Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Кафедра прикладной информатики и математики

**ОТЧЕТ**

О прохождении учебной (эксплуатационной) практики

4 семестр

Студент Орлов Алексей Максимович

(фамилия имя отчество)

направления 09.03.03 Прикладная информатика

профиль «Прикладная информатика в экономике»

Группа ПИ-21

Руководитель практики от вуза: доцент кафедры ПИМ Яковлева Л.Л.

Сроки практики: с 03.07.23 по 16.07.23 г.

Оценка за практику:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Чита2023

Оглавление

[ВВЕДЕНИЕ 3](#_Toc146755459)

[1. КРАТКОЕ ОПИСАНИЕ ВЫБРАННОЙ ОРГАНИЗАЦИИ 4](#_Toc146755460)

[2. ОПИСАНИЕ ТРУДОВЫХ ФУНКЦИЙ И АВТОМАТИЗИРУЕМЫХ БИЗНЕС-ПРОЦЕССОВ СПЕЦИАЛИСТА 5](#_Toc146755461)

[3. ОПИСАНИЕ ПРОЦЕССА СОЗДАНИЯ АРМ 7](#_Toc146755462)

[ЗАКЛЮЧЕНИЕ 42](#_Toc146755463)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc146755464)

# ВВЕДЕНИЕ

Методист в образовательной организации играет большую роль в эффективном управлении учебными материалами. Для облегчения работы и обеспечение более эффективных процессов методист нуждается в автоматизированном рабочем месте (АРМ).

Введение АРМ в работу методиста приносит целый ряд преимуществ. Оно значительно повышает производительность и оптимизирует рутинные задачи, освобождая его время для более важных задач. Система автоматически упорядочивает отчёты по пройдённым курсам.

Внедрение АРМ для методиста образовательной организации является важным шагом в направлении эффективного управления обучение студентов. Оно позволяет автоматизировать рутинные задачи, упорядочивать работу с учебными материалами.

# 1. КРАТКОЕ ОПИСАНИЕ ВЫБРАННОЙ ОРГАНИЗАЦИИ

Методист — это специалист, занимающийся организацией образовательного процесса. Именно он оттачивает учебные и образовательные программы, составляет и следит за выполнением годового плана, контролирует соблюдение профессиональных стандартов, создает рекомендации для учителей и педагогов.

На самом деле методист — это достаточно старая профессия, которая образовалась в России еще при Петре I. В современном мире, при развитии дистанционного образования данная профессия стала очень востребована и актуальна. Теперь такие специалисты могут работать дистанционно из любой точки планеты, что дает много новых возможностей для развития.

**Профессиональный стандарт**

Должностные обязанности методиста. Выполняет методическую работу по планированию и организации учебного процесса. Организует формирование учебных планов по направлениям (специальностям) подготовки студентов (бакалавров, специалистов и магистров) в соответствии с образовательными стандартами. Осуществляет распределение учебной нагрузки между факультетами, отделениями и кафедрами. Составляет ведомости выполнения почасовой нагрузки кафедр. Составляет учебное расписание преподавателей, расписание выполнения лабораторных, практических работ, семинаров, консультаций обучающихся (студентов, слушателей). Контролирует наличие и выполнение учебных планов кафедр, индивидуальных планов преподавателей, программы читаемых курсов лекций на предмет соответствия образовательному стандарту. Проверяет наличие учебной документации на кафедрах: рабочих программ курсов, расчета учебной нагрузки на учебный год (семестр), планов работы учебно-вспомогательного персонала, протоколов заседаний кафедры, журналов учета лабораторных работ, домашних заданий, курсового проектирования, списка тем дипломных проектов, списка руководителей дипломных и курсовых проектов, расписаний лабораторных работ, приема домашних заданий, проведения консультаций. Осуществляет оперативное регулирование выполнения учебного процесса. Фиксирует отклонения в выполнении учебной нагрузки с целью урегулирования расписания занятий и недопущения отклонения от образовательного стандарта. Корректирует ведомости выполнения учебной (почасовой) нагрузки. Осуществляет компьютерную обработку получаемой информации. Руководит разработкой системы рейтингов обучающихся (студентов, слушателей), составляет рейтинги, отвечает за обеспечение доступа к ним. Координирует освещение образовательного процесса на сайте структурного подразделения образовательного учреждения. Анализирует и представляет руководству сведения о причинах академической задолженности обучающихся (студентов, слушателей), отсева обучающихся (студентов, слушателей), а также отчеты по выполнению учебной нагрузки преподавателями. Готовит предложения по величине государственного задания и государственного заказа по подготовке бакалавров, специалистов и магистров и иных обучающихся на будущие периоды, по распределению численности профессорско-преподавательского (педагогического) персонала по кафедрам в зависимости от учебной нагрузки, выполняемой кафедрой. Составляет график использования аудиторного фонда для занятий с обучающимися (студентами, слушателями).

Методист должен знать: законы и иные нормативные правовые акты Российской Федерации в области образования и науки; локальные нормативные акты образовательного учреждения; основные технологические процессы, правила и приемы работы по направлениям деятельности образовательного учреждения; методические и нормативные документы по организации учебного процесса, составлению учебно-методической документации и обеспечению учебного процесса; требования образовательного стандарта по направлениям подготовки (специальностям) образовательного учреждения; правила по охране труда и пожарной безопасности.

# 2. ОПИСАНИЕ ТРУДОВЫХ ФУНКЦИЙ И АВТОМАТИЗИРУЕМЫХ БИЗНЕС-ПРОЦЕССОВ СПЕЦИАЛИСТА

Методист является ключевой фигурой в образовательной организации. Он выполняет широкий спектр задач, связанных с контролем профстандартов и соответствием программ обучения нормам ФГОС. Некоторые из трудовых функций методиста включают в себя:

* Разработка и актуализация образовательных программ
* Ведение нормативно-методической базы
* Контроль соответствия учебных планов требованиям законодательства в сфере образования
* Коммуникация с педагогами и учащимися
* Ведение и оформление документации
* Координация деятельности педагогов
* Планирование работы сотрудников учебной организации

В ходе анализа трудовых функций сотрудника с данной должностью я выявил такие, которые, на мой взгляд, требуют автоматизации:

1. Ведение учёт образовательный программ- у каждого преподавателя есть программы обучения по той или иной темы и методист должен знать какие программы есть и соответствуют ли они всем стандартам.
2. Хранения данных о студентах- структурированное хранение данных – одна из самых важных частей автоматизации. Помнить какую программу написал какой преподаватель и кто из групп и студентов учился по ней. Грамотно спроектированная база поможет структурировать и сделать уникальными данные о записях, что поможет реализовать быстрые поиск и фильтрацию данных.
3. Редактирование данных- преподаватель в любое время может изменить свои пособия, а изменения отдать методисту, а листочке или отправить на почту. Этот листочек или письмо еще необходимо найти, и не спутать с другими похожими. При помощи автоматизации это реализуется очень просто, так как каждому курсу в программе присвоен собственный уникальный номер, что позволяет его быстро найти и, в случае необходимости, быстро изменить.
4. Удаление данных – преподаватель в любой момент может как полностью переписать свою методичку, так и уйти из этой образовательной организации**,** так что для такого случая должно быть организовано удаление записи.

# 3. ОПИСАНИЕ ПРОЦЕССА СОЗДАНИЯ АРМ

Для создания автоматизированного рабочего места были выбраны такие инструменты как: VS (Visual Studio) - мощное средство разработчика, которое можно использовать для выполнения всего цикла разработки в одном месте; язык программирования C#; DB Browser for SQLite – быстрая и легкая встраиваемая однофайловая СУБД.

