СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Темой данной курсовой работы является «Проектирование и разработка реляционной базы данных информационной системы средней школы».

Еще с давних времен человек старался упорядочить какие-то данные для того, чтобы было удобно использовать информацию. На сегодняшний день применение баз данных приобрело весьма важное значение для многих сфер жизни. От работы системы структурирования и обработки информации на сегодняшний день многое зависит: работа предприятий, магазинов и других объектов, на которых ведется учет товара, продукции, материалов и т. д. Базы данных занимают одно из первых мест среди различных структур хранения данных. Базы данных используются в приложениях, написанных для облегчения работы мелких и крупных предприятий, учреждений. База данных является эффективно организованной структурой хранения данных, которая предоставляет пользователю значительные возможности при работе с информацией, находящейся в ней.

В настоящее время в учебных учреждениях существует необходимость в программных средствах, автоматизирующих работу по накоплению, систематизации и анализу информации. Требуется средства, позволяющие хранить информацию о учащихся, родителях, сотрудниках, расписании занятий, а также оценок учащихся, систематизировать её и обобщать, составлять заявления по различным причинам через приложение.

В различных учреждениях образования используются разные методы для организации учета и ведения документации. Однако очевидно, что наиболее оптимальным являлось бы создание базы данных под конкретные цели и задачи конкретного учреждения. Это позволило бы хранить всю необходимую информацию в одном месте, проводить анализ и находить требуемые данные, а также составлять отчеты по различным параметрам. Такая база данных позволила бы сократить затраты времени и труда на поиск информации, оформление документов, анализа текущей ситуации в школе.

Целью данной курсовой работы является создание такой системы, которая могла бы обеспечить должный уровень учета контингента средней школы и позволила бы проводить необходимые манипуляции над расписанием и оценками учащихся хранимой в информационной системе.

1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

Инфологическая модель данных – описание, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающих над проектированием базы данных.

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

1.1 Анализ предметной области и выявление необходимого набора сущностей

Перед началом разработки базы данных, необходимо определить основные цели, задачи и правила для решаемой задачи, после чего приступать к проектированию. Поэтому сформулируем краткое описание поставленной задачи.

Наименование задачи – ведение учета контингента средней школы, а также управление расписанием и оценками, каждым сотрудником школы.

Сотрудники в школе разделяются на 5 типов:

1. Сотрудник, не имеющий отношение к учебному процессу.
2. Учитель.
3. Заведующий учебной частью.
4. Заместитель директора, директор.
5. Системный администратор.

Так же требуется разделение базовых пользователей на 2 типа:

1. Ученик
2. Родитель

Основные бизнес-правила:

– Вся информация храниться до её явного удаления из системы.

– После составления управляющих документов, необходимо изменять соответствующие свойства на вновь определенные.

– Учащийся может иметь одну учетную запись и несколько родителей.

– Каждая учетная запись должна иметь уникальный номер, не повторяющийся в пределах одного типа содержания.

– У учетных записей должен быть номер телефона.

– У учетных записей должен быть уникальный логин для входа

– Учетным записям сотрудников устанавливаются привилегии вручную системным администратором и только с имеющимся заявлением с подписью директора.

Перечень печатных заявлений:

– Заявление об изменении привилегий сотруднику.

– Заявление о привязке ученику соответствующего класса.

– Заявление о привязке ученика к его родителю.

– Заявление об изменении оценки.

Для построения информационной системы требуется для начала выделить необходимы набор сущностей, которые описывают эту систему. Данный набор должен удовлетворять всем условиям на проектирование системы.

Определим минимальный набор сущностей, необходимый для проектирования информационной системы средней школы. Для определения первичного набора сущностей будет проведён анализ технического задания и предметной области.

Для учета данных необходимо описание такой сущности как *ученик*. В данной сущности должны быть заложены описательные характеристики студента, однозначно идентифицирующие его и его класс.

Все студенты обучаются в классах, поэтому следует выделить так же сущность – *класс*.

У всех учащихся должны быть родители, поэтому выделяется следующая сущность – *Родитель*.

Всей базой данных должны управлять сотрудники, следующая сущность – Сотрудник.

Так как сотрудник, родитель и ученик — это все какие-то люди то надо сделать обобщающую сущность, которая хранит основную информации об этих 3 сущностях. Требуется выделить сущность – *Person*.

Каждая персона — это отдельная учетная запись, чтоб облегчить доступ к данным вынесем данные для входа, а также уровень доступ. Требуется выделить сущность – *LogIn.*

Некоторые сотрудники могут быть классными руководителями, следовательно, новая сущность – *Классные рук-ли.*

По заданию требуется работать с расписанием уроков в школе. С этим справится сущность – *Расписание.*

Чтобы можно корректно составлять расписание, данной сущности требуется знать о дне, классе, предмете, учителе, кабинете. У нас уже есть некоторые сущность, поэтому создадим недостающие сущности – *дни, предметы, кабинеты*.

В заключении требуется создать сущность, которая будет отвечать за работу с оценками учащихся. Для этого будет сущность – *Журнал*.

1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Атрибут – поименованная характеристика сущности.

Атрибутом сущности является любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей.

Для каждой сущности, выделенной в пункте 1.1. необходимо определить атрибуты.

1 Сущность – «LogIn»:

Атрибуты: *id* (уникальный идентификационный номер), *username* (логин), *password* (пароль), *level* (уровень доступа).

2 Сущность – «Person»:

Атрибуты: *id*, *LastName* (фамилия), *Name* (имя), *Patronymic* (отчество), *Sex* (пол), *Birthday* (день рождения), *Number* (мобильный номер), *E-mail* (адрес электронной почты), *position* (стандартная должность персоны), *Secret word* (кодовое слово).

Атрибут «*position*» предназначен для хранения основной информации о должности в информационной системе. В данном поле может быть только статус: «Учащийся», «Родитель», «Сотрудник».

Атрибут «*Secret word*» будет использоваться для того, чтобы пользователь в случае утери пароля, смог самостоятельно его восстановить.

3 Сущность – «Учащийся»:

Атрибуты: *id* (Название), *Фамилия, Имя, Отчество, Код класса*.

4 Сущность – «Родители»:

Атрибуты: *id*, Фамилия, Имя, Отчество, Код ребенка.

5 Сущность – «Сотрудник»:

Атрибуты: *id, Фамилия, Полная должность, стаж*.

Атрибут «*Полная должность*» расширяет атрибут «*Должность*» из таблицы «Person». В этом атрибуте будет хранится полная должность сотрудника в средней школе.

6 Сущность – «Классные рук-ли»:

Атрибуты: *Код классного рук* (уникальный номер сотрудника), *Код класса* (уникальный номер класса).

7 Сущность – «Классы»:

Атрибуты: *Код класса* (уникальный номер класса), *название* (полное название класса).

8 Сущность – «Предметы»:

Атрибуты: *Код предмета* (уникальный идентификатор предмета), *Название предмета, От (от какого класса идет обучение предметом), До (до какого класса идет обучение этим предметом)*.

9 Сущность – «Журнал»:

Атрибуты: *Код оценки* (уникальный идентификатор оценки), *Код ученика, Код предмета, Код учителя, оценка, дата оценки*.

10 Сущность – «Дни»:

Атрибуты: *Код дня* (уникальный идентификатор дня), *название*.

11 Сущность – «Кабинеты»:

Атрибуты: *Код кабинета* (уникальный идентификатор кабинета), *номер*, *этаж*.

12 Сущность – «Расписание»:

Атрибуты: *Код расписания* (уникальный идентификатор расписания), *Код дня*, *Код класса*, *Код предмета*, *Код учителя*, *Код кабинета*.

1.3 Определение связей между объектами

Следующим этапом в проектировании инфологической модели является установление связи между сущностями.

Связь – это ассоциирование двух или более сущностей. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). В любой связи выделяются два конца (в соответствии с

существующей парой связываемых сущностей), на каждом из которых указывается имя конца связи, степень конца связи (сколько экземпляров данной сущности связывается), обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи).

Для реализации информационной системы средней школы необходимо установить все связи между объектами. А именно, нужно рассмотреть всю информационную систему в совокупности и определить взаимное влияние объектов, составляющих систему.

Этот процесс изображен на рис. 1.1

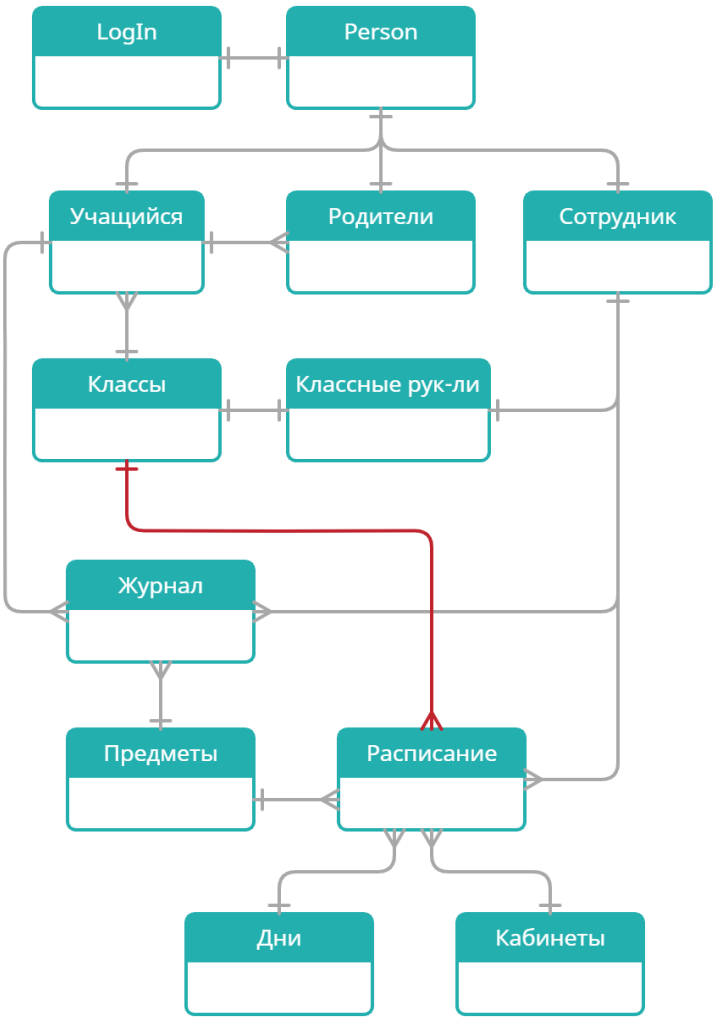

Рисунок 1.1 – Концептуальная схема базы данных

Проследить отношения, в которых состоят таблицы базы данных можно по схеме, изображенной на в приложении А на рис. А.1.

1.4 Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (*англ.* entity-relationship model).

По правилам построения ER-диаграмм в нотации Crow's Foot (рус. «воронья лапка») сущность изображается в виде прямоугольника. Связь изображается линией, которая связывает две сущности, участвующие в отношении. Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Атрибуты сущности записываются внутри прямоугольника, изображающего сущность.

На основе проведенного проектирования, в частности на основе инфологической схемы, приведенной на рис. 1.1, получим ER-диаграмму, проектируемой базы данных, представленную в приложении А на рис. А.1.

2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

2.1 Построение набора необходимых отношений базы данных

Для построения схемы реляционной базы данных необходимо определить совокупность отношений, составляющих базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

На основе полученной в первом пункте концептуальной модели можно определить набор необходимых отношений базы данных.

На рисунке 2.1 представлены отношения для базы данных информационной системы средней школы.

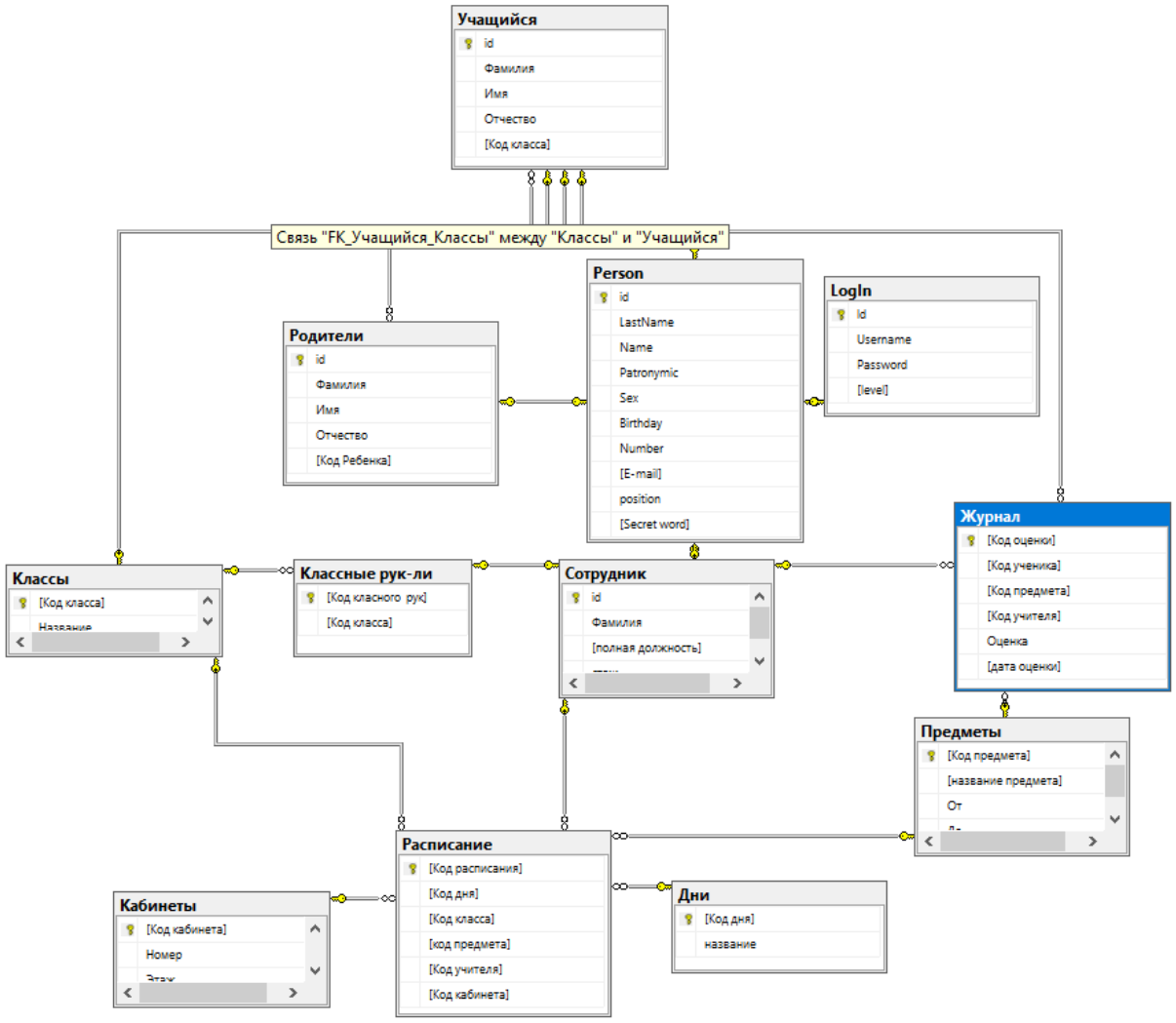

Рисунок 2.1 – Набор необходимых отношений базы данных

Схема реляционной базы данных представлена в приложении В.

