

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**   
**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | КТ |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Г.В. Аверин |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе бакалавра 3 курса

на тему:

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА ОБЩЕОБРАЗОВАТЕЛЬНОЙ ШКОЛЫ

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  |  | С.О. Подлужный |

подпись

|  |  |  |
| --- | --- | --- |
| **Направление** | **09.03.01** | **Информатика и вычислительная техника** |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст.преп. Ю.В. Котенко

подпись

Консультанты по разделам:

Моделирование бизнес-процессов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент В.И. Бондаренко

подпись

Проектирование системы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент В.И. Бондаренко

подпись

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. лаборант В.Г. Медведева

подпись

Курсовая работа защищена \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата итоговая оценка комиссии

Подписи членов комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Донецк  
2021

ГОУ ВПО «Донецкий национальный университет»

кафедра Компьютерных технологий

Утверждаю

Зав. кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

**ЗАДАНИЕ**

на курсовую работу студента 3 курса Подлужного С.О.

*Тема курсовой работы*: Автоматизированная информационная системаобщеобразовательной школы.

*Краткая постановка задачи*: 1. Изучить и проанализировать предметную область –деятельность школы. 2. Разработать техническое задание на создание приложения. 3. Разработать базу данных «Школа». 4. Разработать проект программного обеспечения информационной системы. 5. Разработать средствами языка C# электронную информационную систему «Школа», которая позволяет обрабатывать информацию таблиц базы данных. 6. Протестировать программное обеспечение. 7. Оформить отчёт.

*Исходные данные*: 1. Руководство по программированию в Windows Forms. 2. Документация по языку программирования C#. 3. Руководство по разработке баз данных 4. Учебно-педагогическая документация.

*Ожидаемые результаты*: электронная информационная система школы.

*Календарный план работы:*

|  |  |  |
| --- | --- | --- |
| Даты консультаций | Этапы выполнения работы | Отметки о выполнении |
| 13.09.2021 | Обсуждение литературы и постановка задачи | выполнено |
| 28.09.2021 | Предварительное утверждение содержания отчёта | выполнено |
| 29.09.2021 | Утверждение проекта, алгоритмов, методов, технологий | выполнено |
| 04.10.2021 | Ход реализации проекта | выполнено |
| 15.11.2021 | Обсуждения организации тестирования программы | выполнено |
| 01.12.2021 | Демонстрация программного продукта руководителю | выполнено |
| 02.12.2021 | Оформление отчёта | выполнено |
| 03.12.2021 | Предоставление отчёта руководителю | выполнено |

Дата выдачи задания 12.09.2021 года

Студент С.О. Подлужный

Руководитель Ю.В. Котенко

**АННОТАЦИЯ**

Отчет о курсовой работе: 93 с., 31 рис., 13 таблицы, 2 приложения, 10 источников.

Объектом исследования является муниципальное общеобразовательное учреждение «Классическая гуманитарная гимназия г. Донецка», его документооборот.

Предмет исследования – информационная система для автоматизации рабочих процессов муниципального общеобразовательного учреждения «Классическая гуманитарная гимназия г. Донецка».

Цель работы – разработка и создание информационной системы «Школа», позволяющая вести базу данных с информацией об учащихся, учителеях, событиях, расписании, успеваемости для муниципального общеобразовательного учреждения «Классическая гуманитарная гимназия г. Донецка».

Метод исследования – изучение особенностей создания автоматизированных информационных систем и анализа возможностей языка С# для реализации поставленной задачи.

В работе были использованы возможности СУБД SQL Server, средства администрирования SQL Server Management Studio, среды разработки Visual Studi и язык программирования C#, а также технологии Windows Forms, .NET Framework, JSON, XML.

В результате решения задач была разработана информационная система «Школа», позволяющая просматривать, изменять, добавлять сведения об учителях, учащихся и их успеваемости; формировать журналы событий и расписание, а также генерировать статистику по успеваемости.

Разработанное приложение может использоваться учителями, учащиямися и администрацией любых общеобразовательных учреждений в их профессиональной деятельности.

С#, WINDOWS FORMS, БАЗА ДАННЫХ, ИНФОРМАЦИОННАЯ СИСТЕМА, ШКОЛА, УЧИТЕЛЯ, УЧАЩИЕСЯ, ЖУРНАЛ СОБЫТИЙ.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc89351506)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc89351507)

[1.1 Состояние вопроса 7](#_Toc89351508)

[1.2 Моделирование существующих бизнес-процессов 8](#_Toc89351509)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 10](#_Toc89351510)

[2.1 Описание области применения и исходных данных приложения 10](#_Toc89351511)

[2.2 Требования к пользовательским интерфейсам 10](#_Toc89351512)

[2.3 Требования к аппаратным и программным интерфейсам 13](#_Toc89351513)

[2.4 Требования к пользователям продукта 13](#_Toc89351514)

[2.5 Требования к адаптации на месте 14](#_Toc89351515)

[2.6 Функции продукта 14](#_Toc89351516)

[2.7 Ограничения 14](#_Toc89351517)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 15](#_Toc89351518)

[4 РАЗРАБОТКА БАЗЫ ДАННЫХ И ПРИЛОЖЕНИЯ 16](#_Toc89351519)

[4.1 Разработка базы данных приложения 16](#_Toc89351520)

[*4.1.1 Концептуальное проектирование базы данных.* 16](#_Toc89351521)

[*4.1.2 Логическое проектирование базы данных.* 17](#_Toc89351522)

[*4.1.3 Физическое проектирование базы данных* 17](#_Toc89351523)

[4.2 Проектирование структуры приложения 21](#_Toc89351524)

[4.3 Описание объектов и их взаимодействия 23](#_Toc89351525)

[4.4 Описание SQL-запросов к базе данных системы 31](#_Toc89351526)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 40](#_Toc89351527)

[5.1 Аппаратные и программные средства создания и эксплуатации приложения 40](#_Toc89351528)

[5.2 Руководство пользователя 40](#_Toc89351529)

[5.3 Описание контрольных примеров 40](#_Toc89351530)

[ЗАКЛЮЧЕНИЕ 43](#_Toc89351531)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 44](#_Toc89351532)

[ПРИЛОЖЕНИЕ А Экранные формы 45](#_Toc89351533)

[ПРИЛОЖЕНИЕ Б Фрагменты листинга 53](#_Toc89351534)

# ВВЕДЕНИЕ

Современные информационные технологии внедряются во многие сферы деятельности человека, в том числе и в образование. Руководят образовательным процессом директор и его заместители. Последние осуществляют контроль результатов работы педагогического коллектива, принимают участие в подборе и расстановке кадров, установлению нагрузки, обеспечивают повышение квалификации педагогов, руководят работой преподавателей, ведут учет рабочего времени и т.д.

Несомненно, управлять образовательным процессом без информации невозможно. Специфика деятельности заместителя директора заключается в том, что все потоки информации об учителях, учащихся стекаются именно к нему, потому что без определенных данных невозможно качественно осуществлять организацию учебно-воспитательного процесса. Информация, которой должна владеть администрация общеобразовательного учреждения, с каждым годом становится все больше. Она быстро меняется. Часто ее необходимо обрабатывать в кратчайшие сроки и принимать решения.

Большое количество времени тратится на поиск, систематизацию информации, формирование выводов, таблиц. Возникает проблема управления данными, которое бы обеспечило наиболее эффективную работу. Поэтому в настоящее время особое внимание следует уделить внедрению новых информационных технологий в управлении школой, образовательным процессом.

Целью курсовой работы является разработка и создание информационной системы «Школа» для муниципального общеобразовательного учреждения «Классическая гуманитарная гимназия г. Донецка», позволяющей хранить и обрабатывать данные об учителях, учащихся, классах, успеваемости, расписании, производить быстрый поиск по вводимым критериям, создавать таблицы с информацией для заявки на курсы и заявки на оздоровление, создавать диаграммы успеваемости. Программа будет способствовать оптимизации работы заместителя директора, уменьшению времени на поиск и обработку информации, предоставит широкие возможности для автоматизации кадровой работы, администрирования учебно-воспитательного процесса.

Для реализации данной цели были поставлены следующие задачи:

1. изучить и проанализировать предметную область – деятельность школы;
2. разработать техническое задание на создание приложения;
3. разработать базу данных «Школа»;
4. разработать проект программного обеспечения информационной системы;
5. разработать средствами языка C# информационную систему «Школа»;
6. протестировать программное обеспечение;
7. оформить отчёт.

Объектом исследования является муниципальное общеобразовательное учреждение «Классическая гуманитарная гимназия г. Донецка», его документооборот.

Предмет исследования – электронная информационная система для муниципального общеобразовательного учреждения «Классическая гуманитарная гимназия г. Донецка».

Новизна работы заключается в том, что данная информационная система не имеет аналогов, она разрабатывалась специально под потребности администрации муниципального общеобразовательного учреждения «Классическая гуманитарная гимназия г. Донецка», при этом учитывалась специфика работы заместителя директора.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Состояние вопроса

В настоящее время существует большое количество информационных систем, разработанных для школ c различным функционалом. Но большинство систем строятся, отталкиваясь от требований заказчиков, а также видов общеобразовательного учреждения.

В результате проведенного анализа были рассмотрены существующие в сети аналоги приложений для общеобразовательных школ. Приведем для примера одну из проанализированных систем.

Рассмотрим информационную систему «Школа»: <http://isch.krasnoyarck.ru/>, и установим ее достоинства и недостатки.

Данная информационная система имеет следующие достоинства:

* формирование отчетов по движению учащихся за учебный год;
* формирование штатного расписания;
* введение архива уволенных сотрудников.

Недостатки данной информационной системы:

* нет функции автоматического собирания таблицы, для документации;
* не ведется журнал событий;
* не ведется учет успеваемости;
* нет функции прохождения курсов и аттестации;
* маленькое количество персональных данных об учителях и учащихся;
* нет системы поиска по таблицам.

Из этого следует вывод: рассмотренная программа имеет ряд преимуществ, однако не может быть использована в муниципальном общеобразовательном учреждении «Классическая гуманитарная гимназия г. Донецка» по следующим причинам:

* отсутствие функций, требуемых заказчиком;
* наличие лишних функций, которые не будут применяться в работе.

1.2 Моделирование существующих бизнес-процессов

IDEF0 ­­– нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции [7].

Произведём оценку системы в терминах бизнес-процессов организации с использованием методологий IDEF0. Первый шаг в построении модели – это определение цели модели, то есть вопросов, на которые призвана ответить модель. Вопросы следуют из формулировки задачи и анализа требований.

Построим контекстную диаграмму (самое общее описание системы и её взаимодействия с внешней средой). Контекстная диаграмма, описывающая работу школы, приведена на рисунке 1.1.

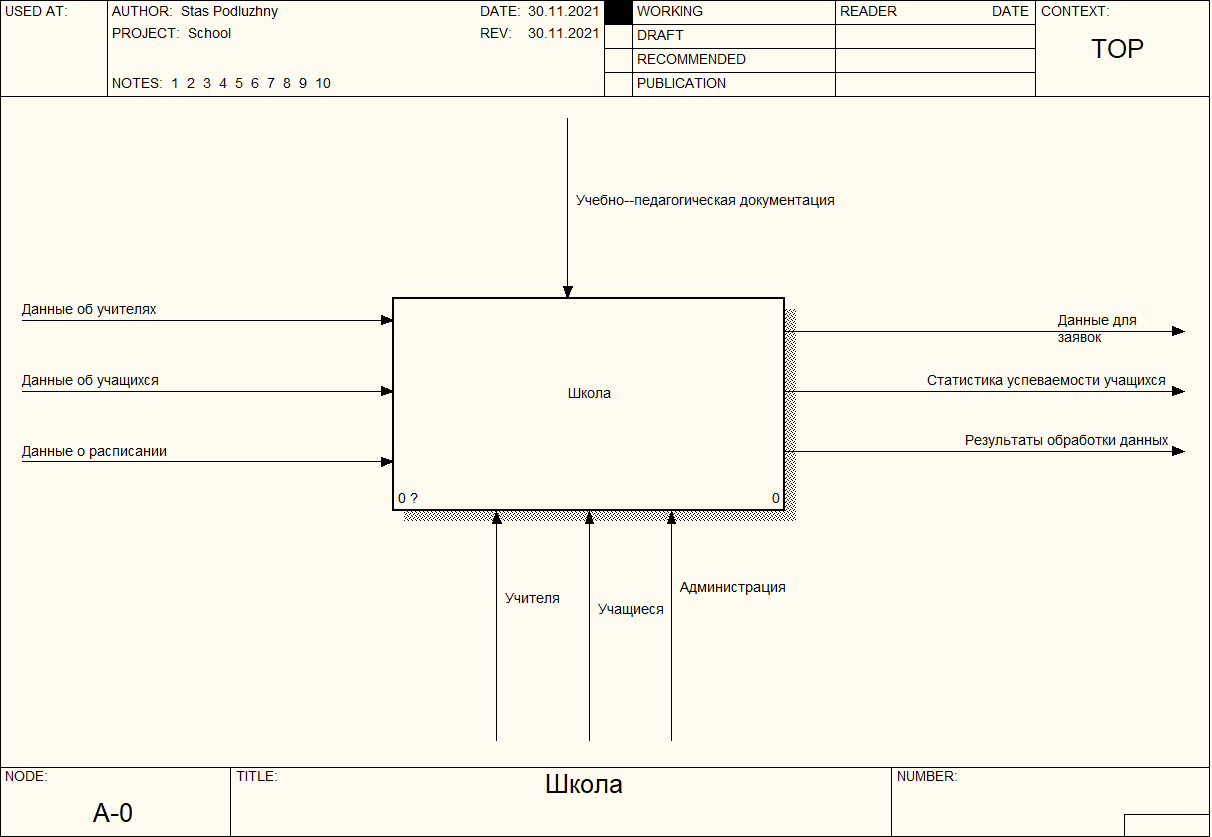


Рисунок 1.1 – Контекстная диаграмма

Проведём функциональную декомпозицию (разбиение сложного процесса на составляющие его функции) для более подробного описания системы. Диаграмма декомпозиции приведена на рисунке 1.2.

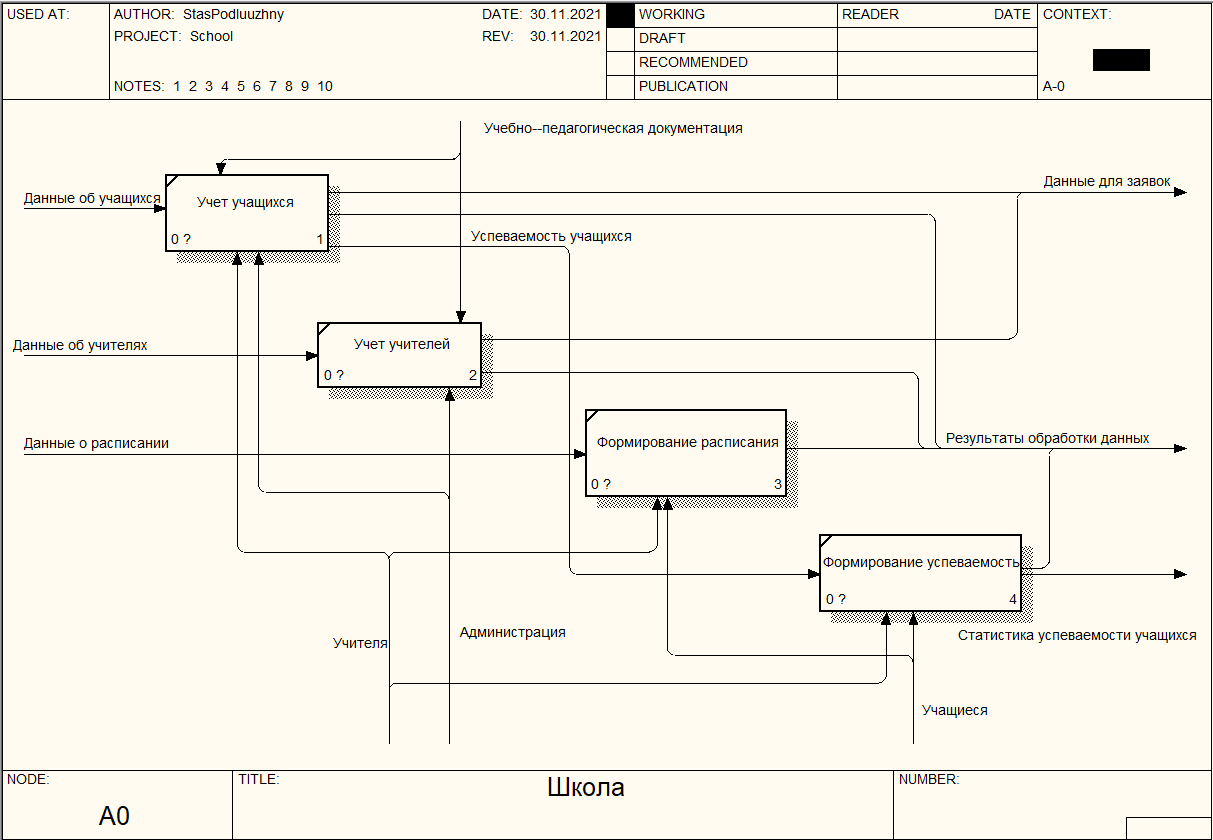


Рисунок 1.2 – Диаграмма декомпозиции

# 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для использования педагогическим составом (директором, завучами, преподавателями) и учащимися школьных общеобразовательных учреждений. Исходными данными могут являться данные об учащихся, учетелей, расписании, а выходными данными – заявки, а также статистические данные.

2.2 Требования к пользовательским интерфейсам

Пользовательский интерфейс должен предполагать стандартную системную цветовую палитру, разрабатываться под разрешение экрана 800x600 и более. Окна должны обладать системным меню с кнопкой закрытия.

Требования к окнам:

Окно авторизации:

1. должно иметь два поля с вводом информации: логин и пароль;
2. CheckBox c функцией отображения пароля;
3. кнопка: «Войти».

Окно «Main»:

1. должно иметь DataGridView с информацией из базы данных «Школа» таблиц Teachers, Pupils, Events.
2. кнопка: «Удалить»;
3. должно иметь поле для поиска в DataGridView;
4. каждое действие с изменением данных в DataGridView должно сопровождаться MessageBox с подтверждением действия пользователя.
5. menuStrip c выбором: «Просмотр» (учителей, учащихся, событий), «Добавить» (учителя, учащегося, событие), «Изменить» (учителя, учащегося, событие), «Успеваемость», «Расписание».