При проектировании базы данных я выявил сущности, необходимые для функционирования программного продукта:

1. Преподаватели-хранит информацию о авторах методических материалов.
2. Курсы-хранит информацию о методичках, которые написали преподаватели.
3. Группы – хранит информацию о группах, которые умчаться по данным методическим материалам.
4. Студенты- хранит информацию о студентах
5. Пройденные курсы-хранит информацию о том, кто, какие, и как учился по данным материалам

Я считаю, что это самые необходимые сущности для автоматизации трудового процесса. База данные была приведена к 3-й нормальной форме (3NF), и соответствует таким требованиям как: отсутствие дублирующих строк в таблицах, атомарное значение в ячейках, в столбцах данные одного типа, таблицы имеют ключи.

Далее я приведу даталогическую модель данных каждой из сущностей в виде таблиц, где указаны типы данных полей и значения по умолчанию.

Таблица №1

Таблица «Преподаватели»

|  |  |
| --- | --- |
| **Преподаватели** | |
| код\_преподавателя(ПК) | Integer (not null) |
| Фамилия | Text (not null) |
| Имя | Text (not null) |
| **Преподаватель** | |
| Отчество | Text (null) |

Таблица №2

Таблица «Курсы»

|  |  |
| --- | --- |
| **Курсы** | |
| код\_курса(ПК) | Integer (not null) |
| название\_курса | Text (not null) |
| код\_преподавателя | Integer (not null) |

Таблица №3

Таблица «Группы»

|  |  |
| --- | --- |
| **Группы** | |
| код\_группы(ПК) | Integer (not null) |
| название\_группы | Text (not null) |

Таблица №4

Таблица «Студенты»

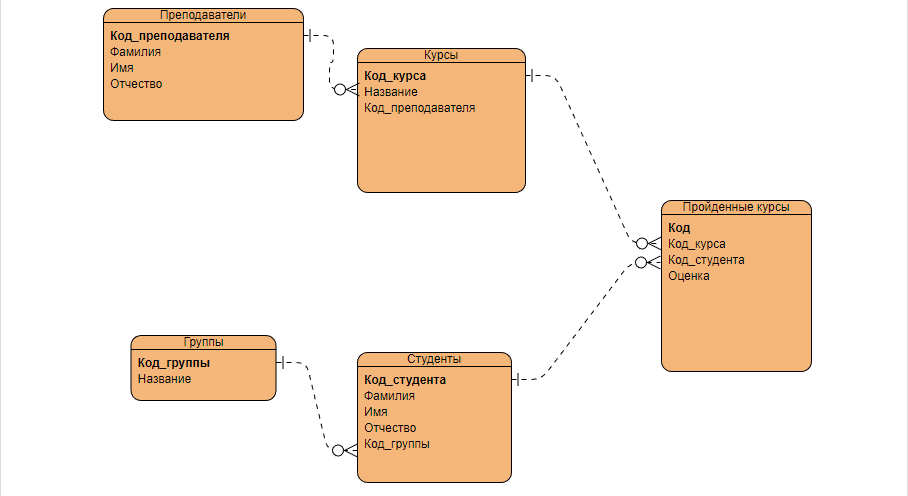
|  |  |
| --- | --- |
| **Студенты** | |
| код\_студента(ПК) | Integer (not null) |
| Фамилия | Text (not null) |
| Имя | Text (not null) |
| Отчество | Text (null) |
| код\_группы | Integer (not null) |

Таблица №5

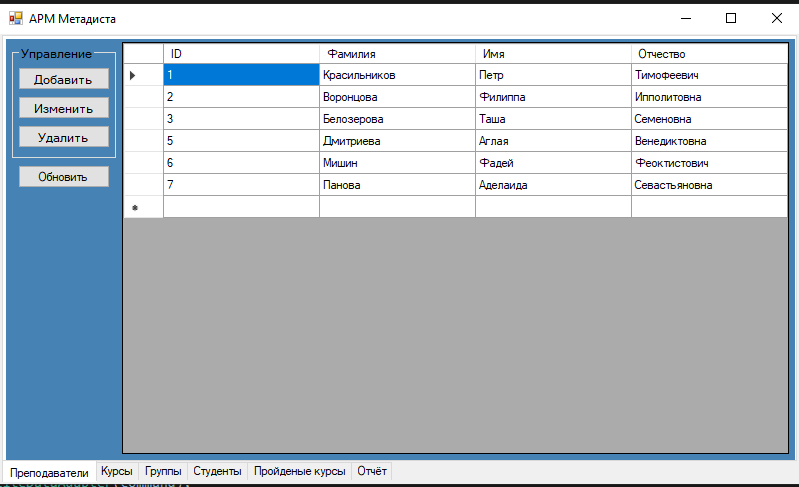
Таблица «Пройденные\_курсы»

|  |  |
| --- | --- |
| **Пройденные\_курсы** | |
| код (ПК) | Integer (not null) |
| код\_курса | Text (not null) |
| код\_студента | Text (not null) |
| оценка | Integer (null) |

Для правильного функционирования базы данных, сущности необходимо связать между друг другом. Для этого построим ER-модель, где наглядно видно связи между сущностями:

 *Рисунок 1 ER-диаграмма*

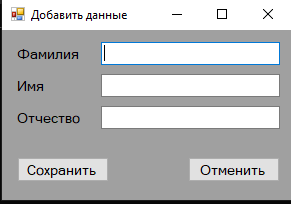
Теперь перейдем к самому автоматизированному месту. При входе в программу, пользователя встречает главное окно на котором расположены вкладки для добавление различной информации.



*Рисунок 2 главное окно программы «Таблица преподавателя» нажата кнопка обновить*