2.2 Задание первичных и внешних ключей определенных отношений

В реляционной базе данных каждому объекту и сущности реального мира соответствуют кортежи отношений. И любое отношение должно обладать первичным ключом. Ключ – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся атрибутам. Каждое отношение должно обладать хотя бы одним ключом. В таблице 2.1 определены первичные и внешние ключи для отношений.[4]

Таблица 2.1 – Первичные и внешние ключи отношений

№ п/п	Название таблицы	Первичный ключ	Внешние ключи
1	LogIn	ID	
2	Person	ID	ID
3	Учащийся	ID	Код класса
4	Родители	ID	Код ребенка
5	Сотрудник	ID	
6	Классы	Код класса	
7	Классные рук-ли	Код классного рук	Код класса
8	Дни	Код дня	
9	Кабинеты	Код кабинета	
10	Предметы	Код предмета	
11	Журнал	Кою оценки	Код ученика Код предмета Код учителя
12	Расписание	Код расписания	Код дня Код класса Код предмета Код учителя Код кабинета

2.3 Приведение отношения БД к третьей нормальной форме

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации.[4]

Для реляционных баз данных необходимо, чтобы все отношения базы данных обязательно находились в 1НФ. Нормальные формы более высокого порядка могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до 3НФ, тем самым, исключив из базы данных избыточность и аномалии обновления.

Определение 3НФ – неключевые атрибуты не должны определять другие неключевые атрибуты.

В спроектированной базе данных почти все отношения находятся в третьей нормальной форме, за исключением таблиц *Сотрудик, Учащийся* и *Родители* их целесообразно оставить во второй нормальной форме.

2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

Ограничение целостности для внешних ключей состоит в том, что значение внешнего ключа должно быть равным значению первичного ключа цели; либо быть полностью неопределенным, т.е. каждое значение атрибута, участвующего во внешнем ключе должно быть неопределенным.

Условиями целостности называется набор правил, используемых для поддержания допустимых межтабличных связей и запрета на случайное изменение или удаление связанных данных. Следует устанавливать целостность данных только при выполнении

следующих условий: связываемое поле из главной таблицы является полем первичного ключа и имеет уникальный индекс, связанные поля имеют один и тот же тип данных.

Для автоматического обновления связанных полей (удаления записей) при обновлении (удалении) в главной таблице, следует устанавливать обеспечение целостности данных и каскадное обновление связанных полей (каскадное удаление связанных записей).

Ограничение целостности, накладываемые на разрабатываемую систему:

- ключевое поле отношения должно быть уникальным;
- внешний ключ должен быть повторяющимся, то есть соответствовать уникальному ключу в своем отношении.

Для удовлетворения требования ограничения целостности для внешних ключей отношений и для отношений в целом необходимо, чтобы выполнялось соответствие между типами вводимых данных и типами столбцов в таблицах, а также чтобы были заполнены все обязательные поля в таблицах, т.е. те поля которые не могут содержать значения NULL.

Для хранения информации о персоне необходимо обязательно заполнить такие данные как, фамилия, имя, отчество, пол, день рождения, номер телефона, статус. После установления статуса, персона заносится в соответствующую таблицу – *учащийся* или *родители,* или *сотрудник*.

Т.к. сотрудник не обязательно может быть классным руководителем, либо отношение к журналу оценок и расписанию, то контроль ограничения целостности для отношений между таблицами Сотрудник и Классные руководители, Сотруник и Журнал, Сотрудник и Расписание – отключен.

2.5 Графическое представление связей между внешними и первичными ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении В. Полученная ER-диаграмма построена по методу Crow's Foot (рус. «воронья лапка») [4]. Средства моделирования Crow's Foot специально разработаны для построения реляционных информационных систем. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами. Первичные ключи обозначаются знаком ключа, внешние ключи обозначаются знаком ссылки.

3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для реализации спроектированной базы данных была выбрана система управления базами данных MS SQL Server 2019. Это обусловлено тем, что, данная СУБД имеет большую функциональность, множество средств для поддержки и работы с ней, развитую инфраструктуру для интеграции баз данных в пользовательские приложения.

В создаваемой базе данных будут использоваться следующие типы данных:

- `INT` – Целочисленный тип. Размер – 4 байта
- `NVARCHAR` – Строковый тип переменной длины
- `DATE` – Тип, определяющий дату.

Опишем все таблицы, которые будут созданы в базе данных.

Таблица *LogIn* содержит список данных для входа пользователей. Ее структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы LogIn

Имя атрибута	**Тип**	**Описание**
ID	INT	Идентификатор учетной записи. Ключевой атрибут
username	NVARCHAR (50)	Логин для входа
password	NVARCHAR (40)	Пароль для входа
level	INT	Уровень доступа в приложение

Таблица *Person* содержит основную информацию о персоне. Ее структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы Person

Имя атрибута	**Тип**	**Описание**
ID	INT	Идентификатор персоны. Ключевой атрибут
LastName	NVARCHAR (50)	Фамилия персоны
Name	NVARCHAR (50)	Имя персоны

Продолжение таблицы 3.2

Patronymic	NVARCHAR (50)	Отчество персоны. Необязательный атрибут
Sex	NVARCHAR (10)	Пол
Birthday	DATE	День рождения персоны
Number	NVARCHAR (15)	Номер телефона
E-mail	NVARCHAR (50)	Адрес электронной почты. Необязательный атрибут
Position	NVARCHAR (15)	Статус в школе
Secret word	NVARCHAR (50)	Кодовое слово, используется для восстановления пароля

Таблица *Учащийся* содержит информацию о учащимся. Ее структура приведена в таблице 3.3.

Таблица 3.3 – Характеристика атрибутов таблицы Учащийся

Имя атрибута	**Тип**	**Описание**
ID	INT	Идентификатор учетной записи. Ключевой атрибут
Фамилия	NVARCHAR (50)	Фамилия ученика
Имя	NVARCHAR (50)	Имя ученика
Отчество	NVARCHAR (50)	Отчество ученика
Код класса	INT	Идентификатор класса, в котором состоит ученик

Таблица *Родители* содержит данные о родителях и привязанным к ним учащимся. Ее структура приведена в таблице 3.4.

Таблица 3.4 – Характеристика атрибутов таблицы Родители

Имя атрибута	**Тип**	**Описание**
ID	INT	Идентификатор учетной записи. Ключевой атрибут
Фамилия	NVARCHAR (50)	Фамилии родителя
Имя	NVARCHAR (50)	Имя родителя
Отчество	NVARCHAR (50)	Отчество родителя
Код ребенка	INT	Идентификатор привязанного ребенка

Таблица *Сотрудник* содержит информацию о сотрудниках школы. Ее структура приведена в таблице 3.5.

Таблица 3.5 – Характеристика атрибутов таблицы Сотрудник

Имя атрибута	**Тип**	**Описание**
ID	INT	Идентификатор учетной записи. Ключевой атрибут
Фамилия	BIT	Фамилия сотрудника
Полная должность	NVARCHAR (50)	Полная должность сотрудника
Стаж	INT	Стаж сотрудника в средней школе

Таблица *Классы* содержит классы, которые присутствуют в школе. Ее структура приведена в таблице 3.6.

Таблица 3.6 – Характеристика атрибутов таблицы Классы

Имя атрибута	**Тип**	**Описание**
Код класса	INT	Идентификатор класса. Ключевой атрибут
Название	NVARCHAR (50)	Название класса

Таблица *Классные рук-ли* служит для соединения класса и сотрудника, определяется классный руководитель для класса. Ее структура приведена в таблице 3.7.

Таблица 3.7 – Характеристика атрибутов таблицы Классные рук-ли

Имя атрибута	**Тип**	**Описание**
Код классного рук	INT	Идентификатор сотрудника. Ключевой атрибут
Код класса	INT	Идентификатор класса

Таблица *Дни* служит хранения информации о рабочих днях школы. Ее структура приведена в таблице 3.8.

Таблица 3.8 – Характеристика атрибутов таблицы Дни

Имя атрибута	**Тип**	**Описание**
Код дня	INT	Идентификатор дня. Ключевой атрибут
Название	NVARCHAR (50)	Название дня в неделе

Таблица *Кабинеты* содержит об используемых кабинетах для обучения. Ее структура приведена в таблице 3.9.

Таблица 3.9 – Характеристика атрибутов таблицы Кабинеты

Имя атрибута	**Тип**	**Описание**
Код кабинета	INT	Идентификатор кабинета. Ключевое поле
Номер	INT	Номер кабинета
Этаж	INT	Этаж, на котором находится кабинет

Таблица *Предметы* служит для хранения обучаемых предметов в определённые года обучения. Ее структура приведена в таблице 3.10.

Таблица 3.10 – Характеристика атрибутов таблицы Предметы

Имя атрибута	**Тип**	**Описание**
Код предмета	INT	Идентификатор предмета. Ключевое поле.
Название предмета	NVARCHAR (30)	Название предмета
От	INT	От какого года обучения изучается предмет
До	INT	До какого года обучения изучается предмет

Таблица Журнал служит для связи студентов и их гражданств. Ее структура приведена в таблице 3.11.

Таблица 3.11 – Характеристика атрибутов таблицы Журнал

Имя атрибута	**Тип**	**Описание**
Код оценки	INT	Идентификатор оценки. Ключевое поле.
Код ученика	INT	Идентификатор ученика
Код предмета	INT	Идентификатор предмета
Код учителя	INT	Идентификатор учителя
Оценка	INT	Оценка учащегося
Дата оценки	DATE	Дата выставления оценки

Таблица *Расписание* служит для связи студентов и их гражданств. Ее структура приведена в таблице 3.12.

Таблица 3.12 – Характеристика атрибутов таблицы Расписание

Имя атрибута	**Тип**	**Описание**
Код расписания	INT	Идентификатор расписания. Ключевое поле.
Код дня	INT	Идентификатор дня
Код класса	INT	Идентификатор класса
Код предмета	INT	Идентификатор предмета
Код учителя	INT	Идентификатор учителя
Код кабинета	INT	Идентификатор кабинета

В спроектированной базе данных реализован контроль целостности и корректности модификаций, путем выполнения триггеров. Так для большинства таблиц в базе данных определены триггеры, которые при добавлении пользователя позволяют его добавить по требуемым правилам. Для таблицы *LogIn* определен, триггер, который не позволяет добавлять учетные записи с одинаковыми логинами.

Например, при добавлении персоны были введены не в том месте большие или маленькие буквы, то триггер это исправит и приведет ФИО персоны к нормальному виду. Реализация данного триггера представлена в листинге 3.1.

Листинг 3.1 – Триггер исправляющий ФИО:

```
1:  create trigger toTrueFIO
2:  on Person
3:  after INSERT
4:  as
5:  begin
6:  declare @ins_ln nvarchar(50)
7:  declare @ins_n nvarchar(50)
8:  declare @ins_p nvarchar(50)
9:  declare @ins_id int
10: set @ins_ln = (select lower(LastName) from inserted)
11: set @ins_n = (select lower(Name) from inserted)
12: set @ins_p = (select lower(Patronymic) from inserted)
13: set @ins_id = (select id from inserted)
14: set @ins_ln = UPPER(substring(@ins_ln,1,1)) +
SUBSTRING(@ins_ln,2,LEN(@ins_ln))
15: set @ins_n = UPPER(substring(@ins_n,1,1)) +
SUBSTRING(@ins_n,2,LEN(@ins_n)) set @ins_p =
UPPER(substring(@ins_p,1,1)) + SUBSTRING(@ins_p,2,LEN(@ins_p))
16: UPDATE Person SET LastName = @ins_ln, Name = @ins_n,
Patronymic= @ins_p WHERE @ins_id = id; print('ФИО выглядят
красиво') end
```

4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

1 Запрос, устанавливающий связь между родителем и студентом.

```
1:  update Родители set [Код Ребенка] = @idC
2:  where id = @idP
```

2 Запрос, возвращающий приемливый вид для пользователя расписание занятий по заданным критериям.

```
1:  select [Код расписания], Дни.название as [День],
Классы.Название as [Класс],
2:  Предметы.[название предмета] as [Предмет], Кабинеты.Номер
as [Кабинет],
3:  Сотрудник.Фамилия as [Учитель]
4:  from Расписание
5:  inner join Дни on Расписание.[Код дня] = Дни.[Код дня] and
Дни.название like @day
6:  inner join Классы on Расписание.[Код класса] = Классы.[Код
класса] and Классы.Название like @class
7:  inner join Предметы on Расписание.[код предмета] =
Предметы.[Код предмета] and Предметы.[название предмета] like
@object
8:  inner join Кабинеты on Расписание.[Код кабинета] =
Кабинеты.[Код кабинета] and Кабинеты.Номер like @cabinet
9:  inner join Сотрудник on Расписание.[Код учителя] =
Сотрудник.id and Сотрудник.Фамилия like @teacher
```

3 Запрос, возвращающий список студентов, переведены из другого ВУЗа, с другого факультета с указанием ликвидации разницы.

```
1:  select id, Фамилия, Имя, Отчество, Классы.Название as
[Класс]
2:  from Учащийся inner join Классы on Учащийся.[Код класса] =
Классы.[Код класса]
3:  and Классы.Название like @nazv
4:  where Учащийся.Фамилия like @famil
```

4 Запрос, выводящий оценки по критериям.

```
1:  select [Код оценки], [дата оценки], Предметы.[название
предмета], Оценка,
2:  Учащийся.Фамилия as [Ученик], Сотрудник.Фамилия as
[Преподаватель]
3:  from Журнал inner join Предметы on Журнал.[Код предмета] =
Предметы.[Код предмета] and Предметы.[название предмета] like
@name
4:  inner join Учащийся on Журнал.[Код ученика] = Учащийся.id
and Учащийся.Фамилия like @familST
```

```
   5:  inner join Сотрудник on Журнал.[Код учителя] = Сотрудник.id
and Сотрудник.Фамилия like @familTC
   6:  where [Код оценки] like @idMark
```

5 Запрос, выводящий список сотрудников по различным критериям.

```
   1:  select person.lastname as [фамилия], person.name as [имя],
   2:  person.patronymic as [отчество], person.birthday as [день
рождения],
   3:  person.number as [мобильный], person.[e-mail],
   4:  сотрудник.[полная должность] as [должность]
   5:  from сотрудник inner join person
   6:  on сотрудник.id = person.id
   7:  where сотрудник.[полная должность] like @pos
   8:  and сотрудник.фамилия like @famil
```

6 Запрос, показывающий отношения родителей и учеников.

```
   1:  select родители.id,родители.фамилия, родители.имя,
   2:  родители.отчество, учащийся.имя as [имя ребенка],
   3:  учащийся.фамилия as [фамилия ребенка], учащийся.отчество as
[отчество ребенка]
   4:  from родители inner join учащийся
   5:  on родители.[код ребенка] = учащийся.id
```

7 Запрос, который возвращает таблицу учеников.

```
   1:  select учащийся.id, учащийся.имя, учащийся.фамилия,
   2:  учащийся.отчество, классы.название as класс
   3:  from учащийся inner join классы
   4:  on учащийся.[код класса] = классы.[код класса]
```

8 Запрос, который возвращает полную должность определнного человека.

```
   1:  select [полная должность]
   2:  from сотрудник
   3:  where id = @id
```

9 Запрос, возвращающий название класса к котому привязан ученик

```
   1:  select классы.название
   2:  from учащийся inner join классы
   3:  on учащийся.[код класса] = классы.[код класса]
   4:  where учащийся.id = @id
```

5 ВЫБОР И ОБОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Выбор СУБД является сложной задачей и должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или и то, и другое).

Система SQL Server 2019 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. SQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. SQL Server 2019 позволяет сократить затраты времени и средств, требуемые на управление и развертывание таких приложений. Также следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД (Express).