Окно «Добавить Учащегося»:

1. должно иметь 10 полей для ввода данных об учащемся (ФИО, дата рождения, возраст, класс, домашний адрес, льготная категория, контактные данные, дата приема, дата выбытия, ФИО родителя, телефон родителя), каждое поле должно проходить на корректность введенных данных, в случае некорректности вывести MessageBox с информацией, какое поле было не корректно и пример правильного ввода;
2. ComboBox с классами;
3. кнопка: «ОК»

Окно «Обновить Учащегося»:

Должно повторять весь функционал окна «Добавить Учащегося» и заполнять все поля информацией об учащемся, которого вы изменяете.

Окно «Добавить Учителя»:

1. должно иметь 15 полей для ввода данных об учащемся (ФИО, должность, категория, дата рождения, возраст, педагогический стаж, год аттестации, год курсов, телефон, почта, нагрузка, отпуск (дата с по), больничный (дата с по)), каждое поле должно проходить на корректность введенных данных, в случае некорректности вывести MessageBox с информацией какое поле было не корректно и пример правильного ввода;
2. кнопка: «ОК».

Окно «Обновить Учителя»:

Должно повторять весь функционал окна «Добавить Учителя» и заполнять все поля информацией об учителе, которого вы изменяете.

Окно «Добавить Событие»:

1. должно иметь 4 поля для ввода данных о событии (важность, срок, место, дополнительная информация), каждое поле должно проходить на корректность введенных данных;
2. два ComboBox с указанием типа и ответственного;
3. кнопка: «ОК».

Окно «Обновить Событие»:

Должно повторять весь функционал окна «Добавить Событие» и заполнять все поля информацией о событии, которое вы изменяете.

Окно «Успеваемость»:

1. должно иметь DataGridView с информацией из базы данных «Школа» таблицы Performance.
2. должно иметь 3 ComboBox с информацией из базы данных «Школа» для поиска из таблицы Performance по учащимся, предмету, классу;
3. каждое действие с изменением данных в DataGridView должно сопровождаться MessageBox с подтверждением действия пользователя.
4. menuStrip c выбором: «Успеваемость» (просмотр, добавить, изменить).
5. кнопка «Диаграмма» c выводом диаграммы успеваемости по классу или предмету.

Окно «Добавить Успеваемость»:

1. должно иметь поле для ввода данных о успеваемости (оценка), каждое поле должно проходить на корректность введенных данных;
2. четыре ComboBox с указанием учащегося, урока, периода, класса;
3. кнопка: «ОК».

Окно «Диаграмма Успеваемости»:

1. должно иметь элемент Chart с выводом количества учащихся по классу и предмету.

Окно «Обновить Успеваемость»:

Должно повторять весь функционал окна «Добавить Успеваемость» и заполнять все поля информацией о событии, которое вы изменяете.

Окно «Расписание»:

1. должно иметь DataGridView с информацией из базы данных «Школа» таблицы Schedule.
2. должно иметь 2 ComboBox с информацией из базы данных «Школа» для поиска из таблицы Schedule по классу, учителям;
3. каждое действие с изменением данных в DataGridView должно сопровождаться MessageBox с подтверждением действия пользователя.
4. menuStrip c выбором: «Расписание» (просмотр, добавить, изменить).

Окно «Добавить Расписание»:

1. должно иметь поле для ввода данных о расписании (урок), каждое поле должно проходить на корректность введенных данных;
2. четыре ComboBox с указанием дня, преподавателя, предметя, класса;
3. кнопка: «ОК».

Окно «Обновить Расписание»:

Должно повторять весь функционал окна «Добавить Расписание» и заполнять все поля информацией о событии, которое вы изменяете.

2.3 Требования к аппаратным и программным интерфейсам

Необходимо обеспечить программное взаимодействие системы c операционными системами Windows XP / Vista / 7 / 8 / 8.1 / 10.

Программа должна занимать не более 180 Мб оперативной памяти.

Модули программы должны занимать также не более 250 Мб памяти на жестком диске.

Для работы программы не требуется подключение к интернету.

2.4 Требования к пользователям продукта

Учитель:

* Владение компьютером на уровне пользователя ОС Windows.
* Понимание принципов работы приложения.
* Умение читать круговые диаграммы.

Учащийся:

* Владение компьютером на уровне пользователя ОС Windows.
* Понимание принципов работы приложения.

Администратор:

* Владение компьютером на уровне опытного пользователя.
* Понимание принципов работы приложения.
* Умение ввести работу по поддержке и наполнению базы данных приложения.
* Умение читать круговые диаграммы.

2.5 Требования к адаптации на месте

Для работы приложения необходимо иметь установленную систему управления базами данных (СУБД) MS SQL Server.

Предпочтительно иметь установленную программу MS Word.

Необходимы программы-установщики для осуществления развертывания модулей на соответствующих целевых компьютерах.

Необходимо предоставить пользователям справочную информацию.

2.6 Функции продукта

Приложение должно уметь работать с таблицами базы данных с информацией об учащихся, учителях, успеваемости, расписании, событиях.

В программе должна присутствовать возможность добавления, изменения, удаления данных об учащихся, учителях, успеваемости, расписании, событиях.

Должна быть реализована функция поиска, вывода расписания по учителям и классам, вывода успеваемости по классам, предмету и учащимся, переводу учащихся в другой класс, вывода диаграммы успеваемости по классу и предмету, формирование таблиц для заявок на оздоровление и курсы.

2.7 Ограничения

* Продукт будет поддерживать только русский язык пользовательского интерфейса.
* Приложение не поддерживает полноэкранный режим работы.

# 3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

Для разработки приложения была выбрана технологи Microsoft .NET Framework 4.7, обладающий огромным функционалом для графического представления данных и построения окон.

Для создания GUI (графического интерфейса пользователя) был выбран интерфейс программирования приложений Windows Forms, являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде.

В качестве системы управления базой данных (СУБД) была использована Microsoft SQL Server. Данная СУБД хорошо подходит для приложений, работающих под операционно системой Windows.

В качестве инструментальной среды разработки (IDE) была выбрана Microsoft Visual Studio 2019 Community и язык C#, который является языком разработки приложений для платформы Microsoft .NET Framework.

Для написания кода была выбрана парадигма «ООП» (объектно- ориентированное программирование), так как C# − это объектно- ориентированный язык и каждое окно формы является экземпляром класса.

# 4 РАЗРАБОТКА БАЗЫ ДАННЫХ И ПРИЛОЖЕНИЯ

4.1 Разработка базы данных приложения

*4.1.1 Концептуальное проектирование базы данных.*

Концептуальное проектирование – построение семантической модели предметной области (информационной модели самого высокого уровня абстракции). Основными компонентами концептуальной модели являются объекты и взаимосвязи [8].

В ходе изучения предметной области выделены следующие сущности, представляющие информационное значение для задачи:

1. *Учащийся* – информация об учащихся;
2. *Класс –* информация о классах;
3. *Успеваемость –* информация об успеваемости учащихся;
4. *Период –* справочник периодов для успеваемости;
5. *Предмет* – справочник предметов;
6. *Учитель* – информация об учителях;
7. *Расписание* – информация о расписании;
8. *День* – справочник учебных дней;
9. *Событие* – информация о событие;
10. *Вид* – вид события.

Концептуальное представление базы данных приведено на рисунке 4.1.

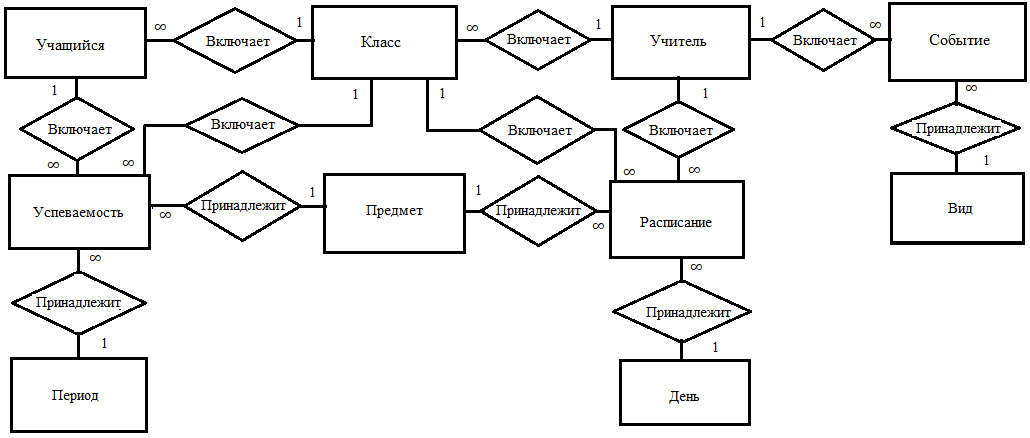


Рисунок 4.1 – Концептуальная модель базы данных

*4.1.2 Логическое проектирование базы данных.*

Логическое проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Основными компонентами логической модели являются сущности, атрибуты и связи.

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Эта модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Логическое представление базы данных приведено на рисунке 4.2.

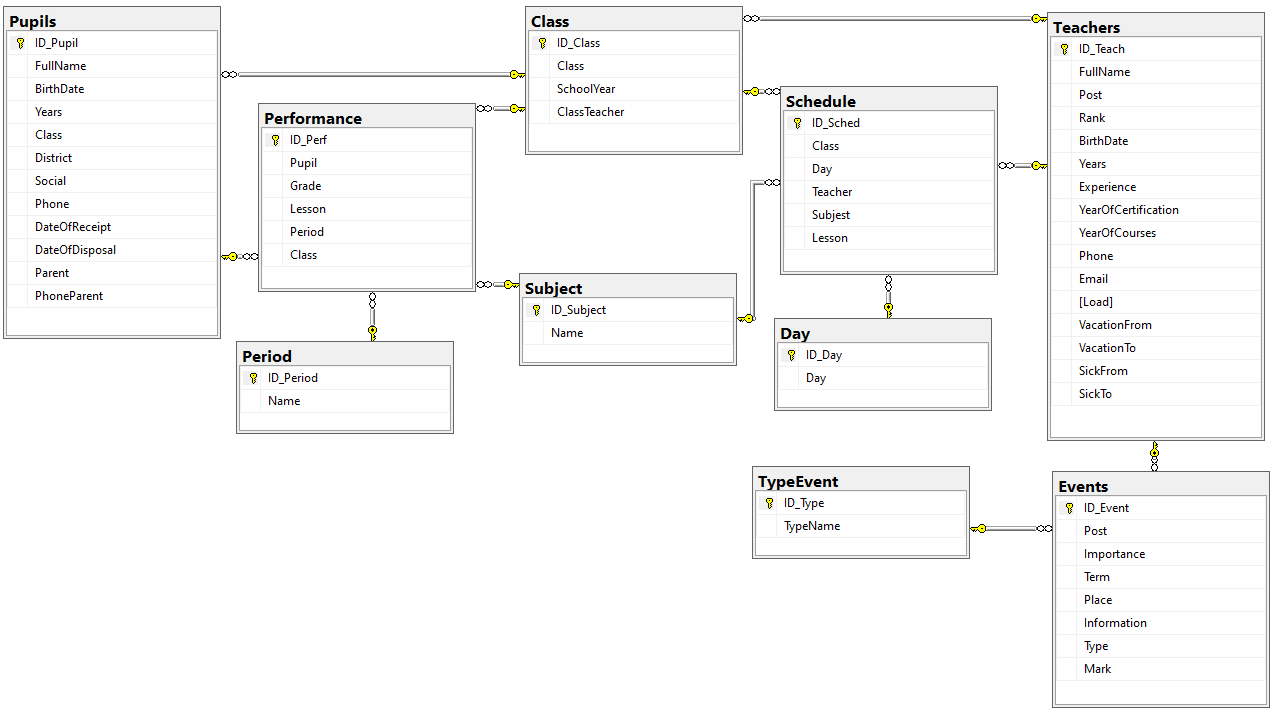


Рисунок 4.2 – Логическая модель базы данных

*4.1.3 Физическое проектирование базы данных*

Физическое проектирование – создание схемы базы данных для конкретной СУБД.

Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах - таблицах, колонках, индексах, процедурах и т. д [9].

Физическое представление таблиц базы данных приведено на рисунках 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12.

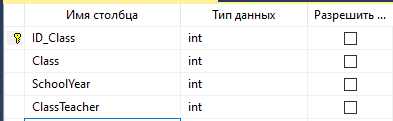


Рисунок 4.3 – Физическая модель таблицы «Class»

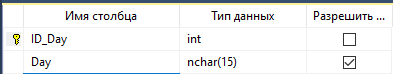


Рисунок 4.4 – Физическая модель таблицы «Day»

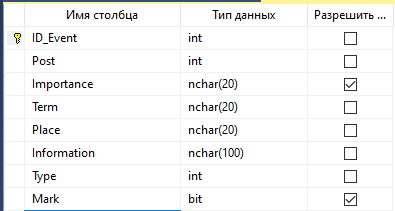


Рисунок 4.5 – Физическая модель таблицы «Events»

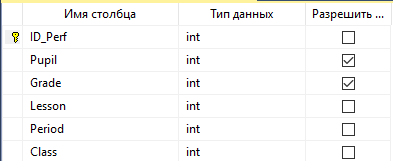


Рисунок 4.6 – Физическая модель таблицы «Performance»

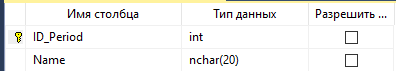


Рисунок 4.7 – Физическая модель таблицы «Period»

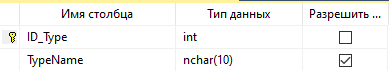


Рисунок 4.8 – Физическая модель таблицы «TypeEvent»

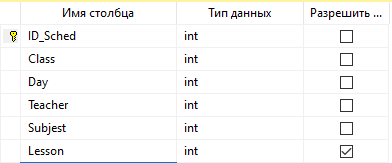


Рисунок 4.9 – Физическая модель таблицы «Schedule»

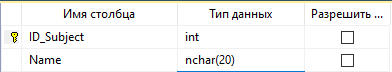


Рисунок 4.10 – Физическая модель таблицы «Subject»

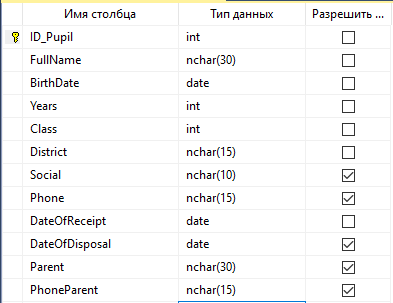


Рисунок 4.11 – Физическая модель таблицы «Pupils»

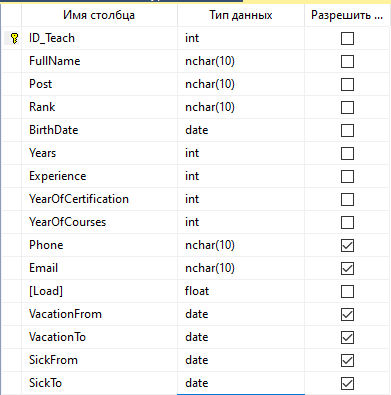


Рисунок 4.12 – Физическая модель таблицы «Teachers»

Физическое представление базы данных изображено на рисунке 4.13.

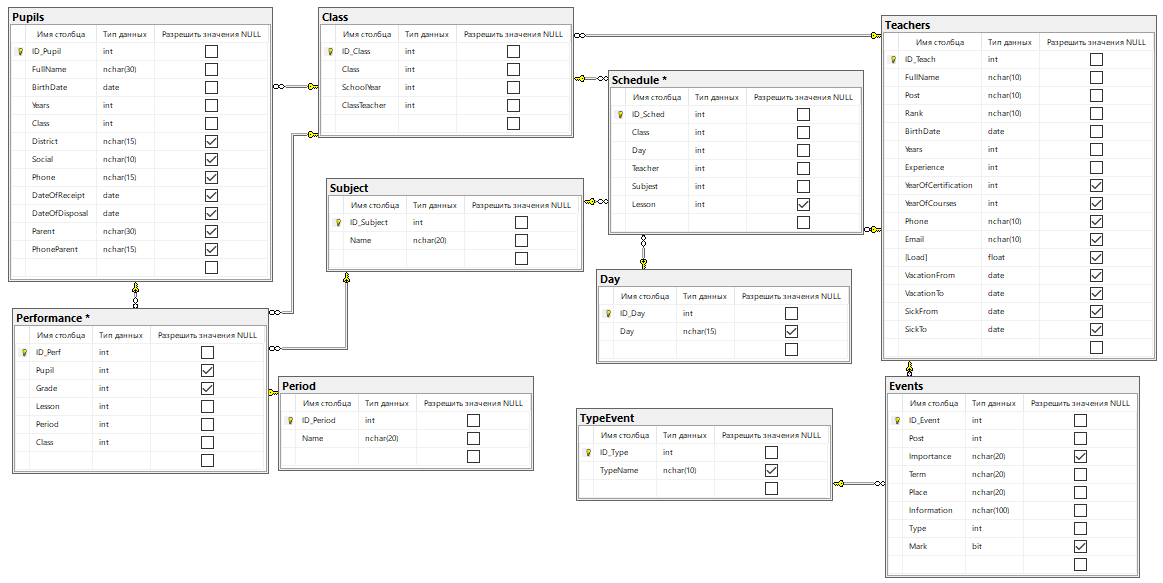


Рисунок 4.13 – Физическая модель базы данных

4.2 Проектирование структуры приложения

Логика приложения (классы и соответствующие файлы) приведена в таблице 4.1.

Таблица 4.1 – Классы и файлы

|  |  |
| --- | --- |
| Program.cs | Класс, являющийся основной точкой входа в программу. |
| ToDoc.cs | Класс, содержащий функции для работы с .docx документами. |
| AuthService.cs | Класс для реализации авторизации пользователя |
| Account.cs | Класс, содержащий данные об аккаунтах. |

Так как в интерфейсе Windows Forms все окна и графические элементы формы являются отдельными классами, в таблице 4.2 представлены файлы окон и их описание.