Form1.cs-Главный экран

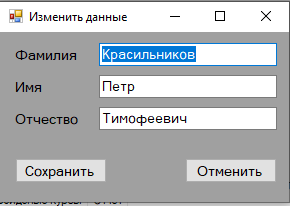
|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SqlClient**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **using** **static** System.Windows.Forms.VisualStyles.VisualStyleElement;    **namespace** **praktika**  {    **public** **partial** **class** **main\_screen** : Form  {  **public** **main\_screen**()  {  InitializeComponent();  grup\_check\_cmb.Visible = **false**;  kurs\_check\_cmb.Visible=**false**;  ocenki\_check\_cmb.Visible = **false**;      }    **private** **void** **button1\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(table\_data\_kurs.CurrentRow.Cells[**0**].Value.ToString());  edit\_kurs windows = **new** edit\_kurs(id);  DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  update\_kurs\_Click(sender, e);  }  }    **private** **void** **update\_prepodavatel\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con =**new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT ID\_prepodavatel [ID],Familia[Фамилия],Name[Имя],Otchestvo[Отчество] FROM prepodavatel";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt =**new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  table\_prepodavatel.DataSource = dt;  con.Close();  }    **private** **void** **comboBox1\_SelectedIndexChanged**(**object** sender, EventArgs e)  {  update\_student\_Click(sender, e);    }    **private** **void** **update\_student\_Click**(**object** sender, EventArgs e)  {    **if** (grup\_filter.Text == "Все"){  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT student.ID\_student [ID],student.Familia[Фамилия],student.Name[Имя],student.Otchestvo[Отчество],grupa.Name\_grup[Название группы] FROM student INNER JOIN grupa ON grupa.ID\_grup=student.ID\_grup";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  table\_student.DataSource = dt;  con.Close();    }  **else**  {  **string** znachenie= grup\_filter.Text.ToString();    SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT student.ID\_student [ID],student.Familia[Фамилия],student.Name[Имя],student.Otchestvo[Отчество],grupa.Name\_grup[Название группы] FROM student INNER JOIN grupa ON grupa.ID\_grup=student.ID\_grup WHERE student.ID\_grup=(SELECT ID\_grup FROM grupa WHERE Name\_grup=\""+znachenie+"\")";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  table\_student.DataSource = dt;  con.Close();    }    }    **private** **void** **table\_kurs\_SelectedIndexChanged**(**object** sender, EventArgs e)  {    **switch** (tab\_kurs.SelectedIndex.ToString())  {    **case** ("0"):  update\_prepodavatel\_Click(sender, e);  **break**;  **case** ("1"):  update\_kurs\_Click(sender, e);  **break**;  **case** ("2"):  update\_grup\_btn\_Click(sender, e);  **break**;  **case** ("3"):  grup\_filter.Items.Clear();  grup\_filter.Items.Add("Все");  grup\_filter.SelectedIndex = **0**;  update\_student\_Click (sender, e);  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT ID\_grup, Name\_grup FROM grupa";  SQLiteCommand command = **new** SQLiteCommand(query, con);    SQLiteDataReader DR = command.ExecuteReader();    **while** (DR.Read())  {  grup\_filter.Items.Add(DR[**1**]);    }        **break**;  **case** ("4"):  fillter\_categorie\_kurs\_completed.SelectedIndex = **0**;  button1\_Click\_1(sender, e);  **break**;  **case** ("5"):      **break**;  }  }    **private** **void** **main\_screen\_Load**(**object** sender, EventArgs e)  {  update\_prepodavatel\_Click(sender, e);  }    **private** **void** **update\_kurs\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT kurs.ID\_kurs[ID курса],kurs.Name[Название курса],prepodavatel.Familia||\" \"|| prepodavatel.Name||\" \"||prepodavatel.otchestvo [ФИО] FROM kurs INNER JOIN prepodavatel on kurs.ID\_prepodavatel=prepodavatel.ID\_prepodavatel ";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  table\_data\_kurs.DataSource = dt;  con.Close();  }    **private** **void** **add\_prepodavatel\_Click**(**object** sender, EventArgs e)  {  add\_prepodavatel windows =**new** add\_prepodavatel();    DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  update\_prepodavatel\_Click(sender, e);  }  }    **private** **void** **edit\_prepodavatel\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(table\_prepodavatel.CurrentRow.Cells[**0**].Value.ToString());  edit\_prepodavatel windows = **new** edit\_prepodavatel(id);  DialogResult result =windows.ShowDialog();  **if** (result==DialogResult.OK)  {  update\_prepodavatel\_Click(sender, e);  }  }    **private** **void** **delete\_prepodavatel\_btn\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(table\_prepodavatel.CurrentRow.Cells[**0**].Value.ToString());  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **var** res = MessageBox.Show("Удалить запись?", "Удаление", MessageBoxButtons.YesNo);  **if** (res == DialogResult.Yes)  {  **string** sql = "DELETE FROM prepodavatel WHERE ID\_prepodavatel =" + id;    SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();    update\_prepodavatel\_Click(sender, e);  }  con.Close();  }    **private** **void** **add\_kurs\_btn\_Click**(**object** sender, EventArgs e)  {  add\_kurs windows = **new** add\_kurs();  DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  update\_kurs\_Click(sender, e);  }  }    **private** **void** **delete\_kurs\_btn\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(table\_data\_kurs.CurrentRow.Cells[**0**].Value.ToString());  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **var** res = MessageBox.Show("Удалить запись?", "Удаление", MessageBoxButtons.YesNo);  **if** (res == DialogResult.Yes)  {  **string** sql = "DELETE FROM kurs WHERE ID\_kurs =" + id;    SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();    update\_kurs\_Click(sender, e);  }  con.Close();  }    **private** **void** **update\_grup\_btn\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT ID\_grup [ID группы], Name\_grup[Название группы] FROM grupa ";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  grupa\_table\_data.DataSource = dt;  con.Close();  }    **private** **void** **add\_grup\_btn\_Click**(**object** sender, EventArgs e)  {  add\_grup windows = **new** add\_grup();  DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  update\_grup\_btn\_Click(sender, e);  }  }    **private** **void** **edit\_student\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(table\_student.CurrentRow.Cells[**0**].Value.ToString());  edit\_student windows = **new** edit\_student(id);  DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  update\_student\_Click(sender, e);  }  }    **private** **void** **edit\_grup\_btn\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(grupa\_table\_data.CurrentRow.Cells[**0**].Value.ToString());  edit\_grup windows = **new** edit\_grup(id);  DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  update\_grup\_btn\_Click(sender, e);  }  }    **private** **void** **delete\_grup\_btn\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(grupa\_table\_data.CurrentRow.Cells[**0**].Value.ToString());  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **var** res = MessageBox.Show("Удалить запись?", "Удаление", MessageBoxButtons.YesNo);  **if** (res == DialogResult.Yes)  {  **string** sql = "DELETE FROM prupa WHERE ID\_grup =" + id;    SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();    update\_grup\_btn\_Click(sender, e);  }  con.Close();  }    **private** **void** **add\_student\_Click**(**object** sender, EventArgs e)  {  add\_student windows = **new** add\_student();  DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  update\_student\_Click(sender, e);  }  }    **private** **void** **delete\_student\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(table\_student.CurrentRow.Cells[**0**].Value.ToString());  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **var** res = MessageBox.Show("Удалить запись?", "Удаление", MessageBoxButtons.YesNo);  **if** (res == DialogResult.Yes)  {  **string** sql = "DELETE FROM student WHERE ID\_student =" + id;    SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();    update\_student\_Click(sender, e);  }  con.Close();  }    **private** **void** **button1\_Click\_1**(**object** sender, EventArgs e)  {  **if** (fillter\_categorie\_kurs\_completed.Text== "По курсам")  {  **if** (fillter\_kurs\_completed.Text== "Все курсы")  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT kurs\_completed.ID [ID],kurs.Name [Название