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2019, в качестве языка программирования – C# .NET Framework. Одной из причин так же послужила совместимость версий MS Visual Studio и MS SQL Server.

Достоинства платформы .NET [1]:

1. Вся платформа .NET основана на единой объектно-ориентированной модели. Все сервисы, интерфейсы и объекты, которые платформа предоставляет разработчику объединены в единую иерархию классов. Другими словами, все, что может вам потребоваться при создании приложений под платформу .NET будет всегда у вас под рукой. Причем, все это сгруппировано очень удобно и интуитивно понятно.

2. Приложение, написанное на любом .NET-совместимом языке, является межплатформенным (в идеале). Почему в идеале? Дело в том, что приложение, написанное, скажем, на том же C#, не зависит от платформы, на которой будет выполняться, но зато зависит от наличия платформы .NET.

3. В состав платформы .NET входит "сборщик мусора", который освобождает ресурсы. Таким образом, приложения защищены от утечки памяти и от необходимости освобождать ресурсы. Это делает программирование более легким и более безопасным.

4. Приложения .NET используют безопасные типы, что повышает их надежность.

6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

6.1 Разработка и построение интерфейса главной и рабочей форм

Главная форма приложения является объектом класса *aunt*, наследуемый от класса `Form`, определенного в .NET Framework. Создание всех компонентов формы, в частности управляющих элементов, окон сообщений, диалогов и др., происходит в методе по мере их выхова, соответствующими им конструкторами.

Все основные таблицы для представления данных были выполнены в виде DataGridView, что упрощает понимание и просмотр информации, т.к. она представляется в табличном виде.

При помощи главной формы приложения пользователь производит вход в приложении с определенными привилегиями. Если это ученик или его родитель, то следует запустить первой подсистеме, в случае если заходит сотрудник запускаются другая версия подсистемы, а следовательно, и другая форма. Затем пользователь сам вызывает нужное ему окошко, в котором будет производить нужные ему действия. Основной упор при проектировании интерфейса приложения был сделан на привлекательность и понятность для конечного пользователя. Поэтому были заменены многие стандартные элементы оформления на доработанные, также установлен задний фон.

При проектировании приложения были учтены все возможные случаи некорректной работы программы, поэтому большинство нештатных ситуаций сопровождается оповещениями с описанием проблемы.

Скриншоты главных окон и диалоговых окон представлены в приложении С.

6.2 Построение главного меню и кнопок панели инструментов

Главное меню программы любой из подсистем содержит семь управляющий элементов: мой личный кабинет, сотрудники, родители, учащиеся, успеваемость, расписание, выход. Данные элементы распределены по всему окну приложения, так чтоб это имело общий дизайн. В подсистеме сотрудника добавлена кнопка управления базой данных, а также кнопку которая позволяет составить автоматическое заявление об изменении привилегий.

При работе с базой данных (просмотре, изменении, удалении) используются элемент textbox и combobox. При помощи данных элементов упрощена работа с базой данных.

6.3 Выполнение программного кода в среде Microsoft Visual C#

Опишем работу приложения с базой данных. Вся необходимая реализация обращения к базе данных вынесена в класс *WorkWithDB*. Данный класс имеет конструктор при помощи которого определяется объект соединения с базой данных. Подключение к базе данных начинается с формирования строки подключения и последующим созданием контекста на основе данной строки. Описанный участок кода приложен в листинге 6.3.1.

Листинг 6.3.1 – поля и конструктор класса WorkWithDB

```
1:  private const string connectionString = @"Data Source=ASUS-
ZENBOOK;Initial Catalog=DBSchool;Integrated Security=True";
2:  private SqlConnection sqlConnection;
3:  public WorkWithDB()
4:  {
5:  sqlConnection = new SqlConnection(connectionString);
6:  }
```

После вызова конструктора для осуществления работы с базой данных требуется вызывать определённые методы. Некоторые методы используют хранимые процедуры при работе с данными. Но если запрос слишком маленький, то составляется запрос уже в самой программе и передается на выполнение серверу. В случае если какой-либо запрос не удался, пользователю выводится сообщение об ошибке. Привилегии в приложении делится на 6 категорий, описанных ранее, поэтому некоторым пользователям запрещены определенные действия. Так допустим ученик не сможет выставить себе оценку сам так как у него не достаточно для этого прав, но он может в случае если учитель ошибся с оценкой составить автоматическое заявление в котором он укажет причину данного заявления. Так же в структуре школы есть разделение на разные категории сотрудников, поэтому сотрудники, которые не имеют отношения к учебному процессу, могут лишь просматривать какие-либо данные. Определение привилегий происходит при авторизации в приложение. Перед запуском приложения программа вызывают метод класса *WorkWithDB, getActualDB ()*. Данная функция возвращает список пользователей типа *Person*, в котором содержится необходимая информация о персоне. Листинг данной функции представлен в листинге 6.3.2

Листинг 6.3.2 – функция getActualDB()

```
1:  public List<Person> getActualDB()
2:  {
3:  List<Person> Persons = new List<Person>();
4:  sqlConnection = new SqlConnection(connectionString);
```

Продолжение листинга 6.3.2

```
   5:  sqlConnection.Open();
   6:  SqlDataReader sqlDataReader = null;
   7:  SqlCommand sqlCommand = new SqlCommand("SELECT LogIn.Id,
LogIn.Username, LogIn.Password, LogIn.level, Person.[E-
mail],Person.Number, Person.[Secret word] FROM[LogIn], Person
where LogIn.Id = Person.id", sqlConnection);
   8:  try
   9:  {
   10: sqlDataReader = sqlCommand.ExecuteReader();
   11: while (sqlDataReader.Read())
   12: {
   13: Person somePerson = new Person();
   14: somePerson.id = Convert.ToInt32(sqlDataReader["Id"]);
   15: somePerson.username = sqlDataReader["Username"].ToString();
   16: somePerson.password = sqlDataReader["Password"].ToString();
   17: somePerson.level = Convert.ToInt32(sqlDataReader["level"]);
   18: somePerson.number = sqlDataReader["Number"].ToString();
   19: somePerson.secretWord = sqlDataReader["Secret
word"].ToString();
   20: somePerson.email = sqlDataReader["E-mail"].ToString();
   21: Persons.Add(somePerson);
   22: }
   23: sqlDataReader.Close();
   24: }
   25: catch (Exception ex)
   26: {
   27: MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
   28: }
   29: finally
   30: {
   31: if (sqlDataReader != null)
   32: {
   33: sqlDataReader.Close();
   34: sqlConnection.Close();
   35: }
   36: }
   37: return Persons;
   38: }
```

Далее после получения списка программа сравнивает введенные данные для входа пользователем с имеющимися в базе данных.

Так же в приложении присутствует окно для просмотра всех сведений своего аккаунта, а также настройка их. Изначально данное окошко должно получать данные из базы данных методом выборки по id персоны. Реализация получения информация из базы данных приведена в листинге 6.3.3.

Листинг 6.3.3 – Метод получаения полной информации

```
   1: public Person getFullInformation(int id)
   2: {
   3: Person somePerson = new Person();
   4: sqlConnection = new SqlConnection(connectionString);
   5: sqlConnection.Open();
   6: SqlDataReader sqlDataReader = null;
   7: SqlCommand sqlCommand = new SqlCommand("SELECT LogIn.Id,
Username, Password, level, " +
   8: "LastName, Name, Patronymic, Sex, Birthday, Number, [E-
mail], position, [Secret word] FROM [LogIn], " +
   9: $"Person where LogIn.Id ={id} AND LogIn.Id = Person.id",
sqlConnection);
   10: try
   11: {
   12: sqlDataReader = sqlCommand.ExecuteReader();
   13: sqlDataReader.Read();
   14: somePerson.id = Convert.ToInt32(sqlDataReader["Id"]);
   15: somePerson.username = sqlDataReader["Username"].ToString();
   16: somePerson.password = sqlDataReader["Password"].ToString();
   17: somePerson.level = Convert.ToInt32(sqlDataReader["level"]);
   18: somePerson.lastName = sqlDataReader["LastName"].ToString();
   19: somePerson.name = sqlDataReader["Name"].ToString();
   20: somePerson.patronymic                                    =
sqlDataReader["Patronymic"].ToString();
   21: somePerson.sex = sqlDataReader["Sex"].ToString();
   22: somePerson.birthday = sqlDataReader["Birthday"].ToString();
   23: somePerson.number = sqlDataReader["Number"].ToString();
   24: somePerson.email = sqlDataReader["E-mail"].ToString();
   25: somePerson.position = sqlDataReader["position"].ToString();
   26: somePerson.secretWord        =        sqlDataReader["Secret
word"].ToString();
   27: sqlDataReader.Close();
   28: sqlCommand     =     new     SqlCommand("getFullPosition",
sqlConnection);
   29: sqlCommand.CommandType                                   =
System.Data.CommandType.StoredProcedure;
   30: // параметр для ввода имени
   31: SqlParameter idParam = new SqlParameter
   32: {
   33: ParameterName = "@id",
   34: Value = somePerson.id
   35: };
   36: sqlCommand.Parameters.Add(idParam);
   37: somePerson.fullPosition = "";
   38: if (somePerson.level >= 3)
   39: {
   40: somePerson.fullPosition                                   =
sqlCommand.ExecuteScalar().ToString();
   41: }
   42: }
   43: catch (Exception ex)
```

Продолжение листинга 6.3.3

```
   44: {
   45: MessageBox.Show(ex.Message,                    ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
   46: }
   47: finally
   48: {
   49: if (sqlDataReader != null)
   50: {
   51: sqlDataReader.Close();
   52: }
   53: sqlConnection.Close();
   54: }
   55: return somePerson;
   56: }
```

После выполнения данного метода происходит заполнение управляющих элементов. После пользователь изменяют в случае надобности какие-либо данные и вызывается соответсвующий метод. Например пользователь решил измениить пароль, то вызовиться метод *UpdatePassword(int id, string password)*, из класса *WorkWithDB*. Данный метод описан в листинге 6.3.4.

Листинг 6.3.4 – UpdatePassword(int id, string password)

```
   1:  public void UpdatePassword(int id, string password)
   2:  {
   3:  sqlConnection.Open();
   4:  SqlCommand sqlCommand = new SqlCommand("", sqlConnection);
   5:  sqlCommand.CommandText  =  $"UPDATE  LogIn  SET  Password  =
'{password}' WHERE id = {id}";
   6:  try
   7:  {
   8:  sqlCommand.ExecuteNonQuery();
   9:  }
   10: catch (Exception e)
   11: {
   12: MessageBox.Show(e.Message,  e.Source,  MessageBoxButtons.OK,
MessageBoxIcon.Error);
   13: }
   14: sqlConnection.Close();
   15: }
```

После выполнения данного метода, пользователь может в следующий раз зайти под другим паролем. Остальной листинг программы представлен в приложении Г.

ЗАКЛЮЧЕНИЕ

В результате выполненной работы, была создана база данных для средней школы, а также эффективно работающее с этой базой данных приложение. Полученная комбинация представляет собой информационную систему средней школы.

Разработанная база данных удовлетворяет всем требованиям, предъявленным в задании, и позволяет без проблем хранить и извлекать требуемую информацию.

Созданное приложение позволяет упростить работу с информацией для работников школы, позволяя систематизировать всю необходимую информацию. Основные функции, которые требуются школьнику, родителю и сотруднику школы были реализованы. Пользователи могут свободно работать с оценками, расписанием, а также составом пользователей приложения. В приложении реализованы запросы, позволяющие пользователю выбрать всю необходимую информацию по заданным критериям, осуществлять поиск данных и формировать заявления.

Разработанная система реагирует на ошибочный ввод данных, а также способна определять возникающие ошибки и уведомлять об этом пользователя, чтобы в любой момент он знал из-за чего или почему произошла ошибка, и оперативно устранил её.

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на языке C#, проектировании баз данных и реализации их в СУБД MS SQL Server 2019.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Шилдт Г. Полный справочник по SQL.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2004. – 752 с.: ил.

2 Раттц Д. язык C# 2008 для профессионалов. Пер. с англ. – М.: Вильямс, 2008. – 645с.: ил.

3 Хернандес М., Вьескас Д. SQL-запросы. Практическое руководство.: Пер. с англ. – М.: Лори, 2003. – 473 с.: ил.

4 Коннолли Т., Бегг К., Базы данных. Проектирование, реализация и сопровождение. Теория и практика.: Пер. с англ. – М.: Вильямс, 2003. – 1500 с.: ил.

5 Jennings R., Professional .NET 3.5 Framework.: – New York.: Wrox, 2009. – 560 с.: ил.

ПРИЛОЖЕНИЕ А

(обязательное)

КОНЦЕПТУАЛЬНАЯ СХЕМА БД

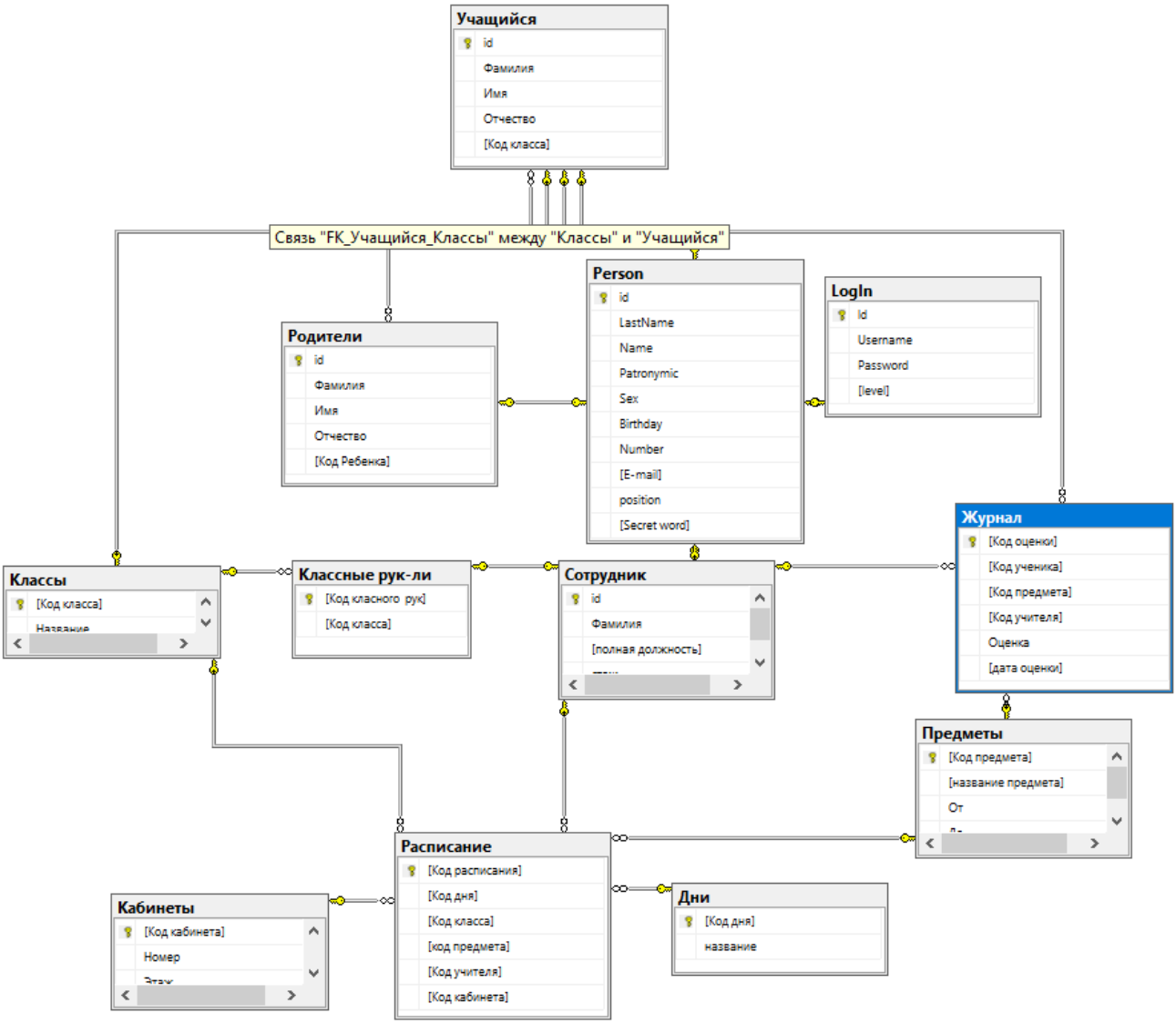

Рисунок А.1 – Инфологическая схема

ПРИЛОЖЕНИЕ Б

(обязательное)