Таблица 4.2 – Классы, отвечающие за графический интерфейс

|  |  |
| --- | --- |
| Main.cs | Основное окно программы, являющееся контейнером для всех элементов управления. И содержащее основные функции. |
| Authorization.cs | Форма для авторизации пользователя. |
| Pupil.cs | Окно для ввода необходимых данных для добавления/изменения учащихся. |
| Teacher.cs | Окно для ввода необходимых данных для добавления/изменения учителей. |
| EventForm.cs | Окно для ввода необходимых данных для добавления/изменения событий. |
| Performance.cs | Окно для вывода данных об успеваемости учащихся. |
| Schedule.cs | Окно для вывода данных о расписании. |
| ScheduleAdd.cs | Окно для ввода необходимых данных для добавления/изменения расписания. |
| PerformanceAdd.cs | Окно для ввода необходимых данных для добавления/изменения успеваемости. |
| PerfDiagram.cs | Окно для вывода диаграммы по успеваемости. |

# 

4.3 Описание объектов и их взаимодействия

На рисунке 4.14 представлена общая структура проекта.

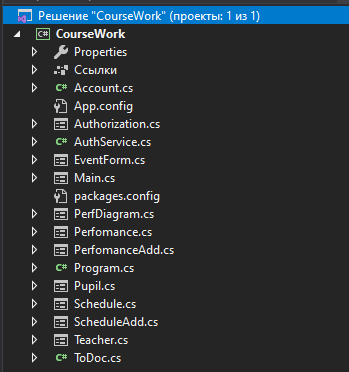


Рисунок 4.14 – Структура проекта

Ниже в таблицах будет рассмотрен каждый класс более детально.

Таблица 4.3 – Методы, используемые в классе Main.

|  |  |
| --- | --- |
| Название метода | Реализация |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Удалить». Вызов метода удаления элемента. |
| private void учителяToolStripView\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Просмотр» – «Учителя». Вызов метода заполнения DataGridView элементами из таблицы Teachers. |
| private void учащиесяToolStripView\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Просмотр» – «Учащиеся». Вызов метода заполнения DataGridView элементами из таблицы Pupils. |
| private void событияToolStripView\_Click(object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Просмотр» – «События». Вызов метода заполнения DataGridView элементами из таблицы Events. |
| private void успеваемостьToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Успеваемость». Вызов формы Performance. |
| private void учителяToolStripAdd\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Добавить» – «Учителя». Вызов формы Teacher. |
| private void учителяToolStripChange\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Изменить» – «Учителя». Вызов формы Teacher. |
| public void FillTeacherGrid () | Метод заполнения DataGridView элементами из таблицы Teachers |
| public void FillPupilGrid () | Метод заполнения DataGridView элементами из таблицы Pupils |
| public void FillEventGrid () | Метод заполнения DataGridView элементами из таблицы Events |
| private void radioButton1\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения свойства Checked. Вызов метода поиска. |
| private void radioButton2\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения свойства Checked. Вызов метода отмены поиска. |
| private void учащегосяToolStripAdd\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Добавить» – «Учащегося». Вызов формы Pupil. |
| private void учащегосяToolStripChange\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Изменить» – «Учащегося». Вызов формы Pupil. |
| private void событиеToolStripAdd\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Добавить» – «Событие». Вызов формы EventForm. |
| private void событиеToolStripChange\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Изменить» – «Событие». Вызов формы EventForm. |
| private void button2\_Click\_1 (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Выполнить событие». Вызов метода выполнения события. |
| private void рассписаниеToolStrip\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Расписание». Вызов формы Schedule. |
| private void заявкаНаКурсыToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент контекстного меню «Создать таблицу». Создание таблицы в формате .docx из выбранных элементов. |
| private void заявкаНаОздоровлениеДетейЛьготныхКатегорийToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент контекстного меню «Создать таблицу». Создание таблицы в формате .docx из выбранных элементов. |
| private void button3\_Click(object sender, EventArgs e) | Обработчик события нажатие на кнопку «Перeвод в другой класс». Вызов метода перевода учащихся в другой класс. |

Таблица 4.4 – Методы, используемые в классе Authorization.

|  |  |
| --- | --- |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатия на кнопку «Войти». Проверка пароля и логина, открытие формы Main в случае удачи. |
| private void checkBox1\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения чек бокса. Функция воспроизведения пароля. |

Таблица 4.5 – Методы, используемые в классе AuthService.

|  |  |
| --- | --- |
| public IEnumerable<Account> GetListAccounts() | Метод получения списка аккаунтов. |
| public IEnumerable<Account> GetAccounts() | Метод для чтения их файла json и загрузка в список аккаунтов. |
| public bool Authorize(string login, string password) | Метод проверки пользователя на существование и выдача роли. |
| public Account AuthUser(string login, string password) | Метод поиска пользователя. |

Таблица 4.6 – Методы, используемые в классе Account.

|  |  |
| --- | --- |
| Название метода | Реализация |
| public int ID { get; set; } | Гетер и сетер для уникального id. |
| public string Login { get; set; } | Гетер и сетер для логина пользователя. |
| public string Password { get; set; } | Гетер и сетер для пароля пользователя. |
| public string Roles { get; set; } | Гетер и сетер для роли пользователя. |

Таблица 4.7 – Методы, используемые в классе EventForm.

|  |  |
| --- | --- |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Ок». Вызов метода добавления/изменения события в таблицы Events. |
| private void Event\_Load(object sender, EventArgs e) | Обработчик события загрузки формы. Вызов метода заполнения полей изменяемого события. |

Таблица 4.8 – Методы, используемые в классе Pupil.

|  |  |
| --- | --- |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Ок». Вызов метода добавления/изменения учащегося в таблице Pupils. |
| private void Pupil\_Load (object sender, EventArgs e) | Обработчик события загрузки формы. Вызов метода заполнения полей изменяемого учащегося. |

Таблица 4.9 – Методы, используемые в классе Teacher.

|  |  |
| --- | --- |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Ок». Вызов метода добавления/изменения учителя в таблицы Teachers. |
| private void Teacher \_Load  (object sender, EventArgs e) | Обработчик события загрузки формы. Вызов метода заполнения полей изменяемого учителя. |

Таблица 4.10 – Методы, используемые в классе PerformanceAdd.

|  |  |
| --- | --- |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Ок». Вызов метода добавления/изменения успеваемости в таблицы Performance. |
| private void PerformanceAdd \_Load  (object sender, EventArgs e) | Обработчик события загрузки формы. Вызов метода заполнения полей изменяемой успеваемости. |

Таблица 4.11 – Методы, используемые в классе ScheduleAdd.

|  |  |
| --- | --- |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Ок». Вызов метода добавления/изменения расписания в таблицы Schedule. |
| private void ScheduleAdd \_Load  (object sender, EventArgs e) | Обработчик события загрузки формы. Вызов метода заполнения полей изменяемого расписания. |

Таблица 4.12 – Методы, используемые в классе Schedule.

|  |  |
| --- | --- |
| public void FillScheduleGrid() | Вызов метода заполнения DataGridView элементами из таблицы Schedule. |
| private void radioButton1\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения свойства Checked. Вызов метода поиска расписания по классам. |
| private void radioButton2\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения свойства Checked. Вызов метода поиска расписания по учителям. |
| private void Schedule\_Load (object sender, EventArgs e) | Обработчик события загрузки формы. Вызов метода заполнения ComboBox. |
| private void button1\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Найти». Вызов метода поиска. |
| private void добавитьToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Расписание» – «Добавить». Вызов формы ScheduleAdd. |
| private void показатьToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Расписание» – «Просмотр». Вызов метода заполнения DataGridView элементами из таблицы Schedule. |
| private void изменитьToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Расписание» – «Изменить». Вызов формы ScheduleAdd. |
| private void button2\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Удалить». Вызов метода удаления элемента. |

Таблица 4.13 – Методы, используемые в классе Performance.

|  |  |
| --- | --- |
| public void FillPerformanceGrid() | Вызов метода заполнения DataGridView элементами из таблицы Performance. |
| private void radioButton1\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения свойства Checked. Вызов метода поиска успеваемости по учащимся. |
| private void radioButton2\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения свойства Checked. Вызов метода поиска успеваемости по предметам. |
| private void radioButton3\_CheckedChanged (object sender, EventArgs e) | Обработчик события изменения свойства Checked. Вызов метода поиска успеваемости по классам. |
| private void добавитьToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Успеваемость» – «Добавить». Вызов формы PerformanceAdd. |
| private void показатьToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Успеваемость» – «Просмотр». Вызов метода заполнения DataGridView элементами из таблицы Performance. |
| private void изменитьToolStripMenuItem\_Click (object sender, EventArgs e) | Обработчик события нажатия на элемент меню панели инструментов  «Успеваемость» – «Изменить». Вызов формы PerformanceAdd. |
| private void button2\_Click (object sender, EventArgs e) | Обработчик события нажатие на кнопку «Удалить». Вызов метода удаления элемента. |
| private void comboBoxPupil\_SelectedIndexChanged (object sender, EventArgs e) | Обработчик события изменения индекса ComboBoxPupil. Вызов метода поиска успеваемости по учащимся. |
| private void comboBoxClass\_SelectedIndexChanged (object sender, EventArgs e) | Обработчик события изменения индекса ComboBoxClass. Вызов метода поиска успеваемости по классам. |
| private void comboBoxLesson\_SelectedIndexChanged (object sender, EventArgs e) | Обработчик события изменения индекса ComboBoxLesson. Вызов метода поиска успеваемости по предметам. |
| private void button1\_Click(object sender, EventArgs e) | Обработчик события нажатие на кнопку «Диаграмма». Вызов формы PerfDiagram, заполнение Chart успеваемости по предмету. |
| private void button2\_Click(object sender, EventArgs e) | Обработчик события нажатие на кнопку «Диаграмма». Вызов формы PerfDiagram, заполнение Chart успеваемости по классу. |

4.4 Описание SQL-запросов к базе данных системы

SQL запросы формы Main (полный листинг файла в приложении Б1):

Заполнение DataGridView информацией из таблицы Pupils:

string sql2 = "SELECT ID\_Pupil, FullName AS ФИО, BirthDate AS [Дата рождения], Years AS Возраст, Class.Class AS Класс" +

", District AS [Район], Social AS [Льготная категория], Phone AS Телефон," +

"DateOfReceipt AS [Дата Приема], DateOfDisposal AS [Дата Выбытия], Parent AS [ФИО родителя], " +

"PhoneParent AS [Телефон родителя] FROM Class INNER JOIN Pupils on Class.ID\_Class = Pupils.Class";

Заполнение DataGridView информацией из таблицы Teachers:

string sql3 = "SELECT ID\_Event, Teachers.FullName AS Ответственный, Importance AS Важность, " +

"Term AS Срок, Place AS Место, Information AS Информация," +

"TypeEvent.TypeName AS Тип, Mark AS Отметка FROM Events INNER JOIN " +

"TypeEvent ON Events.Type = TypeEvent.ID\_Type INNER JOIN Teachers ON Events.Post = Teachers.ID\_Teach";

Заполнение DataGridView информацией из таблицы Events:

string sql1 = "SELECT ID\_Teach, FullName AS ФИО, Post AS Должность, Rank AS Категория, BirthDate AS [Дата рождения], Years AS Возраст, Experience AS [Пед. стаж]" +

", YearOfCertification AS [Год Аттестации], YearOfCourses AS [Год Курсов], Phone AS Телефон, Email AS Почта, Load AS Нагрузка, " +

"VacationFrom, VacationTo, SickFrom, SickTo FROM Teachers";

Удаление информацией из таблицы Pupils:

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

command.CommandText = "DELETE FROM Pupils WHERE ID\_Pupil=@id";

command.Parameters.AddWithValue("@id", id);

Удаление информацией из таблицы Teachers:

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

command.CommandText = "DELETE FROM Teachers WHERE ID\_Teach=@id";

command.Parameters.AddWithValue("@id", id);

Удаление информацией из таблицы Events:

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

command.CommandText = "DELETE FROM Events WHERE ID\_Event=@id";

command.Parameters.AddWithValue("@id", id);

Поиск информации из таблицы Pupils:

string sql2 = "SELECT ID\_Pupil, FullName AS ФИО, BirthDate AS [Дата рождения], Years AS Возраст, Class.Class AS Класс" +

", District AS [Район], Social AS [Льготная категория], Phone AS Телефон," +

"DateOfReceipt AS [Дата Приема], DateOfDisposal AS [Дата Выбытия], Parent AS [ФИО родителя], " +

"PhoneParent AS [Телефон родителя] FROM Class INNER JOIN Pupils on Class.ID\_Class = Pupils.Class";

sql = string.Format("SELECT ID\_Pupil, FullName AS ФИО, BirthDate AS [Дата рождения], Years AS Возраст, Class.Class AS Класс" +

", District AS [Район], Social AS [Льготная категория], Phone AS Телефон," +

"DateOfReceipt AS [Дата Приема], DateOfDisposal AS [Дата Выбытия], Parent AS [ФИО родителя], PhoneParent AS [Телефон родителя] " +

"FROM Class INNER JOIN Pupils on Class.ID\_Class = Pupils.Class WHERE concat(FullName, BirthDate, Years, Class.Class, District, Social, Phone," +

"DateOfReceipt, DateOfDisposal, Parent, PhoneParent) LIKE '%{0}%'", FindtextBox.Text);

Поиск информации из таблицы Teachers:

string sql3 = "SELECT ID\_Event, Teachers.FullName AS Ответственный, Importance AS Важность, " +

"Term AS Срок, Place AS Место, Information AS Информация," +

"TypeEvent.TypeName AS Тип, Mark AS Отметка FROM Events INNER JOIN " +

"TypeEvent ON Events.Type = TypeEvent.ID\_Type INNER JOIN Teachers ON Events.Post = Teachers.ID\_Teach";

sql = string.Format("SELECT ID\_Teach, FullName AS ФИО, Post AS Должность, Rank AS Категория, BirthDate AS [Дата рождения], Years AS Возраст, Experience AS [Пед. стаж]" +

", YearOfCertification AS [Год Аттестации], YearOfCourses AS [Год Курсов], Phone AS Телефон, Email AS Почта, Load AS Нагрузка, " +

"VacationFrom, VacationTo, SickFrom, SickTo FROM Teachers WHERE concat(FullName, Post, Rank, BirthDate, Years, Experience" +

", YearOfCertification, YearOfCourses, Phone, Email, Load, " +

"VacationFrom, VacationTo, SickFrom, SickTo) LIKE '%{0}%'", FindtextBox.Text);

Поиск информации из таблицы Events:

string sql1 = "SELECT ID\_Teach, FullName AS ФИО, Post AS Должность, Rank AS Категория, BirthDate AS [Дата рождения], Years AS Возраст, Experience AS [Пед. стаж]" +

", YearOfCertification AS [Год Аттестации], YearOfCourses AS [Год Курсов], Phone AS Телефон, Email AS Почта, Load AS Нагрузка, " +

"VacationFrom, VacationTo, SickFrom, SickTo FROM Teachers";

sql = string.Format("SELECT ID\_Event, Teachers.FullName AS Ответственный, Importance AS Важность, " +

"Term AS Срок, Place AS Место, Information AS Информация," +

"TypeEvent.TypeName AS Тип, Mark AS Отметка FROM Events INNER JOIN " +

"TypeEvent ON Events.Type = TypeEvent.ID\_Type INNER JOIN Teachers ON Events.Post = Teachers.ID\_Teach WHERE concat(Teachers.FullName," +

" Importance, Term, Place, Information, TypeEvent.TypeName, Mark) LIKE '%{0}%'", FindtextBox.Text);

Выполнение событие из таблицы Events:

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

command.CommandText = "UPDATE Events SET Mark = 1 WHERE ID\_Event=@ID";

command.Parameters.AddWithValue("@id", id);

Перевод учащегося в другой класс:

int index = dataGridView1.SelectedRows[i].Index;

int id = int.Parse(dataGridView1.Rows[index].Cells[0].Value.ToString());

command.CommandText = "UPDATE Pupils SET Class = Class+1 WHERE ID\_Pupil=@ID";

command.Parameters.AddWithValue("@id", id);

SQL запросы формы Pupil (полный листинг файла в приложении Б2):

Изменение учащегося:

command.CommandText = "UPDATE Pupils SET FullName = @fn, BirthDate = @bd, Years = @y, Class = @cl" +

", District = @dist, Social = @s, Phone = @phone," +

"DateOfReceipt = @dor, DateOfDisposal = @dod, Parent = @p, PhoneParent = @pp WHERE ID\_Pupil=@ID";

command.Parameters.AddWithValue("@fn", textBoxFullName.Text);

command.Parameters.AddWithValue("@bd", textBoxBirthDate.Text);

command.Parameters.AddWithValue("@y", textBoxYears.Text);

command.Parameters.AddWithValue("@cl", comboBoxClass.SelectedValue);

command.Parameters.AddWithValue("@dist", textBoxDistrict.Text);

command.Parameters.AddWithValue("@s", textBoxSocial.Text);

command.Parameters.AddWithValue("@phone", textBoxPhone.Text);

command.Parameters.AddWithValue("@dor", textBoxDateOfReceipt.Text);

if (textBoxDateOfDisposal.Text == null)

{

command.Parameters.AddWithValue("@dod", DBNull.Value);

}

else

{

command.Parameters.AddWithValue("@dod", textBoxDateOfDisposal.Text);

}

command.Parameters.AddWithValue("@p", textBoxParent.Text);

command.Parameters.AddWithValue("@pp", textBoxPhoneParent.Text);

command.Parameters.AddWithValue("@id", IDPupil);

SQL запросы формы Teacher (полный листинг файла в приложении Б3):

Изменение учителя:

command.CommandText = "UPDATE Teachers SET FullName = @fn, Post = @p, Rank = @r, BirthDate = @bd, Years = @y, Experience = @exp" +

", YearOfCertification = @yocert, YearOfCourses = @yocours, Phone = @phone, Email = @email, Load = @l, " +