курса],student.Familia||\" \" ||student.Name||\" \" ||student.Otchestvo [ФИО студента],grupa.Name\_grup [Группа],kurs\_completed.ocenka [Оценка] FROM kurs\_completed INNER JOIN kurs on kurs.ID\_kurs= kurs\_completed.ID\_kurs INNER JOIN student on student.ID\_student=kurs\_completed.ID\_student INNER JOIN grupa on student.ID\_grup=grupa.ID\_grup";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  kurs\_completed\_table.DataSource = dt;  con.Close();  }  **else**  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT kurs\_completed.ID [ID],kurs.Name [Название курса],student.Familia||\" \" ||student.Name||\" \" ||student.Otchestvo [ФИО студента],grupa.Name\_grup [Группа],kurs\_completed.ocenka [Оценка] FROM kurs\_completed INNER JOIN kurs on kurs.ID\_kurs= kurs\_completed.ID\_kurs INNER JOIN student on student.ID\_student=kurs\_completed.ID\_student INNER JOIN grupa on student.ID\_grup=grupa.ID\_grup WHERE kurs.Name='"+fillter\_kurs\_completed.Text+"'";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  kurs\_completed\_table.DataSource = dt;  con.Close();  }    }  **else**  {  **if** (fillter\_kurs\_completed.Text == "Все группы")  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT kurs\_completed.ID [ID],kurs.Name [Название курса],student.Familia||\" \" ||student.Name||\" \" ||student.Otchestvo [ФИО студента],grupa.Name\_grup [Группа],kurs\_completed.ocenka [Оценка] FROM kurs\_completed INNER JOIN kurs on kurs.ID\_kurs= kurs\_completed.ID\_kurs INNER JOIN student on student.ID\_student=kurs\_completed.ID\_student INNER JOIN grupa on student.ID\_grup=grupa.ID\_grup";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  kurs\_completed\_table.DataSource = dt;  con.Close();  }  **else**  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT kurs\_completed.ID [ID],kurs.Name [Название курса],student.Familia||\" \" ||student.Name||\" \" ||student.Otchestvo [ФИО студента],grupa.Name\_grup [Группа],kurs\_completed.ocenka [Оценка] FROM kurs\_completed INNER JOIN kurs on kurs.ID\_kurs= kurs\_completed.ID\_kurs INNER JOIN student on student.ID\_student=kurs\_completed.ID\_student INNER JOIN grupa on student.ID\_grup=grupa.ID\_grup WHERE grupa.Name\_grup='" + fillter\_kurs\_completed.Text + "'";  SQLiteCommand cmd = **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);    adapter.Fill(dt);  kurs\_completed\_table.DataSource = dt;  con.Close();  }  }    }    **private** **void** **fillter\_categorie\_kurs\_completed\_SelectedIndexChanged**(**object** sender, EventArgs e)  {  **if** (fillter\_categorie\_kurs\_completed.Text == "По курсам")  {  fillter\_kurs\_completed.Items.Clear();  fillter\_kurs\_completed.Items.Add("Все курсы");  fillter\_kurs\_completed.SelectedIndex = **0**;  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT ID\_kurs, Name FROM kurs";  SQLiteCommand command = **new** SQLiteCommand(query, con);  DataTable dt = **new** DataTable();  SQLiteDataReader DR = command.ExecuteReader();    **while** (DR.Read())  {  fillter\_kurs\_completed.Items.Add(DR[**1**]);    }  }  **else**  {  fillter\_kurs\_completed.Items.Clear();  fillter\_kurs\_completed.Items.Add("Все группы");  fillter\_kurs\_completed.SelectedIndex = **0**;  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT ID\_grup, Name\_grup FROM grupa";  SQLiteCommand command = **new** SQLiteCommand(query, con);  DataTable dt = **new** DataTable();  SQLiteDataReader DR = command.ExecuteReader();    **while** (DR.Read())  {  fillter\_kurs\_completed.Items.Add(DR[**1**]);    }  }    }    **private** **void** **fillter\_kurs\_completed\_SelectedIndexChanged**(**object** sender, EventArgs e)  {  button1\_Click\_1(sender, e);  }    **private** **void** **add\_kurs\_complited\_btn\_Click**(**object** sender, EventArgs e)  {  add\_kurs\_completed windows = **new** add\_kurs\_completed();    DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {    button1\_Click\_1(sender, e);  }  }    **private** **void** **edit\_kurs\_completed\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(kurs\_completed\_table.CurrentRow.Cells[**0**].Value.ToString());  edit\_kurs\_completed windows = **new** edit\_kurs\_completed(id);  DialogResult result = windows.ShowDialog();  **if** (result == DialogResult.OK)  {  button1\_Click\_1(sender, e);  }  }    **private** **void** **delete\_kurs\_completed\_Click**(**object** sender, EventArgs e)  {  **int** id = **int**.Parse(kurs\_completed\_table.CurrentRow.Cells[**0**].Value.ToString());  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **var** res = MessageBox.Show("Удалить запись?", "Удаление", MessageBoxButtons.YesNo);  **if** (res == DialogResult.Yes)  {  **string** sql = "DELETE FROM kurs\_completed WHERE ID =" + id;    SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();    button1\_Click\_1(sender, e);  }  con.Close();  }    **private** **void** **groupBox6\_Enter**(**object** sender, EventArgs e)  {    }    **private** **void** **check\_grup\_CheckedChanged**(**object** sender, EventArgs e)  {  **if** (check\_grup.Checked)  {  grup\_check\_cmb.Items.Clear();  //grup\_check\_cmb.Items.Add("Все группы");    SQLiteConnection con1 = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con1.Open();  **string** query1 = "SELECT ID\_grup, Name\_grup FROM grupa";  SQLiteCommand command1 = **new** SQLiteCommand(query1, con1);    SQLiteDataReader DR1 = command1.ExecuteReader();    **while** (DR1.Read())  {  grup\_check\_cmb.Items.Add(DR1[**1**]);    }  grup\_check\_cmb.SelectedIndex = **0**;  grup\_check\_cmb.Visible = **true**;  }  **else**  {  grup\_check\_cmb.Visible = **false**;  }  }    **private** **void** **check\_kurs\_CheckedChanged**(**object** sender, EventArgs e)  {  **if** (check\_kurs.Checked)  {  kurs\_check\_cmb.Items.Clear();  //kurs\_check\_cmb.Items.Add("Все курсы");    SQLiteConnection con1 = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con1.Open();  **string** query1 = "SELECT ID\_kurs, Name FROM kurs";  SQLiteCommand command1 = **new** SQLiteCommand(query1, con1);    SQLiteDataReader DR1 = command1.ExecuteReader();    **while** (DR1.Read())  {  kurs\_check\_cmb.Items.Add(DR1[**1**]);    }  kurs\_check\_cmb.SelectedIndex = **0**;  kurs\_check\_cmb.Visible = **true**;  }  **else**  {  kurs\_check\_cmb.Visible = **false**;  }  }    **private** **void** **check\_ocenki\_CheckedChanged**(**object** sender, EventArgs e)  {  **if** (check\_ocenki.Checked)  {  ocenki\_check\_cmb.SelectedIndex = **0**;  ocenki\_check\_cmb.Visible = **true**;  }  **else**  {  ocenki\_check\_cmb.Visible = **false**;  }  }    **private** **void** **grup\_check\_cmb\_SelectedIndexChanged**(**object** sender, EventArgs e)  {      }    **private** **void** **main\_screen\_FormClosed**(**object** sender, FormClosedEventArgs e)  {  **this**.DialogResult = DialogResult.OK;    }    **private** **void** **ocenki\_check\_cmb\_SelectedIndexChanged**(**object** sender, EventArgs e)  {  **if** (ocenki\_check\_cmb.Text == "Интервал")  {    }  }    **private** **void** **button1\_Click\_2**(**object** sender, EventArgs e)  {  **bool** check = **false**;  **string** cmd = "";  **if** (check\_grup.Checked == **true**)  {  cmd ="WHERE grupa.Name\_grup='"+grup\_check\_cmb.Text+"'";  check = **true**;  }  **if** (check\_kurs.Checked == **true** && check == **false**)  {  cmd= "WHERE kurs.Name='" + kurs\_check\_cmb.Text+"'";  check = **true**;  }  **else** **if**(check\_kurs.Checked == **true** && check ==**true**)  {  cmd = cmd + "and kurs.Name='" + kurs\_check\_cmb.Text + "'";  }  **if** (check\_ocenki.Checked == **true** && check == **false**)  {  cmd = "WHERE kurs\_completed.ocenka='" + ocenki\_check\_cmb.Text + "'";  check = **true**;  }  **else** **if** (check\_ocenki.Checked == **true** && check == **true**)  {  cmd = cmd + "and kurs\_completed.ocenka='" + ocenki\_check\_cmb.Text + "'";  }    SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** query = "SELECT kurs\_completed.ID [ID],kurs.Name [Название курса],student.Familia||\" \" ||student.Name||\" \" ||student.Otchestvo [ФИО студента],grupa.Name\_grup [Группа],kurs\_completed.ocenka [Оценка] FROM kurs\_completed INNER JOIN kurs on kurs.ID\_kurs= kurs\_completed.ID\_kurs INNER JOIN student on student.ID\_student=kurs\_completed.ID\_student INNER JOIN grupa on student.ID\_grup=grupa.ID\_grup "+cmd;  SQLiteCommand command= **new** SQLiteCommand(query, con);    DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(command);    adapter.Fill(dt);  otchet\_table.DataSource = dt;  con.Close();  }  }  } |



