием технологий ADO.NET Entity Framework, Microsoft SQL Server, LINQ. Также были использованы паттерны «Repository» и «Unit of Work».

В ходе выполнения курсовой работы было создано приложение для управления персоналом предприятия, которое выполняет следующие функции:

— добавление сотрудников в базу данных;

— добавление сведений о сотрудниках;

— поиск сотрудников;

— удаление сотрудников из базы данных;

— запись сведений о сотрудниках в Excel-файл.

Данное программное средство имеет удобный и понятный интерфейс, реагирует на ошибочный ввод данных (например, ввод данных не того типа, проверка на пустоту вводимых значений).

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список используемых источников

1. MSDN сеть разработчиков в Microsoft [Электронный ресурс] —<http://msdn.microsoft.com/library/rus/>
2. Рихтер Д Программирование на платформе Microsoft .NET Framework СПб.: Русская редакция, 2005. 512 с
3. Язык C# и .NET Framework [Электронный ресурс] — <http://professorweb.ru/my/csharp/charp_theory/level1/infonet.php>
4. Рихтер Д Программирование на платформе Microsoft .NET Framework СПб.: Русская редакция, 2005. 512 с
5. Мэтью Макдональд — WPF: Windows Presentation Foundation в .NET 4.0 с примерами на C#

# Приложение А

Функция записи данных в Excel-файл.

private void ExportToExcel()

{

string path = "" ;

SaveFileDialog saveFileDialog = new SaveFileDialog();

if (saveFileDialog.ShowDialog () == true)

path=saveFileDialog.FileName;

using (var package = new ExcelPackage())

{

var workbook = package.Workbook;

var worksheet = workbook.Worksheets.Add("Sheet1");

worksheet.Column(3).Width = 50;

worksheet.Column(4).Width = 50;

worksheet.Column(5).Width = 50;

worksheet.Column(6).Width = 50;

worksheet.Column(7).Width = 50;

worksheet.Column(8).Width = 50;

worksheet.Column(9).Width = 50;

worksheet.Column(10).Width = 50;

worksheet.Column(11).Width = 50;

worksheet.Column(12).Width = 50;

worksheet.Column(13).Width = 50;

worksheet.Column(14).Width = 50;

foreach (personale\_data a1 in a)

{

worksheet.Cells[i, 1].Value = a1.Табельный\_номер;

worksheet.Cells[i, 2].Value = a1.Пол;

worksheet.Cells[i, 3].Value = a1.Имя;

worksheet.Cells[i, 4].Value = a1.Отчество;

worksheet.Cells[i, 5].Value = a1.Фамилия;

worksheet.Cells[i, 6].Value = Convert.ToString(a1.Дата\_рождения);

worksheet.Cells[i, 7].Value = a1.Место\_рождения;

worksheet.Cells[i, 8].Value = a1.Контакты;

worksheet.Cells[i, 9].Value = a1.Домашний\_адрес;

worksheet.Cells[i, 10].Value = a1.Семейное\_положение;

worksheet.Cells[i, 11].Value = a1.Серия\_паспорта;

worksheet.Cells[i, 12].Value = Convert.ToString(a1.Номер\_паспорта);

worksheet.Cells[i, 13].Value = a1.Выдан;

worksheet.Cells[i, 14].Value = Convert.ToString(a1.Дата\_выдачи);

i++;

}

package.SaveAs(new System.IO.FileInfo(path));

} }

Функция добавления сотрудника в БД

private void Click(object sender, RoutedEventArgs e)

{

using (var uow = new WpfEm.Common.UnitOfWork())

{

//создаем сотрудника

var E = new personale\_data()

{

Табельный\_номер = Convert.ToInt32(text\_box.Text),

Пол = gender,

Фамилия = textBox1.Text,

Имя = textBox2.Text,

Отчество = textBox3.Text,

Семейное\_положение = marital\_status,

Домашний\_адрес = textBox.Text,

Контакты = textBox6.Text,

Дата\_рождения = Convert.ToDateTime(date\_bir.SelectedDate),

Место\_рождения = textBox5.Text,

Номер\_паспорта = Convert.ToInt32(textBox28.Text),

Серия\_паспорта = textBox27.Text,

Выдан = textBox30.Text,

Дата\_выдачи = Convert.ToDateTime(data\_pas.SelectedDate)

};

var EEEE = new educationn()

{

Табельный\_номер = Convert.ToInt32(text\_box.Text),

Уровень\_образования = textBox20.Text,

Название\_учебного\_заведения= textBox23.Text,

Форма\_получения\_образования= Convert.ToString(comboBox.SelectionBoxItem),

Специальность= textBox25.Text,

Квалификация= textBox26.Text,

Дата\_завершения\_обучения= Convert.ToDateTime(data\_end\_stud.SelectedDate)

};

uow.Personale\_dataRepository.Insert(E);

uow.Commit();

uow.EducationnRepository.Insert(EEEE);

uow.Commit();

MessageBox.Show("Данные занесены в базу данных");

this.Close();

}

}