СХЕМА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

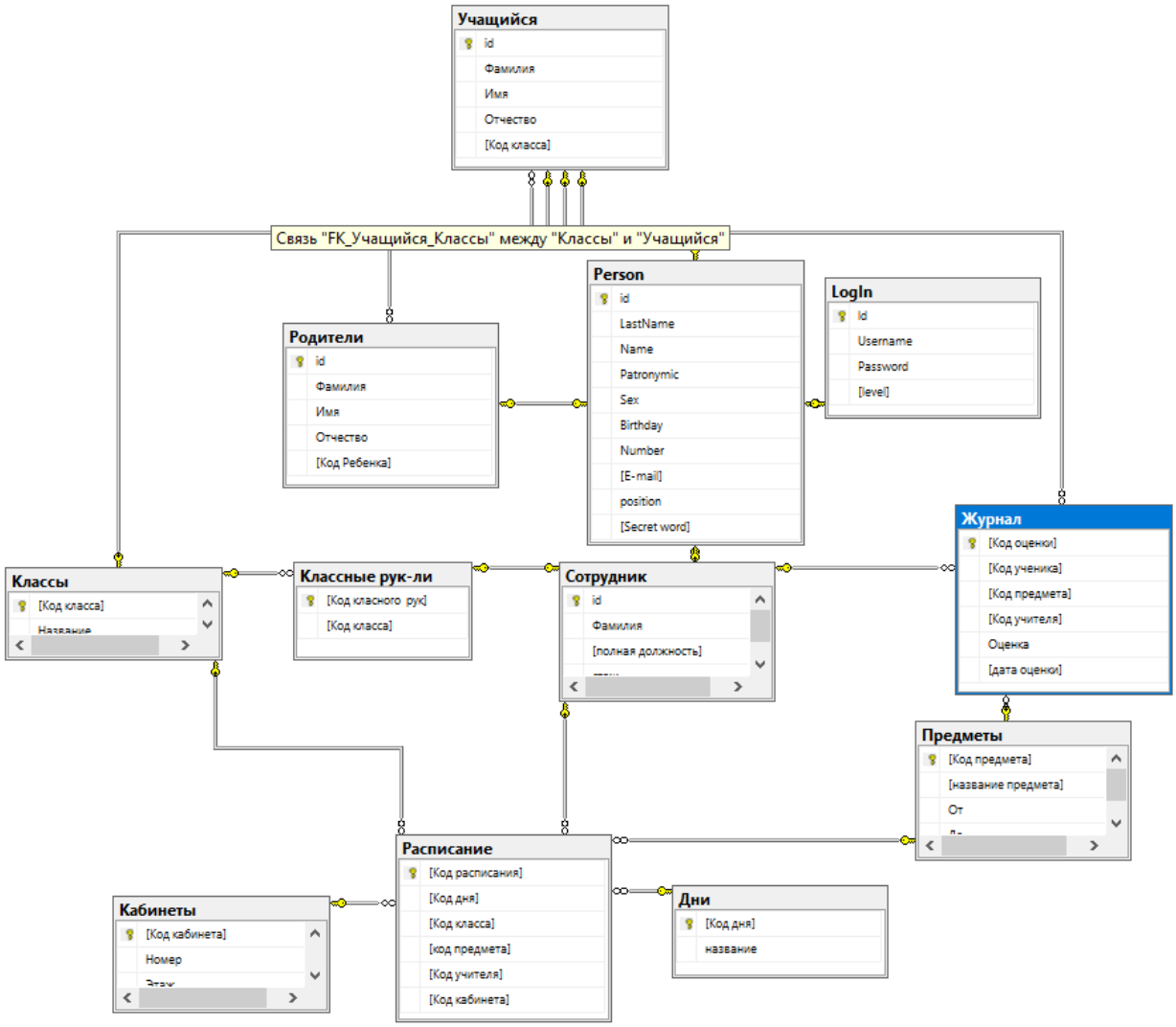

Рисунок Б.1 – Схема реляционной базы данных

ПРИЛОЖЕНИЕ В

(обязательное)

ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ

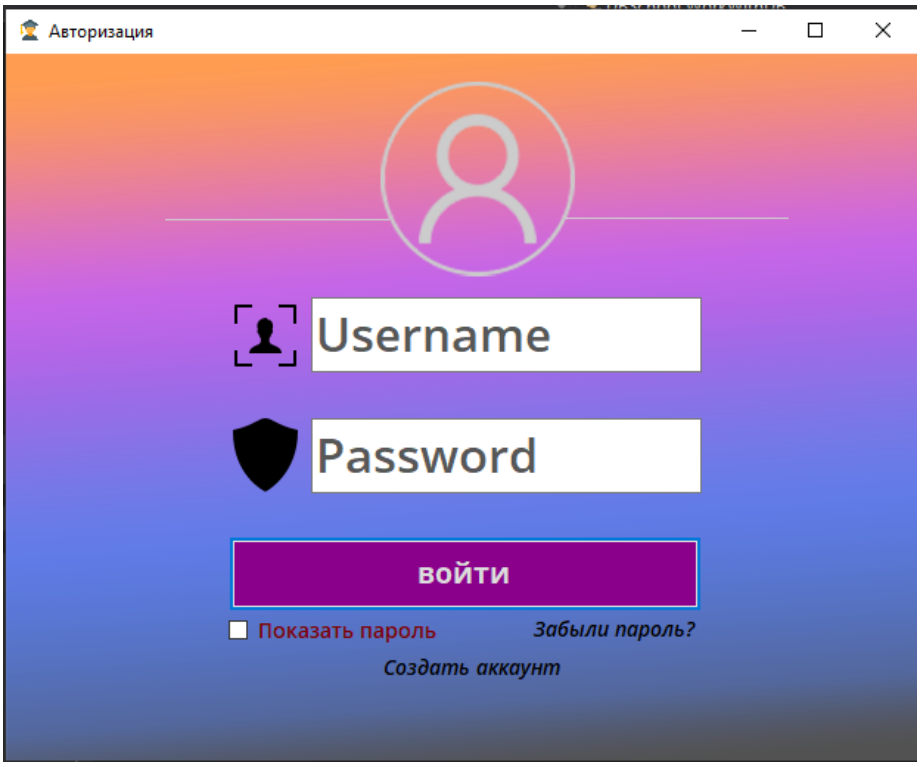

Рисунок В.1 – Главная форма приложения

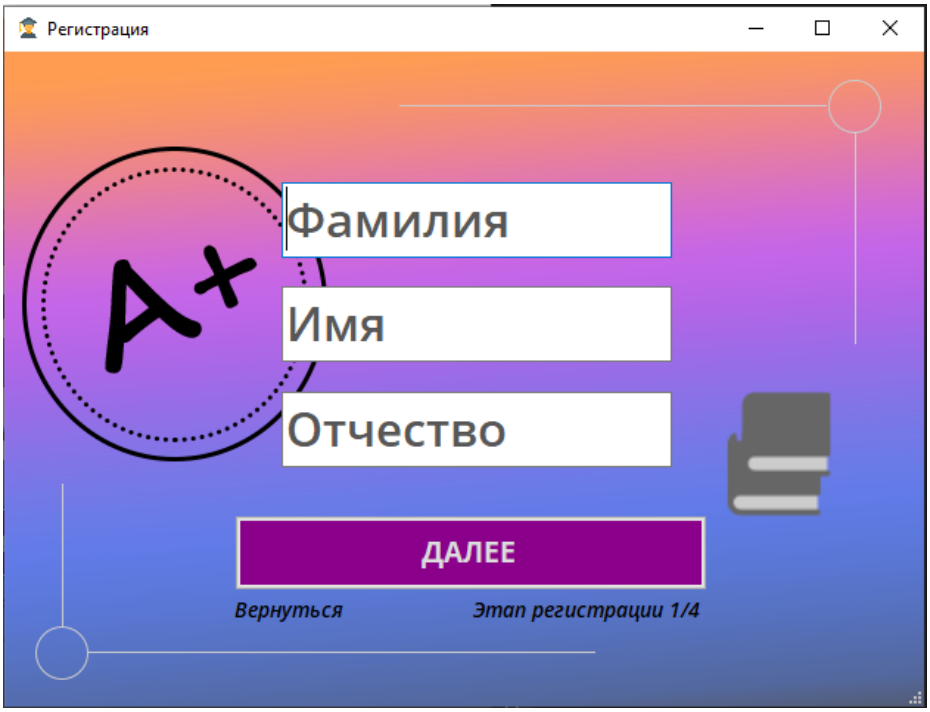

Рисунок В.2 – Диалоговое окно «Регистрация этап 1»

Рисунок В.3 – Подсистема сотрудника

Рисунок В.4 Подсистема учащегося и родителя

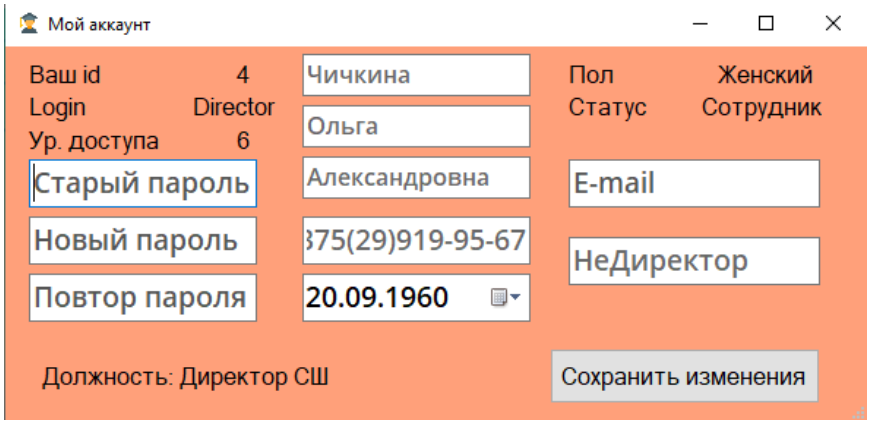

Рисунок В.5 Настройка аккаунта

Рисунок В.6 Просмотр сотрудников школы

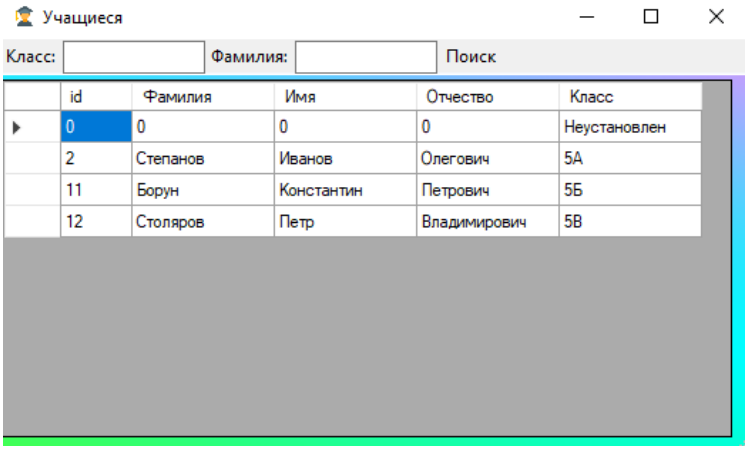

Рисунок В.7 Просмотр учащихся школы

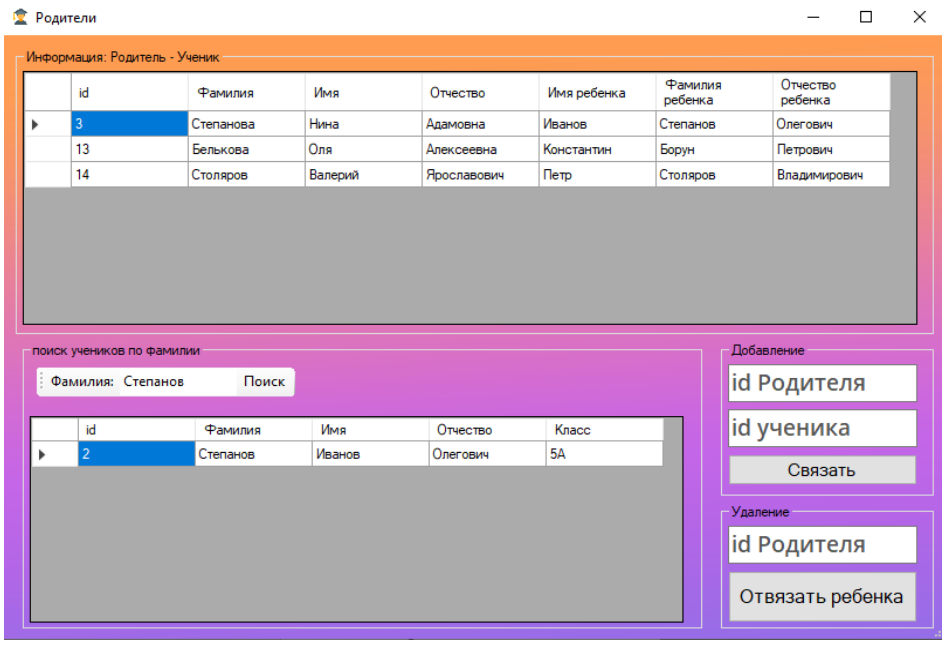

Рисунок В.8 Просмотр и настройка связей между родителями и учениками

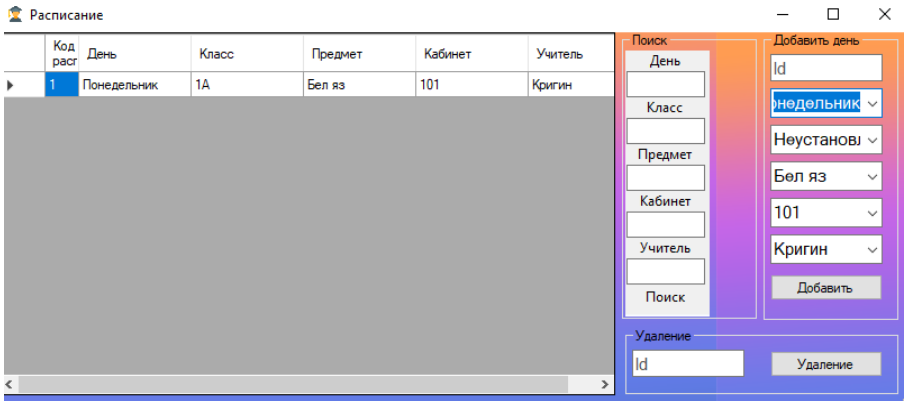

Рисунок В.9 Просмотр и настройка расписания уроков

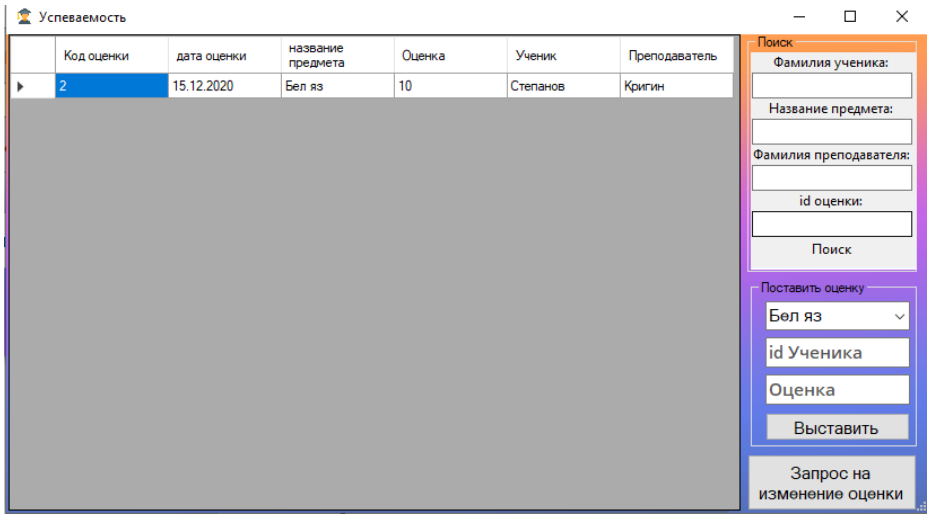

Рисунок В.19 Просмотр и выставление оценок ученикам

ПРИЛОЖЕНИЕ Г

(обязательное)

