АННОТАЦИЯ

Отчет о курсовой работе: 39 с., 16 рис., 1 табл., 2 приложение, 6 источников.

Объектом исследования являются методы и способы построения баз данных.

Цель работы – разработка базы данных и приложения поиска, просмотра и редактирования данных детских учреждений.

Метод исследования – изучение принципов разработки и проектирования приложений средствами .NET Framework, принципов работы с базой данных MS SQL.

В работе были использованы технологии: C#, .NET, Visual Studio 2015, Windows Form, MSSQL.

В результате решения задачи была спроектирована база данных для дошкольного учреждения, а также windows-приложение для добавления, редактирования и удаления данных в вышеупомянутой базе.

Дальнейшее развитие программы связано с расширением ее возможностей, увеличением числа опций.

БАЗА ДАННЫХ, ПРИЛОЖЕНИЕ, WINDOWS FORM, VISUAL STUDIO, MS SQL, MS SQL VISUAL STUDIO TOOLS, .NET, FRAMEWORK, СУБД, ADO.NET, SQL

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мы живём в век информационных технологий, в век в котором техника может выполнять задачи, которые обременяют нас в повседневности. В настоящий момент существует масса предприятий и процессов, которые можно было бы автоматизировать и данных в них, которые бы можно было хранить. С этого следует, что процесс создания приложений и баз для каких-то предприятий становится все актуальнее с каждым днем. Поэтому целесообразным будет получить практические навыки в этой области. Отличным инструментом для этого будет задача создания базы данных и приложения для ее управления, которая бы позволила хранить данные дошкольного учреждения.

Целью курсовой работы является разработка базы данных для детского сада. Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Проанализировать предметную область.
2. Изучить технологии: .NET, Windows Forms, C#, Visual Studio 2015, MS SQL.
3. Составить техническое задание.
4. Разработать структуру программы и базы данных, разработать функционал программы, соответствующий техническому заданию.

1 Анализ предметной области

1.1 Состояние вопроса

На данный момент автоматизацией и информатизацией процессов занимаются уже давно. Этому процессу подвергаются все сферы, которые только могут. Не исключением стали и дошкольные учреждения, однако, и при существовании стабильных и отработанных ситам, как «1С: Дошкольное учреждение» все-равно большее количество детских садов использует бумажную документацию или бумажную документацию, перемешанную с электронной. При этом часто случаются ошибки, зачастую связанные с потерей документов или с невозможностью отследить историю записей и операций. При бумажном хранении данных сложно найти архивные данные, а также хранить их.

1.2 Описание существующих бизнес-процессов

Моделирование бизнес-процессов, в некотором смысле, определяют структуру нашей базы, а соответственно и приложения, поэтому является крайне важным этапом разработки. Также, в общем, моделирование позволяет бизнес-процессов позволяет оценить работу предприятия и в дальнейшем оптимизировать его работу.

Для описания текущих бизнес-процессов детского сада была выбрана нотация idef0, которая позволяет описать процессы в их функциональном аспекте. Главными компонентами моделей являются блоки и дуги. В свою очередь дуги и их вхождение обуславливают тип интерфейса:

- Если дуга входит в блок сверху – это управляющая информация
- Если дуга входит в блок слева – это входная информация
- Если дуга выходит из блока справа – это результаты
- Если дуга выходит из блока снизу – это механизм

В основе выбранной нотации лежит принцип о том, что каждый компонент модели может быть разложен на мелкие детали (декомпозирован) с целью детализации процесса, наглядного представления.

На рисунке 1 представлена контекстная диаграмма. Ее задача – отображение в целом процессов, которые описывают структуру и функции проекта. На данной диаграмме реализованы следующие интерфейсы:

- Управляющая информация – обучающий материал
- Вход – дети
- Механизмы – воспитатель, детский сад
- Результат – ребенок, готовый к обучению в школе

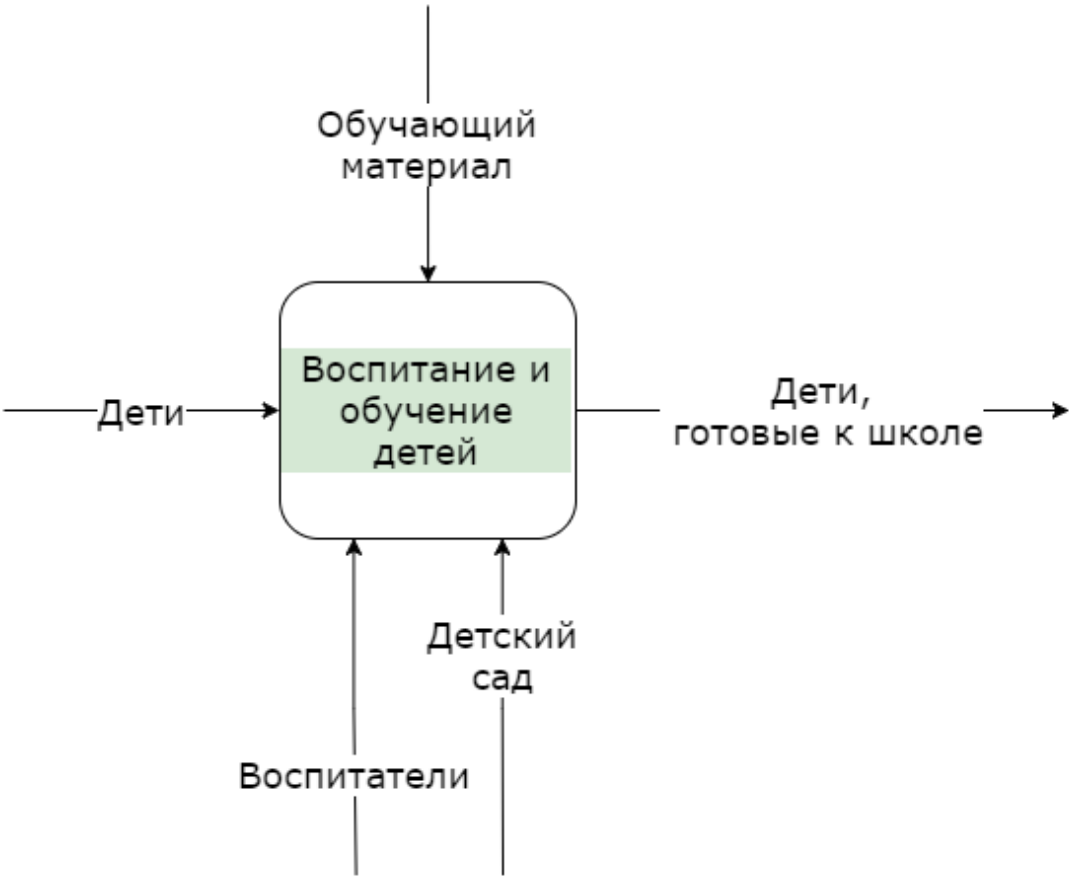

Рисунок 1

На рисунке 2 отображена декомпозиция контекстной диаграммы. Общий блок разбивается на следующие дочерние: медосмотр, тестирование определение в группу и обучение. Сначала ребенок проходит медосмотр, для этого необходимы документы и договор, после прохода медосмотра ребенок получает справку, которая является необходимым документом для тестирования, которое проводит психолог. Далее механизм распределения с участием родителей на основании тестов определяет ребенка в группу, а после определения в группу можно начинать обучение.

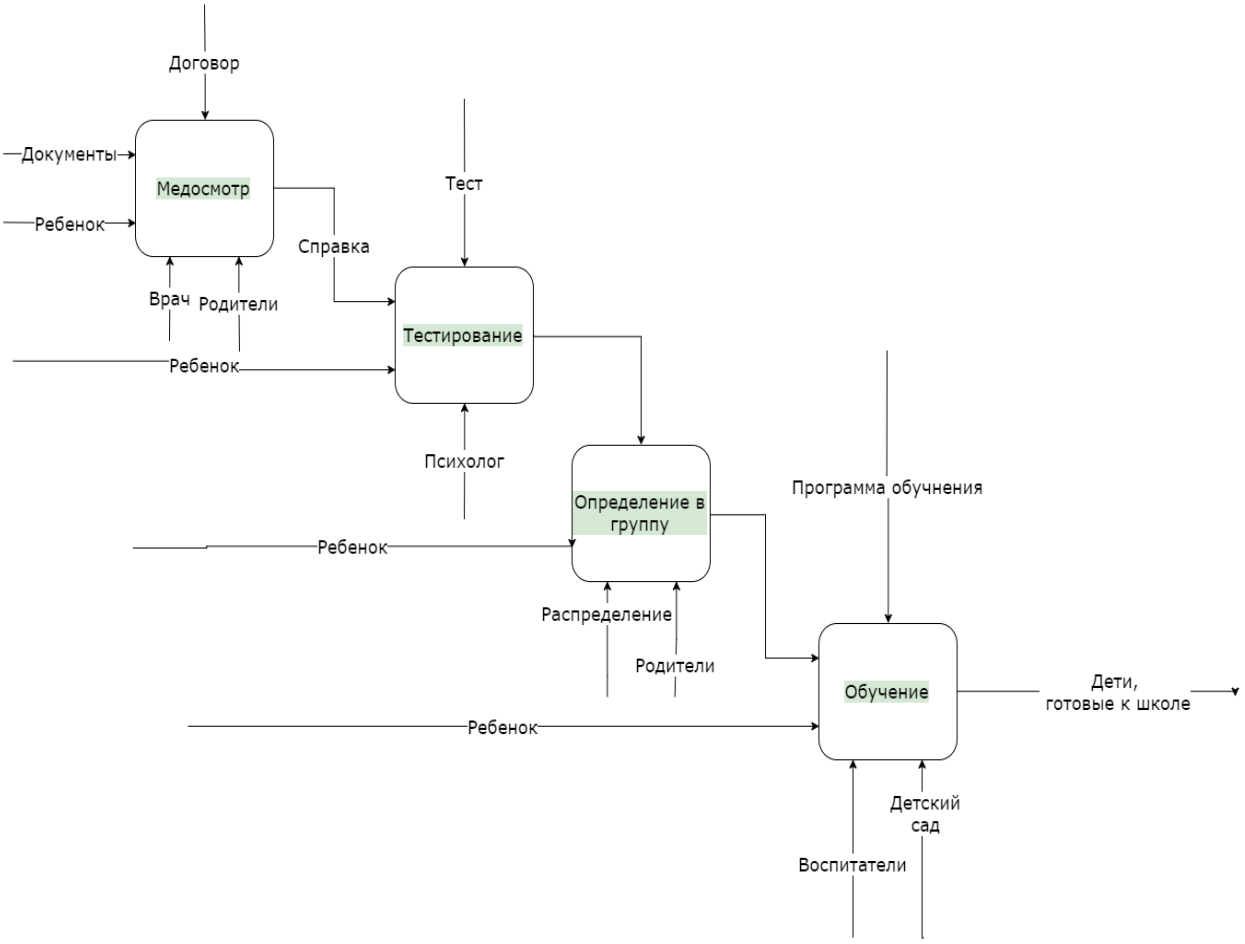

Рисунок 2

1.3 Актуальность и цель работы

На данный момент задача автоматизации бумажного документооборота важна и актуальна. Поскольку, еще большее количество дошкольных учреждений не пользуются преимуществами разработанных систем управления, то актуальным будет создать продукт, который бы в целом позволил облегчить манипулирование данными и при этом был бы легким для внедрения, поскольку, в целом, не менял бы бизнес-процессов системы.