*Рисунок 3 окно «Добавить преподавателя»*

add\_prepodavatel.cs-форма для добавления преподавателя

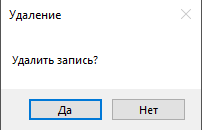
|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **add\_prepodavatel** : Form  {  **public** **add\_prepodavatel**()  {  InitializeComponent();  }  **private** **void** **label2\_Click**(**object** sender, EventArgs e)  {  }  **private** **void** **save\_btn\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "INSERT INTO prepodavatel (Familia,Name,Otchestvo) VALUES ('" + add\_familia.Text + "', '" + add\_name.Text + "','" + add\_otchestvo.Text + "')";  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  **private** **void** **exit\_btn\_Click**(**object** sender, EventArgs e)  {  **this**.Close();  }  }  } |



*Рисунок 4 окно «Изменения преподавателя»*

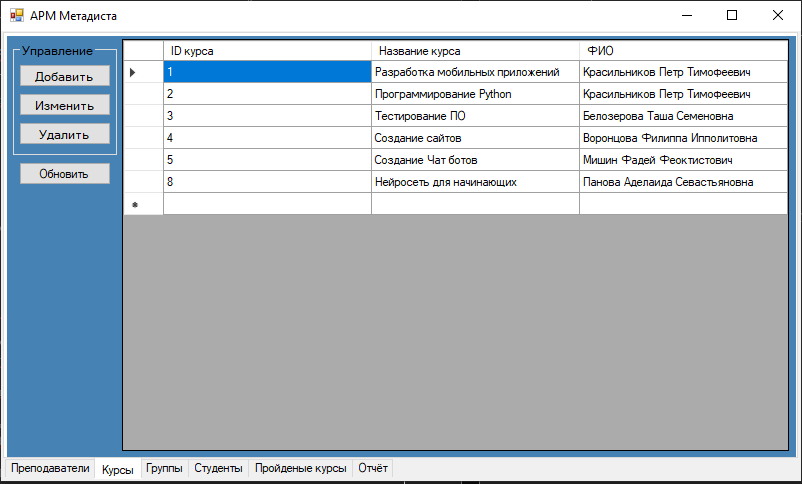
edit\_prepodavatel.cs-форма для изменения преподавателя

|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SqlClient**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **edit\_prepodavatel** : Form  {  **int** id { **get**; **set**; }  **public** **edit\_prepodavatel**(**int** id\_con)  {  InitializeComponent();  id = id\_con;  SQLiteConnection con=**new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "SELECT Familia,Name,Otchestvo FROM prepodavatel WHERE ID\_prepodavatel=" + id;  SQLiteCommand cmd = **new** SQLiteCommand(sql,con);  using(SQLiteDataReader reader = cmd.ExecuteReader())  {  **while** (reader.Read())  {  edit\_familia.Text = reader.GetString(**0**);  edit\_name.Text = reader.GetString(**1**);  edit\_otchestvo.Text = reader.GetString(**2**);  }  }  }  **private** **void** **save\_btn\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "UPDATE prepodavatel SET Familia='" + edit\_familia.Text + "',Name='" + edit\_name.Text + "',Otchestvo='" + edit\_otchestvo.Text + "' WHERE ID\_prepodavatel=" + id;  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  }  } |

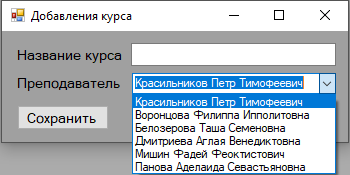


*Рисунок 5 окно «Удаления»*

Окно удаления идентично для всех таблиц



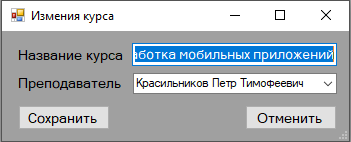
*Рисунок 6 главное окно программы «Таблица курсы» нажата кнопка обновить*



*Рисунок 7 окно «Добавления курса»*

add\_kurs.cs-форма для добавления курса

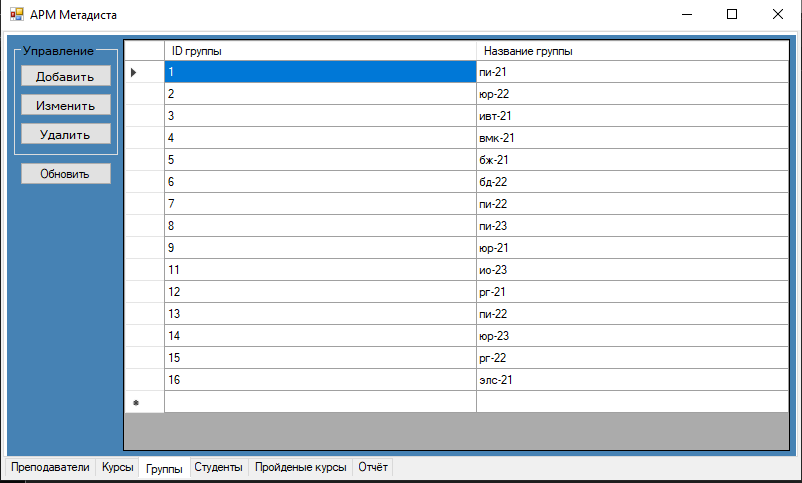
|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **add\_kurs** : Form  {  **public** **add\_kurs**()  {  InitializeComponent();  SQLiteConnection connection = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  connection.Open();  **string** sql = "SELECT prepodavatel.ID\_prepodavatel [ID],prepodavatel.Familia||\" \"|| prepodavatel.Name||\" \"||prepodavatel.otchestvo [name] FROM prepodavatel";  SQLiteCommand command = **new** SQLiteCommand(sql, connection);  DataTable dt =**new** DataTable();  SQLiteDataAdapter adapter=**new** SQLiteDataAdapter(command);  adapter.Fill(dt);  kurs\_prepodovatel.DataSource = dt;  kurs\_prepodovatel.DisplayMember = "name";  kurs\_prepodovatel.ValueMember = "ID";  connection.Close();  }  **private** **void** **save\_btn\_Click**(**object** sender, EventArgs e)  {      SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "INSERT INTO kurs (Name, ID\_prepodavatel) VALUES ('" + name\_kurs.Text + "', '" + kurs\_prepodovatel.SelectedValue.ToString() + "')";  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();    }  **private** **void** **button1\_Click**(**object** sender, EventArgs e)  {  **this**.Close();  }  **private** **void** **add\_kurs\_Load**(**object** sender, EventArgs e)  {  }  }  } |