ЛИСТИНГ ПРОГРАММЫ

Исходный код класса WorkWithDB

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Deployment.Application;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DBSchool
{
    class WorkWithDB
    {
        private const string connectionString = @"Data
Source=ASUS-ZENBOOK;Initial Catalog=DBSchool;Integrated
Security=True";

        private SqlConnection sqlConnection;

        public WorkWithDB()
        {
            sqlConnection = new
SqlConnection(connectionString);
        }

        public List<Person> getActualDB()
        {
            List<Person> Persons = new List<Person>();

            sqlConnection = new
SqlConnection(connectionString);

            sqlConnection.Open();

            SqlDataReader sqlDataReader = null;
            SqlCommand sqlCommand = new SqlCommand("SELECT
LogIn.Id, LogIn.Username, LogIn.Password, LogIn.level, Person.[E-
mail],Person.Number, Person.[Secret word] FROM[LogIn], Person
where LogIn.Id = Person.id", sqlConnection);

            try
            {
                sqlDataReader = sqlCommand.ExecuteReader();
```

```
                while (sqlDataReader.Read())
                {
                    Person somePerson = new Person();
                    somePerson.id                                   =
Convert.ToInt32(sqlDataReader["Id"]);
                    somePerson.username                             =
sqlDataReader["Username"].ToString();
                    somePerson.password                             =
sqlDataReader["Password"].ToString();
                    somePerson.level                                =
Convert.ToInt32(sqlDataReader["level"]);
                    somePerson.number                               =
sqlDataReader["Number"].ToString();
                    somePerson.secretWord                           =
sqlDataReader["Secret word"].ToString();
                    somePerson.email    =    sqlDataReader["E-
mail"].ToString();
                    Persons.Add(somePerson);
                }
                sqlDataReader.Close();


            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlDataReader != null)
                {
                    sqlDataReader.Close();
                    sqlConnection.Close();
                }
            }


            return Persons;
        }

        public Person getFullInformation(int id)
        {
            Person somePerson = new Person();

            sqlConnection                =                  new
SqlConnection(connectionString);

            sqlConnection.Open();

            SqlDataReader sqlDataReader = null;
```

```
            SqlCommand sqlCommand = new SqlCommand("SELECT
LogIn.Id, Username, Password, level, " +
                "LastName, Name, Patronymic, Sex, Birthday,
Number, [E-mail], position, [Secret word] FROM [LogIn], " +
                $"Person where LogIn.Id ={id} AND LogIn.Id =
Person.id", sqlConnection);

            try
            {
                sqlDataReader = sqlCommand.ExecuteReader();
                sqlDataReader.Read();
                somePerson.id                                =
Convert.ToInt32(sqlDataReader["Id"]);
                somePerson.username                          =
sqlDataReader["Username"].ToString();
                somePerson.password                          =
sqlDataReader["Password"].ToString();
                somePerson.level                             =
Convert.ToInt32(sqlDataReader["level"]);
                somePerson.lastName                          =
sqlDataReader["LastName"].ToString();
                somePerson.name                              =
sqlDataReader["Name"].ToString();
                somePerson.patronymic                        =
sqlDataReader["Patronymic"].ToString();
                somePerson.sex                               =
sqlDataReader["Sex"].ToString();
                somePerson.birthday                          =
sqlDataReader["Birthday"].ToString();
                somePerson.number                            =
sqlDataReader["Number"].ToString();
                somePerson.email      =      sqlDataReader["E-
mail"].ToString();
                somePerson.position                          =
sqlDataReader["position"].ToString();
                somePerson.secretWord  =  sqlDataReader["Secret
word"].ToString();
                sqlDataReader.Close();


                sqlCommand = new SqlCommand("getFullPosition",
sqlConnection);
                sqlCommand.CommandType                       =
System.Data.CommandType.StoredProcedure;
                // параметр для ввода имени
                SqlParameter idParam = new SqlParameter
                {
                    ParameterName = "@id",
                    Value = somePerson.id
                };
                sqlCommand.Parameters.Add(idParam);
```

```
                somePerson.fullPosition = "";
                if (somePerson.level >= 3)
                {
                    somePerson.fullPosition                    =
sqlCommand.ExecuteScalar().ToString();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlDataReader != null)
                {
                    sqlDataReader.Close();

                }
                sqlConnection.Close();
            }

            return somePerson;
        }
        public bool addUser(Person somePerson)
        {

            SqlConnection       sqlConnection       =       new
SqlConnection(connectionString);

            sqlConnection.Open();

            SqlCommand   sqlCommand   =   new   SqlCommand("",
sqlConnection);

            sqlCommand.CommandText   =   $"INSERT   INTO   LogIn
(Username,           Password,           level)           VALUES
('{somePerson.username}','{somePerson.password}',
{somePerson.level})";

            try
            {
                sqlCommand.ExecuteNonQuery();
            }
            catch(Exception exp)
            {
                MessageBox.Show("Не могу вас добавить\nПричина:
"     +     exp.Message,     "Error",     MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }
            SqlDataReader sqlDataReader = null;
```

```
            sqlCommand.CommandText = $"SELECT [Id] FROM LogIn
WHERE Username='{somePerson.username}'";
            sqlDataReader = sqlCommand.ExecuteReader();
            sqlDataReader.Read();
            somePerson.id
=Convert.ToInt32(sqlDataReader["Id"]);
            sqlDataReader.Close();

            sqlCommand.CommandText = $"INSERT INTO Person
(Id,LastName, Name, Patronymic, Sex, Birthday, Number, [E-mail],
position, [Secret word]) " +
                $"VALUES
({somePerson.id},'{somePerson.lastName}','{somePerson.name}', " +
                $"'{somePerson.patronymic}',
'{somePerson.sex}'," +
                $"'{somePerson.birthday}',
'{somePerson.number}'," +
                $"'{somePerson.email}',
'{somePerson.position}'," +
                $"'{somePerson.secretWord}')";

            try
            {
                sqlCommand.ExecuteNonQuery();
            }
            catch (Exception exp)
            {
                MessageBox.Show("Не могу вас добавить\nПричина:
"     +     exp.Message,     "Error",     MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }

            string sql = "";
            switch (somePerson.level)
            {
                case 1:
                    sql   =   $"INSERT   INTO   [Учащийся]   (id,
[Фамилия], [Имя], [Отчество], [Код класса]) " +
                        $"VALUES              ({somePerson.id},
'{somePerson.lastName}',                    '{somePerson.name}',
'{somePerson.patronymic}', 0)";
                    break;
                case 2:
                    sql   =   $"INSERT   INTO   [Родители]   (id,
[Фамилия], [Имя], [Отчество], [Код ребенка]) " +
                        $"VALUES              ({somePerson.id},
'{somePerson.lastName}',                    '{somePerson.name}',
'{somePerson.patronymic}', 0)";
                    break;
                case 3:
```

```
                        sql = $"INSERT INTO [Сотрудник] (id,
[Фамилия], [полная должность], [стаж]) " +
                              $"VALUES ({somePerson.id},
'{somePerson.lastName}', '{somePerson.position}',0)";
                        break;
                }
                sqlCommand.CommandText = sql;

                try
                {
                    sqlCommand.ExecuteNonQuery();
                }
                catch (Exception exp)
                {
                    MessageBox.Show("Не могу вас добавить\nПричина:
"     +     exp.Message,     "Error",     MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }

                sqlConnection.Close();

                string text = "<h2>Вы были зарегестрированы в DB
School</h2><p>Это автоматическое сообщение о регистрации вашей
почты в приложении DB School</p>" +
                    $"<hr><p>{somePerson.lastName}
{somePerson.name}, не потеряйте свои данные для входа!</p>" +
                    $"<p>Ваш                                    логин:
<i><b>{somePerson.username}</b></i></p>" +
                    $"<p>Ваш                                  пароль:
<i><b>{somePerson.password}</b></i></p>" +
                    $"<h4>А так же не разглашайте эти данные и
лучше удалите данное сообщение</h4>";
                try
                {
                    mailbot.send(somePerson.email,   "Регистрация",
text);
                }
                catch (Exception exp)
                {
                    MessageBox.Show("Вы зарегестрированы но я не
могу вам отправить данные на E-mail\nПодробнее" + exp.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return true;
                }
                MessageBox.Show("Регистрация успешно завершена!",
"Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);
                return true;
            }
            public bool updateUser(Person somePerson)
            {
                bool result = false;
```

```
            string sql = $"Update Person set LastName =
'{somePerson.lastName}', Name = '{somePerson.name}', Patronymic =
'{somePerson.patronymic}', " +
                $"Number = '{somePerson.number}', Birthday =
'{somePerson.birthday}', [E-mail] = '{somePerson.email}', [Secret
word] = '{somePerson.secretWord}' " +
                $"where id = {somePerson.id}";
            sqlConnection.Open();
            SqlCommand sqlCommand = new SqlCommand(sql,
sqlConnection);
            try
            {
                sqlCommand.ExecuteNonQuery();
                result = true;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                sqlConnection.Close();
            }
            return result;
        }
        public string getClass(int id)
        {
            string Class = "";
            sqlConnection                =                new
SqlConnection(connectionString);

            sqlConnection.Open();
            try
            {

                SqlDataReader sqlDataReader = null;
                SqlCommand       sqlCommand       =       new
SqlCommand("getClassName", sqlConnection);
                sqlCommand.CommandType                          =
System.Data.CommandType.StoredProcedure;
                // параметр для ввода имени
                SqlParameter idParam = new SqlParameter
                {
                    ParameterName = "@id",
                    Value = id
                };
                sqlCommand.Parameters.Add(idParam);

                Class = sqlCommand.ExecuteScalar().ToString();

            }
```

```
            catch
            {
                try
                {
                    SqlCommand      sqlCommand      =      new
SqlCommand("getidChild", sqlConnection);
                    sqlCommand.CommandType                    =
System.Data.CommandType.StoredProcedure;
                    // параметр для ввода имени
                    SqlParameter idParam = new SqlParameter
                    {
                        ParameterName = "@idC",
                        Value = id
                    };
                    sqlCommand.Parameters.Add(idParam);

                    int                 newId                 =
Convert.ToInt32(sqlCommand.ExecuteScalar().ToString());
                    sqlConnection.Close();
                    Class = getClass(newId);
                }
                catch(Exception ex)
                {
                    MessageBox.Show(ex.Message,      ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);

                }


            }

            return Class;
        }
        public void UpdatePassword(int id, string password)
        {
              sqlConnection.Open();

              SqlCommand   sqlCommand   =   new   SqlCommand("",
sqlConnection);

              sqlCommand.CommandText   =   $"UPDATE   LogIn   SET
Password = '{password}' WHERE id = {id}";
              try
              {
                   sqlCommand.ExecuteNonQuery();
              }
              catch (Exception e)
            {
                   MessageBox.Show(e.Message,          e.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
              sqlConnection.Close();
```

```
        }
        public int UpdateParentChild(int idParent, int idChild)
        {
            int res = 0;
            sqlConnection = new
SqlConnection(connectionString);

            sqlConnection.Open();
            try
            {

                SqlDataReader sqlDataReader = null;
                SqlCommand sqlCommand = new
SqlCommand("updateParentChild", sqlConnection);
                sqlCommand.CommandType =
System.Data.CommandType.StoredProcedure;
                // параметр для ввода имени
                SqlParameter idP = new SqlParameter
                {
                    ParameterName = "@idP",
                    Value = idParent
                };
                sqlCommand.Parameters.Add(idP);
                SqlParameter idC = new SqlParameter
                {
                    ParameterName = "@idC",
                    Value = idChild
                };
                sqlCommand.Parameters.Add(idC);

                res = sqlCommand.ExecuteNonQuery();


            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                sqlConnection.Close();
            }
            return res;
        }
        public void addMark(int idPeople, int idObject, int
idTeacher, int mark)
        {
            sqlConnection.Open();
            try
            {
```

```
                string sql = $"Insert into Журнал
Values({idPeople}, {idObject}, {idTeacher}, {mark}, GETDATE())";
                SqlCommand sqlCommand = new SqlCommand(sql,
sqlConnection);
                sqlCommand.ExecuteNonQuery();
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                sqlConnection.Close();
            }

        }
        public int getIdBySQL(string sql)
        {
            int id = -1;
            sqlConnection.Open();
            SqlCommand sqlCommand = new SqlCommand(sql,
sqlConnection);
            try
            {
                id                                              =
Convert.ToInt32(sqlCommand.ExecuteScalar());
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                sqlConnection.Close();
            }
            return id;
        }
        public void addTimeTable(int day, int classOfSchool,
int Object, int cabinet, int teacher)
        {
            string sql = $"Insert into Расписание Values({day},
{classOfSchool}, {Object}, {teacher}, {cabinet})";
            sqlConnection.Open();

            SqlCommand sqlCommand = new SqlCommand(sql,
sqlConnection);
            try
            {
                sqlCommand.ExecuteNonQuery();
            }
```

```
                catch(Exception ex)
                {
                    MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    sqlConnection.Close();
                }
            }
            public  void  updateTimeTable(int  id,  int  day,  int
classOfSchool, int Object, int cabinet, int teacher )
            {
                string sql = $"Update Расписание set [Код дня] =
{day}, [Код класса] = {classOfSchool}, [код предмета] = {Object},
[Код учителя] = {teacher}, [Код кабинета] = {cabinet} " +
                    $"where [Код расписания] = {id}";
                sqlConnection.Open();

                SqlCommand   sqlCommand   =   new   SqlCommand(sql,
sqlConnection);
                try
                {
                    sqlCommand.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    sqlConnection.Close();
                }
            }
            public void deleteFromTimeTable(int id)
            {
                string  sql  =  $"DELETE  FROM  Расписание  WHERE  [Код
расписания] = {id}";
                sqlConnection.Open();
                SqlCommand   sqlCommand   =   new   SqlCommand(sql,
sqlConnection);
                try
                {
                    sqlCommand.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message,          ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
```