Внедренное приложение позволит сократить канцелярские расходы на бумагу для печати, папок для хранения и так далее, а также может разграничить доступ к информации, при этом вся информация будет четко структурирован.

Целью курсовой работы является разработка базы данных дошкольного учреждения. Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Проанализировать предметную область.
2. Изучить технологии: MS SQL,.NET, Windows Forms, C#, Visual Studio 2015, Visual Studio MS SQL Tools.
3. Составить техническое задание.
4. Спроектировать и разработать базу данных, заполнить ее начальными данными.
5. Согласно ТЗ разработать приложение для работы с базой дошкольного учреждения.
6. Проанализировать результаты работы, оформить отчет.

2 Техническое задание

2.1 Общие требования к продукту

Поскольку, приложение с базой берутся хранить данные, то следует определиться с данными, которые будут храниться. Можно выделить следующие данные, которое можно хранить в базе:

- Данные о родителях
- Данные о детях
- Данные о сотрудниках
- Список должностей
- Список групп

При запуске приложения пользователя встречает окно для авторизации, где он может зайти либо как гость и просматривать данные, либо как сотрудник, но для этого нужно ввести логин и пароль, который можно получить у заведующей. Не каждый сотрудник должен иметь доступ ко всем данным в базе (возможность редактировать данные), поэтому база хранить данные для авторизации пользователя в приложении.

2.2 Позиционирование продукта

2.2.1 Требование к пользовательским интерфейсам

Интерфейс приложения должен обладать простым, понятным интерфейсом, который должен быть понятен для людей, которые имеют небольшой опыт работы с компьютером. Это делает обязательным наличие:

- Группировки элементов в зависимости от функциональной нагрузки
- Наличие инструкции использования программы
- Наличие подсказок при наведении на элементы управления

2.2.2 Требования к программным интерфейсам

Код программы должен:

- Быть понятным для чтения другими разработчиками
- Содержать комментарии
- Содержать минимум повторяющегося кода
- Давать названия переменным так, чтобы можно было определить функциональную нагрузку

Следует отделить работу с базой данных от логики приложения в отдельный класс.

2.2.3 Требования к базе данных

База данных должна содержать данные о родителях, детях, группах, должностях и сотрудниках. Также потребуется обеспечить связи между таблицами. База данных должна иметь возможность хранить большие объемы данных и обеспечить удовлетворительную скорость работы.

2.2.4 Требования к пользователям продукта

Пользователи продукта должны иметь базовые знания компьютерной грамотности, понимать свои обязанности и в тайне хранить выданные им комбинации логина и пароля для безопасности данных. Также, пользователи должны знать, куда можно обратиться в случаях некорректной работы программы или базы.

2.3 Функции продукта

Приложение должно стабильно работать и предоставлять следующие функции:

- Вход с авторизацией и без
- Удобный просмотр информации, с возможность сортировки, группировки и поиска
- Добавление, удаление и хранение данных

2.4 Входные и выходные данные

Входными данными для создания приложения является схема существующих бизнес-процессов и механизмы регистрации и обучения в детском саду, а входными данными для приложения является пользовательский ввод, который представляет из себя данные в виде строк, чисел и файлов.

Выходными данными разработки является само приложение с базой, а для пользователя выходными данными являются отображение данных, существующих в базе.

3 Реализация программного продукта

3.1 Обоснование средств разработки

3.1.1 Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Одним из используемых инструментов Visual Studio в данной КР является инструмент «Microsoft Visual Studio 2015 Installer Projects». Данный инструмент позволит создавать установочный файл для данного приложения.

3.1.2 .NET Framework

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

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL). 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. Данные процессы видны на рисунке 3.

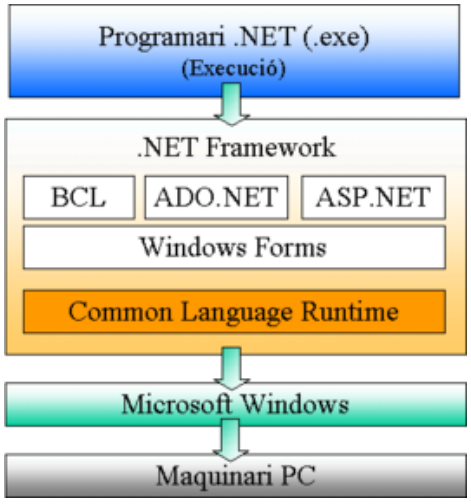

Рисунок 3

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.

3.1.3 Windows Forms

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.

3.1.4 ADO.NET

Платформа .NET определяет ряд пространств имен, которые позволяют непосредственно взаимодействовать с локальными и удаленными базами данных. Вместе эти пространства имен известны как ADO.NET. Библиотеки ADO.NET можно применять тремя концептуально различными способами: в подключенном режиме, в автономном режиме и с помощью технологии Entity Framework. При использовании подключенного уровня (connected layer), кодовая база явно подключается к соответствующему хранилищу

данных и отключается от него. При таком способе использования ADO.NET обычно происходит взаимодействие с хранилищем данных с помощью объектов подключения, объектов команд и объектов чтения данных.

Таблица 1 – Основные объекты ADO.NET

Тип объекта	Базовый класс	Соответствующие интерфейсы	Назначение
Connection	DbConnection	IDbConnection	Позволяет подключаться к хранилищу данных и отключаться от него. Кроме того, объекты подключения обеспечивают доступ к соответствующим объектам транзакций
Command	DbCommand	IDbCommand	Представляет SQL-запрос или хранимую процедуру. Кроме того, объекты команд предоставляют доступ к объекту чтения данных конкретного поставщика данных
DataReader	DbDataReader	IDataReader, IDataRecord	Предоставляет доступ к данным только для чтения в прямом направлении с помощью курсора на стороне сервера

Окончание таблицы 1

DataAdapter	DbDataAdapter	IDataAdapter, IDbDataAdapter	Пересылает наборы данных из хранилища данных к вызывающему процессу и обратно. Адаптеры данных содержат подключение и набор из четырех внутренних объектов команд для выборки, вставки, изменения и удаления информации в хранилище данных
Parameter	DbParameter	IDataParameter, IDbDataParameter	Представляет именованный параметр в параметризованном запросе
Transaction	DbTransaction	IDbTransaction	Инкапсулирует транзакцию в базе данных

В ADO.NET термин "объект подключения" на самом деле относится к конкретному типу, порожденному от DbConnection; объекта подключения "вообще" нет. То же можно сказать и об "объекте команды", "объекте адаптера данных" и т.д. По соглашению имена объектов в конкретном поставщике данных имеют префиксы соответствующей СУБД (например, SqlConnection, OracleConnection, SqlDataReader и т.д.).

3.1.5 Использование языка программирования C#

В данной курсовой работе был выбран благодаря тому, что язык имеет компонент Windows Form, может быть использован в .Net Framework, и в общем позволяет быстро создавать приложения под windows и не только.

По мимо вышесказанных достоинств, ЯП C# имеет ряд других преимуществ. Вот, например, преимущества C# по книге Биллига:

- создавался параллельно с каркасом Framework .NET и в полной мере учитывает все его возможности - как FCL, так и CLR;
- является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
- является мощным объектным языком с возможностями наследования и универсализации;
- является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от C++ к C#;
- сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства C++ как указатели, адресация, разыменование, адресная арифметика;
- благодаря каркасу Framework .NET, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;
- мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-

службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;

- реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

3.1.6 SQL Server Data Tools

SQL Server Data Tools (SSDT) - это окончательное имя продукта, который ранее назывался SQL Server Developer Tools с рабочим названием «Juneau». Средства SSDT предоставляют передовые возможности работы для разработчиков баз данных SQL Server и SQL Azure. Появление SQL Server Data Tools (SSDT) изменило разработку баз данных благодаря внедрению универсальной декларативной модели, охватывающей все этапы разработки базы данных в среде Visual Studio (рисунок 14). Возможности SSDT Transact-SQL помогают в сборке, отладке, обслуживании и реструктурировании баз данных. Можно работать как с проектом базы данных, так и непосредственно с подключенным экземпляром базы данных (как на собственной площадке, так и в облаке).