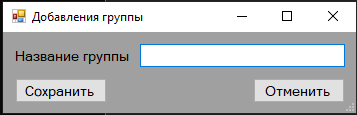
*Рисунок 8 окно «Изменения курса»*

edit\_kurs.cs-форма для изменения курса

|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **edit\_kurs** : Form  {  **int** id { **get**; **set**; }  **public** **edit\_kurs**(**int** id\_con)  {  InitializeComponent();  id = id\_con;  SQLiteConnection connection = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  connection.Open();  **string** sql = "SELECT prepodavatel.ID\_prepodavatel [ID],prepodavatel.Familia||\" \"|| prepodavatel.Name||\" \"||prepodavatel.otchestvo [name] FROM prepodavatel";  SQLiteCommand command = **new** SQLiteCommand(sql, connection);  DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(command);  adapter.Fill(dt);  edit\_kurs\_prepodovatel.DataSource = dt;  edit\_kurs\_prepodovatel.DisplayMember = "name";  edit\_kurs\_prepodovatel.ValueMember = "ID";  **string** sql1 = "SELECT Name, ID\_prepodavatel FROM kurs WHERE ID\_kurs= "+id;  SQLiteCommand command1 = **new** SQLiteCommand( sql1, connection);  using(SQLiteDataReader reader = command1.ExecuteReader())  {  **while** (reader.Read())  {  edit\_name\_kurs.Text = reader.GetString(**0**);  edit\_kurs\_prepodovatel.SelectedValue = reader.GetDecimal(**1**);  }  }  }  **private** **void** **save\_btn\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "UPDATE kurs SET Name='" + edit\_name\_kurs.Text + "',ID\_prepodavatel='" + edit\_kurs\_prepodovatel.SelectedValue.ToString() + "' WHERE ID\_kurs=" + id;  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  **private** **void** **btn\_close\_Click**(**object** sender, EventArgs e)  {  **this**.Close();  }  **private** **void** **edit\_kurs\_Load**(**object** sender, EventArgs e)  {  }  }  } |



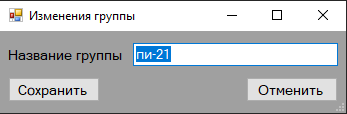
*Рисунок 9 главное окно программы «Таблица группы» нажата кнопка обновить*



*Рисунок 10- окно «Добавления группы»*

Add\_grup.cs-форма для добавления группы

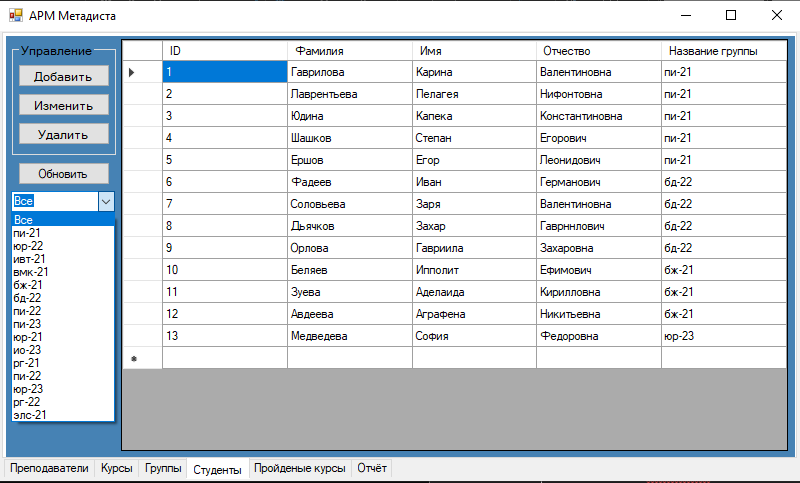
|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **add\_grup** : Form  {  **public** **add\_grup**()  {  InitializeComponent();  }  **private** **void** **exit\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  **this**.Close();  }  **private** **void** **save\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "INSERT INTO grupa (Name\_grup) VALUES ('" + name\_grup.Text + "')";  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  }  } |



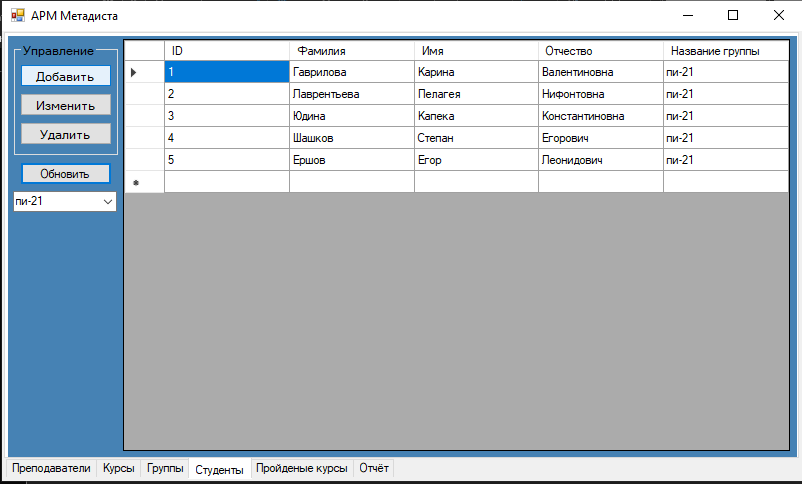
*Рисунок 11- окно «Изменения группы»*

edit\_grup.cs-форма для изменения группы

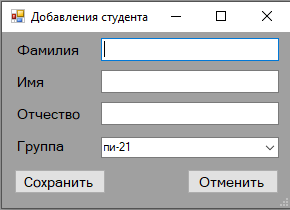
|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **edit\_grup** : Form  {  **int** id { **get**; **set**; }  **public** **edit\_grup**(**int** id\_con)  {  InitializeComponent();  id = id\_con;  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "SELECT Name\_grup FROM grupa WHERE ID\_grup=" + id;  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  **using** (SQLiteDataReader reader = cmd.ExecuteReader())  {  **while** (reader.Read())  {  name\_grup.Text = reader.GetString(**0**);    }  }  }  **private** **void** **edit\_grup\_Load**(**object** sender, EventArgs e)  {  }  **private** **void** **save\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "UPDATE grupa SET Name\_grup='" + name\_grup.Text + "' WHERE ID\_grup=" + id;  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  **private** **void** **exit\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  **this**.Close();  }  }  } |



*Рисунок 12 главное окно программы «Таблица студенты» нажата кнопка обновить с параметром «все»*



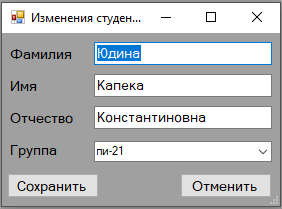
*Рисунок 12 главное окно программы «Таблица студенты» нажата кнопка обновить с параметром «пи-21»*