```
            {
                sqlConnection.Close();
            }
        }
    }
}
```

Исходный код класса Person

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DBSchool
{
    public class Person
    {
        public int id { set; get; }
        public int level { set; get; }
        public string lastName { set; get; }
        public string name { set; get; }
        public string patronymic { set; get; }
        public string sex { set; get; }
        public string birthday { set; get; }
        public string number { set; get; }
        public string email { set; get; }
        public string position { set; get; }
        public string secretWord { set; get; }
        public string username { set; get; }
        public string password { set; get; }
        public string fullPosition { set; get; }

        public Person()
        {
            id = 0;
            level = 0;
            lastName = "";
            name = "";
            patronymic = "";
            sex = "";
            birthday = "";
            number = "";
            email = "";
            position = "";
            secretWord = "";
            username = "";
            password = "";
        }
```

```
        }
    }
```

Исходный код класса mailBot

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Text;
using System.Threading.Tasks;

namespace DBSchool
{
    class mailbot
    {
        public mailbot()
        {

        }

        public static void send(string Email, string caption,
string text)
        {
            // отправитель - устанавливаем адрес и отображаемое в
письме имя
            MailAddress from = new
MailAddress("dbschool.krigin.psu@gmail.com", "DB School");
            // кому отправляем
            MailAddress to = new MailAddress(Email);
            // создаем объект сообщения
            MailMessage m = new MailMessage(from, to);
            // тема письма
            m.Subject = caption;
            // текст письма
            m.Body = text;
            //"<h2>Вы были зарегестрированы в DB
School</h2><p>Это автоматическое сообщение о регистрации вашей
почты в приложении DB School</p>";
            // письмо представляет код html
            m.IsBodyHtml = true;
            // адрес smtp-сервера и порт, с которого будем
отправлять письмо
            SmtpClient smtp = new SmtpClient("smtp.gmail.com",
587);
            // логин и пароль
            smtp.Credentials = new
NetworkCredential("dbschool.krigin.psu@gmail.com", "Q2PrsWeU");
            smtp.EnableSsl = true;
            smtp.Send(m);
        }
```

```
    }
}
```

Исходный код класса student

```
using System;
using DBSchool.DBForms;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DBSchool
{
    public partial class student : Form
    {
        private Person person;
        public student(object person)
        {
            InitializeComponent();
            this.person = (Person)person;
            WorkWithDB workWithDB = new WorkWithDB();
            this.person                                          =
workWithDB.getFullInformation(this.person.id);
            NamePosition_l.Text = this.person.lastName + " " +
this.person.name + ", " + this.person.position;
            //lvl_l.Text = this.person.level.ToString();
            Class_l.Text = workWithDB.getClass(this.person.id);
        }

        private    void    student_FormClosed(object    sender,
FormClosedEventArgs e)
        {
            Environment.Exit(0);
        }

        private void MyAcc_l_Click(object sender, EventArgs e)
        {
            myAccount account = new myAccount(person.id);
            account.ShowDialog();
        }

        private    void    MyAcc_l_MouseEnter(object    sender,
EventArgs e)
        {
            MyAcc_l.ForeColor = Color.Orange;
        }
```

```
        private    void    MyAcc_l_MouseLeave(object    sender,
EventArgs e)
        {
            MyAcc_l.ForeColor = Color.Empty;
        }

        private  void  parents_b_Click(object  sender,  EventArgs
e)
        {
            Parents parents = new Parents(person.level);
            parents.ShowDialog();
        }

        private void Exit_l_Click(object sender, EventArgs e)
        {
            aunt a = new aunt();
            a.Show();
            this.Dispose();
        }

        private void Exit_l_MouseEnter(object sender, EventArgs
e)
        {
            Exit_l.ForeColor = Color.Orange;
        }

        private void Exit_l_MouseLeave(object sender, EventArgs
e)
        {
            Exit_l.ForeColor = Color.Empty;

        }

        private  void  studiers_b_Click(object  sender,  EventArgs
e)
        {
            Studiers studiers = new Studiers();
            studiers.ShowDialog();
        }

        private void Employers_b_Click(object sender, EventArgs
e)
        {
            employers emp = new employers();
            emp.ShowDialog();
        }

        private void marks_b_Click(object sender, EventArgs e)
        {
            marks marks = new marks(person);
            marks.ShowDialog();
        }
```

```
            private    void    toTimeTable_b_Click(object    sender,
EventArgs e)
            {
                TimeTable table = new TimeTable(person);
                table.ShowDialog();
            }
        }
    }
```

Исходный код класса singin

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;
    using System.Text.RegularExpressions;
    using System.Net.Mail;
    using System.Data;
    using System.Data.SqlClient;
    using DBSchool.DBForms;

    namespace DBSchool
    {
        public partial class singIn : Form
        {
             Person somePerson = new Person();
             int stage = 1;
             public singIn()
             {
                  InitializeComponent();
                  updateStage();
             }
             private void updateStage()
             {
                  stage_l.Text    =    "Этап    регистрации    "    +
stage.ToString() + "/4";
                  switch (stage)
                  {
                       case 1:
                            {
                                 Sex_cb.Visible = false;
                                 Bithday_dtp.Visible = false;
                                 PhoneNumber_tb.Visible = false;
                                 tb_2.UseSystemPasswordChar        =
false;
                                 tb_3.UseSystemPasswordChar        =
false;
```

```
                                tb_1.Visible = true;
                                tb_1.Text = somePerson.lastName;
                                tb_1.Cue = "Фамилия";

                                tb_2.Visible = true;
                                tb_2.Text = somePerson.name;
                                tb_2.Cue = "Имя";

                                tb_3.Visible = true;
                                tb_3.Text = somePerson.patronymic;
                                tb_3.Cue = "Отчество";
                                tb_3.Font  =  new  Font("Open  Sans
Semibold", 24);
                                NextStep_b.Text = "ДАЛЕЕ";
                                break;
                           }
                      case 2:
                           {
                                tb_1.Visible = false;
                                Sex_cb.Visible = true;
                                tb_2.UseSystemPasswordChar        =
false;
                                tb_3.UseSystemPasswordChar        =
false;

                                Sex_cb.Items.Clear();
                                Sex_cb.Items.Add("Мужской");
                                Sex_cb.Items.Add("Женский");
                                Sex_cb.Text = "Пол";
                                Sex_cb.SelectedIndex = -1;

                                tb_2.Visible = false;
                                Bithday_dtp.Visible = true;
                                Bithday_dtp.Value                 =
somePerson.birthday               !=              ""              ?
Convert.ToDateTime(somePerson.birthday) : Bithday_dtp.Value;

                                tb_3.Visible = false;
                                PhoneNumber_tb.Visible = true;
                                NextStep_b.Text = "ДАЛЕЕ";
                                break;
                           }
                      case 3:
                           {
                                Bithday_dtp.Visible = false;
                                PhoneNumber_tb.Visible = false;
                                tb_2.UseSystemPasswordChar        =
false;
                                tb_3.UseSystemPasswordChar        =
false;
```

```
                                    Sex_cb.Items.Clear();
                                    Sex_cb.Items.Add("Учащийся");
                                    Sex_cb.Items.Add("Родитель");
                                    Sex_cb.Items.Add("Сотрудник");
                                    Sex_cb.Text = "Должность";
                                    Sex_cb.SelectedIndex = -1;

                                    tb_2.Visible = true;
                                    tb_2.Cue = "E-mail";
                                    tb_2.Text = "";

                                    tb_3.Visible = true;
                                    tb_3.Text = "";
                                    tb_3.Cue = "Кодовое слово";
                                    tb_3.Font  =  new  Font("Open  Sans
Semibold", 24);
                                    NextStep_b.Text = "ДАЛЕЕ";
                                    break;
                               }
                          case 4:
                               {
                                    Sex_cb.Visible = false;
                                    Bithday_dtp.Visible = false;

                                    tb_1.Visible = true;
                                    tb_1.Cue = "Username";
                                    tb_1.Text = "";

                                    tb_2.Visible = true;
                                    tb_2.Cue = "Password";
                                    tb_2.Text = "";
                                    tb_2.UseSystemPasswordChar = true;

                                    tb_3.Visible = true;
                                    tb_3.Cue = "Confirm Password";
                                    tb_3.Text = "";
                                    tb_3.UseSystemPasswordChar = true;

                                    NextStep_b.Text = "ГОТОВО";
                                    break;
                               }
                     }

               }

               private void NextStep_b_Click(object sender, EventArgs
e)
               {
                     switch (stage)
                     {
                          case 1:
                               {
```

```
                                    Regex rgx = new Regex(@"^[А-Я][а-
я]+$");
                                    if (tb_1.Text.Length < 2)
                                    {
                                         MessageBox.Show("В     фамилии
слишком мало символов!", "Ошибка!");
                                         return;
                                    }
                                    if (!rgx.IsMatch(tb_1.Text))
                                    {
                                         MessageBox.Show("В     фамилии
найдены неверные символы!", "Ошибка!");
                                         return;
                                    }
                                    if (tb_2.Text.Length < 2)
                                    {
                                         MessageBox.Show("В       имени
слишком мало символов!", "Ошибка!");
                                         return;
                                    }
                                    if (!rgx.IsMatch(tb_2.Text))
                                    {
                                         MessageBox.Show("В       имени
найдены неверные символы!", "Ошибка!");
                                         return;
                                    }
                                    if(tb_3.Text.Length != 0)
                                    {
                                         if (!rgx.IsMatch(tb_3.Text))
                                         {
                                              MessageBox.Show("В
отчестве найдены неверные символы!", "Ошибка!");
                                              return;
                                         }
                                    }
                                    somePerson.lastName = tb_1.Text;
                                    somePerson.name = tb_2.Text;
                                    somePerson.patronymic =tb_3.Text ;
                                    stage = 2;
                                    updateStage();
                                    break;
                               }
                          case 2:
                               {
                                    if   (Sex_cb.SelectedIndex.Equals(-
1))
                                    {
                                         MessageBox.Show("Вы         не
выбрали ваш пол!", "Ошибка!");
                                         return;
                                    }
```

```
                                    somePerson.sex                      =
Sex_cb.SelectedItem.ToString();
                                    if((DateTime.Now.Year               -
Bithday_dtp.Value.Year) < 6)
                                    {
                                         MessageBox.Show("Ваш   возраст
слишком мал для регистрации на данном сервисе", "Ошибка!");
                                         return;
                                    }
                                    somePerson.birthday                 =
Bithday_dtp.Value.Day + "." +
                                       Bithday_dtp.Value.Month + "."+
                                       Bithday_dtp.Value.Year;
                                    if (!PhoneNumber_tb.MaskCompleted)
                                    {
                                         MessageBox.Show("Номер
телефона это обязательное поле!", "Ошибка!");
                                         return;
                                    }

  if(!PhoneNumber_tb.Text.Contains("(25)") &&

  !PhoneNumber_tb.Text.Contains("(29)")&&

  !PhoneNumber_tb.Text.Contains("(33)")&&

  !PhoneNumber_tb.Text.Contains("(44)"))
                                    {
                                         MessageBox.Show("Нет    такого
кода оператора", "Ошибка!");
                                         return;
                                    }


                                    somePerson.number                   =
PhoneNumber_tb.Text;
                                    stage = 3;
                                    updateStage();
                                    break;
                               }
                          case 3:
                               {
                                    if   (Sex_cb.SelectedIndex.Equals(-
1))
                                    {
                                         MessageBox.Show("Вы         не
выбрали вашу должность!", "Ошибка!");
                                         return;
                                    }
                                    somePerson.position                 =
Sex_cb.SelectedItem.ToString();
```

```
                                somePerson.level                    =
Sex_cb.SelectedIndex + 1 ;
                                try
                                {

                                     MailAddress     m     =     new
MailAddress(tb_2.Text);
                                }
                                catch (FormatException)
                                {
                                     MessageBox.Show("Вы      ввели
неверный E-mail!", "Ошибка!");
                                     return;
                                }
                                catch (ArgumentException)
                                {

                                }
                                somePerson.email = tb_2.Text ;
                                if(tb_3.Text.Length < 2)
                                {
                                     MessageBox.Show("Кодовое
слово обязательно!\nоно будет использовано при утере пароля",
"Ошибка!");
                                     return;
                                }
                                somePerson.secretWord = tb_3.Text;
                                stage = 4;
                                updateStage();
                                break;
                           }
                      case 4:
                           {
                                if (tb_1.Text.Length < 3)
                                {
                                     MessageBox.Show("Username
должен состоять как минимум из 3 символов");
                                     return;
                                }
                                Regex  regex  =  new  Regex(@"[A-Za-
z0-9]+$");
                                if (!regex.IsMatch(tb_1.Text))
                                {
                                     MessageBox.Show("username
долен состоять из англиских букв и цифр", "Ошибка!");
                                     return;
                                }
                                somePerson.username = tb_1.Text;
                                if(tb_2.Text.Length < 8)
                                {
                                     MessageBox.Show("Пароль
должен содержать не менее 8 символов", "Ошибка!");
```

```
                                        return;
                                    }
                                    if (!regex.IsMatch(tb_2.Text))
                                    {
                                        MessageBox.Show("Пароль  долен
состоять из англиских букв и цифр", "Ошибка!");
                                        return;
                                    }
                                    if (!tb_3.Text.Equals(tb_2.Text))
                                    {
                                        MessageBox.Show("Пароли
должны совпадать!", "Ошибка!");
                                        return;
                                    }
                                    somePerson.password = tb_2.Text;

                                    WorkWithDB   workWithDB   =   new
WorkWithDB();
                                    if
(!workWithDB.addUser(somePerson)) return;
                                    if(somePerson.level == 3)
                                {
                                        ChangeFullPositions
changeFullPositions = new ChangeFullPositions(somePerson);

  changeFullPositions.ShowDialog();
                                }
                                        else if(somePerson.level == 2)
                                {
                                        MessageBox.Show("Родитель");
                                }
                                        else
                                {
                                        MessageBox.Show("Ученик");
                                }
                                        this.Close();
                                        break;
                                }
                        }


                }


                #region Animation
                private void back_l_Click(object sender, EventArgs e)
                {
                        if(stage !=1)
                                stage--;
                        updateStage();

                }
```

```
            private    void    back_l_MouseEnter(object    sender,
EventArgs e)
            {
                back_l.ForeColor = Color.Orange;
            }

            private    void    back_l_MouseLeave(object    sender,
EventArgs e)
            {
                back_l.ForeColor = Color.Black;
            }
            #endregion

            private   void   NextStep_b_MouseEnter(object   sender,
EventArgs e)
            {
                NextStep_b.BackColor = Color.Magenta;
            }

            private   void   NextStep_b_MouseLeave(object   sender,
EventArgs e)
            {
                NextStep_b.BackColor = Color.DarkMagenta;
            }
        }
    }
```