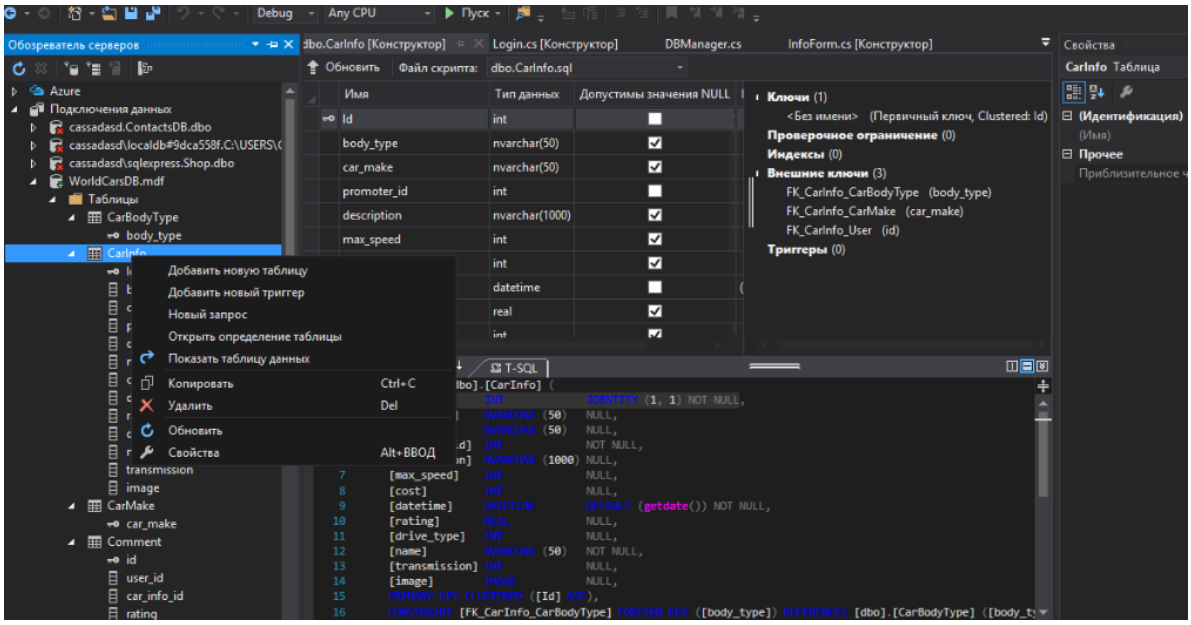

Рисунок 4

3.2 Проектирование БД

3.2.1 Концептуальное проектирование

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД или модель данных.

Назначение концептуальной модели состоит в том, чтобы представить формализованную информацию о предметной области таким образом, чтобы она было достаточно емкой для оценки глубины и корректности проработки проекта базы данных. В данном проекте были выделены следующие сущности, необходимые для решения задачи:

- Родитель
- Ребенок
- Сотрудник
- Группа
- Должность

Родители воспитывают детей, дети состоят в группах и имеют родителей, группы характеризуются учебным планом и сотрудниками (воспитателями и нянями), а сотрудники – должностью. Графически это можно увидеть на рисунке 5.

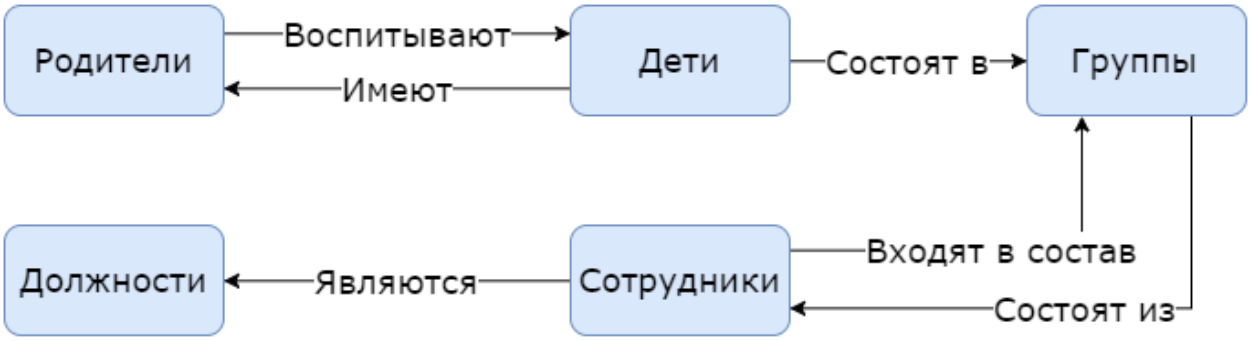

Рисунок 5

3.2.2 Логическое проектирование

В результате концептуального проектирования была создана схема базы данных с выделенными в ней объектами и отношениями между ними. Можно заметить, что между некоторыми объектами двухсторонняя связь, при проектировании логической базы это значит, что данные объекты связаны между собой отношением многие-ко-многим, это означает, что требуется создать дополнительные таблицы для организации данной связи.

Для хранения пользователей нужно создать дополнительную таблицу, в которой будут хранится логин и пароль для авторизации в приложении, что привело к результатам, которые можно увидеть на рисунке 6.

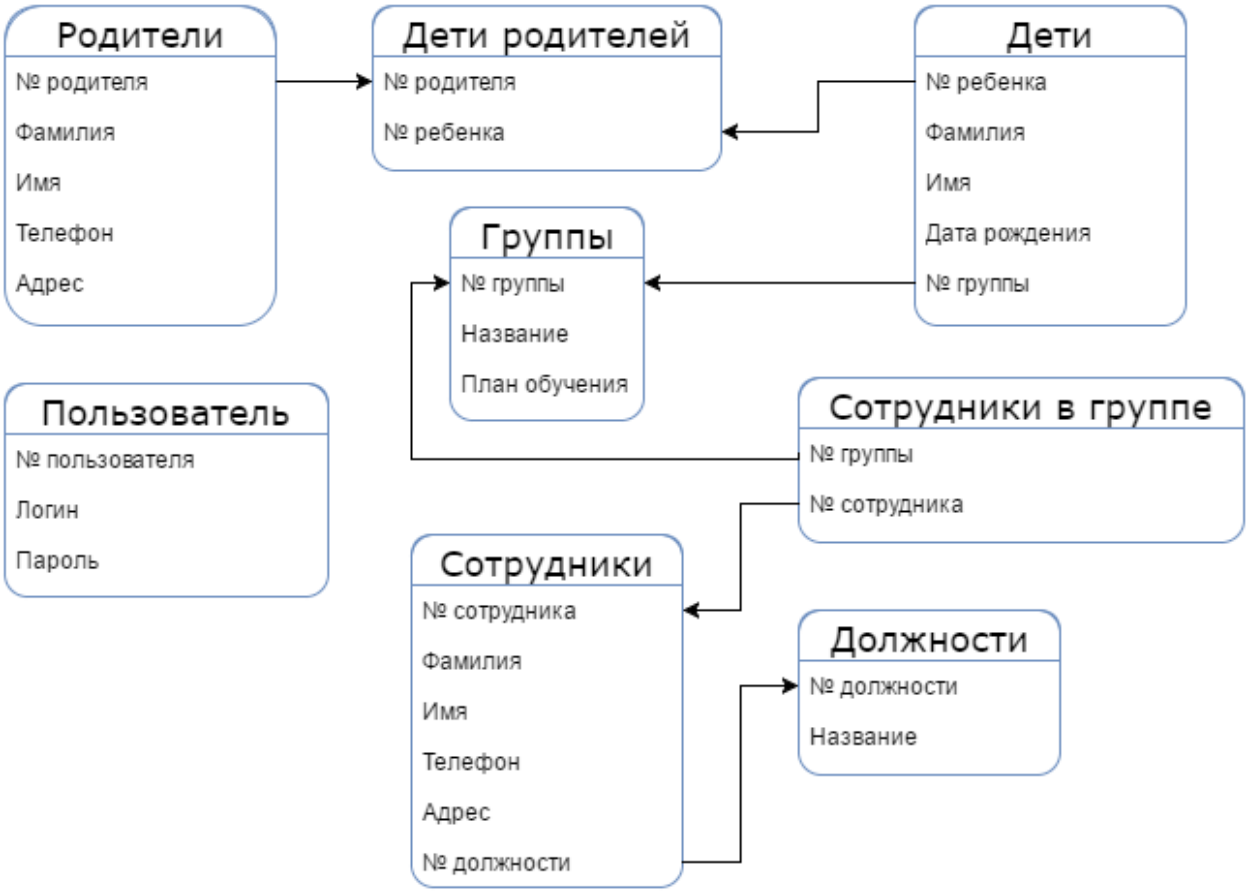

Рисунок 6

3.2.3 Физическое проектирование

Физическое проектирование базы данных подразумевает процесс подготовки переноса реализации базы данных на определенную СУБД. На изображении ниже представлена физическая схема базы данных для курсового проекта. Для построения данной схемы было выбрано веб-приложение DbDesigner. На картинке 7 изображена физическая схема базы данных.