*Рисунок 13- окно «Добавления студента»*

Add\_student.cs-форма для добавления студента

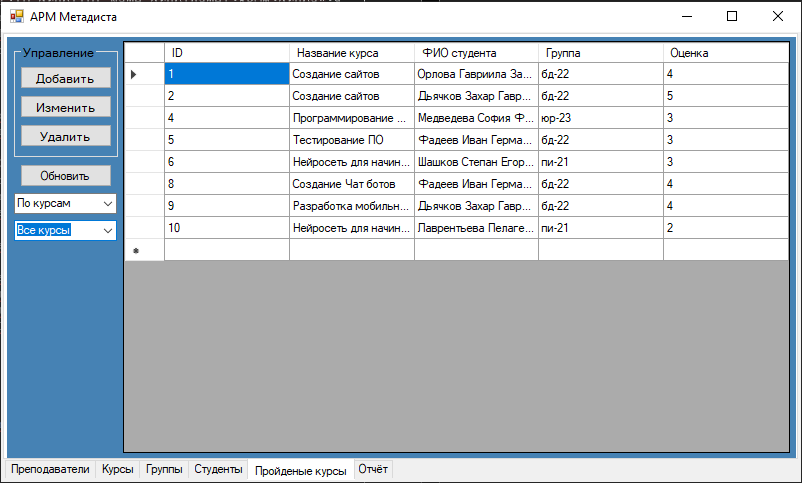
|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **add\_student** : Form  {  **public** **add\_student**()  {  InitializeComponent();  SQLiteConnection connection = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  connection.Open();  **string** sql = "SELECT ID\_grup[ID],Name\_grup[name] FROM grupa";  SQLiteCommand command = **new** SQLiteCommand(sql, connection);  DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(command);  adapter.Fill(dt);  grup\_box.DataSource = dt;  grup\_box.DisplayMember = "name";  grup\_box.ValueMember = "ID";  connection.Close();  }  **private** **void** **label4\_Click**(**object** sender, EventArgs e)  {  }  **private** **void** **textBox1\_TextChanged**(**object** sender, EventArgs e)  {  }  **private** **void** **add\_student\_Load**(**object** sender, EventArgs e)  {  }  **private** **void** **save\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "INSERT INTO student (Familia,Name,Otchestvo,ID\_grup) VALUES ('" + add\_familia\_student.Text + "', '" + add\_name\_student.Text + "','" + add\_otchestvo\_student.Text + "','" + grup\_box.SelectedValue.ToString() + "')";  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  **private** **void** **exit\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  **this**.Close();  }  }  } |

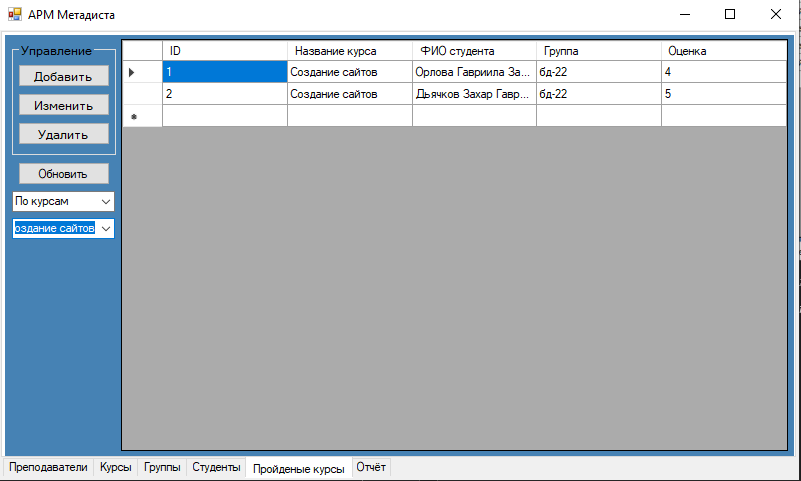


*Рисунок 13- окно «Изменения студента»*

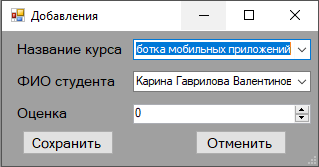
Edit\_student.cs-форма для изменения студента

|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **edit\_student** : Form  {  **int** id { **get**; **set**; }  **public** **edit\_student**(**int** id\_con)  {  id = id\_con;  InitializeComponent();  SQLiteConnection connection = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  connection.Open();  **string** sql = "SELECT ID\_grup[ID],Name\_grup[name] FROM grupa";  SQLiteCommand command = **new** SQLiteCommand(sql, connection);  DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(command);  adapter.Fill(dt);  grup\_box.DataSource = dt;  grup\_box.DisplayMember = "name";  grup\_box.ValueMember = "ID";  **string** sql1 = "SELECT Familia,Name,Otchestvo,ID\_grup FROM student WHERE ID\_student= " + id;  SQLiteCommand command1 = **new** SQLiteCommand(sql1, connection);  **using** (SQLiteDataReader reader = command1.ExecuteReader())  {  **while** (reader.Read())  {  edit\_familia\_student.Text = reader.GetString(**0**);  edit\_name\_student.Text = reader.GetString(**1**);  edit\_otchestvo\_student.Text = reader.GetString(**2**);  grup\_box.SelectedValue = reader.GetDecimal(**3**);  }  }  }  **private** **void** **edit\_student\_Load**(**object** sender, EventArgs e)  {  }  **private** **void** **exit\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  **this**.Close();  }  **private** **void** **save\_btn\_grup\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "UPDATE student SET Familia='" + edit\_familia\_student.Text + "',Name='" + edit\_name\_student.Text + "',Otchestvo='" + edit\_otchestvo\_student.Text + "',ID\_grup='" + grup\_box.SelectedValue.ToString() + "' WHERE ID\_student=" + id;  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  }  } |

 *Рисунок 14 главное окно программы «Таблица проденные курсы» нажата кнопка обновить сортировка по курсам «Все курсы»*



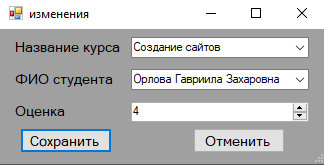
*Рисунок 15 главное окно программы «Таблица проденные курсы» нажата кнопка обновить сортировка по курсам «Создание сайтов»*



*Рисунок 16- окно «Добавление пройдённого курса»*

Add\_kurs\_completed.cs-форма для добавления пройдённого курса

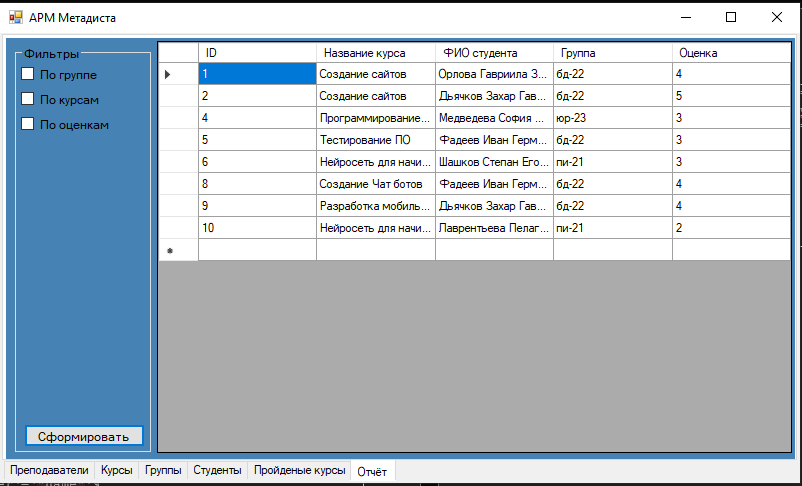
|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **add\_kurs\_completed** : Form  {  **public** **add\_kurs\_completed**()  {  InitializeComponent();  SQLiteConnection con = **new** SQLiteConnection("data source=../../../praktika\_BD.db");  con.Open();  **string** name\_kurs = "SELECT ID\_kurs [ID], Name FROM kurs";  **string** name\_tovara = "SELECT ID\_student [ID],Name||\" \"||Familia||\" \"||Otchestvo[name] FROM student";  SQLiteCommand cmd = **new** SQLiteCommand(name\_kurs, con);  DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);  adapter.Fill(dt);  name\_kurs\_cmb.DataSource = dt;  name\_kurs\_cmb.DisplayMember = "name";  name\_kurs\_cmb.ValueMember = "ID";  SQLiteCommand cmd1 = **new** SQLiteCommand(name\_tovara, con);  DataTable dt1 = **new** DataTable();  SQLiteDataAdapter adapter1 = **new** SQLiteDataAdapter(cmd1);  adapter1.Fill(dt1);  FIO\_cmb.DataSource = dt1;  FIO\_cmb.DisplayMember = "name";  FIO\_cmb.ValueMember = "ID";  }    **private** **void** **save\_btn\_kurs\_completed\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection("data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "INSERT INTO kurs\_completed (ID\_kurs,ID\_student,ocenka) VALUES ('" + name\_kurs\_cmb.SelectedValue.ToString() + "', '" + FIO\_cmb.SelectedValue.ToString() + "','" + ocenka.Value.ToString() + "')";  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  }  } |