Исходный код класса myAccount

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Net.Mail;
    using System.Text;
    using System.Text.RegularExpressions;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace DBSchool
    {
        public partial class myAccount : Form
        {
            Person person;
            public myAccount(int id)
            {
                InitializeComponent();
                person = new Person();
                person.id = id;
            }
```

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

                WorkWithDB workWithDB = new WorkWithDB();
                person = workWithDB.getFullInformation(person.id);
                id_l.Text = person.id.ToString();
                login_l.Text = person.username;
                lvl_l.Text = person.level.ToString();

                LastName_tb.Text = person.lastName;
                Name_tb.Text = person.name;
                patr_tb.Text = person.patronymic;
                PhoneNumber_tb.Text = person.number;
                Birthday_dtp.Value =
DateTime.Parse(person.birthday);

                sex_l.Text = person.sex;
                status_l.Text = person.position;
                email_ctm.Text = person.email;
                secretWord_ctb.Text = person.secretWord;

                full_posit_l.Text = person.fullPosition == "" ? ""
: "Должность: " + person.fullPosition;

            }

            private void save_b_Click(object sender, EventArgs e)
            {
                if (old_pass_ctb.Text.Equals(person.password))
                {
                    Regex regex = new Regex(@"[A-Za-z0-9]+$");
                    if (!regex.IsMatch(new_pass_ctb.Text))
                    {
                        MessageBox.Show("Пароль должен состоять из
английских букв и цифр", "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        return;
                    }
                    if
(!new_pass_ctb.Text.Equals(repeat_pass_tb.Text))
                    {
                        MessageBox.Show("Пароли не совпадают",
"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return;
                    }
                    if(new_pass_ctb.Text.Length < 8)
                    {
                        MessageBox.Show("СЛишком короткий пароль",
"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return;
                    }
```

```
                    WorkWithDB work = new WorkWithDB();
                    work.UpdatePassword(person.id,
new_pass_ctb.Text);
                }
                else if(old_pass_ctb.Text.Length != 0)
                {
                    MessageBox.Show("Введен неверный старый
пароль!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                person.lastName = LastName_tb.Text;
                person.name = Name_tb.Text;
                person.patronymic = patr_tb.Text;
                if (!PhoneNumber_tb.MaskCompleted)
                {
                    MessageBox.Show("Номер телефона указан
неверно", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                person.number = PhoneNumber_tb.Text;

                if ((DateTime.Now.Year - Birthday_dtp.Value.Year) <
6)
                {
                    MessageBox.Show("Ваш возраст слишком мал для
регистрации на данном сервисе", "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }
               person.birthday = Birthday_dtp.Value.Day + "." +
                   Birthday_dtp.Value.Month + "." +
                   Birthday_dtp.Value.Year;
                try
                {

                    MailAddress m = new
MailAddress(email_ctm.Text);
                }
                catch (FormatException)
                {
                    MessageBox.Show("Вы ввели неверный E-mail!",
"Ошибка!", MessageBoxButtons.OK,MessageBoxIcon.Error);
                    return;
                }
                person.email = email_ctm.Text;
                person.secretWord = secretWord_ctb.Text;
                WorkWithDB workWithDB = new WorkWithDB();
                if (workWithDB.updateUser(person))
                {
                    this.Close();
                }
```

```
            }
        }
    }
```

Исходный код класса employer

```
    using System;
    using DBSchool.DBForms;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace DBSchool
    {
        public partial class employer : Form
        {
            private Person person;

            public employer(object person)
            {
                InitializeComponent();
                this.person = (Person)person;
                WorkWithDB workWithDB = new WorkWithDB();
                this.person                                          =
workWithDB.getFullInformation(this.person.id);
                NamePosition_l.Text = this.person.lastName + " " +
this.person.name + ", " + this.person.position;
                lvl_l.Text  =  this.person.level.ToString()  +  "  из
7";
                FullPosition_l.Text = this.person.fullPosition;
                if (this.person.level >= 6) toDB_b.Enabled = true;
            }

            private   void   employer_FormClosing(object   sender,
FormClosingEventArgs e)
            {
                Environment.Exit(0);
            }

            private    void    MyAcc_l_MouseEnter(object    sender,
EventArgs e)
            {
                MyAcc_l.ForeColor = Color.Orange;
            }

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

```
            MyAcc_l.ForeColor = Color.Empty;
        }

        private void Exit_l_MouseEnter(object sender, EventArgs
e)
        {
            Exit_l.ForeColor = Color.Orange;
        }

        private void Exit_l_MouseLeave(object sender, EventArgs
e)
        {
            Exit_l.ForeColor = Color.Empty;
        }

        private void Exit_l_Click(object sender, EventArgs e)
        {
            aunt a = new aunt();
            a.Show();
            this.Dispose();
        }

        private  void  parents_b_Click(object  sender,  EventArgs
e)
        {
            Parents p = new Parents(person.level);
            p.ShowDialog();
        }

        private void MyAcc_l_Click(object sender, EventArgs e)
        {
            myAccount account = new myAccount(person.id);
            account.ShowDialog();
        }

        private  void  studiers_b_Click(object  sender,  EventArgs
e)
        {
            Studiers studiers = new Studiers();
            studiers.ShowDialog();
        }

        private void Employers_b_Click(object sender, EventArgs
e)
        {
            employers emp = new employers();
            emp.ShowDialog();
        }

        private void marks_Click(object sender, EventArgs e)
        {
            marks m = new marks(person);
```

```
            m.ShowDialog();
        }

        private    void    time_table_b_Click(object    sender,
EventArgs e)
        {
            TimeTable table = new TimeTable(person);
            table.ShowDialog();
        }

        private  void  changeFullPosition_b_Click(object  sender,
EventArgs e)
        {
            ChangeFullPositions   changeFullPositions   =   new
ChangeFullPositions(person);
            changeFullPositions.ShowDialog();
        }

        private void toDB_b_Click(object sender, EventArgs e)
        {
            ControlDB controlDB = new ControlDB();
            controlDB.ShowDialog();
        }
    }
}
```

Исходный код класса aunt

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;

namespace DBSchool
{
    public partial class aunt : Form
    {
        SqlConnection sqlConnection;
        List<Person> Persons;
        private bool remember;

        public aunt()
        {
            InitializeComponent();
            WorkWithDB workWithDB = new WorkWithDB();
            Persons = workWithDB.getActualDB();
```

```
                remember = false;
            }

            private string CreatePassword()
            {
                int[] arr = new int[16]; // сделаем длину пароля в
16 символов
                Random rnd = new Random();
                string Password = "";

                for (int i = 0; i < arr.Length; i++)
                {
                    arr[i] = rnd.Next(48, 122);
                    if (arr[i] == 96) arr[i]++;
                    Password += (char)arr[i];
                }
                return Password;
            }

            private void CreateAcc_L_Click(object sender, EventArgs
e)
            {
                singIn s = new singIn();
                s.ShowDialog();
                WorkWithDB workWithDB = new WorkWithDB();
                Persons = workWithDB.getActualDB();
            }
            private void showPass_cb_CheckedChanged(object sender,
EventArgs e)
            {
                if (showPass_cb.Checked)
                {
                    password_tb.UseSystemPasswordChar = false;
                }
                else
                {
                    password_tb.UseSystemPasswordChar = true;
                }
            }
            private void Log_in_b_Click(object sender, EventArgs e)
            {
                Person foundPerson = new Person();
                foundPerson = null;
                badLogIn_l.Visible = true;


                if (remember)
                {
                    if (username_tb.Cue.Equals("Username"))
                    {
                        foreach (Person tempPerson in Persons)
                        {
```

```
                            if
(username_tb.Text.Equals(tempPerson.username) &&
password_tb.Text.Equals(tempPerson.secretWord))
                            {
                                badLogIn_l.Visible = false;
                                foundPerson = tempPerson;
                            }
                        }
                    }
                    else
                    {
                        foreach (Person tempPerson in Persons)
                        {
                            if
(username_tb.Text.Equals(tempPerson.number) &&
password_tb.Text.Equals(tempPerson.secretWord))
                            {
                                badLogIn_l.Visible = false;
                                foundPerson = tempPerson;
                            }
                        }

                    }

                    string newPassword = CreatePassword();
                    if (!badLogIn_l.Visible &&
!foundPerson.email.Equals(""))
                    {
                        WorkWithDB workWithDB = new WorkWithDB();
                        workWithDB.UpdatePassword(foundPerson.id,
newPassword);
                        Persons = workWithDB.getActualDB();
                        try
                        {
                            string text = "<h2>Заявка на новый
пароль одобрена!</h2>" +
                                $"<p>Наша система сгененировала вам
новый пароль: {newPassword}</p>" +
                                $"<hr><p>Постарайтесь больше не
забывать ваш пароль. <i>Совет: храните пароль в специальном
приложении для менеджера паролей.</i></p>";
                            mailbot.send(foundPerson.email,
"Восстановление пароля", text);
                            MessageBox.Show("Все отлично, ваш
пароль был отправлен вам на почту.", "Восстановление пароля",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        }
                        catch(Exception exp)
                        {
                            MessageBox.Show("По каким либо причинам
мы не можем отправить вам пароль на E-mail. Ваш новый пароль: " +
newPassword +
```

```
                                    " Пароль скопирован в буфер
обмена.\nОшибка: " + exp.Message, exp.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                Clipboard.SetText(newPassword);
                            }
                            dontRemember_l_Click(null, null);
                            return;

                        }
                        else if (!badLogIn_l.Visible)
                        {
                            WorkWithDB workWithDB = new WorkWithDB();
                            workWithDB.UpdatePassword(foundPerson.id,
newPassword);
                            Persons = workWithDB.getActualDB();
                            MessageBox.Show("Если бы у вас был E-mail
мы бы выслали пароль туда. ЭТО БЫЛО БЫ БЕЗОПАСНЕЕ!!!\nA пока вы не
завели почту вот ваш пароль:\n" +
                                $"{newPassword} Он скопирован в буфер
обмена вашего ПК\nСовет: заведите Электронную почту, " +
                                $"и храните пароли в специальных
приложениях", "Восстановление пароля", MessageBoxButtons.OK,
MessageBoxIcon.Warning) ;
                            Clipboard.SetText(newPassword);
                            dontRemember_l_Click(null, null);
                            return;
                        }
                    }
                    else
                    {
                        foreach (Person tempPerson in Persons)
                        {
                            if
(username_tb.Text.Equals(tempPerson.username) &&
password_tb.Text.Equals(tempPerson.password))
                            {
                                badLogIn_l.Visible = false;
                                foundPerson = tempPerson;
                            }
                        }
                    }

                    if (foundPerson != null)
                    {
                        if(foundPerson.level < 3)
                        {
                            student st = new student(foundPerson);
                            st.Show();
                            this.Hide();
                        }
                        else
                        {
```

```
                    employer em = new employer(foundPerson);
                    em.Show();
                    this.Hide();
                }
            }
        }
        private void dontRemember_l_Click(object sender,
EventArgs e)
        {
            if (dontRemember_l.Text.Equals("Отмена"))
            {
                dontRemember_l.Text = "Забыли пароль?";
                username_tb.Text = "";
                username_tb.Cue = "Username";
                password_tb.Text = "";
                password_tb.Cue = "Password";
                showPass_cb.Enabled = true;
                showPass_cb.Checked = false;
                showPass_cb_CheckedChanged(null, null);
                remember = false;
                return;
            }
            DialogResult dialogResult =
MessageBox.Show("Помните ли вы ваш username?", "Уведомление",
MessageBoxButtons.YesNoCancel);

            switch (dialogResult)
            {
                case DialogResult.Yes:
                    {
                        username_tb.Text = "";
                        username_tb.Cue = "Username";

                        break;
                    }
                case DialogResult.No:
                    {
                        username_tb.Text = "";
                        username_tb.Cue = "Номер телефона";
                        break;
                    }
                case DialogResult.Cancel:
                    {
                        return;
                    }
            }
            dontRemember_l.Text = "Отмена";
            password_tb.Text = "";
            password_tb.Cue = "Кодовое слово";
            password_tb.UseSystemPasswordChar = false;
            showPass_cb.Enabled = false;
            remember = true;
```

```
            }


            #region Animation
            private void username_tb_TextChanged(object sender,
EventArgs e)
            {
                badLogIn_l.Visible = false;
            }
            private void password_tb_TextChanged(object sender,
EventArgs e)
            {
                badLogIn_l.Visible = false;
            }
            private void dontRemember_l_MouseEnter(object sender,
EventArgs e)
            {
                dontRemember_l.ForeColor = Color.Orange;
            }
            private void dontRemember_l_MouseLeave(object sender,
EventArgs e)
            {
                dontRemember_l.ForeColor = Color.Black;
            }
            private void Log_in_b_MouseEnter(object sender,
EventArgs e)
            {
                Log_in_b.BackColor = Color.Magenta;
            }
            private void Log_in_b_MouseLeave(object sender,
EventArgs e)
            {
                Log_in_b.BackColor = Color.DarkMagenta;
            }
            private void CreateAcc_L_MouseEnter(object sender,
EventArgs e)
            {
                CreateAcc_L.ForeColor = Color.Orange;
            }
            private void CreateAcc_L_MouseLeave(object sender,
EventArgs e)
            {
                CreateAcc_L.ForeColor = Color.Black;
            }

            #endregion

            private void aunt_FormClosing(object sender,
FormClosingEventArgs e)
            {
                Environment.Exit(0);
```

```
            }
        }
    }
```

Исходный код класса TimeTable

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace DBSchool.DBForms
    {
        public partial class TimeTable : Form
        {
            public TimeTable(Person person)
            {
                InitializeComponent();
                if(person.level < 5)
                {
                    groupBox2.Enabled = false;
                    groupBox3.Enabled = false;
                }
            }

            private void fillToolStripButton_Click(object sender,
EventArgs e)
            {
                if (day_tb.Text.Equals("")) day_tb.Text = "%";
                if (class_tb.Text.Equals("")) class_tb.Text = "%";
                if (object_tb.Text.Equals("")) object_tb.Text =
"%";
                if (cabinet_tb.Text.Equals("")) cabinet_tb.Text =
"%";
                if (teacher_tb.Text.Equals("")) teacher_tb.Text =
"%";
                try
                {

this.getTimeTableTableAdapter.Fill(this.dBSchoolDataSet.getTimeTab
le, day_tb.Text, class_tb.Text, object_tb.Text, cabinet_tb.Text,
teacher_tb.Text);
                }
                catch (System.Exception ex)
                {

System.Windows.Forms.MessageBox.Show(ex.Message);
                }
```

```
            if (day_tb.Text.Equals("%")) day_tb.Text = "";
            if (class_tb.Text.Equals("%")) class_tb.Text = "";
            if (object_tb.Text.Equals("%")) object_tb.Text =
"";
            if (cabinet_tb.Text.Equals("%")) cabinet_tb.Text =
"";
            if (teacher_tb.Text.Equals("%")) teacher_tb.Text =
"";
        }

        private void TimeTable_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.getTeacher". При необходимости
она может быть перемещена или удалена.

this.getTeacherTableAdapter.Fill(this.dBSchoolDataSet.getTeacher);
            // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.getCabinet". При необходимости
она может быть перемещена или удалена.

this.getCabinetTableAdapter.Fill(this.dBSchoolDataSet.getCabinet);
            // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.getObjects". При необходимости
она может быть перемещена или удалена.

this.getObjectsTableAdapter.Fill(this.dBSchoolDataSet.getObjects);
            // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.getClasses". При необходимости
она может быть перемещена или удалена.

this.getClassesTableAdapter.Fill(this.dBSchoolDataSet.getClasses);
            // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.getDays". При необходимости она
может быть перемещена или удалена.

this.getDaysTableAdapter.Fill(this.dBSchoolDataSet.getDays);

        }

        private void id_ctb_TextChanged(object sender,
EventArgs e)
        {
            if(id_ctb.Text.Length > 0)
            {
                add_tt_b.Text = "Изменить";
            }
            else
            {
                add_tt_b.Text = "Добавить";
            }
        }
```