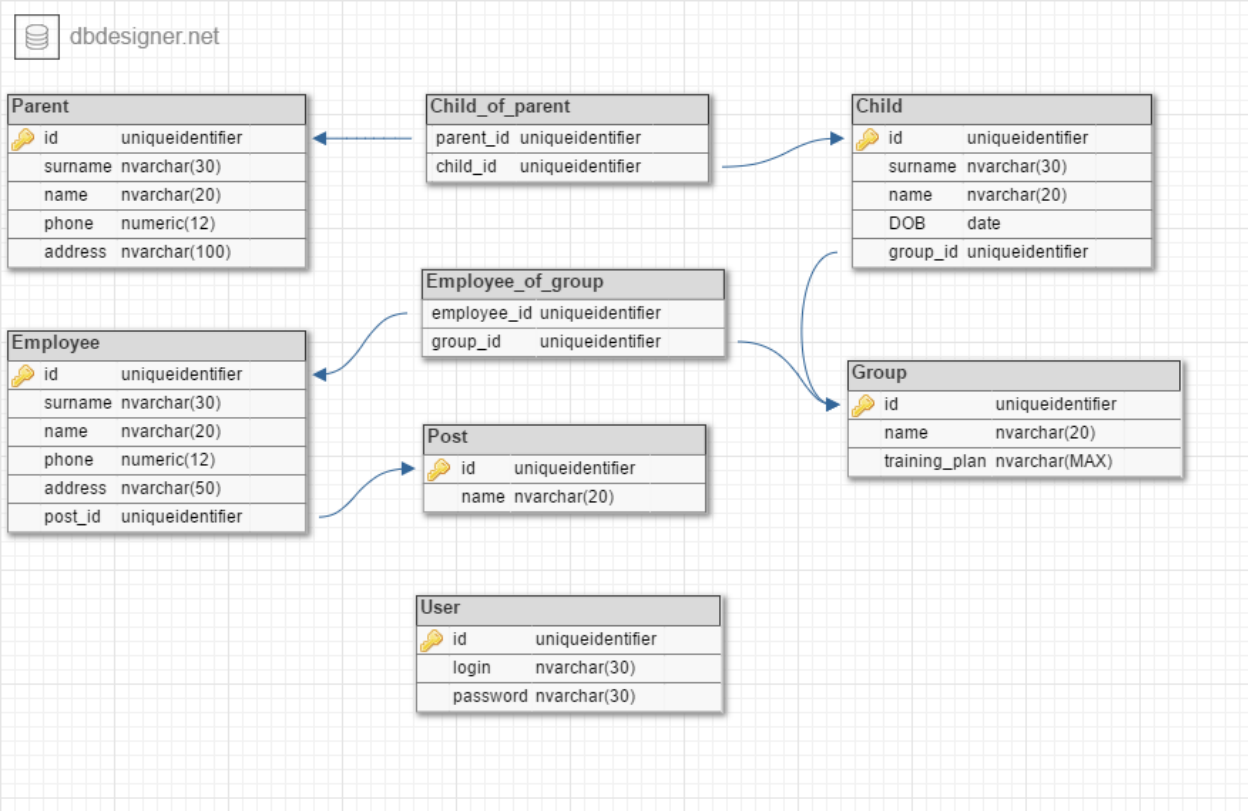

Рисунок 7

Также, данный веб-сайт имеет меню импорта (рисунок 8), которое позволяет создавать скрипты создания и удаления базы для СУБД MS SQL на основе имеющейся схемы БД.

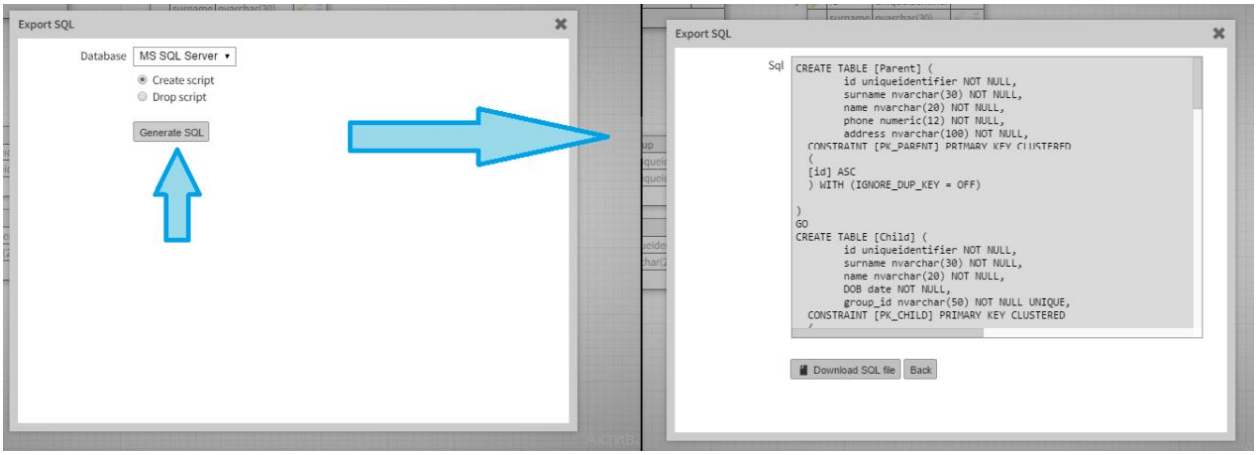

Рисунок 8

После создания скрипта на веб-сайте, создаем базу данных. Для этого в проекте создадим новый элемент «База данных, основанная на службах», после чего открываем двойным кликом и база будет открыта с помощью MS SQL Visual Studio Tools. Нажимаем на базу в обозревателе серверов правой кнопкой и создаем запрос, в который копируем текст, сгенерированный сайтом. Выглядит это как на картинке 9. После этого в папке с проектом находится готовая для использования база данных, которую мы спроектировали.

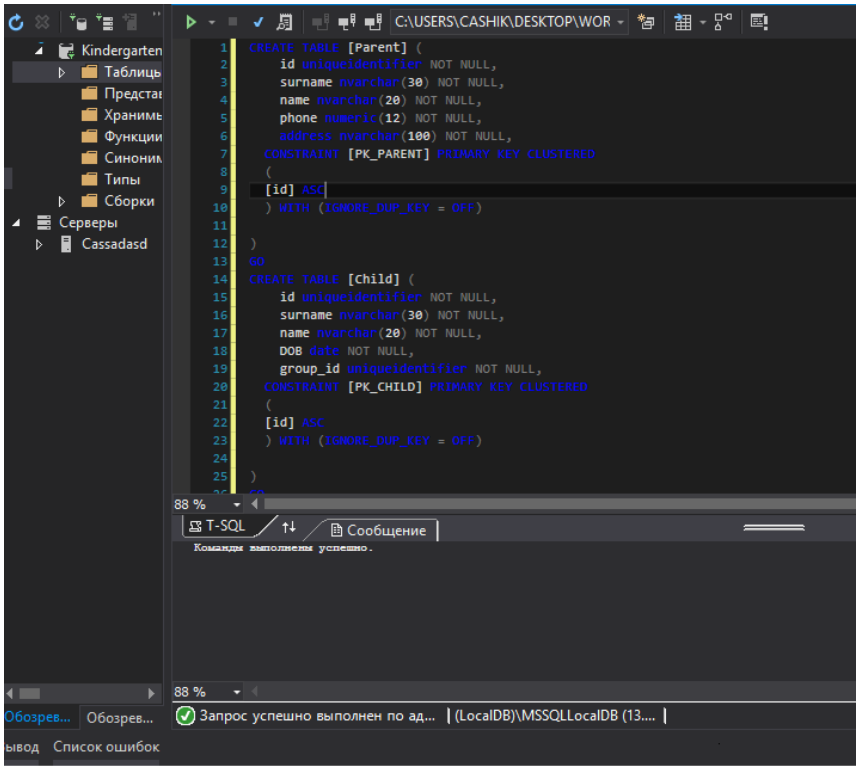

Рисунок 9

3.3 Проектирование приложения

3.3.1 Создание интерфейса

Поскольку была использована технология Windows Form, для создания интерфейсной части приложения требовалось только создать необходимые формы и настроить их свойства.

Для начала, следует создать главную форму приложения. В ней создадим панель управления (MenuStrip), в которой будет меню «Справочники» и «Справка», при этом первое меню будет иметь дочерние меню: Дети, Родители, Группы, Сотрудники, Должности. Чтобы добавить их, нужно нажать на пустые места, которые видно только при редактировании, нажав на которое можно добавить еще элементов меню (рисунок 10).

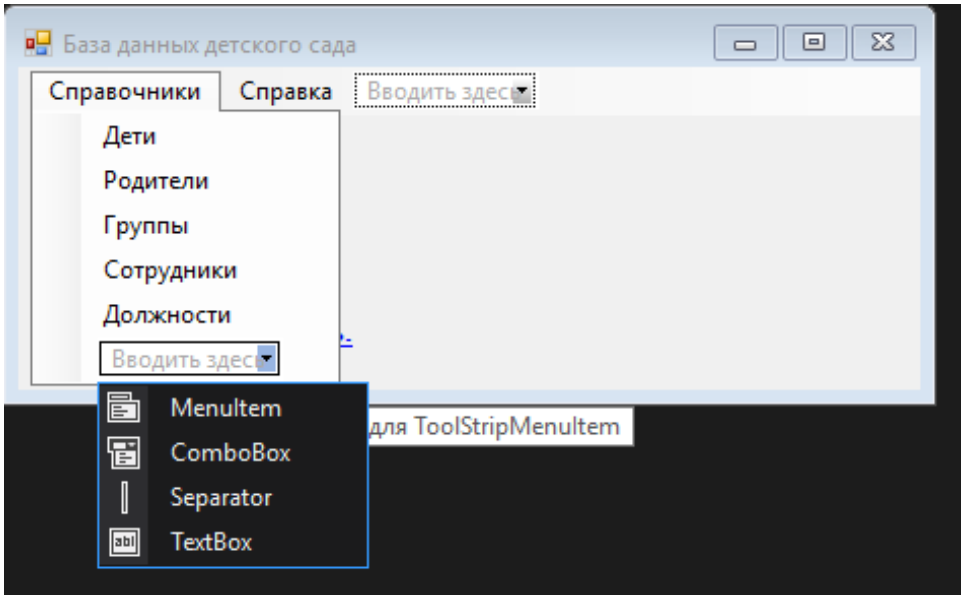

Рисунок 10

Каждое из дочерних меню будет вызывать отдельное диалоговое окно для редактирования соответствующих данных. Это обусловлено тем, как WF работает с базой данных.

Для отображения текущего пользователя создадим два текстовых

элемента, один из которых будет содержать приветствие, а другой – авторизирован ли пользователь.

После этого создадим окна для работы с данными. Для работы с данными групп добавим элемент DataGridViev в окно, и привяжем его к таблице нашей базы данных. После чего настроем отображаемые поля: поставим соответствия между полями таблицы в приложении и базе, спрячем поля идентификаторов. Для окна по работе с должностями делаем аналогичные действия.

Для окон сотрудников и детей сделаем те же действия, что и для предыдущих окон, однако для них вид таблицы должен усложниться, поскольку нужно организовать в них привязку к другой таблице, то есть сделать возможность пользователя выбрать группу для ребенка или должность для сотрудника. Для этого сделаем новое поле в таблице приложения типа DataGridViewComboBoxColumn, и укажем какие поля служат для связывания таблиц, и какое поле с внешней таблицы нужно отображать (рисунок 11).