"VacationFrom = @vf, VacationTo = @vt, SickFrom = @sf, SickTo = @st WHERE ID\_Teach=@ID";

command.Parameters.AddWithValue("@fn", textBoxName.Text);

command.Parameters.AddWithValue("@P", textBoxPost.Text);

command.Parameters.AddWithValue("@r", textBoxRank.Text);

command.Parameters.AddWithValue("@bd", textBoxBirthDate.Text);

command.Parameters.AddWithValue("@y", textBoxYears.Text);

command.Parameters.AddWithValue("@exp", textBoxExperience.Text);

command.Parameters.AddWithValue("@yocert", textBoxYearOfCertification.Text);

command.Parameters.AddWithValue("@yocours", textBoxYearOfCourses.Text);

command.Parameters.AddWithValue("@phone", textBoxPhone.Text);

command.Parameters.AddWithValue("@email", textBoxEmail.Text);

command.Parameters.AddWithValue("@l", textBoxLoad.Text);

command.Parameters.AddWithValue("@vf", textBoxVacationFrom.Text);

command.Parameters.AddWithValue("@vt", textBoxVacationTo.Text);

command.Parameters.AddWithValue("@sf", textBoxSickFrom.Text);

command.Parameters.AddWithValue("@st", textBoxSickTo.Text);

command.Parameters.AddWithValue("@id", IDTeacher);

SQL запросы формы EventForm (полный листинг файла в приложении Б4):

Изменение события:

command.CommandText = "UPDATE Events SET Post = @p, Importance = @imp, Term = @t, Place = @place" +

", Information = @inf, Mark = @m, Type = @t WHERE ID\_Event=@ID";

command.Parameters.AddWithValue("@p", comboBoxTeacher.SelectedValue);

command.Parameters.AddWithValue("@imp", textBox2.Text);

command.Parameters.AddWithValue("@t", textBox3.Text);

command.Parameters.AddWithValue("@place", textBox4.Text);

command.Parameters.AddWithValue("@inf", textBox5.Text);

command.Parameters.AddWithValue("@m", 0);

command.Parameters.AddWithValue("@t", comboBoxType.SelectedValue);

command.Parameters.AddWithValue("@id", IDEvent);

SQL запросы формы Schedule (полный листинг файла в приложении Б5):

Заполнение DataGridView информацией из таблицы Schedule:

string sqlSched = "SELECT Schedule.ID\_Sched, Class.Class AS Класс, Day.Day AS День, Teachers.FullName AS Учитель, Subject.Name AS Предмет, " +

"Schedule.Lesson AS Урок FROM Day INNER JOIN " +

"Schedule ON dbo.Day.ID\_Day = Schedule.Day INNER JOIN " +

"Class ON Schedule.Class = Class.ID\_Class INNER JOIN " +

"Subject ON Schedule.Subjest = Subject.ID\_Subject INNER JOIN " +

"Teachers ON Schedule.Teacher = Teachers.ID\_Teach";

Вывод информации из таблицы Schedule по классу:

string sql1 = "SELECT Schedule.Lesson AS Урок, Subject.Name AS Предмет , Schedule.Class AS Класс " +

"FROM Day INNER JOIN " +

"Schedule ON Day.ID\_Day = Schedule.Day INNER JOIN " +

"Class ON Schedule.Class = Class.ID\_Class INNER JOIN " +

"Teachers ON Schedule.Teacher = Teachers.ID\_Teach INNER JOIN " +

"Subject ON Schedule.Subjest = Subject.ID\_Subject";

sqlFind = string.Format(sql + " WHERE Schedule.Class = {0} AND Schedule.Day = {1}", comboBoxClass.SelectedValue, comboBoxDay.SelectedValue);

Вывод информации из таблицы Schedule по учителю:

string sql = "SELECT Schedule.Lesson AS Урок, Subject.Name AS Предмет " +

"FROM Day INNER JOIN " +

"Schedule ON Day.ID\_Day = Schedule.Day INNER JOIN " +

"Class ON Schedule.Class = Class.ID\_Class INNER JOIN " +

"Teachers ON Schedule.Teacher = Teachers.ID\_Teach INNER JOIN " +

"Subject ON Schedule.Subjest = Subject.ID\_Subject";

sqlFind = string.Format(sql1 + " WHERE Schedule.Teacher = {0} AND Schedule.Day = {1}", comboBoxTeacher.SelectedValue, comboBoxDay.SelectedValue);

Удаление информацией из таблицы Schedule:

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

command.CommandText = "DELETE FROM Schedule WHERE ID\_Sched=@id";

command.Parameters.AddWithValue("@id", id);

SQL запросы формы ScheduleAdd (полный листинг файла в приложении Б6):

Изменение расписания:

command.CommandText = "UPDATE Schedule SET Class = @cl, Day = @d, Teacher = @t, Subjest = @s" +

", Lesson = @l WHERE ID\_Sched=@ID";

command.Parameters.AddWithValue("@cl", comboBoxClass.SelectedValue);

command.Parameters.AddWithValue("@d", comboBoxDay.SelectedValue);

command.Parameters.AddWithValue("@t", comboBoxTeacher.SelectedValue);

command.Parameters.AddWithValue("@s", comboBoxSubject.SelectedValue);

command.Parameters.AddWithValue("@l", textBoxLesson.Text);

command.Parameters.AddWithValue("@id", IDSchedule);

Помимо запросов для работы с БД «Школа» были созданы представления.

Представление MinGrade (нахождение минимальной оценки учащихся по классу):

SELECT TOP (100) PERCENT MIN(dbo.Performance.Grade) AS [Мин Оценка], dbo.Pupils.ID\_Pupil, dbo.Class.ID\_Class

FROM dbo.Performance INNER JOIN

dbo.Pupils ON dbo.Performance.Pupil = dbo.Pupils.ID\_Pupil INNER JOIN

dbo.Class ON dbo.Performance.Class = dbo.Class.ID\_Class AND dbo.Pupils.Class = dbo.Class.ID\_Class

GROUP BY dbo.Pupils.ID\_Pupil, dbo.Class.ID\_Class

Представление MinGrade2 (нахождение минимальной оценки учащихся по предмету):

SELECT TOP (100) PERCENT MIN(dbo.Performance.Grade) AS [Мин Оценка], dbo.Pupils.ID\_Pupil, dbo.Subject.ID\_Subject

FROM dbo.Pupils INNER JOIN

dbo.Performance ON dbo.Pupils.ID\_Pupil = dbo.Performance.Pupil INNER JOIN

dbo.Subject ON dbo.Performance.Lesson = dbo.Subject.ID\_Subject

GROUP BY dbo.Pupils.ID\_Pupil, dbo.Subject.ID\_Subject

Представление PerfomanceClass (нахождение успеваемости по классу):

SELECT dbo.Class.Class, CASE WHEN dbo.MinGrade.[Мин Оценка] = 2 THEN COUNT(dbo.MinGrade.ID\_Pupil) END AS Низкий, CASE WHEN dbo.MinGrade.[Мин Оценка] = 3 THEN COUNT(dbo.MinGrade.ID\_Pupil) END AS Средний,

CASE WHEN dbo.MinGrade.[Мин Оценка] = 4 THEN COUNT(dbo.MinGrade.ID\_Pupil) END AS Достаточный, CASE WHEN dbo.MinGrade.[Мин Оценка] = 5 THEN COUNT(dbo.MinGrade.ID\_Pupil) END AS Высокий

FROM dbo.MinGrade INNER JOIN

dbo.Class ON dbo.MinGrade.ID\_Class = dbo.Class.ID\_Class

GROUP BY dbo.MinGrade.[Мин Оценка], dbo.Class.Class

Представление PerfomanceSubject(нахождение успеваемости по предмету):

SELECT dbo.Subject.ID\_Subject, CASE WHEN dbo.MinGrade2.[Мин Оценка] = 2 THEN COUNT(dbo.MinGrade2.ID\_Pupil) END AS Низкий, CASE WHEN dbo.MinGrade2.[Мин Оценка] = 3 THEN COUNT(dbo.MinGrade2.ID\_Pupil)

END AS Средний, CASE WHEN dbo.MinGrade2.[Мин Оценка] = 4 THEN COUNT(dbo.MinGrade2.ID\_Pupil) END AS Достаточный, CASE WHEN dbo.MinGrade2.[Мин Оценка] = 5 THEN COUNT(dbo.MinGrade2.ID\_Pupil)

END AS Высокий

FROM dbo.MinGrade2 INNER JOIN

dbo.Subject ON dbo.MinGrade2.ID\_Subject = dbo.Subject.ID\_Subject

GROUP BY dbo.MinGrade2.[Мин Оценка], dbo.Subject.ID\_Subject

Представление PerfomancePupil (вывод успеваемости учащихся):

SELECT dbo.Pupils.FullName AS Учащийся, dbo.Performance.Grade AS Оценка, dbo.Subject.Name AS Предмет, Period\_1.Name AS Период, dbo.Performance.Pupil

FROM dbo.Class INNER JOIN

dbo.Performance ON dbo.Class.ID\_Class = dbo.Performance.Class INNER JOIN

dbo.Period AS Period\_1 ON dbo.Performance.Period = Period\_1.ID\_Period INNER JOIN

dbo.Subject ON dbo.Performance.Lesson = dbo.Subject.ID\_Subject INNER JOIN

dbo.Pupils ON dbo.Performance.Pupil = dbo.Pupils.ID\_Pupil

# 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

5.1 Аппаратные и программные средства создания и эксплуатации приложения

Аппаратные требования для работы интернет-приложения:

* процессор с тактовой частотой 1.0 ГГц.
* оперативная память 512 Мб и более;
* видеокарта с объёмом памяти 128 Мб и выше;
* монитор 1200х800 или с более высоким разрешением.

Программные требования к приложению.

Установленные на компьютере:

* операционная система Windows 7 или новее.
* СУБД MS SQL Server.
* Установленный пакет MS Office.

5.2 Руководство пользователя

Для работы приложения необходимо установить приложение с помощью инсталлятора, следуя инструкциям. После этого следует ознакомится с логической моделью базы данных. Затем создать базу данных на своем сервере, используя скрипт.

5.3 Описание контрольных примеров

После запуска программы пользователю откроется окно приложения.

Для начала пользователю надо пройти авторизацию (рисунок А.1), введя все данные и нажав кнопку «Войти», если все данные будут корректны и в файле, ему откроется главное меню (рисунок А.2) с оповещение о типе пользователя, в случае неудачи пользователь получит сообщение о неудачи.

Так как данное приложение поддерживает разных видов пользователей (администратор школы, учитель, учащийся) функционал у них различается.

В главном меню пользователь может выбрать 3 таблицы, нажав на ToolStripMenu «Просмотр» («Учителя», «Учащиеся» (рисунок А.3), «События» (рисунок А.4)) (для пользователя с ролью учащийся это функция не доступна). При выборе «Просмотр» – «Учащиеся» будет доступна кнопка «Перевод в другой класс» при нажатии на которую выбранные учащиеся будут переведены на класс вперед. При выборе «Просмотр» – «События» будет доступна кнопка «Выполнить событие» при нажатии на которую выбранное событие будет выполнено. Нажав на ToolStripMenu «Добавить» («Учителя», «Учащегося», «Событие») (для пользователя с рольюю учащийся это функция не доступна, для пользователя с ролью учитель функция «Добавить» – «Учителя» не доступна) откроется форма для добавления учителя, учащегося, событие в соответствии. Нажав на ToolStripMenu «Изменить» («Учителя», «Учащегося», «Событие») (для пользователя с ролью учащийся это функция не доступна, для пользователя с ролью учитель функция «Изменить» – «Учителя» не доступна) откроется форма для изменения учителя (рисунок А.5), учащегося (рисунок А.6), событие (рисунок А.7) в соответствии, с заполнением формы добавления, выбранной информацией для изменения. Нажав на ToolStripMenu «Успеваемость» откроется форма с успеваемостью (рисунок А.8). Нажав на ToolStripMenu «Расписание» откроется форма с расписанием (рисунок А.9). Нажав на кнопку «Удалить» (для пользователя с ролью учащийся это функция не доступна) выбранный элемент будет удален из БД «Школа». При заполнении поля для поиска и нажатии переключателя «Найти» в таблице будут показаны только те данные, которые удовлетворяют полю (рисунок А.10), при нажатии на переключатель «Отмена» результат поиска обнулится.

В форме успеваемости при нажатии на ToolStripMenu «Успеваемость» – «Добавить» (для пользователя с ролью учащийся это функция не доступна) откроется форма добавления успеваемости. При нажатии на ToolStripMenu «Успеваемость» – «Изменить» (для пользователя с ролью учащийся это функция не доступна) откроется форма добавления успеваемости (рисунок А.11) с заполнением формы добавления, выбранной информацией. При нажатии на переключатель «Включить» в секции по ученикам, при изменении значения в комбобоксе, будет выведена успеваемость по ученику. При нажатии на переключатель «Включить» в секции по предмету, при изменении значения в комбобоксе, будет выведена успеваемость по предмету, при нажатии на кнопку «Диаграмма» будет выведена форма с диаграммой успеваемости по предмету (рисунок А.12) (для пользователя с ролью учащийся это функция не доступна). При нажатии на переключатель «Включить» в секции по классу, при изменении значения в комбобоксе, будет выведена успеваемость по предмету, при нажатии на кнопку «Диаграмма» будет выведена форма с диаграммой успеваемости по классу (рисунок А.13) (для пользователя с ролью учащийся это функция не доступна).

В форме расписания при нажатии на ToolStripMenu «Расписание» – «Добавить» (для пользователя с ролью учащийся это функция не доступна) откроется форма добавления расписание. При нажатии на ToolStripMenu «Расписание» – «Изменить» (для пользователя с ролью учащийся это функция не доступна) откроется форма добавления расписания (рисунок А.14) с заполнением формы добавления, выбранной информацией. При нажатии на кнопку «Найти» будет выведено расписание по учителю (для пользователя с ролью учащийся это функция не доступна) или по классу (рисунок А.15). Нажав на кнопку «Удалить» (для пользователя с ролью учащийся это функция не доступна) выбранный элемент будет удален из базы данных «Школа».

# ЗАКЛЮЧЕНИЕ

В результате написания курсовой работы спроектирована база данных и приложение, автоматизирующее работу школы, позволяющая просматривать, изменять, искать, добавлять сведения об учителях, учащихся, расписании, успеваемости; формировать журналы событий; создавать таблицы в формате .doxc на основе данных из таблиц Teachers, Pupils; выполнять поставленные задачи в таблице Events, переводить учащихся в другой класс. Разработанное приложение является модифицируемым, то есть при необходимости может быть дополнено различными функциями и использоваться в любой школе.

Данное приложение значительно упрощает работу администрации школы и учителей, сокращая время работы, тем самым, повышая производительность труда.

В ходе курсовой работы в полном объеме решены все поставленные задачи.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Windows Forms Metanit // Сайт Metanit. URL: <https://metanit.com/sharp/windowsforms> (дата обращения 26.11.2021).
2. Работа с JSON // Сайт Metanit. URL: <https://metanit.com/sharp/tutorial/6.5.php> (дата обращения 26.11.2021).
3. Информационная система // Сайт Википедия. URL: [https://usnd.to/IYZR](https://usnd.to/IYZR%20) (дата обращения 19.11.2021).
4. Система управления // Сайт Википедия. URL: [https://usnd.to/IYZB](https://usnd.to/IYZB%20) (дата обращения 19.11.2021).
5. Школьная документация // Сайт Blanker. URL: <https://blanker.ru/doc/shkolnaya-dokumentacia> (дата обращения 23.11.2021)
6. Официальный сайт МОУ ДКГГ // Сайт gimnazija-dkgg. URL: <https://gimnazija-dkgg.mozello.ru/> (дата обращения 23.11.2021).
7. Нотация IDEF0 // Сайт Businessstudio. URL: [https://www.businessstudio.ru/wiki/docs/current/doku.php/ru/csdesign/bpmodeling/idef0](https://www.businessstudio.ru/wiki/docs/current/doku.php/ru/csdesign/bpmodeling/idef0%20) (дата обращения 18.11.2021).
8. Концептуальная модель предметной области // Сайт Studbooks. URL: [https://studbooks.net/2321704/informatika/kontseptualnaya\_model\_predmetnoy\_oblasti](https://studbooks.net/2321704/informatika/kontseptualnaya_model_predmetnoy_oblasti%20) (дата обращения 18.11.2021).
9. Физическая модель базы данных // Сайт Leally. URL: <https://leally.ru/good-to-know/logicheskaya-model-logicheskaya-model-dannyh-obekty-atributy-i/> (дата обращения 22.11.2021).
10. MS SQL Server Metanit // Сайт Metanit. URL: [https://metanit.com/sql/sqlserver/](https://metanit.com/sql/sqlserver/%20) (дата обращения 24.11.2021).