```
            private void add_tt_b_Click(object sender, EventArgs e)
            {
                WorkWithDB workWithDB = new WorkWithDB();
                int day = workWithDB.getIdBySQL($"select Дни.[Код
дня] from Дни where Дни.название =
'{getDaysComboBox.SelectedValue.ToString()}'");
                int classOfSchool = workWithDB.getIdBySQL($"select
Классы.[Код класса] from Классы where Классы.Название =
'{getClassesComboBox.SelectedValue.ToString()}'");
                int obj = workWithDB.getIdBySQL($"select
Предметы.[Код предмета] from Предметы where Предметы.[название
предмета] = '{getObjectsComboBox.SelectedValue.ToString()}'");
                int cabinet = workWithDB.getIdBySQL($"select
Кабинеты.[Код кабинета] from Кабинеты where Кабинеты.Номер =
{getCabinetComboBox.SelectedValue.ToString()}");
                int teacher = workWithDB.getIdBySQL($"select
Сотрудник.id from Сотрудник where Сотрудник.Фамилия =
'{getTeacherComboBox.SelectedValue.ToString()}'");
                if (add_tt_b.Text.Equals("Добавить"))
                {
                    workWithDB.addTimeTable(day, classOfSchool,
obj, cabinet, teacher);
                    MessageBox.Show("Добавление завершено");
                }
                else
                {
                    int id = -1;
                    try
                    {
                       id = Convert.ToInt32(id_ctb.Text);
                    }
                    catch(Exception ex)
                    {
                        MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return;
                    }
                    workWithDB.updateTimeTable(id, day,
classOfSchool, obj, cabinet, teacher);
                    MessageBox.Show("Изменения сохранены");
                }
                fillToolStripButton_Click(null, null);
            }

            private void delete_Click(object sender, EventArgs e)
            {
                DialogResult dialogResult =
                MessageBox.Show("Вы уверены что хотите удалить эту
запись из расписания?", "Подтвердите",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning);
```

```
                if(dialogResult.Equals(DialogResult.No) ||
dialogResult.Equals(DialogResult.Cancel))
                {
                    return;
                }
                int id = -1;
                try
                {
                    id = Convert.ToInt32(idForDelete_ctb.Text);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                WorkWithDB workWithDB = new WorkWithDB();
                workWithDB.deleteFromTimeTable(id);
                MessageBox.Show("Удалено");
            }
        }
    }
```

Исходный код класса Studyers

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace DBSchool.DBForms
    {
        public partial class Studiers : Form
        {
            public Studiers()
            {
                InitializeComponent();
            }


            private void fillToolStripButton_Click_3(object sender,
EventArgs e)
            {
                if (ClassName_tb.Text.Equals("")) ClassName_tb.Text
= "%";
                if (LastName_tb.Text.Equals("")) LastName_tb.Text =
"%";
                try
```

```
                {

this.getPeopleByClassTableAdapter.Fill(this.dBSchoolDataSet.getPeo
pleByClass, ClassName_tb.Text, LastName_tb.Text);
                }
                catch (System.Exception ex)
                {

System.Windows.Forms.MessageBox.Show(ex.Message);
                }
                if (ClassName_tb.Text.Equals("%"))
ClassName_tb.Text = "";
                if (LastName_tb.Text.Equals("%")) LastName_tb.Text
= "";

            }
        }
    }
```

Исходный код класса Parents

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace DBSchool
    {
        public partial class Parents : Form
        {
            int lvl;
            public Parents(int lvl)
            {
                InitializeComponent();
                this.lvl = lvl;
            }

            private void updateinfo()
            {
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.getParets".При необходимости она
может быть перемещена или удалена.

this.getParetsTableAdapter.Fill(this.dBSchoolDataSet.getParets);
            }
            private void Parents_Load(object sender, EventArgs e)
            {
                updateinfo();
```

```
                bool permission = false;
                if (lvl > 3) permission = true;
                idParent_tb.Enabled = permission;
                idPeople_tb.Enabled = permission;
                unite_b.Enabled = permission;

                idParetn_del_tb.Enabled = permission;
                divide_b.Enabled = permission;


            }

            private void fillToolStripButton_Click_1(object sender,
EventArgs e)
            {
                try
                {

this.getPeopleByFamilTableAdapter.Fill(this.dBSchoolDataSet.getPeo
pleByFamil, familToolStripTextBox.Text);
                }
                catch (System.Exception ex)
                {

System.Windows.Forms.MessageBox.Show(ex.Message);
                }

            }

            private void unite_b_Click(object sender, EventArgs e)
            {
                int.TryParse(idParent_tb.Text, out int idP);
                if(idP < 1)
                {
                    MessageBox.Show("Неверный id Родителя",
"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                int.TryParse(idPeople_tb.Text, out int idC);
                if (idC < 1)
                {
                    MessageBox.Show("Неверный id Ребенка",
"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                WorkWithDB workWithDB = new WorkWithDB();
                if(workWithDB.UpdateParentChild(idP, idC) == 0)
                {
                    MessageBox.Show("Неудалось связать родителя и
ребенка\nПроверьте данные", "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
```

```
                }
                updateinfo();
                idParent_tb.Text = "";
                idPeople_tb.Text = "";
            }

            private void divide_b_Click(object sender, EventArgs e)
            {
                int.TryParse(idParetn_del_tb.Text, out int idP);
                if (idP < 1)
                {
                    MessageBox.Show("Неверный id Родителя",
"Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                DialogResult result = MessageBox.Show("Вы
дейсвительно хотите отвязать ребенка от этого родителя?",
"Подтвердите", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
                if (!result.Equals(DialogResult.Yes)) return;
                WorkWithDB workWithDB = new WorkWithDB();
                if (workWithDB.UpdateParentChild(idP, 0) == 0)
                {
                    MessageBox.Show("Неудалось связать родителя и
ребенка\nПроверьте данные", "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }
                updateinfo();
                idParetn_del_tb.Text = "";
            }
        }
    }
```

Исходный код класса Parents

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace DBSchool.DBForms
    {
        public partial class marks : Form
        {
            Person Im;
            public marks(object person)
            {
                InitializeComponent();
```

```
                Im = (Person)person;
                if (Im.level == 1)
                {
                    lastName_st_tb.Text = Im.lastName;
                    lastName_st_tb.Enabled = false;
                }
                if(Im.level < 4)
                {
                    groupBox2.Enabled = false;
                }
            }

            private void fillToolStripButton_Click(object sender,
EventArgs e)
            {
                if (lastName_st_tb.Text.Equals(""))
lastName_st_tb.Text = "%";
                if (NameObject_tb.Text.Equals(""))
NameObject_tb.Text = "%";
                if (LastName_tc_tb.Text.Equals(""))
LastName_tc_tb.Text = "%";
                if (idMark_tb.Text.Equals("")) idMark_tb.Text =
"%";
                try
                {

this.getMarkTableAdapter.Fill(this.dBSchoolDataSet.getMark,
lastName_st_tb.Text, NameObject_tb.Text, LastName_tc_tb.Text,
idMark_tb.Text);
                }
                catch (System.Exception ex)
                {

System.Windows.Forms.MessageBox.Show(ex.Message);
                }
                if (lastName_st_tb.Text.Equals("%"))
lastName_st_tb.Text = "";
                if (NameObject_tb.Text.Equals("%"))
NameObject_tb.Text = "";
                if (LastName_tc_tb.Text.Equals("%"))
LastName_tc_tb.Text = "";
                if (idMark_tb.Text.Equals("%")) idMark_tb.Text =
"";
            }

            private void marks_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.getObjects". При необходимости
она может быть перемещена или удалена.

this.getObjectsTableAdapter.Fill(this.dBSchoolDataSet.getObjects);
```

```
            }

            private void button1_Click(object sender, EventArgs e)
            {
                int id = -1;
                int int_mark = -1;
                WorkWithDB workWithDB = new WorkWithDB();
                try
                {
                    id = Convert.ToInt32(idPeople.Text);
                    int_mark = Convert.ToInt32(mark.Text);
                    if (int_mark < 1 || int_mark > 10) throw new
Exception("Неверно задана оценка");
                }
                catch(Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                workWithDB.addMark(id,
getObjectsComboBox.SelectedIndex+1, Im.id, int_mark);

            }

            private void toChangeMark_Click(object sender,
EventArgs e)
            {
                ChangeMark changeMark = new ChangeMark(Im);
                changeMark.ShowDialog();
            }
        }
    }
```

Исходный код класса employers

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;

    namespace DBSchool.DBForms
    {
        public partial class employers : Form
        {
            public employers()
            {
```

```
            InitializeComponent();
        }

        private void fillToolStripButton_Click(object sender,
EventArgs e)
        {
            if (Position_tb.Text.Equals("")) Position_tb.Text =
"%";
            if (Familiya_tb.Text.Equals("")) Familiya_tb.Text =
"%";
            try
            {

this.getEmployersByPosFamilTableAdapter.Fill(this.dBSchoolDataSet.
getEmployersByPosFamil, Position_tb.Text, Familiya_tb.Text);
            }
            catch (System.Exception ex)
            {

System.Windows.Forms.MessageBox.Show(ex.Message);
            }
            Position_tb.Text = "";
            Familiya_tb.Text = "";
        }
    }
}
```

Исходный код класса controlDB

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DBSchool.DBForms
{
    public partial class ControlDB : Form
    {
        public ControlDB()
        {
            InitializeComponent();
        }

        private void logInBindingNavigatorSaveItem_Click(object
sender, EventArgs e)
        {
            this.Validate();
            this.logInBindingSource.EndEdit();
```

```
this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);

            }

            private void ControlDB_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Журнал". При необходимости она
может быть перемещена или удалена.

this.журналTableAdapter.Fill(this.dBSchoolDataSet.Журнал);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Расписание". При необходимости
она может быть перемещена или удалена.

this.расписаниеTableAdapter.Fill(this.dBSchoolDataSet.Расписание);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Кабинеты". При необходимости она
может быть перемещена или удалена.

this.кабинетыTableAdapter.Fill(this.dBSchoolDataSet.Кабинеты);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Дни". При необходимости она
может быть перемещена или удалена.

this.дниTableAdapter.Fill(this.dBSchoolDataSet.Дни);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Предметы". При необходимости она
может быть перемещена или удалена.

this.предметыTableAdapter.Fill(this.dBSchoolDataSet.Предметы);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet._Классные_рук_ли". При
необходимости она может быть перемещена или удалена.

this.классные_рук_лиTableAdapter.Fill(this.dBSchoolDataSet._Классн
ые_рук_ли);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Классы". При необходимости она
может быть перемещена или удалена.

this.классыTableAdapter.Fill(this.dBSchoolDataSet.Классы);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Сотрудник". При необходимости
она может быть перемещена или удалена.

this.сотрудникTableAdapter.Fill(this.dBSchoolDataSet.Сотрудник);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Родители". При необходимости она
может быть перемещена или удалена.
```

```
this.родителиTableAdapter.Fill(this.dBSchoolDataSet.Родители);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Учащийся". При необходимости она
может быть перемещена или удалена.

this.учащийсяTableAdapter.Fill(this.dBSchoolDataSet.Учащийся);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.Person". При необходимости она
может быть перемещена или удалена.

this.personTableAdapter.Fill(this.dBSchoolDataSet.Person);
                // TODO: данная строка кода позволяет загрузить
данные в таблицу "dBSchoolDataSet.LogIn". При необходимости она
может быть перемещена или удалена.

this.logInTableAdapter.Fill(this.dBSchoolDataSet.LogIn);
            }

            private void
logInBindingNavigatorSaveItem_Click_1(object sender, EventArgs e)
            {
                this.Validate();
                this.logInBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            private void toolStripButton7_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.personBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
```

```
            private void toolStripButton14_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.учащийсяBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            private void toolStripButton21_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.родителиBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            private void toolStripButton28_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.сотрудникBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
```

```
            private void toolStripButton35_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.классыBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            private void toolStripButton42_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.классные_рук_лиBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            private void toolStripButton49_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.предметыBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
```

```
            private void toolStripButton56_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.дниBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            private void toolStripButton63_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.кабинетыBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }

            private void toolStripButton70_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.расписаниеBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
```

```
            private void toolStripButton77_Click(object sender,
EventArgs e)
            {
                this.Validate();
                this.журналBindingSource.EndEdit();
                try
                {

this.tableAdapterManager.UpdateAll(this.dBSchoolDataSet);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
```

Исходный код класса changeMark

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;
    using Word = Microsoft.Office.Interop.Word;

    namespace DBSchool.DBForms
    {
        public partial class ChangeMark : Form
        {
            Person person;
            public ChangeMark(Person person)
            {
                InitializeComponent();
                this.person = person;
            }

            private void ChangeMark_Load(object sender, EventArgs
e)
            {

            }

            private void sumbit_Click(object sender, EventArgs e)
            {
                try
```

```
                {
                    var wordApp = new Word.Application();
                    wordApp.Visible = true;
                    string p = Application.StartupPath.ToString();
                    var wordDocument = wordApp.Documents.Open(p +
"//documents//ChangeMarkTemplate.dotx");

                    ReplaceWordStub("{name}", person.name,
wordDocument);
                    ReplaceWordStub("{lastname}", person.lastName,
wordDocument);
                    ReplaceWordStub("{patronymic}",
person.patronymic, wordDocument);
                    ReplaceWordStub("{ID}", person.id.ToString(),
wordDocument);
                    ReplaceWordStub("{idMark}", IdMark.Text,
wordDocument);
                    ReplaceWordStub("{newMark}", newMark.Text,
wordDocument);


                    ReplaceWordStub("{reason}", reason.Text,
wordDocument);


wordDocument.SaveAs2($"{person.id}{person.lastName}{DateTime.Now.D
ay}{DateTime.Now.Month}{DateTime.Now.Year}.docx");
                    wordApp.Visible = true;
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            private void ReplaceWordStub(string stubToReplase,
string text, Word.Document wordDocument)
            {
                var range = wordDocument.Content;
                range.Find.ClearFormatting();
                range.Find.Execute(FindText: stubToReplase,
ReplaceWith: text, Replace: 2);
            }
        }
    }
```

Исходный код класса changeFullPosition

```
    using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Word = Microsoft.Office.Interop.Word;

namespace DBSchool.DBForms
{
    public partial class ChangeFullPositions : Form
    {
        Person person;
        public ChangeFullPositions(Person person)
        {
            InitializeComponent();
            this.person = person;
        }


        private void sumbit_Click(object sender, EventArgs e)
        {
            try
            {
                var wordApp = new Word.Application();
                wordApp.Visible = true;
                string p = Application.StartupPath.ToString();
                var wordDocument = wordApp.Documents.Open(p +
"//documents//ChangePositionTemplate.dotx");

                ReplaceWordStub("{name}", person.name,
wordDocument);
                ReplaceWordStub("{lastname}", person.lastName,
wordDocument);
                ReplaceWordStub("{patronymic}",
person.patronymic, wordDocument);
                ReplaceWordStub("{ID}", person.id.ToString(),
wordDocument);
                ReplaceWordStub("{full position}",
fullPosition.Text, wordDocument);

                ReplaceWordStub("{reason}", reason.Text,
wordDocument);


wordDocument.SaveAs2($"{person.id}{person.lastName}{DateTime.Now.D
ay}{DateTime.Now.Month}{DateTime.Now.Year}.docx");
                wordApp.Visible = true;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, ex.Source,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
            }

            private void ReplaceWordStub(string stubToReplase,
string text, Word.Document wordDocument)
            {
                var range = wordDocument.Content;
                range.Find.ClearFormatting();
                range.Find.Execute(FindText: stubToReplase,
ReplaceWith: text, Replace: 2);
            }
        }
    }
```