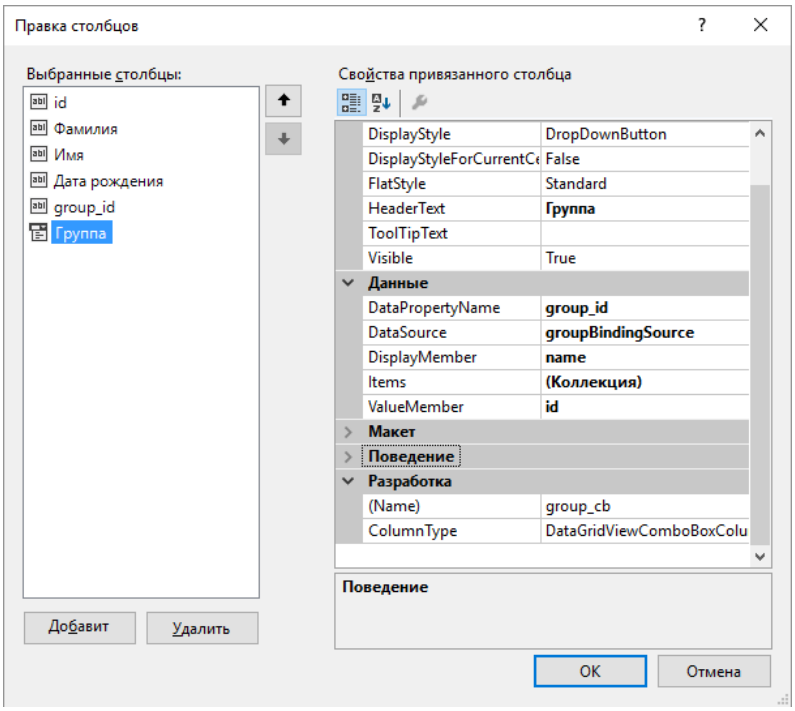

Рисунок 11

Для манипулирования данными родителей не хватит вышеуказанных элементов, так как нужно организовать связь между родителями и детьми. Поскольку, у родителя может быть несколько детей, то создадим дополнительную таблицу в приложении, в которой будут отображаться дети родителя. Для того, чтобы можно было как-то связать родителя и ребенка добавим кнопку связывания и отвязывания ребенка и родителя из разных DataGridView окна. Также, создадим группу из двух переключателей RadioButton, при переключении которых будут изменяться вариант отображения детей родителя (дети выбранного родителя и не его дети), что позволит связывать родителя и ребенка и избавится при ошибке связать уже связанных.

Для окна справки создадим элемент TabControl и в нем две TabPage, которые позволят содержать информацию об использовании приложения и разработке на разных страницах, при этом можно будет переключатся между ними и не созывать отдельных окон. Информация в этих окнах будет содержаться статически из текстовых элементов Lable.

Также, во всех кнопках для работы с базой нужно создать кнопку, которая будет сохранять данные в базу, а также текстовые элементы, для подписи окон. Финальный вид всех окон приложения можно посмотреть в приложении Б.

3.3.2 Создание логики приложения

Для авторизации, создадим обработчик клика (двойным нажатием на текст) и пропишем там создание модального окна авторизации. Переменная authorization поменяет значение на true, если модальное окно авторизации вернет результат «OK», значение которого прописано в классе ModalResult. Также, при удачной авторизации текст «Гость» измениться на «Авторизированный пользователь». При повторном нажатии пользователем на этот текстовый элемент значение authorization станет false, а значение

текста в данном текстовом поле изменится на «Гость».

При клике любое дочернее меню в «Справочники» создается модальное окно, в конструктор которого отправляется значение переменной authorization. Если переданное значение равно false, то в конструкторе окна все кнопки сохранения и кнопка связывания изменяют свойство Visible на false, а все DataGridView – свойство ReadOnly на true, что избавит пользователя от возможности редактировать данные в приложении и базе.

При нажатии на кнопку сохранения происходят следующие действия: в элементах управления оканчивается ввод и данные меняются на актуальные, в случае существования изменений данные отправляются измененные данные. Фрагмент подобного кода для окна с данными о детях можно увидеть на рисунке 12.

```
ссылка: 1
private void saveChildren_Click(object sender, EventArgs e)
{
    // редактирование заканчиваются
    // и в таблицу заносятся актуальные данные
    children_dsv.EndEdit();
    childBindingSource.EndEdit();

    // получаем все изменения
    DataSet changes = this.kindergartenDBDataSet.GetChanges();

    if (changes != null)
    {
        // в случае сущестования изменений
        // отсылаем изменения в базу через адаптер
        // и перезаполняем таблицу в приложении
        childTableAdapter.Adapter.Update(changes);
        childTableAdapter.Fill(kindergartenDBDataSet.Child);
    }
}
```

Рисунок 12

Логика в окне управления данными о родителях построена на двух элементах выбора «Дети родителя», «Остальные дети». От них зависит логика заполнения данных во втором DataGridView окна и логика работы кнопки «Связать/Отвязать выбранных родителя и ребенка». Последняя, добавляет запись в таблицу Child_of_parent, когда выбран первый переключатель и удалят, если включен второй. Также, при выборе любого из них таблицы с детьми заполняются новыми данными.

4 Описание программного продукта

4.1 Описание объектов и их взаимодействия

В программе существуют объекты форм и объекты для работы с базой данных. Все эти объекты содержаться в библиотека фреймворка .NET, и описаны в главе 3.1. Их взаимодействие отображено на рисунке 13.

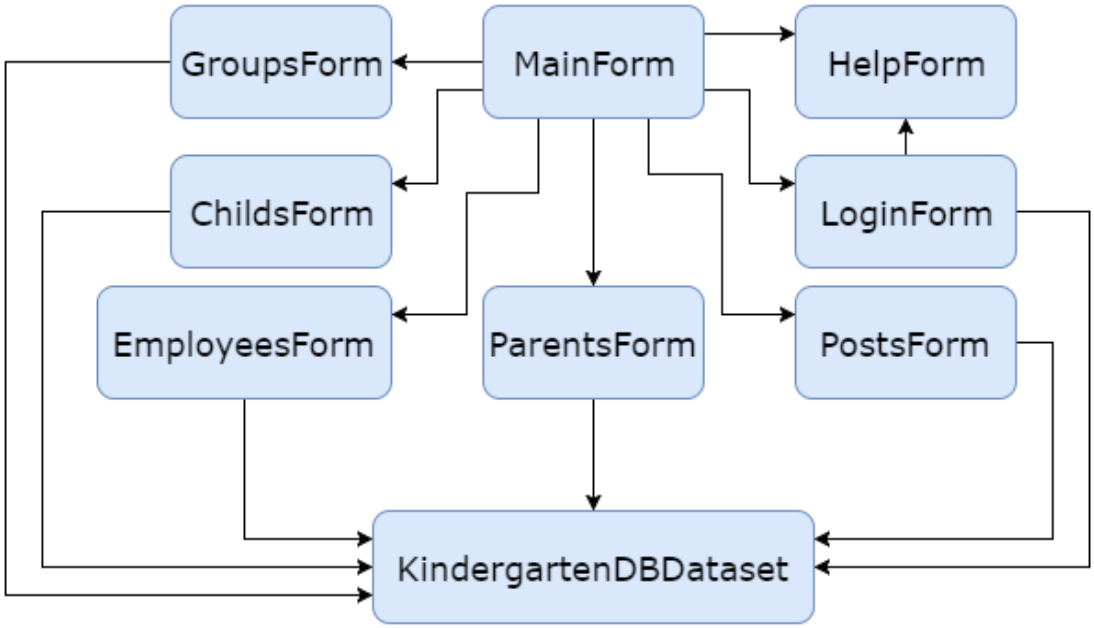

Рисунок 13

Класс KindergartenDBDataSet был создан автоматически, и состоит из компонентов классов: DataTable и TableAdapter. Структура данного класса, которую можно рассмотреть на рисунке 14, обусловлена структурой базы данных. т.е. для каждой таблицы в базе были созданы пара DataTable и TableAdapter объектов. Так же, была создана дополнительная пара Child_of_parent2, которая используется для заполнения таблицы с данными в окне редактирования родителей.

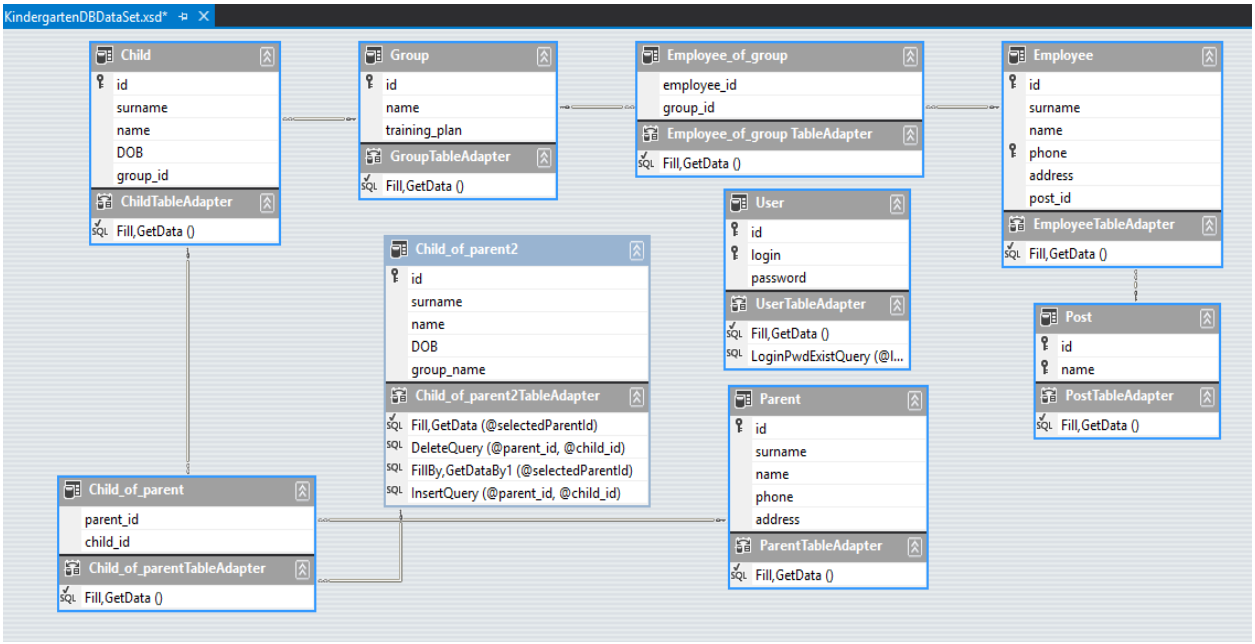

Рисунок 14

4.2 Описание SQL-запросов

При создании класса KindergartenDBDataSet в нем были автоматически созданы объекты TableAdapter, в которых есть стандартные методы insert, delete, update для DataTable, к которому они привязаны. Эти запросы и используются для заполнения элементов DataGrid. При необходимости, можно создать еще запросы и вызывать их, как методы объекта, как и было сделано для Child_of_parent2TableAdapter. Данный адаптер имеет два метода заполнения таблицы приложения, а также два метода для создания и удаления связи между ребенком и родителем. Запрос на получение данных о детях родителя выглядит как остальные запросы и создан с помощью конструктора запросов, а вот запрос на получение остальных детей имеет более сложный вид (рисунок 15).