# 

# ПРИЛОЖЕНИЕ А Экранные формы

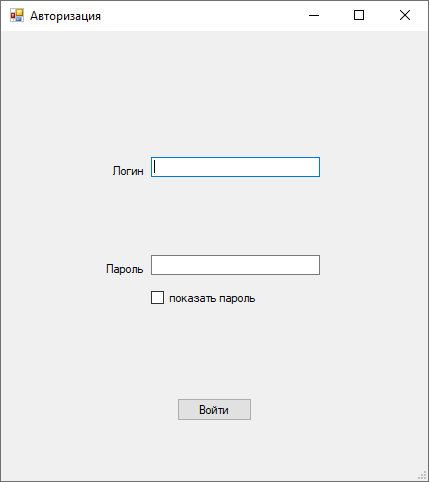


Рисунок A.1 – Окон авторизации

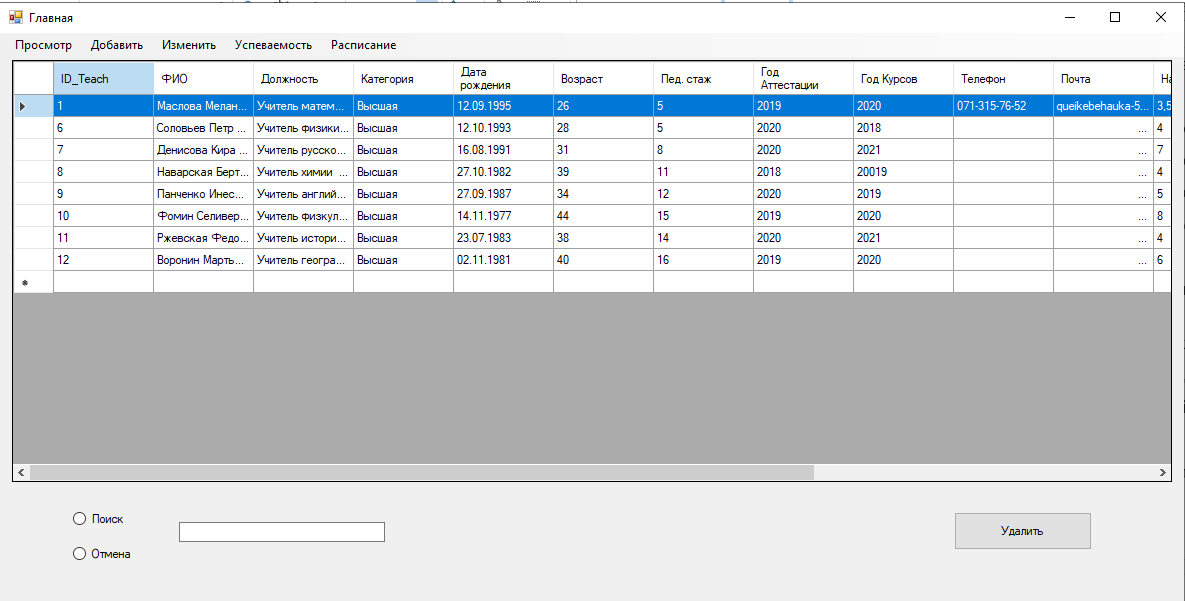


Рисунок A.2 – Главное окно приложения

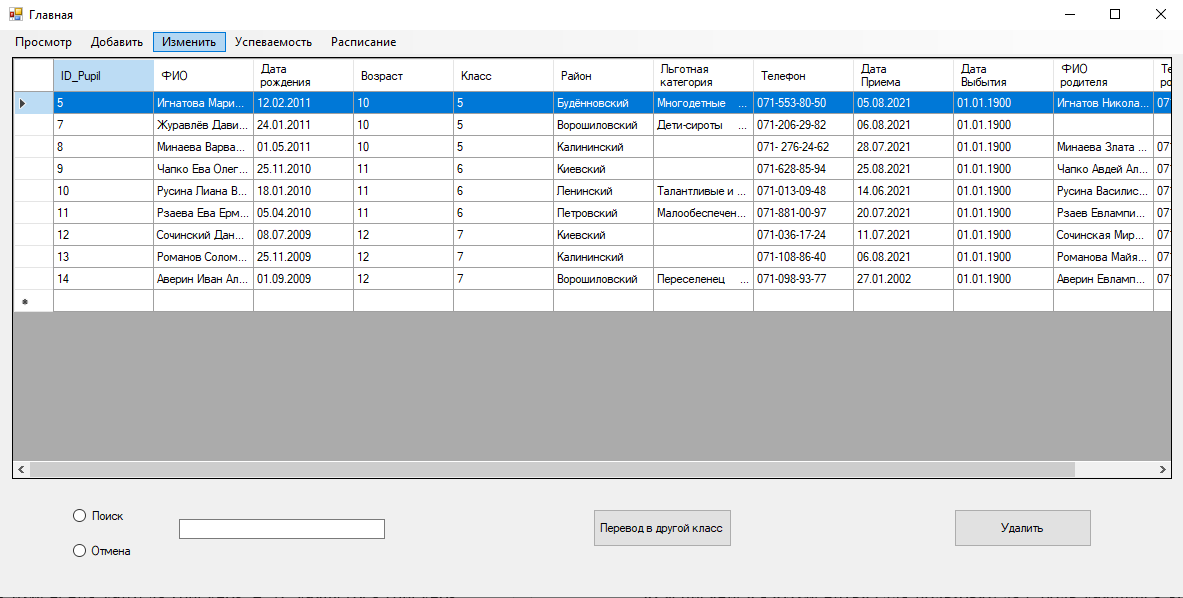


Рисунок A.3 – Таблица «Учащиеся»

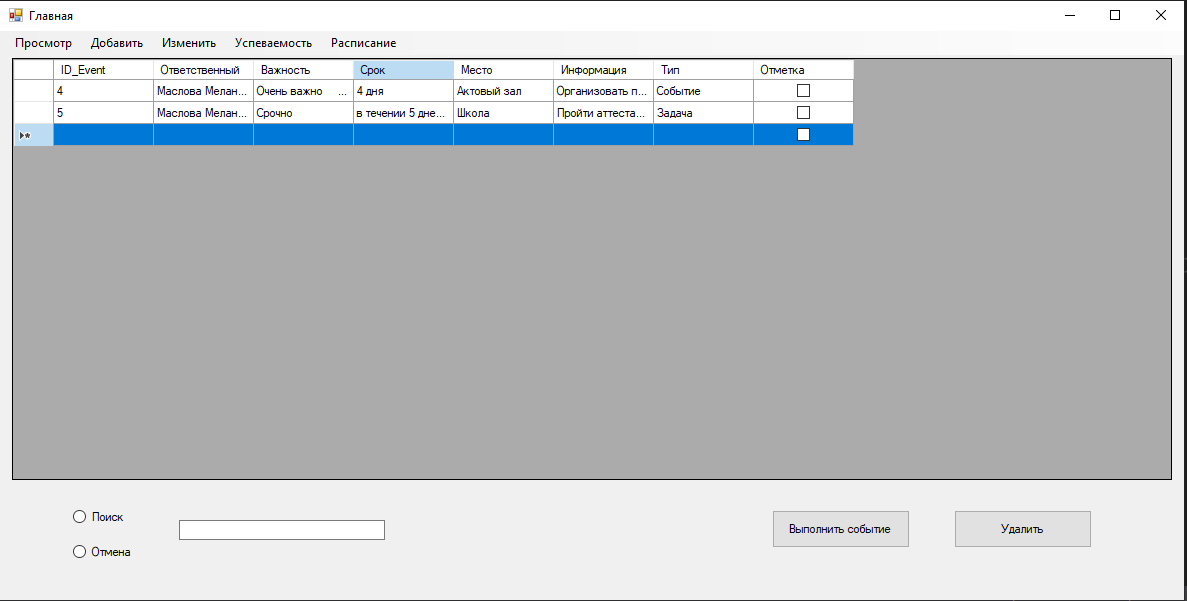


Рисунок A.4 – Таблица «События»