*Рисунок 17- окно «Изменения пройдённого курса»*

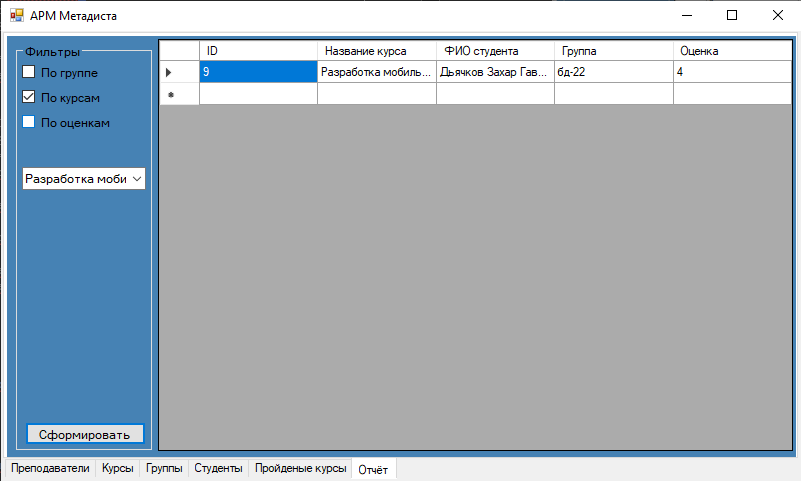
edit\_kurs\_completed.cs-форма для добавления пройдённого курса

|  |
| --- |
| **using** **System**;  **using** **System.Collections.Generic**;  **using** **System.ComponentModel**;  **using** **System.Data**;  **using** **System.Data.SQLite**;  **using** **System.Drawing**;  **using** **System.Linq**;  **using** **System.Text**;  **using** **System.Threading.Tasks**;  **using** **System.Windows.Forms**;  **namespace** **praktika**  {  **public** **partial** **class** **edit\_kurs\_completed** : Form  {  **int** id { **get**; **set**; }  **public** **edit\_kurs\_completed**(**int** id\_con)  {  id= id\_con;  InitializeComponent();  SQLiteConnection con = **new** SQLiteConnection("data source=../../../praktika\_BD.db");  con.Open();  **string** name\_kurs = "SELECT ID\_kurs [ID\_kurs], Name FROM kurs";  **string** name\_tovara = "SELECT ID\_student [ID],Familia||\" \"||Name||\" \"||Otchestvo[name] FROM student";  SQLiteCommand cmd = **new** SQLiteCommand(name\_kurs, con);  DataTable dt = **new** DataTable();  SQLiteDataAdapter adapter = **new** SQLiteDataAdapter(cmd);  adapter.Fill(dt);    name\_kurs\_cmb.DataSource = dt;  name\_kurs\_cmb.DisplayMember = "Name";  name\_kurs\_cmb.ValueMember = "ID\_kurs";  SQLiteCommand cmd1 = **new** SQLiteCommand(name\_tovara, con);  DataTable dt1 = **new** DataTable();  SQLiteDataAdapter adapter1 = **new** SQLiteDataAdapter(cmd1);  adapter1.Fill(dt1);  FIO\_cmb.DataSource = dt1;  FIO\_cmb.DisplayMember = "name";  FIO\_cmb.ValueMember = "ID";  **string** sql = "SELECT ID,ID\_kurs,ID\_student,ocenka FROM kurs\_completed WHERE ID=" + id;  SQLiteCommand cmd2 = **new** SQLiteCommand(sql, con);  using(SQLiteDataReader reader = cmd2.ExecuteReader())  {  **while** (reader.Read())  {  name\_kurs\_cmb.SelectedValue = reader.GetDecimal(**1**);    FIO\_cmb.SelectedValue= reader.GetDecimal(**2**);  ocenka.Value=reader.GetDecimal(**3**);  }  }  con.Close();  }  **private** **void** **edit\_kurs\_completed\_Load**(**object** sender, EventArgs e)  {  }  **private** **void** **save\_btn\_kurs\_completed\_Click**(**object** sender, EventArgs e)  {  SQLiteConnection con = **new** SQLiteConnection(@"data source=../../../praktika\_BD.db");  con.Open();  **string** sql = "UPDATE kurs\_completed SET ID\_kurs='" + name\_kurs\_cmb.SelectedValue.ToString() + "',ID\_student='" + FIO\_cmb.SelectedValue.ToString() + "',ocenka='"+ocenka.Value.ToString()+"' WHERE ID=" + id;  SQLiteCommand cmd = **new** SQLiteCommand(sql, con);  cmd.ExecuteNonQuery();  **this**.DialogResult = DialogResult.OK;  con.Close();  **this**.Close();  }  }  } |



*Рисунок 18 главное окно программы «Таблица проденные курсы» нажата кнопка обновить без параметров*

В разделе отчёты мы можем вывести таблицу по различным фильтрам. Такими как по группе, по курсу, и по оценки которую получили студенты за этот курс все параметры можно использовать одновременно или по отдельности.



*Рисунок 19 главное окно программы «Таблица проденные курсы» нажата кнопка обновить с фильтрацией по курам «Разработка мобильных приложений»*

# ЗАКЛЮЧЕНИЕ

Таким образом, создание автоматизированного рабочего места сотрудника – сложная и кропотливая работа, требующая анализа и учета многих моментов при проектировании и реализации. Автоматизация позволяет существенно повысить эффективность и производительность работы методиста. Данный проект неидеален и ожидает некоторых доработок, чтобы была возможность внедрения подобного АРМ. Однако, для первого созданного АРМ на языке C# - весьма неплохой и рабочий продукт.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://learn.microsoft.com/ru-ru/dotnet/csharp/> - Документация по C# - [Электронный ресурс] - Дата обращения - 15.07.2023
2. <https://docs.microsoft.com/ru-ru/dotnet/csharp/> - Документация C# - [Электронный ресурс] – Дата обращения – 15.07.2023
3. <https://proglib.io/p/5-shagov-dlya-sozdaniya-prostoy-formy-vhoda-na-c-2020-03-24> - Инструкция по создания формы входа - [Электронный ресурс] - Дата обращения - 15.07.2023
4. <https://www.youtube.com/watch?v=gp2rA0rgq_0&list=PL0lO_mIqDDFWOMqSKFaLypANf1W7-o87q> – Блог Гоша Дударь - [Электронный ресурс] - Дата обращения – 15.07.2023
5. <https://www.bestprog.net/ru/2016/01/08/005-пример-программирования-события-в-с/> - Инструкция по программированию в C# - [Электронный ресурс] - Дата обращения - 15.07.2023
6. <https://www.youtube.com/watch?v=l77oxzJUhMQ&list=RDCMUCtLKO1Cb2GVNrbU7Fi0pM0w&start_radio=1&rv=l77oxzJUhMQ&t=446> – Блог #SimpleCode - [Электронный ресурс] - Дата обращения - 15.07.2023
7. <https://www.cyberforum.ru/> - Форум - [Электронный ресурс] - Дата обращения - 15.07.2023
8. <https://metanit.com/sharp/tutorial/1.2.php> - Основы программирования на C# - [Электронный ресурс] - Дата обращение - 15.07.2023