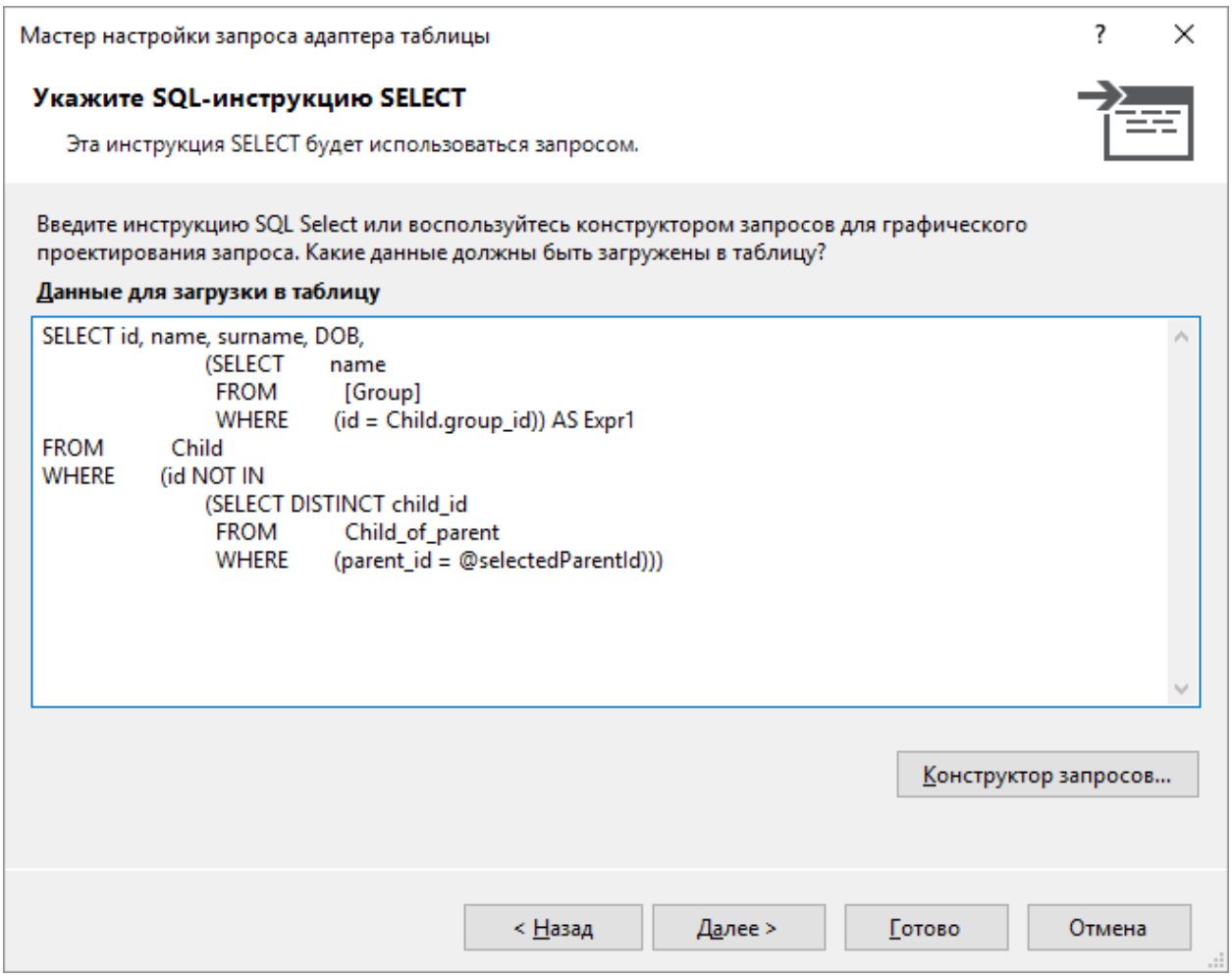

Рисунок 15

Данный запрос выделяет нужные параметры, а также название группы. При этом, если значение Expr1 равно NULL, то приложение распознает его, как пустую строку. Оператор where выбирает из данных только те, id которых не находится в выборке всех связей между ребенком и родителем, id которого было послано в качестве параметра. Слово DISTINCT позволяет выделять только уникальные значения, что делает выборку для поиска меньше.

Остальные запросы были написаны с помощью конструктора запросов. Его функционал позволяет создавать запросы графически. Он имеет окно с таблицами и окно с их атрибутами, в которых можно настраивать разные параметры. На рисунке 16 можно увидеть результат создания запроса на удаления связи с помощью конструктора.

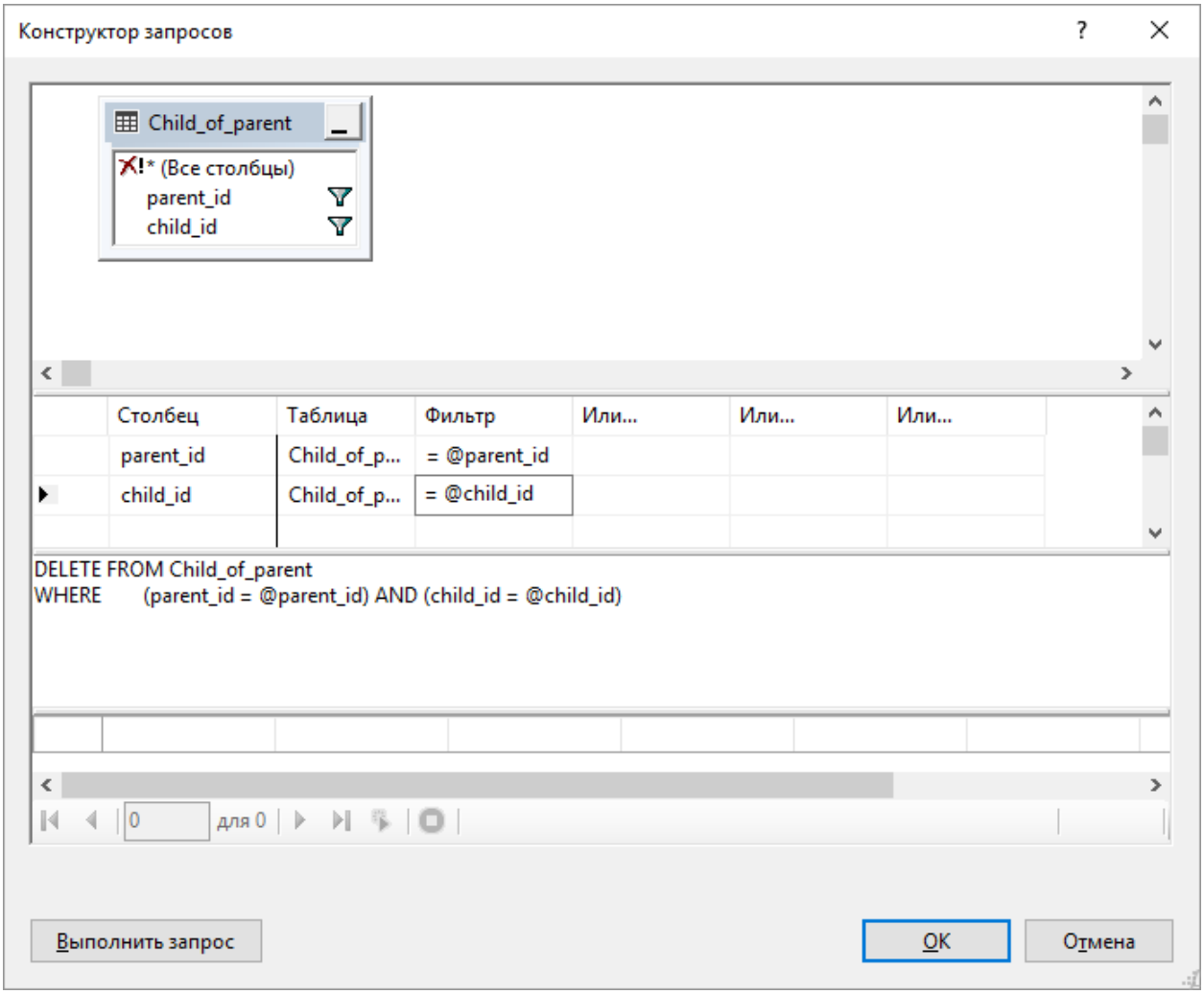

Рисунок 16

4.3 Установка программы

Для использования разрабатываемого продукта действуют следующие системные и программные требования:

- Не менее 5 mb памяти на жестком диске
- Операционная система Windows XP/ Windows7/8/10
- Процессор – Intel Pentium CPU 2117U@ 1.80GHz (2 ядра).
- Видеокарта от 256 mb памяти
- Оперативная память от 1 GB и выше
- Требуется установленная SqlLocalDB
- .NET framework версии 4.5.2

ЗАКЛЮЧЕНИЕ

В результате работы над курсовой работой выполнены все поставленные задачи. Спроектированные и разработанные база данных для дошкольных учреждений и приложение для управления ним. Был изучен ряд технологий для создания проекта: WindowsForms, C#, VisualStudio 2015, MS SQL. Также было написано обоснование выбора данных технологий в курсовом проекте. Функционал приложения был утвержден согласно разработанной структуре программы и базы данных. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий интерфейсной частью и базой данных для хранения данных о пользователях, родителях, детях, группах и должностях.