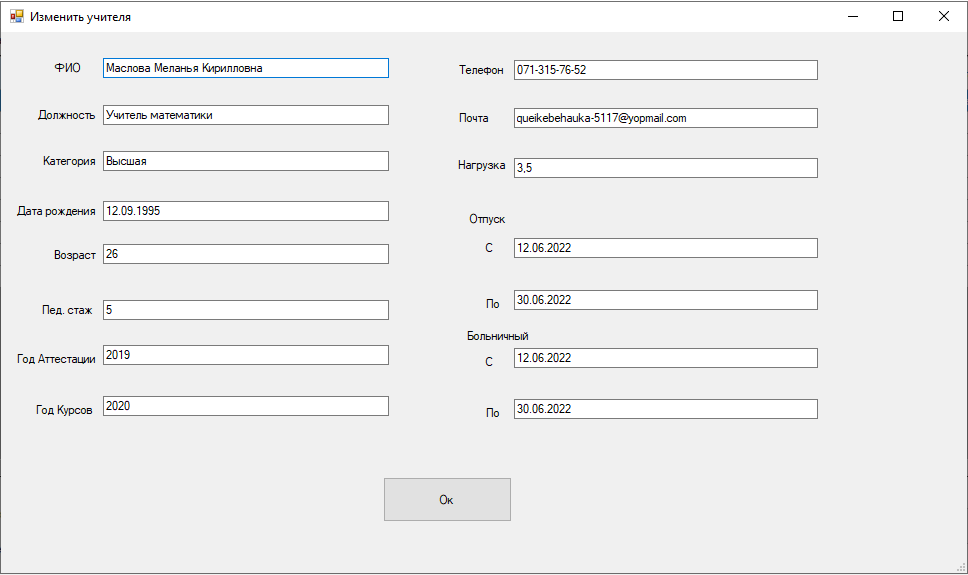


Рисунок A.5 – Окно изменения учителя

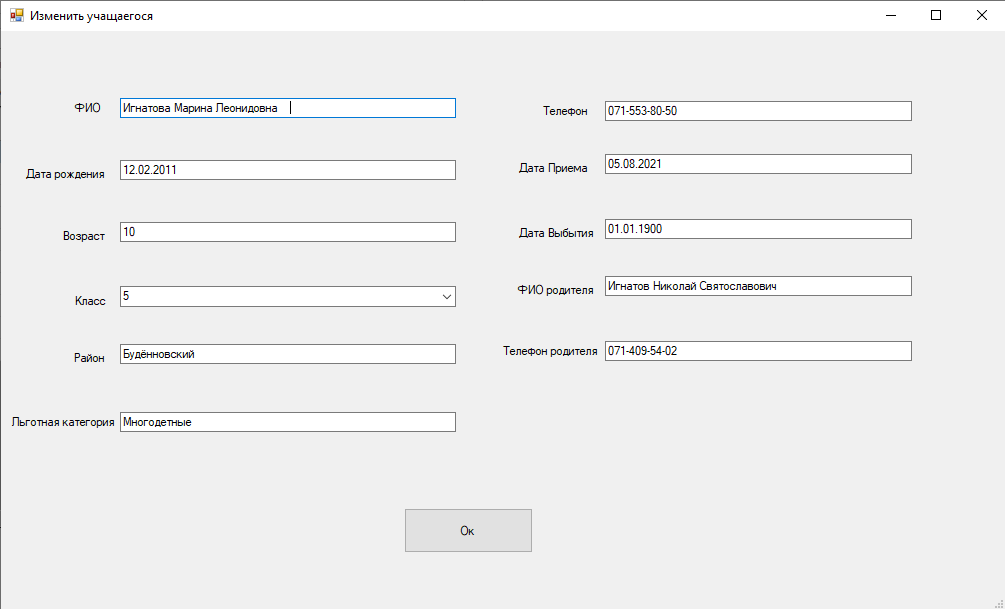


Рисунок A.6 – Окно изменения учащегося

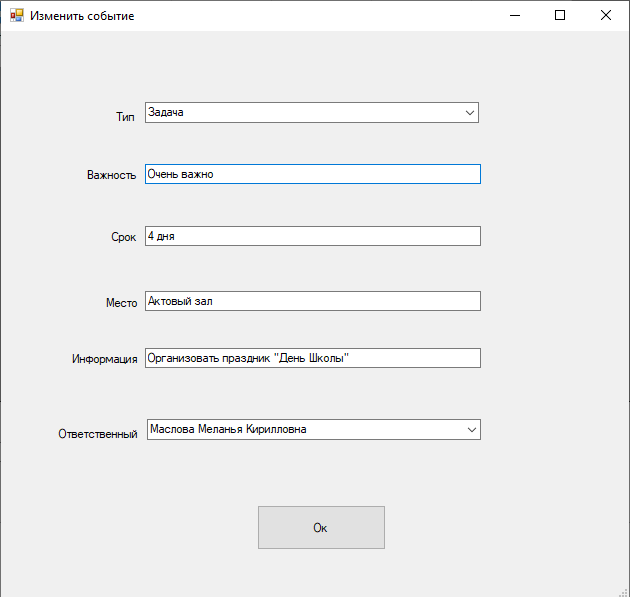


Рисунок A.7 – Окно изменения события

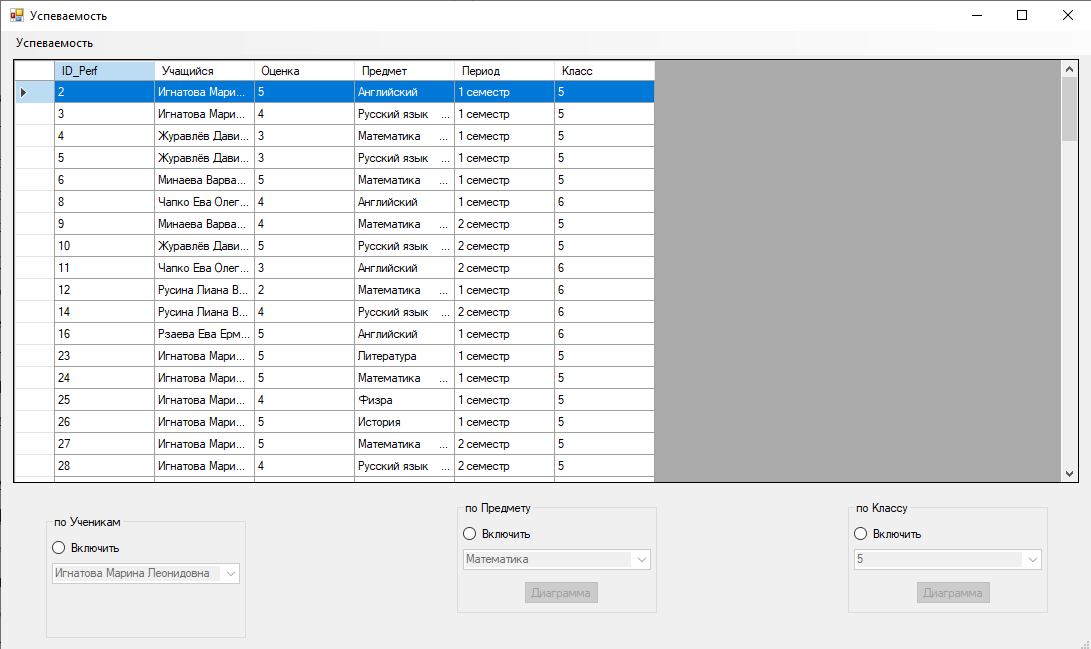


Рисунок A.8 – Окно успеваемости

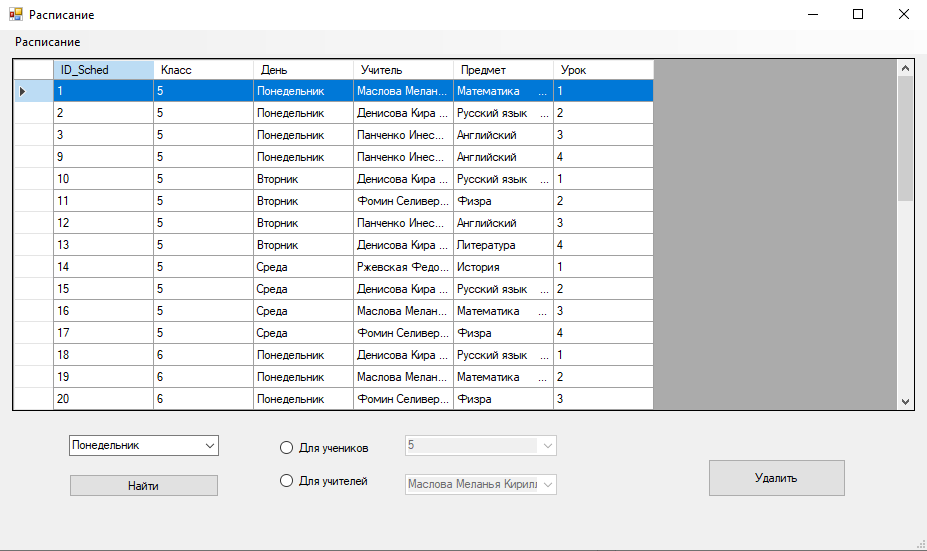


Рисунок A.9 – Окно расписания

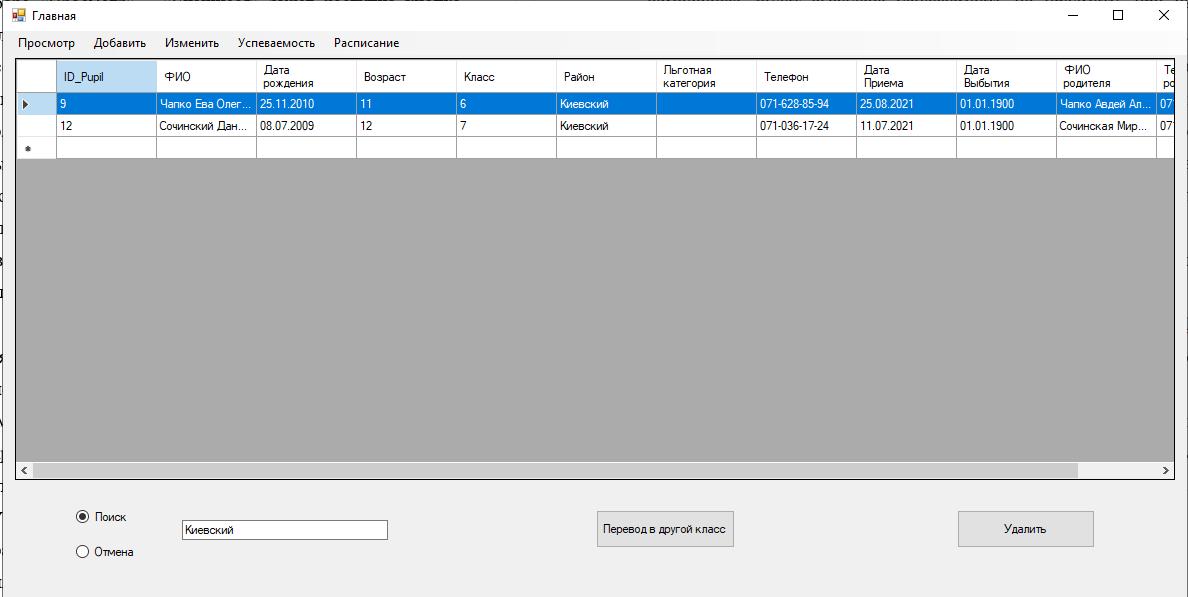


Рисунок A.10 – Результат поиска в главном окне

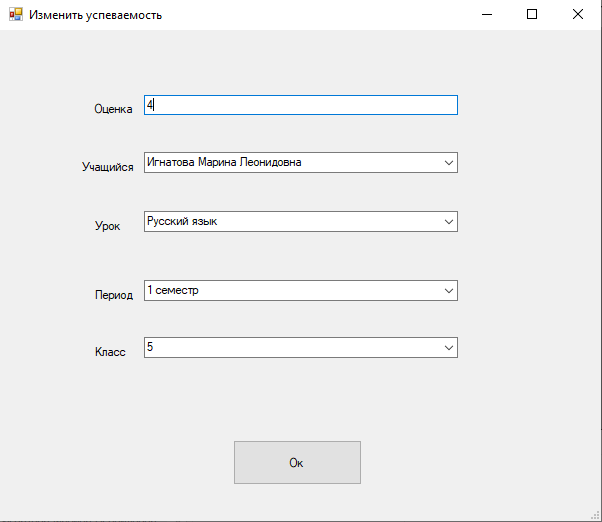


Рисунок A.11 – Окно изменения успеваемости

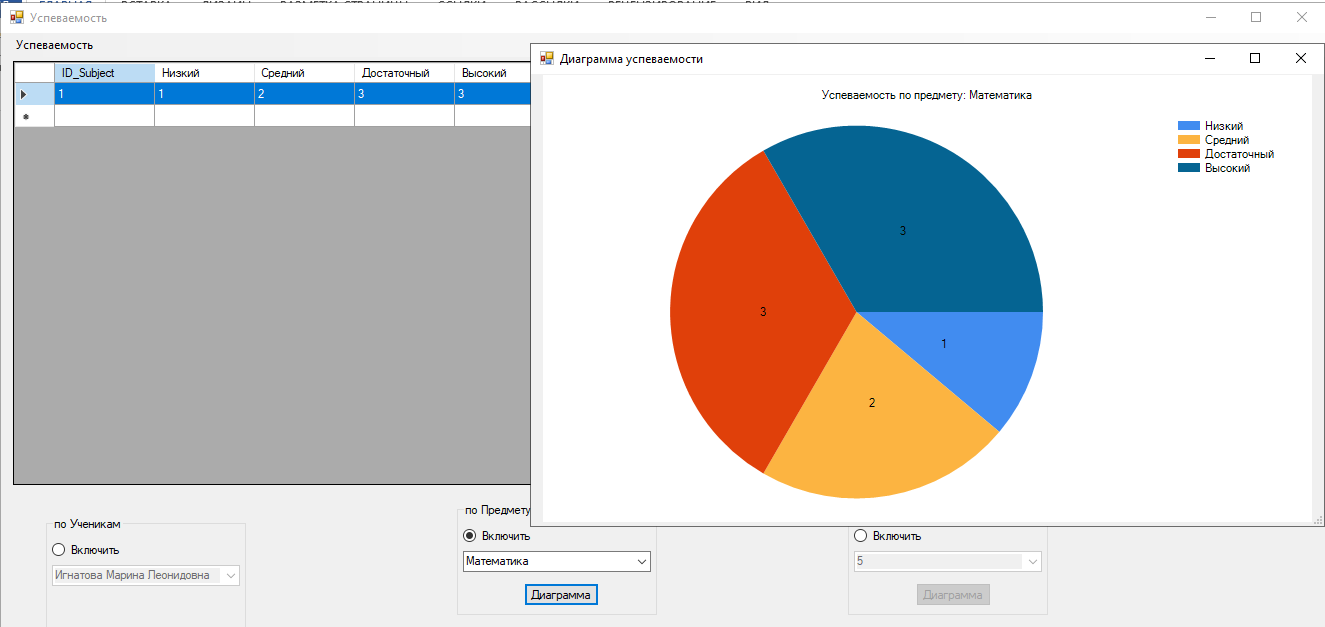


Рисунок A.12 – Диаграмма успеваемости по предмету

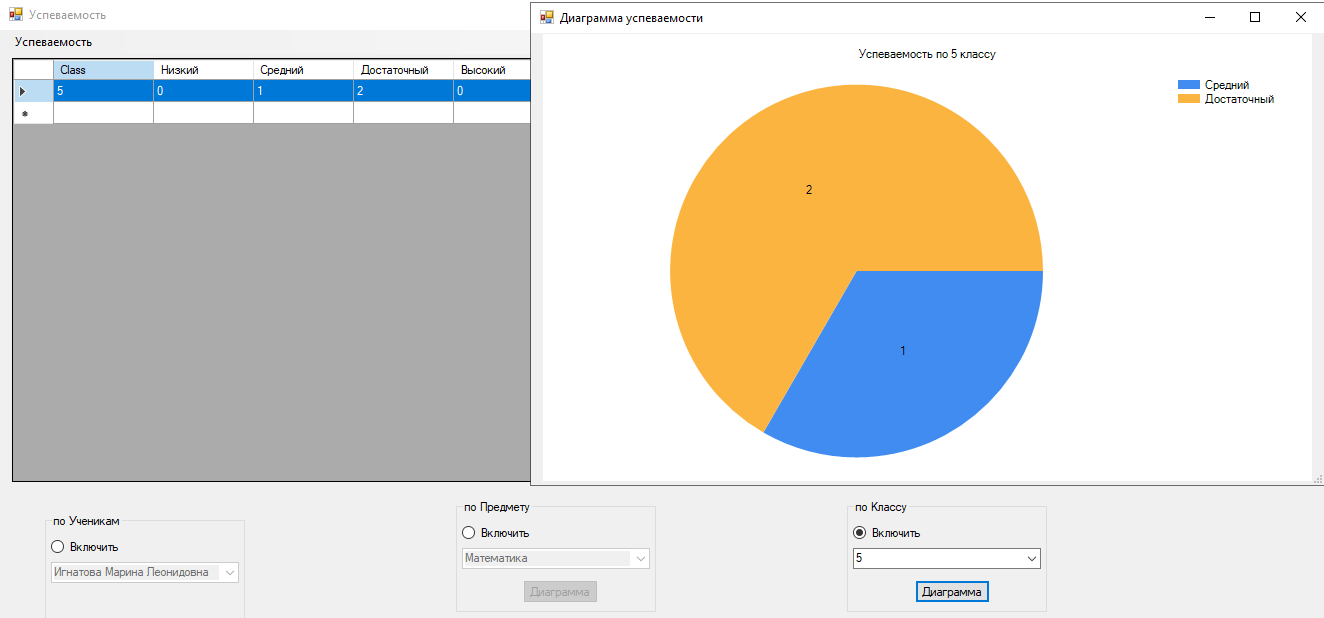


Рисунок A.13 – Диаграмма успеваемости по классу

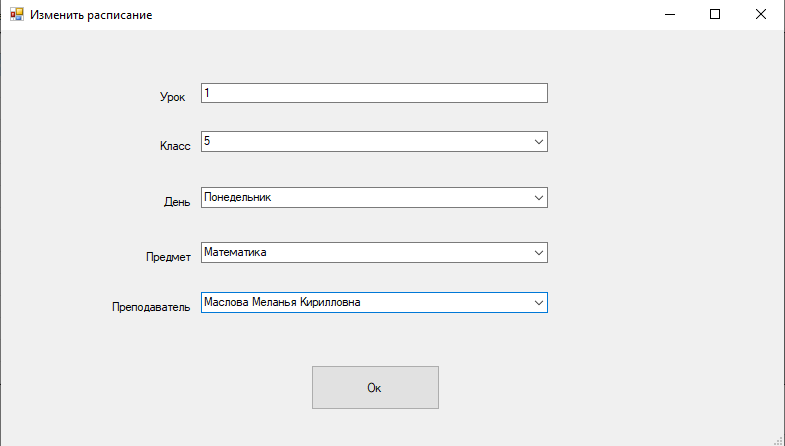


Рисунок A.14 – Окно изменения расписания

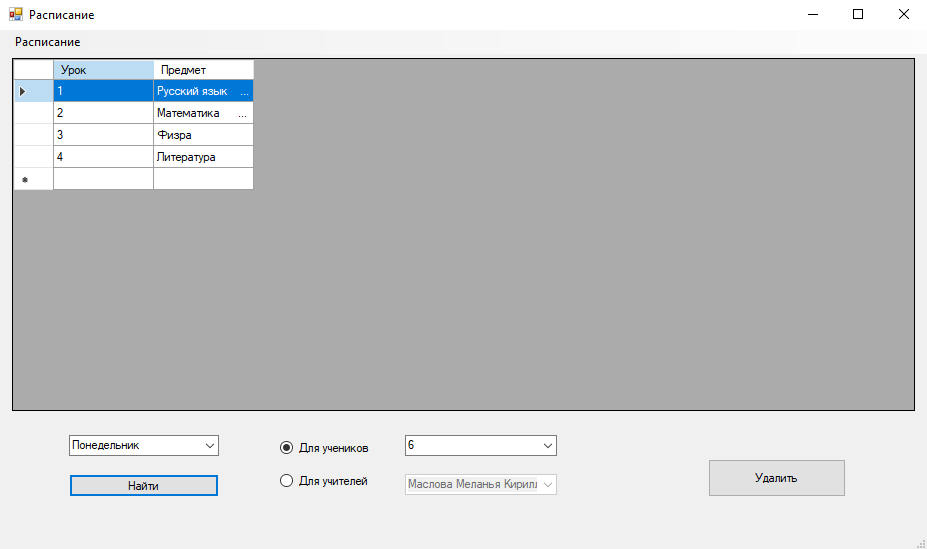


Рисунок A.15 – Результат поиска расписания по классам

# ПРИЛОЖЕНИЕ Б Фрагменты листинга

Листинг Б1 – Файл «Main.cs»

using System;

using System.Data;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace CourseWork

{

public partial class Main : Form

{

DataSet ds;

SqlDataAdapter adapter;

private ToDoc doc = new ToDoc();

string sql1 = "SELECT ID\_Teach, FullName AS ФИО, Post AS Должность, Rank AS Категория, BirthDate AS [Дата рождения], Years AS Возраст, Experience AS [Пед. стаж]" +

", YearOfCertification AS [Год Аттестации], YearOfCourses AS [Год Курсов], Phone AS Телефон, Email AS Почта, Load AS Нагрузка, " +

"VacationFrom, VacationTo, SickFrom, SickTo FROM Teachers";

string sql2 = "SELECT ID\_Pupil, FullName AS ФИО, BirthDate AS [Дата рождения], Years AS Возраст, Class.Class AS Класс" +

", District AS [Район], Social AS [Льготная категория], Phone AS Телефон," +

"DateOfReceipt AS [Дата Приема], DateOfDisposal AS [Дата Выбытия], Parent AS [ФИО родителя], " +

"PhoneParent AS [Телефон родителя] FROM Class INNER JOIN Pupils on Class.ID\_Class = Pupils.Class";

string sql3 = "SELECT ID\_Event, Teachers.FullName AS Ответственный, Importance AS Важность, " +

"Term AS Срок, Place AS Место, Information AS Информация," +

"TypeEvent.TypeName AS Тип, Mark AS Отметка FROM Events INNER JOIN " +

"TypeEvent ON Events.Type = TypeEvent.ID\_Type INNER JOIN Teachers ON Events.Post = Teachers.ID\_Teach";

char table = 't';

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public Main()

{

InitializeComponent();

if (Status.Value == Role.Admin)

{

FillTeacherGrid();

}

if (Status.Value == Role.Teacher)

{

FillTeacherGrid();

учителяToolStripChange.Enabled = false;

учителяToolStripAdd.Enabled = false;

учащегосяToolStripChange.Enabled = false;

событиеToolStripChange.Enabled = false;

}

if (Status.Value == Role.Pupil)

{

изменитьToolStripMenuItem.Enabled = false;

добавитьToolStripMenuItem1.Enabled = false;

просмотрToolStripMenuItem1.Enabled = false;

radioButton2.Enabled = false;

button1.Visible = false;

}

учащегосяToolStripChange.Enabled = false;

событиеToolStripChange.Enabled = false;

}

private void button1\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите удалить эту запись",

"Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

foreach (DataGridViewRow row in dataGridView1.SelectedRows)

{

dataGridView1.Rows.Remove(row);

}

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

if (table == 't')

{

command.CommandText = "DELETE FROM Teachers WHERE ID\_Teach=@id";

}

if (table == 'p')

{

command.CommandText = "DELETE FROM Pupils WHERE ID\_Pupil=@id";

}

if (table == 'e')

{

command.CommandText = "DELETE FROM Events WHERE ID\_Event=@id";

}

command.Parameters.AddWithValue("@id", id);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

}

}

private void учителяToolStripView\_Click(object sender, EventArgs e)

{

button2.Visible = false;

button3.Visible = false;

FillTeacherGrid();

table = 't';

учителяToolStripChange.Enabled = true;

учащегосяToolStripChange.Enabled = false;

событиеToolStripChange.Enabled = false;

}

private void учащиесяToolStripView\_Click(object sender, EventArgs e)

{

button2.Visible = false;

button3.Visible = true;

FillPupilGrid();

table = 'p';

учителяToolStripChange.Enabled = false;

учащегосяToolStripChange.Enabled = true;

событиеToolStripChange.Enabled = false;

}

private void событияToolStripView\_Click(object sender, EventArgs e)

{

button2.Visible = true;

button3.Visible = false;

FillEventGrid();

table = 'e';

учителяToolStripChange.Enabled = false;

учащегосяToolStripChange.Enabled = false;

событиеToolStripChange.Enabled = true;

}

private void успеваемостьToolStripMenuItem\_Click(object sender, EventArgs e)

{

Hide();

Performance perf = new Performance();

perf.ShowDialog();

Show();

}

private void учителяToolStripAdd\_Click(object sender, EventArgs e)

{

Teacher tech = new Teacher();

tech.ShowDialog();

FillTeacherGrid();

}

private void учителяToolStripChange\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите изменить эту запись",

"Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

Teacher tech = new Teacher(id);

tech.ShowDialog();

FillTeacherGrid();

}

}

}

public void FillTeacherGrid()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sql1, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID\_Teach"].ReadOnly = true;

}

}

public void FillPupilGrid()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sql2, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID\_Pupil"].ReadOnly = true;

}

}

public void FillEventGrid()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sql3, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID\_Event"].ReadOnly = true;

dataGridView1.Columns["Отметка"].ReadOnly = true;

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

radioButton2.Checked = false;

if (FindtextBox.Text == "")

{

MessageBox.Show("Поле для пойска пустое",

"Notification", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

string sql = "";

if (table == 't')

{

sql = string.Format("SELECT ID\_Teach, FullName AS ФИО, Post AS Должность, Rank AS Категория, BirthDate AS [Дата рождения], Years AS Возраст, Experience AS [Пед. стаж]" +

", YearOfCertification AS [Год Аттестации], YearOfCourses AS [Год Курсов], Phone AS Телефон, Email AS Почта, Load AS Нагрузка, " +

"VacationFrom, VacationTo, SickFrom, SickTo FROM Teachers WHERE concat(FullName, Post, Rank, BirthDate, Years, Experience" +

", YearOfCertification, YearOfCourses, Phone, Email, Load, " +

"VacationFrom, VacationTo, SickFrom, SickTo) LIKE '%{0}%'", FindtextBox.Text);

}

if (table == 'p')

{

sql = string.Format("SELECT ID\_Pupil, FullName AS ФИО, BirthDate AS [Дата рождения], Years AS Возраст, Class.Class AS Класс" +

", District AS [Район], Social AS [Льготная категория], Phone AS Телефон," +

"DateOfReceipt AS [Дата Приема], DateOfDisposal AS [Дата Выбытия], Parent AS [ФИО родителя], PhoneParent AS [Телефон родителя] " +

"FROM Class INNER JOIN Pupils on Class.ID\_Class = Pupils.Class WHERE concat(FullName, BirthDate, Years, Class.Class, District, Social, Phone," +

"DateOfReceipt, DateOfDisposal, Parent, PhoneParent) LIKE '%{0}%'", FindtextBox.Text);

}

if (table == 'e')

{

sql = string.Format("SELECT ID\_Event, Teachers.FullName AS Ответственный, Importance AS Важность, " +

"Term AS Срок, Place AS Место, Information AS Информация," +

"TypeEvent.TypeName AS Тип, Mark AS Отметка FROM Events INNER JOIN " +

"TypeEvent ON Events.Type = TypeEvent.ID\_Type INNER JOIN Teachers ON Events.Post = Teachers.ID\_Teach WHERE concat(Teachers.FullName," +

" Importance, Term, Place, Information, TypeEvent.TypeName, Mark) LIKE '%{0}%'", FindtextBox.Text);

}

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql,connectionString);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

}

}

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton2.Checked)

{

radioButton1.Checked = false;

FindtextBox.Text = "";

if (table == 't')

{

FillTeacherGrid();

}

if (table == 'p')

{

FillPupilGrid();

}

if (table == 'e')

{

FillEventGrid();

}

}

}

private void учащегосяToolStripAdd\_Click(object sender, EventArgs e)

{

Pupil pupil = new Pupil();

pupil.ShowDialog();

FillPupilGrid();

}

private void учащегосяToolStripChange\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите изменить эту запись",

"Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

Pupil pupil = new Pupil(id);

pupil.ShowDialog();

FillPupilGrid();

}

}

}

private void событиеToolStripAdd\_Click(object sender, EventArgs e)

{

EventForm eve = new EventForm();

eve.ShowDialog();

FillEventGrid();

}

private void событиеToolStripChange\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите изменить эту запись",

"Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

EventForm eve = new EventForm(id);

eve.ShowDialog();

FillEventGrid();

}

}

}

private void button2\_Click\_1(object sender, EventArgs e)

{

if (table == 'e')

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "UPDATE Events SET Mark = 1 WHERE ID\_Event=@ID";

command.Parameters.AddWithValue("@id", id);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

FillEventGrid();

}

}

}

private void рассписаниеToolStrip\_Click(object sender, EventArgs e)

{

Schedule schedule = new Schedule();

Hide();

schedule.ShowDialog();

Show();

}

private void заявкаНаКурсыToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (table != 't') { return; }

if (dataGridView1.SelectedRows.Count > 0)

{

SaveFileDialog sfd = new SaveFileDialog

{

Filter = "Word Documents (\*.docx)|\*.docx",

FileName = "export.docx"

};

if (sfd.ShowDialog() == DialogResult.OK)

{

Int32 selectedRowCount = dataGridView1.Rows.GetRowCount(DataGridViewElementStates.Selected);

string[,] items = new string[dataGridView1.SelectedRows.Count + 1, 10];

items[0, 0] = "№"; items[0, 1] = "ФИО\n(полностью)"; items[0, 2] = "Место работы (полное название)"; items[0, 3] = "Должность";

items[0, 4] = "Уровень профессионального образования, согласно диплому"; items[0, 5] = "Предметная специализация";

items[0, 6] = "Квалификация согласно диплому"; items[0, 7] = "Педагогический стаж";

items[0, 8] = "Контактный телефон оператора \"Феникс\""; items[0, 9] = "Адрес электронной почты";

int rows = 1;

for(int i=0;i<selectedRowCount;i++)