На данный момент при работе с приложением доступны следующие возможности: регистрирование других пользователей, авторизация, просмотр данных, изменение, удаление и добавление данных.

Дальнейшее развитие программы связано с расширением ее возможностей, улучшением уровня безопасности, подключением к удаленной БД, созданием новых объектов и упрощением текущих бизнес-процессов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Wikipedia – информационный ресурс:
URL: https://ru.wikipedia.org/wiki/.NET_Framework
(дата обращения: 02.10.2016).
URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio
(дата обращения: 02.10.2016).
URL: https://ru.wikipedia.org/wiki/Windows_Forms
(дата обращения: 05.10.2016).

2. ProfessionalC# 5.0 and .NET 4.5 / Автор: НейгелК.,ИвьенБ. / Издательство - М.: Вильямс, Год: 2014.

3. DependencyInjectionin .NET Букинистическое издание / Автор: Марк Симан, А. Барышнев, Евгений Зазноба, издательство: Питер.

4. Draw.io // [web-приложение для создания графических схем]
URL: https://www.draw.io/ (дата обращения: 05.11.2016).

5. DBDesigner.net // [web-приложение для создания схемы базы данных]
URL: http://dbdesigner.net/ (дата обращения: 12.11.2016).

6. Professorweb.ru// [Интернет-ресурс с материалами для работы с ADO.NET].
URL: http://professorweb.ru/my/ADO_NET/base/level1/ado_net_index.php
(дата посещения: 22.11.2016).

ПРИЛОЖЕНИЕ А

(обязательное)

Фрагменты листинга

Листинг 1 – Код формы класса ParentForm

```
public partial class ParentsForm : Form
{
    public ParentsForm(bool auth)
    {
        InitializeComponent();
        if (!auth)
        {
            Parents_dgv.ReadOnly = true;
            MakeParentBtn.Visible = false;
            saveChildren.Visible = false;
        }
    }
    private void ParentsForm_Load(object sender, EventArgs e)
    {
this.groupTableAdapter.Fill(this.kindergartenDBDataSet.Group);
this.childTableAdapter.Fill(this.kindergartenDBDataSet.Child);

this.child_of_parentTableAdapter.Fill(this.kindergartenDBDataSet
.Child_of_parent);
this.parentTableAdapter.Fill(this.kindergartenDBDataSet.Parent);
        ChildOfParentRB.Select();
    }
    private void dataGridView1_CurrentCellChanged(object sender,
EventArgs e)
    {
        UpdateChildDGV();
    }
    private void Parents_dgv_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
    {
        DataGridViewRow ThisDataRow =
((DataGridView)sender).CurrentRow;
        if (ThisDataRow != null && ThisDataRow.IsNewRow)
            ThisDataRow.Cells["id"].Value = Guid.NewGuid();
    }
    private void saveChildren_Click(object sender, EventArgs e)
    {
        childrenOfParent_dsv.EndEdit();
        parentBindingSource.EndEdit();

        DataSet changes =
this.kindergartenDBDataSet.GetChanges();
        if (changes != null)
```

```
        {
            //Data has changes.
            //use update method in the adapter. it should update
your datasource
            int updatedRows =
parentTableAdapter.Adapter.Update(changes);

parentTableAdapter.Fill(kindergartenDBDataSet.Parent);
        }
    }
    // метод обновения детей
    private void UpdateChildDGV()
    {
        if (Parents_dgv.CurrentRow != null
            && Parents_dgv.CurrentRow.Index <
(Parents_dgv.RowCount - 1)
            &&
Parents_dgv.Rows[Parents_dgv.CurrentRow.Index].Cells["id"].Value
!= null
            &&
Parents_dgv.Rows[Parents_dgv.CurrentRow.Index].Cells["id"].Value
.ToString() != "")
        {
            if (ChildOfParentRB.Checked)
            {
child_of_parent2TableAdapter.Fill((KindergartenDBDataSet.Child_o
f_parent2DataTable)kindergartenDBDataSet.Tables["Child_of_parent
2"],

(Guid)Parents_dgv.Rows[Parents_dgv.CurrentRow.Index].Cells["id"]
.Value);
            }
            else
            {
child_of_parent2TableAdapter.FillBy((KindergartenDBDataSet.Child
_of_parent2DataTable)kindergartenDBDataSet.Tables["Child_of_pare
nt2"],

(Guid)Parents_dgv.Rows[Parents_dgv.CurrentRow.Index].Cells["id"]
.Value);
            }
        }
    }

    private void ChildOfParentRB_CheckedChanged(object sender,
EventArgs e)
    {
        UpdateChildDGV();
    }

    private void NotChildOfParentRB_CheckedChanged(object
sender, EventArgs e)
    {
```

```
            UpdateChildDGV();
        }
        private void Parents_dgv_RowStateChanged(object sender,
DataGridViewRowStateChangedEventArgs e)
        {
            //if ( Parents_dgv.CurrentRow.Index <
(Parents_dgv.RowCount - 1))
            UpdateChildDGV();
        }

        private void MakeParentBtn_Click(object sender, EventArgs e)
        {
            if (NotChildOfParentRB.Checked)
            {
                try
                {
child_of_parent2TableAdapter.InsertQuery((Guid)Parents_dgv.Curre
ntRow.Cells["id"].Value,

(Guid)childrenOfParent_dsv.CurrentRow.Cells["id_cop2"].Value);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ("Ошибка связывания!
"));
                }
            }
            else
            {
                try
                {

child_of_parent2TableAdapter.DeleteQuery((Guid)Parents_dgv.Curre
ntRow.Cells["id"].Value,

(Guid)childrenOfParent_dsv.CurrentRow.Cells["id_cop2"].Value);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ("Ошибка отвязки!
"));
                }
            }
            UpdateChildDGV();
        }
        private void Parents_dgv_UserRowsAdded(object sender,
DataGridViewRowEventArgs e)
        {
            DataGridViewRow ThisDataRow =
((DataGridView)sender).CurrentRow;
            ThisDataRow.Cells["id"].Value = Guid.NewGuid();
        }
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Экранные формы

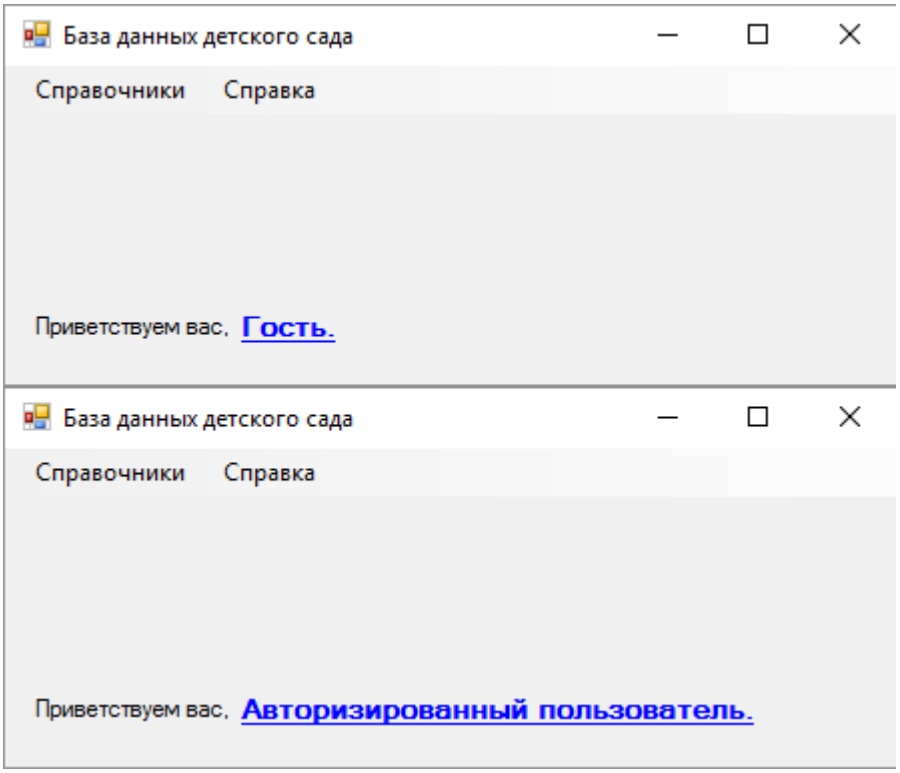

Рисунок Б.1 - Главная форма приложения

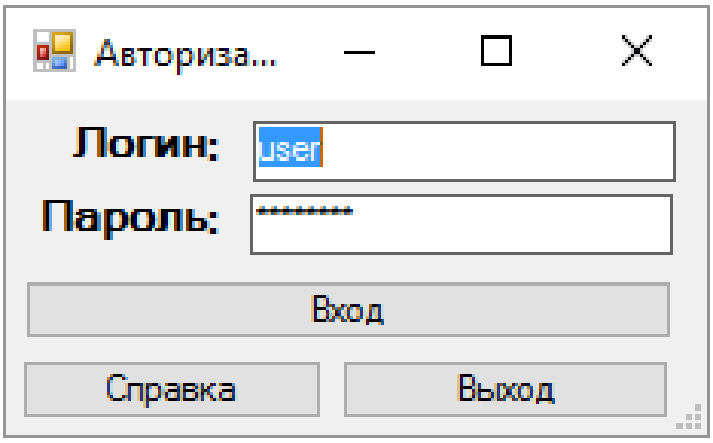

Рисунок Б.2 – Окно авторизации

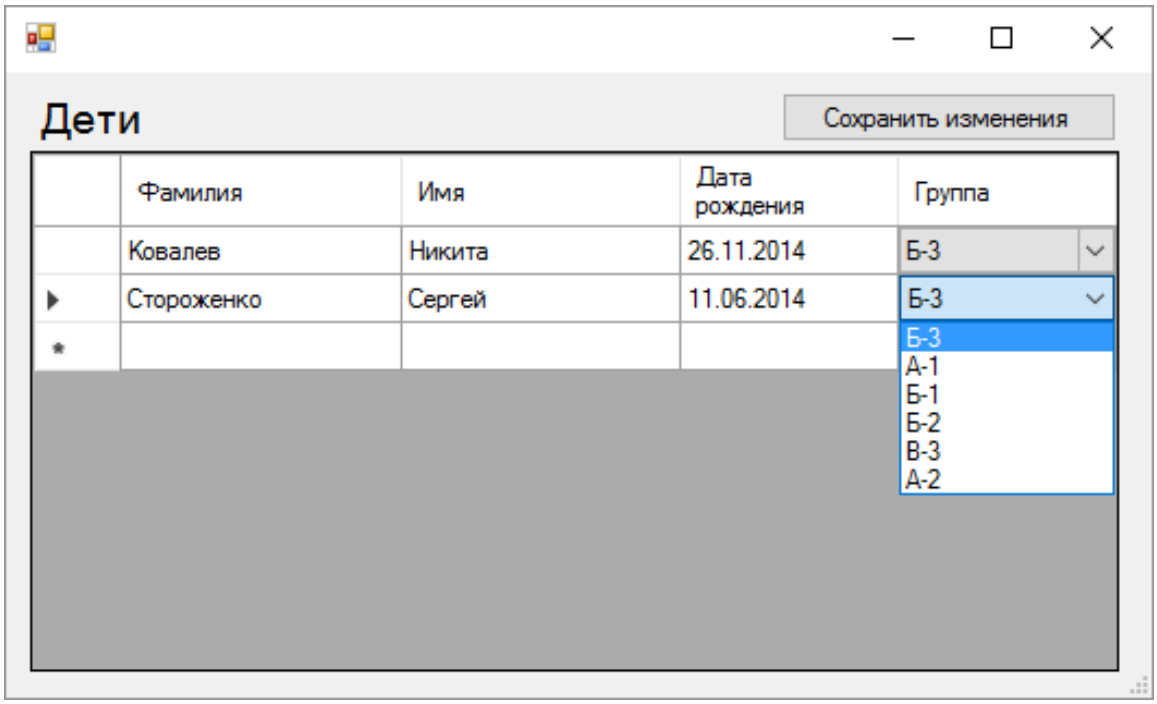

Рисунок Б.3 – Окно редактирования информации о детях

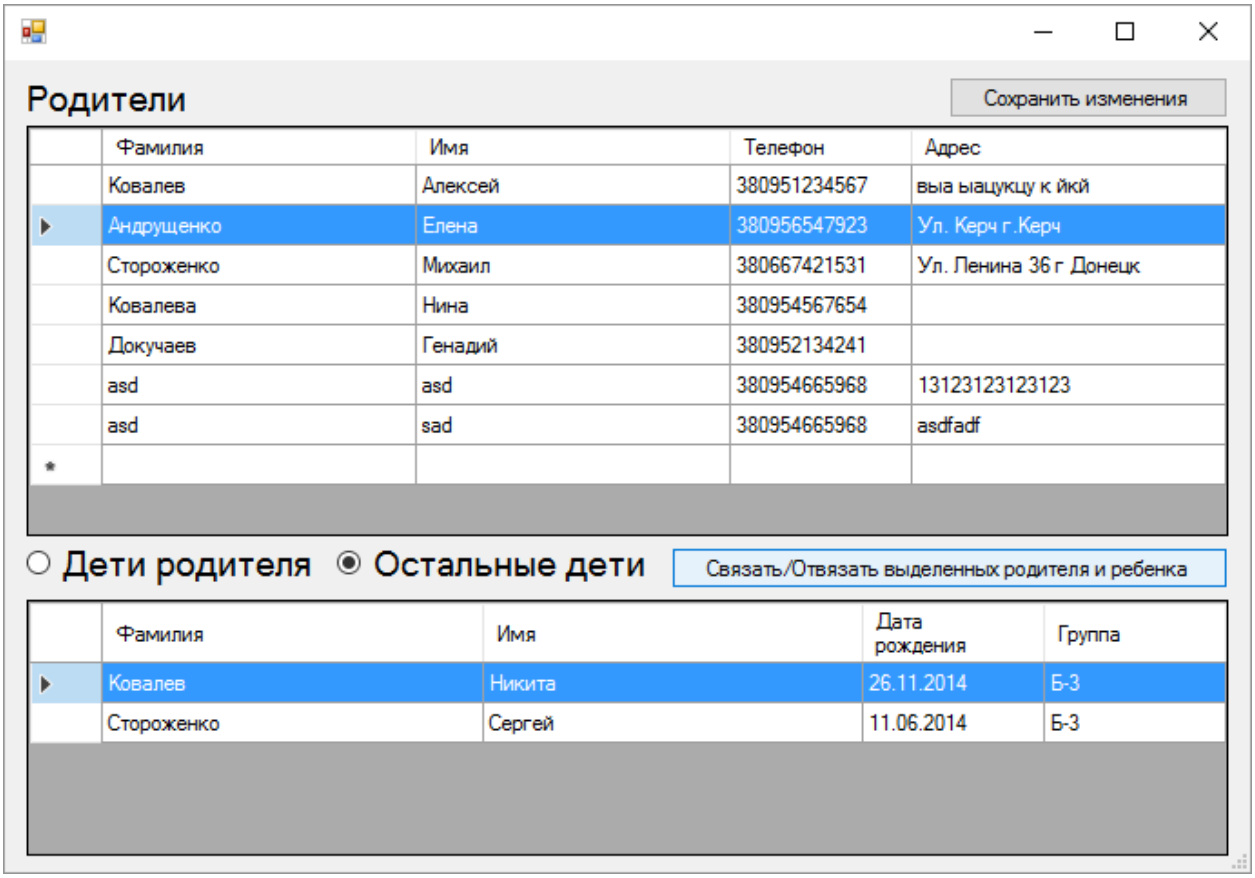

Рисунок Б.4 - Окно редактирования информации о родителях

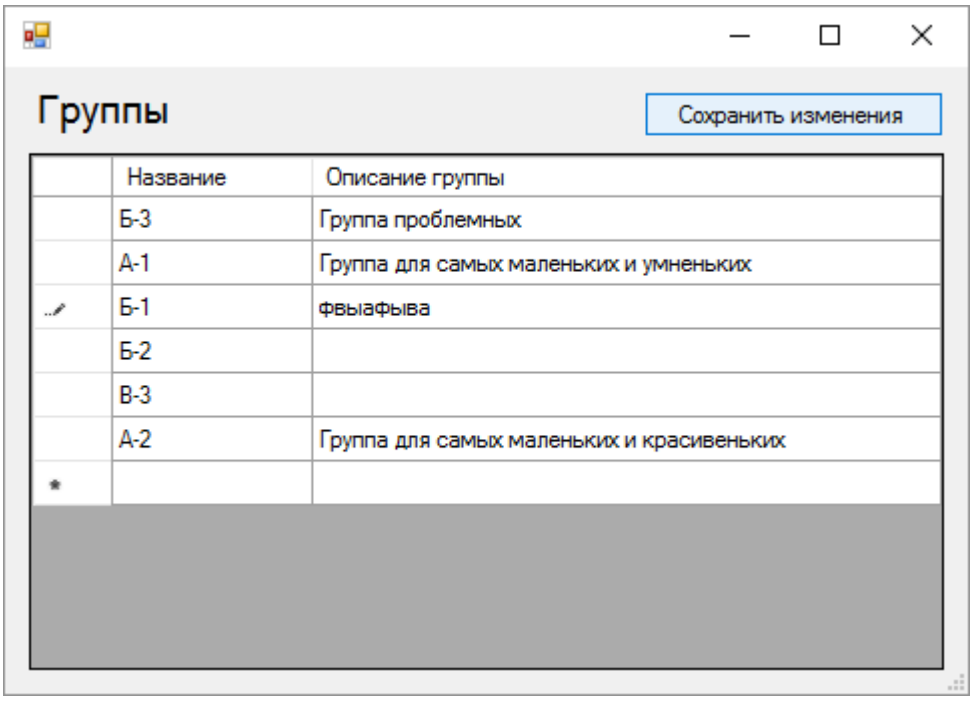

Рисунок Б.5 - Окно редактирования информации о группах

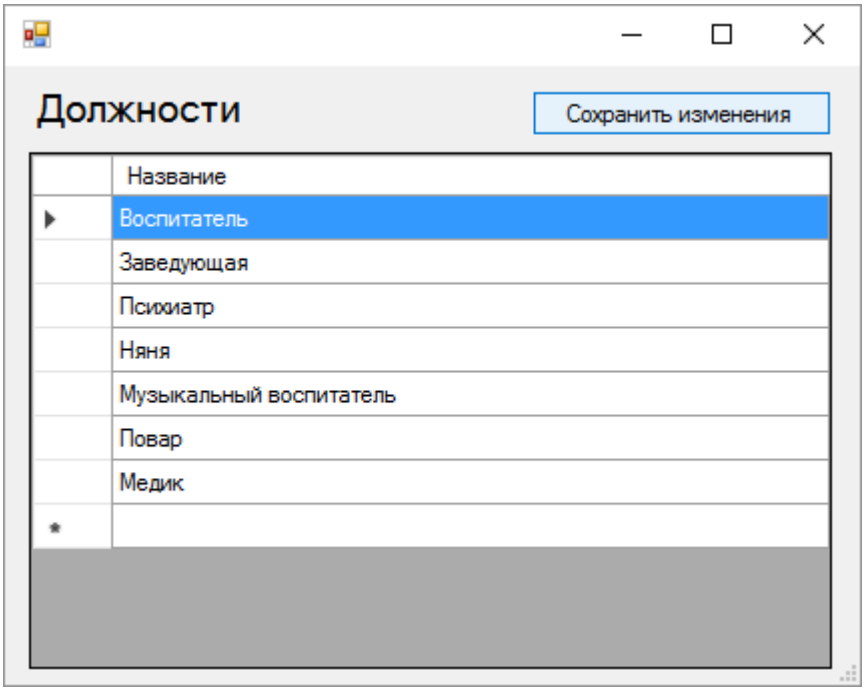

Рисунок Б.6 - Окно редактирования информации о должностях