{

int index = dataGridView1.SelectedRows[i].Index;

items[rows, 0] = rows.ToString();

items[rows, 1] = dataGridView1.Rows[index].Cells[1].Value.ToString();

items[rows, 2] = "МУНИЦИПАЛЬНОЕ ОБШЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ \"КЛАССИЧЕСКАЯ ГУМАНИТАРНАЯ ГИМНАЗИЯ ГОРОДА ДОНЕЦКА\"";

items[rows, 3] = dataGridView1.Rows[index].Cells[2].Value.ToString();

items[rows, 4] = "Высшее профессиональное, специалист";

items[rows, 5] = "---";

items[rows, 6] = "---";

items[rows, 7] = dataGridView1.Rows[index].Cells[6].Value.ToString();

items[rows, 8] = dataGridView1.Rows[index].Cells[9].Value.ToString();

items[rows, 9] = dataGridView1.Rows[index].Cells[10].Value.ToString();

rows++;

}

if (!doc.IsFileInUse(sfd.FileName))

{

doc.CreateWordprocessingDocument(sfd.FileName, 0, items);

}

else

{

MessageBox.Show("Закройте файл в который вы хотите добавить таблицу!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

else

{

MessageBox.Show("Выберите записи которые хотите добавить в таблицу.",

"Notification", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void заявкаНаОздоровлениеДетейЛьготныхКатегорийToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (table != 'p') { return; }

if (dataGridView1.SelectedRows.Count > 0)

{

SaveFileDialog sfd = new SaveFileDialog

{

Filter = "Word Documents (\*.docx)|\*.docx",

FileName = "export.docx"

};

if (sfd.ShowDialog() == DialogResult.OK)

{

DateTime currDate = DateTime.Now;

DateTime season = new DateTime();

if (currDate.Month == 12 || currDate.Month == 1 || currDate.Month == 2)

{

if (currDate.Month == 12) { season = new DateTime(currDate.Year + 1, 2, 1); }

else { season = new DateTime(currDate.Year, 2, 1); }

}

if (currDate.Month > 2 && currDate.Month < 6) { season = new DateTime(currDate.Year, 5, 1); }

if (currDate.Month > 5 && currDate.Month < 9) { season = new DateTime(currDate.Year, 8, 1); }

if (currDate.Month > 8 && currDate.Month < 12) { season = new DateTime(currDate.Year, 11, 1); }

Int32 selectedRowCount = dataGridView1.Rows.GetRowCount(DataGridViewElementStates.Selected);

string[,] items = new string[dataGridView1.SelectedRows.Count + 1, 9];

items[0, 0] = "№"; items[0, 1] = "ФИО ребенка"; items[0, 2] = "Дата рождения"; items[0, 3] = $"Возраст(полных лет на {season:dd.MM.yyyy})";

items[0, 4] = "Место учебы, класс"; items[0, 5] = "Место проживания";

items[0, 6] = "Родители или лица их заменяющие (ФИО, контактный телефон)"; items[0, 7] = "Реквизиты свидетельства о рождении, дата выдачи";

items[0, 8] = "Категория";

int rows = 1;

for (int i = 0; i < selectedRowCount; i++)

{

int index = dataGridView1.SelectedRows[i].Index;

TimeSpan Birthday = season - DateTime.Parse(dataGridView1.Rows[index].Cells[2].Value.ToString());

items[rows, 0] = rows.ToString();

items[rows, 1] = dataGridView1.Rows[index].Cells[1].Value.ToString();

items[rows, 2] = dataGridView1.Rows[index].Cells[2].Value.ToString();

items[rows, 3] = (Birthday.Days / 365).ToString();

items[rows, 4] = $"МОУ «ДКГГ», {dataGridView1.Rows[index].Cells[4].Value.ToString()} класс";

items[rows, 5] = dataGridView1.Rows[index].Cells[5].Value.ToString();

items[rows, 6] = dataGridView1.Rows[index].Cells[10].Value.ToString() + "\n т." + dataGridView1.Rows[index].Cells[11].Value.ToString();

items[rows, 7] = "---";

items[rows, 8] = dataGridView1.Rows[index].Cells[6].Value.ToString();

rows++;

}

if (!doc.IsFileInUse(sfd.FileName))

{

doc.CreateWordprocessingDocument(sfd.FileName, 1, items);

}

else

{

MessageBox.Show("Закройте файл в который вы хотите добавить таблицу!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

else

{

MessageBox.Show("Выберите записи которые хотите добавить в таблицу.",

"Notification", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (table == 'p')

{

if (dataGridView1.SelectedRows.Count > 0)

{

Int32 selectedRowCount = dataGridView1.Rows.GetRowCount(DataGridViewElementStates.Selected);

for (int i = 0; i < selectedRowCount; i++)

{

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

int index = dataGridView1.SelectedRows[i].Index;

int id = int.Parse(dataGridView1.Rows[index].Cells[0].Value.ToString());

command.CommandText = "UPDATE Pupils SET Class = Class+1 WHERE ID\_Pupil=@ID";

command.Parameters.AddWithValue("@id", id);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

FillPupilGrid();

}

}

}

}

}

Листинг Б2 – Файл «Pupil.cs»

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Pupil : Form

{

int IDPupil = 0;

DataSet ds;

SqlDataAdapter adapter;

string sql = "SELECT \* FROM Pupils";

string sql1 = "SELECT \* FROM Class ORDER BY Class";

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public Pupil()

{

InitializeComponent();

Text = "Добавить учащаегося";

}

public Pupil(int id)

{

InitializeComponent();

IDPupil = id;

Text = "Изменить учащаегося";

}

private void button1\_Click(object sender, EventArgs e)

{

if (IDPupil == 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

DataSet ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

// добавим новую строку

DataRow newRow = dt.NewRow();

newRow["FullName"] = textBoxFullName.Text;

newRow["BirthDate"] = textBoxBirthDate.Text;

newRow["Years"] = textBoxYears.Text;

newRow["Class"] = comboBoxClass.SelectedValue ;

newRow["District"] = textBoxDistrict.Text;

newRow["Social"] = textBoxSocial.Text;

newRow["Phone"] = textBoxPhone.Text;

newRow["DateOfReceipt"] = textBoxDateOfReceipt.Text;

newRow["DateOfDisposal"] = textBoxDateOfDisposal.Text;

newRow["Parent"] = textBoxParent.Text;

newRow["PhoneParent"] = textBoxPhoneParent.Text;

dt.Rows.Add(newRow);

// создаем объект SqlCommandBuilder

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(ds);

ds.Clear();

}

}

else

{

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "UPDATE Pupils SET FullName = @fn, BirthDate = @bd, Years = @y, Class = @cl" +

", District = @dist, Social = @s, Phone = @phone," +

"DateOfReceipt = @dor, DateOfDisposal = @dod, Parent = @p, PhoneParent = @pp WHERE ID\_Pupil=@ID";

command.Parameters.AddWithValue("@fn", textBoxFullName.Text);

command.Parameters.AddWithValue("@bd", textBoxBirthDate.Text);

command.Parameters.AddWithValue("@y", textBoxYears.Text);

command.Parameters.AddWithValue("@cl", comboBoxClass.SelectedValue);

command.Parameters.AddWithValue("@dist", textBoxDistrict.Text);

command.Parameters.AddWithValue("@s", textBoxSocial.Text);

command.Parameters.AddWithValue("@phone", textBoxPhone.Text);

command.Parameters.AddWithValue("@dor", textBoxDateOfReceipt.Text);

if (textBoxDateOfDisposal.Text == null)

{

command.Parameters.AddWithValue("@dod", DBNull.Value);

}

else

{

command.Parameters.AddWithValue("@dod", textBoxDateOfDisposal.Text);

}

command.Parameters.AddWithValue("@p", textBoxParent.Text);

command.Parameters.AddWithValue("@pp", textBoxPhoneParent.Text);

command.Parameters.AddWithValue("@id", IDPupil);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

DialogResult = System.Windows.Forms.DialogResult.OK;

Close();

}

private void Pupil\_Load(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

adapter = new SqlDataAdapter(sql1, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxClass.DataSource = ds.Tables[0];

comboBoxClass.DisplayMember = "Class";

comboBoxClass.ValueMember = "ID\_Class";

}

if (IDPupil != 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string cmdStr = "select \* from Pupils where ID\_Pupil=@ID";

SqlCommand command = new SqlCommand(cmdStr, connection);

command.Parameters.AddWithValue("@id", IDPupil);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

reader.Read();

textBoxFullName.Text = reader.GetString(1);

textBoxBirthDate.Text = reader.GetDateTime(2).ToString("dd/MM/yyyy");

textBoxYears.Text = reader.GetInt32(3).ToString();

comboBoxClass.SelectedValue = reader.GetInt32(4);

textBoxDistrict.Text = reader.GetString(5);

textBoxSocial.Text = reader.GetString(6);

textBoxPhone.Text = reader.GetString(7);

textBoxDateOfReceipt.Text = reader.GetDateTime(8).ToString("dd/MM/yyyy");

textBoxDateOfDisposal.Text = reader.GetDateTime(9).ToString("dd/MM/yyyy");

textBoxParent.Text = reader.GetString(10);

textBoxPhoneParent.Text = reader.GetString(11);

connection.Close();

}

}

}

}

}

Листинг Б3 – Файл «Teacher.cs»

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Teacher : Form

{

int IDTeacher = 0;

string sql = "SELECT \* FROM Teachers";

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public Teacher()

{

InitializeComponent();

Text ="Добавить учителя";

}

public Teacher(int id) // конструктор с параметром - кодом книги для редактирования

{

InitializeComponent();

IDTeacher = id;

Text = "Изменить учителя";

using (SqlConnection connection = new SqlConnection(connectionString))

{

string cmdStr = string.Format("SELECT \* FROM Teachers WHERE ID\_Teach = {0}", IDTeacher);

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(cmdStr, connection);

DataSet ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

DataRow[] row = dt.Select();

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (IDTeacher == 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

DataSet ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

// добавим новую строку

DataRow newRow = dt.NewRow();

newRow["FullName"] = textBoxName.Text;

newRow["Post"] = textBoxPost.Text;

newRow["Rank"] = textBoxRank.Text;

newRow["BirthDate"] = textBoxBirthDate.Text;

newRow["Years"] = textBoxYears.Text;

newRow["Experience"] = textBoxExperience.Text;

newRow["YearOfCertification"] = textBoxYearOfCertification.Text;

newRow["YearOfCourses"] = textBoxYearOfCourses.Text;

newRow["Phone"] = textBoxPhone.Text;

newRow["Email"] = textBoxEmail.Text;

newRow["Load"] = textBoxLoad.Text;

newRow["VacationFrom"] = textBoxVacationFrom.Text;

newRow["VacationTo"] = textBoxVacationTo.Text;

newRow["SickFrom"] = textBoxSickFrom.Text;

newRow["SickTo"] = textBoxSickTo.Text;

dt.Rows.Add(newRow);

// создаем объект SqlCommandBuilder

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(ds);

ds.Clear();

}

}

else

{

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "UPDATE Teachers SET FullName = @fn, Post = @p, Rank = @r, BirthDate = @bd, Years = @y, Experience = @exp" +

", YearOfCertification = @yocert, YearOfCourses = @yocours, Phone = @phone, Email = @email, Load = @l, " +

"VacationFrom = @vf, VacationTo = @vt, SickFrom = @sf, SickTo = @st WHERE ID\_Teach=@ID";

command.Parameters.AddWithValue("@fn", textBoxName.Text);

command.Parameters.AddWithValue("@P", textBoxPost.Text);

command.Parameters.AddWithValue("@r", textBoxRank.Text);

command.Parameters.AddWithValue("@bd", textBoxBirthDate.Text);

command.Parameters.AddWithValue("@y", textBoxYears.Text);

command.Parameters.AddWithValue("@exp", textBoxExperience.Text);

command.Parameters.AddWithValue("@yocert", textBoxYearOfCertification.Text);

command.Parameters.AddWithValue("@yocours", textBoxYearOfCourses.Text);

command.Parameters.AddWithValue("@phone", textBoxPhone.Text);

command.Parameters.AddWithValue("@email", textBoxEmail.Text);

command.Parameters.AddWithValue("@l", textBoxLoad.Text);

command.Parameters.AddWithValue("@vf", textBoxVacationFrom.Text);

command.Parameters.AddWithValue("@vt", textBoxVacationTo.Text);

command.Parameters.AddWithValue("@sf", textBoxSickFrom.Text);

command.Parameters.AddWithValue("@st", textBoxSickTo.Text);

command.Parameters.AddWithValue("@id", IDTeacher);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

DialogResult = System.Windows.Forms.DialogResult.OK;

Close();

}

private void Teacher\_Load(object sender, EventArgs e)

{

if (IDTeacher != 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string cmdStr = "select \* from Teachers where ID\_Teach=@ID";

SqlCommand command = new SqlCommand(cmdStr, connection);

command.Parameters.AddWithValue("@id", IDTeacher);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

reader.Read();

textBoxName.Text = reader.GetString(1);

textBoxPost.Text = reader.GetString(2);

textBoxRank.Text = reader.GetString(3);

textBoxBirthDate.Text = reader.GetDateTime(4).ToString("dd/MM/yyyy");

textBoxYears.Text = reader.GetInt32(5).ToString();

textBoxExperience.Text = reader.GetInt32(6).ToString();

textBoxYearOfCertification.Text = reader.GetInt32(7).ToString();

textBoxYearOfCourses.Text = reader.GetInt32(8).ToString();

textBoxPhone.Text = reader.GetString(9);

textBoxEmail.Text = reader.GetString(10);

textBoxLoad.Text = reader.GetDouble(11).ToString();

textBoxVacationFrom.Text = reader.GetDateTime(12).ToString("dd/MM/yyyy");

textBoxVacationTo.Text = reader.GetDateTime(13).ToString("dd/MM/yyyy");

textBoxSickFrom.Text = reader.GetDateTime(14).ToString("dd/MM/yyyy");

textBoxSickTo.Text = reader.GetDateTime(15).ToString("dd/MM/yyyy");

connection.Close();

}

}

}

}

}

Листинг Б4 – Файл «EventForm.cs»

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseWork

{

public partial class EventForm : Form

{

int IDEvent = 0;

DataSet ds;

SqlDataAdapter adapter;

string sql = "SELECT \* FROM Events";

string sql1 = "SELECT \* FROM TypeEvent";

string sql2 = "SELECT \* FROM Teachers";

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public EventForm()

{

InitializeComponent();

Text = "Добавить событие";

}

public EventForm(int ID)

{

InitializeComponent();

IDEvent = ID;

Text = "Изменить событие";

}

private void button1\_Click(object sender, EventArgs e)

{

if (IDEvent == 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

DataSet ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

// добавим новую строку

DataRow newRow = dt.NewRow();

newRow["Post"] = comboBoxTeacher.SelectedValue;

newRow["Importance"] = textBox2.Text;

newRow["Term"] = textBox3.Text;

newRow["Place"] = textBox4.Text;

newRow["Information"] = textBox5.Text;

newRow["Mark"] = false;

newRow["Type"] = comboBoxType.SelectedValue;

dt.Rows.Add(newRow);

// создаем объект SqlCommandBuilder

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(ds);

ds.Clear();

}

}

else

{

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "UPDATE Events SET Post = @p, Importance = @imp, Term = @t, Place = @place" +

", Information = @inf, Mark = @m, Type = @type WHERE ID\_Event=@ID";

command.Parameters.AddWithValue("@p", comboBoxTeacher.SelectedValue);

command.Parameters.AddWithValue("@imp", textBox2.Text);

command.Parameters.AddWithValue("@t", textBox3.Text);

command.Parameters.AddWithValue("@place", textBox4.Text);

command.Parameters.AddWithValue("@inf", textBox5.Text);

command.Parameters.AddWithValue("@m", 0);

command.Parameters.AddWithValue("@type", comboBoxType.SelectedValue);

command.Parameters.AddWithValue("@id", IDEvent);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

DialogResult = System.Windows.Forms.DialogResult.OK;

Close();

}

private void Event\_Load(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

adapter = new SqlDataAdapter(sql1, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxType.DataSource = ds.Tables[0];

comboBoxType.DisplayMember = "TypeName";

comboBoxType.ValueMember = "ID\_Type";

adapter = new SqlDataAdapter(sql2, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxTeacher.DataSource = ds.Tables[0];

comboBoxTeacher.DisplayMember = "FullName";

comboBoxTeacher.ValueMember = "ID\_Teach";

}

if (IDEvent != 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string cmdStr = "select \* from Events where ID\_Event=@ID";

SqlCommand command = new SqlCommand(cmdStr, connection);

command.Parameters.AddWithValue("@id", IDEvent);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

reader.Read();

comboBoxType.SelectedValue = reader.GetInt32(1);

textBox2.Text = reader.GetString(2);

textBox3.Text = reader.GetString(3);

textBox4.Text = reader.GetString(4);

textBox5.Text = reader.GetString(5);

comboBoxTeacher.SelectedValue = reader.GetInt32(6);

connection.Close();

}

}

}

}

}

Листинг Б5 – Файл «Schedule.cs»

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Schedule : Form

{

DataSet ds;

SqlDataAdapter adapter;

string sql = "SELECT Schedule.Lesson AS Урок, Subject.Name AS Предмет " +

"FROM Day INNER JOIN " +

"Schedule ON Day.ID\_Day = Schedule.Day INNER JOIN " +

"Class ON Schedule.Class = Class.ID\_Class INNER JOIN " +

"Teachers ON Schedule.Teacher = Teachers.ID\_Teach INNER JOIN " +

"Subject ON Schedule.Subjest = Subject.ID\_Subject";

string sql1 = "SELECT Schedule.Lesson AS Урок, Subject.Name AS Предмет , Class.Class AS Класс " +

"FROM Day INNER JOIN " +

"Schedule ON Day.ID\_Day = Schedule.Day INNER JOIN " +

"Class ON Schedule.Class = Class.ID\_Class INNER JOIN " +

"Teachers ON Schedule.Teacher = Teachers.ID\_Teach INNER JOIN " +

"Subject ON Schedule.Subjest = Subject.ID\_Subject";

string sql2 = "SELECT \* FROM Class ORDER BY Class";

string sql3 = "SELECT \* FROM Day ORDER BY ID\_Day";

string sql4 = "SELECT \* FROM Teachers";

string sqlSched = "SELECT Schedule.ID\_Sched, Class.Class AS Класс, Day.Day AS День, Teachers.FullName AS Учитель, Subject.Name AS Предмет, " +

"Schedule.Lesson AS Урок FROM Day INNER JOIN " +

"Schedule ON dbo.Day.ID\_Day = Schedule.Day INNER JOIN " +

"Class ON Schedule.Class = Class.ID\_Class INNER JOIN " +

"Subject ON Schedule.Subjest = Subject.ID\_Subject INNER JOIN " +

"Teachers ON Schedule.Teacher = Teachers.ID\_Teach";

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public Schedule()

{

InitializeComponent();

FillScheduleGrid();

if (Status.Value == Role.Pupil)

{

radioButton2.Enabled = false;

рассписаниеToolStripMenuItem.Enabled = false;

}

}

public void FillScheduleGrid()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sqlSched, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID\_Sched"].ReadOnly = true;

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

radioButton2.Checked = false;

comboBoxClass.Enabled = true;

comboBoxTeacher.Enabled = false;

}

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton2.Checked)

{

radioButton1.Checked = false;

comboBoxClass.Enabled = false;

comboBoxTeacher.Enabled = true;

}

}

private void Schedule\_Load(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

adapter = new SqlDataAdapter(sql2, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxClass.DataSource = ds.Tables[0];

comboBoxClass.DisplayMember = "Class";

comboBoxClass.ValueMember = "ID\_Class";

adapter = new SqlDataAdapter(sql3, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxDay.DataSource = ds.Tables[0];

comboBoxDay.DisplayMember = "Day";

comboBoxDay.ValueMember = "ID\_Day";

adapter = new SqlDataAdapter(sql4, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxTeacher.DataSource = ds.Tables[0];

comboBoxTeacher.DisplayMember = "FullName";

comboBoxTeacher.ValueMember = "ID\_Teach";

}

}

private void button1\_Click(object sender, EventArgs e)

{

string sqlFind = "";

using (SqlConnection connection = new SqlConnection(connectionString))

{

if (radioButton1.Checked == true)

{

sqlFind = string.Format(sql + " WHERE Schedule.Class = {0} AND Schedule.Day = {1}", comboBoxClass.SelectedValue, comboBoxDay.SelectedValue);

}

else

{

sqlFind = string.Format(sql1 + " WHERE Schedule.Teacher = {0} AND Schedule.Day = {1}", comboBoxTeacher.SelectedValue, comboBoxDay.SelectedValue);

}

connection.Open();

adapter = new SqlDataAdapter(sqlFind, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

}

}

private void добавитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

ScheduleAdd SchedAdd = new ScheduleAdd();

SchedAdd.ShowDialog();

FillScheduleGrid();

}

private void показатьToolStripMenuItem\_Click(object sender, EventArgs e)

{

FillScheduleGrid();

}

private void изменитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите изменить эту запись",

"Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

ScheduleAdd SchedChange = new ScheduleAdd(id);

SchedChange.ShowDialog();

FillScheduleGrid();

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите удалить эту запись",

"Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

foreach (DataGridViewRow row in dataGridView1.SelectedRows)

{

dataGridView1.Rows.Remove(row);

}

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Schedule WHERE ID\_Sched=@id";

command.Parameters.AddWithValue("@id", id);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

}

}

}

}

Листинг Б6 – Файл «ScheduleAdd.cs»

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseWork

{

public partial class ScheduleAdd : Form

{

int IDSchedule = 0;

DataSet ds;

SqlDataAdapter adapter;

string sql4 = "SELECT \* FROM Subject";

string sql1 = "SELECT \* FROM Class";

string sql2 = "SELECT \* FROM Teachers";

string sql3 = "SELECT \* FROM Day";

string sql = "SELECT \* FROM Schedule";

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public ScheduleAdd()

{

InitializeComponent();

Text = "Добавить расписание";

}

public ScheduleAdd(int id)

{

InitializeComponent();

IDSchedule = id;

Text = "Изменить расписание";

}

private void ScheduleAdd\_Load(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

adapter = new SqlDataAdapter(sql1, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxClass.DataSource = ds.Tables[0];

comboBoxClass.DisplayMember = "Class";

comboBoxClass.ValueMember = "ID\_Class";

adapter = new SqlDataAdapter(sql2, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxTeacher.DataSource = ds.Tables[0];

comboBoxTeacher.DisplayMember = "FullName";

comboBoxTeacher.ValueMember = "ID\_Teach";

adapter = new SqlDataAdapter(sql3, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxDay.DataSource = ds.Tables[0];

comboBoxDay.DisplayMember = "Day";

comboBoxDay.ValueMember = "ID\_Day";

adapter = new SqlDataAdapter(sql4, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxSubject.DataSource = ds.Tables[0];

comboBoxSubject.DisplayMember = "Name";

comboBoxSubject.ValueMember = "ID\_Subject";

}

if (IDSchedule != 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string cmdStr = "select \* from Schedule where ID\_Sched=@ID";

SqlCommand command = new SqlCommand(cmdStr, connection);

command.Parameters.AddWithValue("@id", IDSchedule);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

reader.Read();

comboBoxClass.SelectedValue = reader.GetInt32(1);

comboBoxDay.SelectedValue = reader.GetInt32(2);

comboBoxTeacher.SelectedValue = reader.GetInt32(3);

comboBoxSubject.SelectedValue = reader.GetInt32(4);

textBoxLesson.Text = reader.GetInt32(5).ToString();

connection.Close();

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (IDSchedule == 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

DataSet ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

// добавим новую строку

DataRow newRow = dt.NewRow();

newRow["Class"] = comboBoxClass.SelectedValue;

newRow["Day"] = comboBoxDay.SelectedValue;

newRow["Teacher"] = comboBoxTeacher.SelectedValue;

newRow["Subjest"] = comboBoxSubject.SelectedValue;

newRow["Lesson"] = textBoxLesson.Text;

dt.Rows.Add(newRow);

// создаем объект SqlCommandBuilder

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(ds);

ds.Clear();

}

}

else

{

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "UPDATE Schedule SET Class = @cl, Day = @d, Teacher = @t, Subjest = @s" +

", Lesson = @l WHERE ID\_Sched=@ID";

command.Parameters.AddWithValue("@cl", comboBoxClass.SelectedValue);

command.Parameters.AddWithValue("@d", comboBoxDay.SelectedValue);

command.Parameters.AddWithValue("@t", comboBoxTeacher.SelectedValue);

command.Parameters.AddWithValue("@s", comboBoxSubject.SelectedValue);

command.Parameters.AddWithValue("@l", textBoxLesson.Text);

command.Parameters.AddWithValue("@id", IDSchedule);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

DialogResult = System.Windows.Forms.DialogResult.OK;

Close();

}

}

}

Листинг Б7 – Файл «Performance.cs»

using System;

using System.Data;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace CourseWork

{

public partial class Performance : Form

{

DataSet ds;

DataSet DiagramDs = new DataSet();

SqlDataAdapter adapter;

string sql1 = "SELECT \* FROM Pupils";

string sql2 = "SELECT \* FROM Subject ORDER BY ID\_Subject";

string sql3 = "SELECT \* FROM Class ";

string sql4 = "SELECT \* FROM PerfomanceClass ";

string sql5 = "SELECT Оценка, Предмет, Период FROM PerfomancePupil ";

string sql6 = "SELECT \* FROM PerfomanceSubject ";

string sqlPerf = "SELECT \* FROM PerfomanceView";

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public Performance()

{

InitializeComponent();

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sql1, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxPupil.DataSource = ds.Tables[0];

comboBoxPupil.DisplayMember = "FullName";

comboBoxPupil.ValueMember = "ID\_Pupil";

adapter = new SqlDataAdapter(sql2, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxLesson.DataSource = ds.Tables[0];

comboBoxLesson.DisplayMember = "Name";

comboBoxLesson.ValueMember = "ID\_Subject";

adapter = new SqlDataAdapter(sql3, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxClass.DataSource = ds.Tables[0];

comboBoxClass.DisplayMember = "Class";

comboBoxClass.ValueMember = "Class";

}

if (Status.Value == Role.Admin)

{

}

if (Status.Value == Role.Teacher)

{

}

if (Status.Value == Role.Pupil)

{

radioButton2.Enabled = false;

radioButton3.Enabled = false;

}

FillPerfomanceGrid();

}

private void Perfomance\_FormClosing(object sender, FormClosingEventArgs e)

{

DialogResult = System.Windows.Forms.DialogResult.OK;

}

private void comboBoxLesson\_SelectedIndexChanged(object sender, EventArgs e)

{

DiagramDs.Tables.Clear();

string sqlwhere = sql6 + string.Format("WHERE ID\_Subject = {0}", comboBoxLesson.SelectedValue);

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sqlwhere, connection);

ds = new DataSet();

adapter.Fill(ds);

int sub = int.Parse(ds.Tables[0].Rows[0].ItemArray[0].ToString());

int low = 0;

int middle = 0;

int sufficient = 0;

int high = 0;

int intValue = 0;

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

{

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[1].ToString(), out intValue) == true)

{

low += int.Parse(ds.Tables[0].Rows[i].ItemArray[1].ToString());

}

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[2].ToString(), out intValue) == true)

{

middle += int.Parse(ds.Tables[0].Rows[i].ItemArray[2].ToString());

}

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[3].ToString(), out intValue) == true)

{

sufficient += int.Parse(ds.Tables[0].Rows[i].ItemArray[3].ToString());

}

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[4].ToString(), out intValue) == true)

{

high += int.Parse(ds.Tables[0].Rows[i].ItemArray[4].ToString());

}

}

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

{

ds.Tables[0].Rows[i].Delete();

}

DataTable dt = ds.Tables[0];

DataRow newRow = dt.NewRow();

newRow["ID\_Subject"] = sub;

newRow["Низкий"] = low;

newRow["Средний"] = middle;

newRow["Достаточный"] = sufficient;

newRow["Высокий"] = high;

dt.Rows.Add(newRow);

DiagramDs.Tables.Add(ds.Tables[0].Copy());

dataGridView1.DataSource = ds.Tables[0];

}

}

private void comboBoxClass\_SelectedIndexChanged(object sender, EventArgs e)

{

DiagramDs.Tables.Clear();

string sqlwhere = sql4 + string.Format(" Where Class = {0}", comboBoxClass.SelectedValue);

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sqlwhere, connection);

ds = new DataSet();

adapter.Fill(ds);

int cl = int.Parse(ds.Tables[0].Rows[0].ItemArray[0].ToString());

int low = 0;

int middle = 0;

int sufficient = 0;

int high = 0;

int intValue = 0;

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

{

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[1].ToString(), out intValue) == true)

{

low += int.Parse(ds.Tables[0].Rows[i].ItemArray[1].ToString());

}

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[2].ToString(), out intValue) == true)

{

middle += int.Parse(ds.Tables[0].Rows[i].ItemArray[2].ToString());

}

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[3].ToString(), out intValue) == true)

{

sufficient += int.Parse(ds.Tables[0].Rows[i].ItemArray[3].ToString());

}

if (int.TryParse(ds.Tables[0].Rows[i].ItemArray[4].ToString(), out intValue) == true)

{

high += int.Parse(ds.Tables[0].Rows[i].ItemArray[4].ToString());

}

}

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)

{

ds.Tables[0].Rows[i].Delete();

}

DataTable dt = ds.Tables[0];

DataRow newRow = dt.NewRow();

newRow["Class"] = cl;

newRow["Низкий"] = low;

newRow["Средний"] = middle;

newRow["Достаточный"] = sufficient;

newRow["Высокий"] = high;

dt.Rows.Add(newRow);

DiagramDs.Tables.Add(ds.Tables[0].Copy());

dataGridView1.DataSource = ds.Tables[0];

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

radioButton2.Checked = false;

radioButton3.Checked = false;

comboBoxPupil.Enabled = true;

button1.Enabled = false;

button2.Enabled = false;

}

else

{

comboBoxPupil.Enabled = false;

}

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton2.Checked)

{

radioButton3.Checked = false;

radioButton1.Checked = false;

comboBoxLesson.Enabled = true;

button1.Enabled = true;

button2.Enabled = false;

}

else

{

comboBoxLesson.Enabled = false;

}

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton3.Checked)

{

radioButton1.Checked = false;

radioButton2.Checked = false;

comboBoxClass.Enabled = true;

button1.Enabled = false;

button2.Enabled = true;

}

else

{

comboBoxClass.Enabled = false;

}

}

private void показатьToolStripMenuItem\_Click(object sender, EventArgs e)

{

FillPerfomanceGrid();

}

public void FillPerfomanceGrid()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sqlPerf, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID\_Perf"].ReadOnly = true;

}

}

private void добавитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

PerformanceAdd PerfAdd = new PerformanceAdd();

PerfAdd.ShowDialog();

FillPerfomanceGrid();

}

private void изменитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Вы действительно хотите изменить эту запись",

"Message", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

if (dataGridView1.SelectedRows.Count == 1)

{

int id = int.Parse(dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

PerformanceAdd PerfChange = new PerformanceAdd(id);

PerfChange.ShowDialog();

FillPerfomanceGrid();

}

}

}

private void comboBoxPupil\_SelectedIndexChanged(object sender, EventArgs e)

{

string sqlwhere = sql5 + string.Format("WHERE Pupil = {0}", comboBoxPupil.SelectedValue);

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

adapter = new SqlDataAdapter(sqlwhere, connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

}

}

private void button1\_Click(object sender, EventArgs e)

{

PerfDiagram diagram = new PerfDiagram(2, DiagramDs.Tables[0].Rows.Count - 1, comboBoxLesson.Text, DiagramDs);

diagram.ShowDialog();

}

private void button2\_Click(object sender, EventArgs e)

{

PerfDiagram diagram = new PerfDiagram(1, DiagramDs.Tables[0].Rows.Count-1,comboBoxClass.Text, DiagramDs);

diagram.ShowDialog();

}

}

}

Листинг Б8 – Файл «PerformanceAdd.cs»

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace CourseWork

{

public partial class PerformanceAdd : Form

{

int IDPerf = 0;

DataSet ds;

SqlDataAdapter adapter;

string sql4 = "SELECT \* FROM Subject";

string sql1 = "SELECT \* FROM Class";

string sql3 = "SELECT \* FROM Period";

string sql = "SELECT \* FROM Pupils";

string sql5 = "SELECT \* FROM Performance";

string connectionString = ConfigurationManager.ConnectionStrings["hhh"].ConnectionString;

public PerformanceAdd()

{

InitializeComponent();

Text = "Добавить успеваемость";

}

public PerformanceAdd(int id)

{

InitializeComponent();

IDPerf = id;

Text = "Изменить успеваемость";

}

private void PerfomanceAdd\_Load(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

adapter = new SqlDataAdapter(sql1, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxClass.DataSource = ds.Tables[0];

comboBoxClass.DisplayMember = "Class";

comboBoxClass.ValueMember = "ID\_Class";

adapter = new SqlDataAdapter(sql, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxPupil.DataSource = ds.Tables[0];

comboBoxPupil.DisplayMember = "FullName";

comboBoxPupil.ValueMember = "ID\_Pupil";

adapter = new SqlDataAdapter(sql3, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxPeriod.DataSource = ds.Tables[0];

comboBoxPeriod.DisplayMember = "Name";

comboBoxPeriod.ValueMember = "ID\_Period";

adapter = new SqlDataAdapter(sql4, connection);

ds = new DataSet();

adapter.Fill(ds);

comboBoxSubject.DataSource = ds.Tables[0];

comboBoxSubject.DisplayMember = "Name";

comboBoxSubject.ValueMember = "ID\_Subject";

}

if (IDPerf != 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string cmdStr = "select \* from Performance where ID\_Perf=@ID";

SqlCommand command = new SqlCommand(cmdStr, connection);

command.Parameters.AddWithValue("@id", IDPerf);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

reader.Read();

comboBoxPupil.SelectedValue = reader.GetInt32(1);

textBoxGrade.Text = reader.GetInt32(2).ToString();

comboBoxSubject.SelectedValue = reader.GetInt32(3);

comboBoxPeriod.SelectedValue = reader.GetInt32(4);

comboBoxClass.SelectedValue = reader.GetInt32(5);

connection.Close();

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (IDPerf == 0)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter(sql5, connection);

DataSet ds = new DataSet();

adapter.Fill(ds);

DataTable dt = ds.Tables[0];

// добавим новую строку

DataRow newRow = dt.NewRow();

newRow["Pupil"] = comboBoxPupil.SelectedValue;

newRow["Grade"] = textBoxGrade.Text;

newRow["Lesson"] = comboBoxSubject.SelectedValue;

newRow["Period"] = comboBoxPeriod.SelectedValue;

newRow["Class"] = comboBoxClass.SelectedValue;

dt.Rows.Add(newRow);

// создаем объект SqlCommandBuilder

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(ds);

ds.Clear();

}

}

else

{

using (SqlConnection connection = new SqlConnection(connectionString))

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "UPDATE Performance SET Pupil = @p, Grade = @g, Lesson = @l, Period = @per" +

", Class = @cl WHERE ID\_Perf=@ID";

command.Parameters.AddWithValue("@p", comboBoxPupil.SelectedValue);

command.Parameters.AddWithValue("@g", textBoxGrade.Text);

command.Parameters.AddWithValue("@l", comboBoxSubject.SelectedValue);

command.Parameters.AddWithValue("@per", comboBoxPeriod.SelectedValue);

command.Parameters.AddWithValue("@cl", comboBoxClass.SelectedValue);

command.Parameters.AddWithValue("@ID", IDPerf);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

DialogResult = System.Windows.Forms.DialogResult.OK;

Close();

}

}

}

Листинг Б9 – Файл «PerfDiagram.cs»

using System.Data;

using System.Windows.Forms;

namespace CourseWork

{

public partial class PerfDiagram : Form

{

public PerfDiagram(int t,int len, string name, DataSet DiagramDs)

{

InitializeComponent();

if (t == 1)

{

chart1.Titles.Add("Успеваемость по "+name+" классу");

chart1.Series["Series1"].IsValueShownAsLabel = true;

}

if (t == 2)

{

chart1.Titles.Add("Успеваемость по предмету: " + name);

chart1.Series["Series1"].IsValueShownAsLabel = true;

}

for (int i = 1; i < DiagramDs.Tables[0].Columns.Count; i++)

{

if (int.Parse(DiagramDs.Tables[0].Rows[len].ItemArray[i].ToString()) == 0)

{

continue;

}

else

{

chart1.Series["Series1"].Points.AddXY(DiagramDs.Tables[0].Columns[i].ColumnName, DiagramDs.Tables[0].Rows[len].ItemArray[i].ToString());

}

}

}

}

}