Одеський національний політехнічний університет

Кафедра «Комп'ютеризовані системи управління»

**КУРСОВА РОБОТА**

з дисципліни «Програмування та теорія алгоритмів»

на тему: «Создание базы данных»

Варіант 3

Студента 1 курсу, групи АТ–181

спеціальності «Автоматизація та комп'ютерно-інтегровані технології»

Прізвище: Босюк Е. М.

Керівник: доц. Сперанський В.О.

Національна шкала:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

м. Одеса – 2019 рік

# ВВЕДЕНИЕ

Постановка задачи

1. Спроектировать структуру информационной базы для своего варианта задания.
2. Создать основную таблицу информационной базы в виде динамического массива структур (использование списка из элементов типа структура оценивается выше).
3. Предусмотреть функции контроля корректности всех вводимых данных (например, неотрицательный возраст, дата продажи не ранее, чем 100 лет назад и т.п.).
4. Заполнить основную таблицу не менее чем 20 записями.
5. Обеспечить сохранение и загрузку данных во внешний файл/из файла.
6. Реализовать меню пользователя, соответствующие функции которого обеспечивают:
7. создание информационной базы той предметной области, которая определена постановкой задачи;
8. добавление новых записей в базу данных (с контролем корректности вводимых значений);
9. поиск записей по *ключевому* полю;
10. редактирование записей базы данных;
11. удаление ненужных записей;
12. выдачу отчётов по запросу.
13. Выполнить оригинальную обработку, указанную в постановке конкретной задачи для вашего варианта.
14. Организовать ввод пароля при выполнении операций изменения данных.
15. Добавить справочную систему пользователя по созданному приложению.
16. Подготовить презентацию по разработанному приложению.

# 

1. Создать базу данных, содержащую результаты экзаменационной сессии студентов курса. Каждая запись хранит:

* фамилию, инициалы студента;
* *номер зачётки;*
* наименование факультета;
* курс;
* результаты сессии для каждого студента; представляет массив структур из 5 элементов; структура включает:

- название предмета;

- фамилию преподавателя;

- оценку;

*Предусмотреть:*

а) выдачу справки о студентах - отличниках;

б) выдачу справки о студентах, имеющих двойки;

в) выдачу справки о названии дисциплины, по которой наибольшее число студентов получили двойки;

г) удаление всех записей, у которых есть двойка хотя бы по одному предмету.

РАЗДЕЛ 1

ПРОЕКТИРОВАНИЕ И ИХ РЕАЛИЗАЦИЯ

Календарное планирование проекта

Поэтапное календарное планирование для выполняемого курсового проекта представлено в табл. 1.1.

Таблица 1.1 – Календарный план курсового проектирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование | Содержание | Дата окончания |
| 1 | Начало | Изучение постановочного задания, предметной области | 26.04.2019 |
| 2 | Теоретическая подготовка | Изучение литературы по дисциплины | 30.04.2019 |
| 3 | Проектирование модулей проекта | Разработка компонентов программы, их функций, свойств, а также связь между ними. | 30.04.2019 |
| 4 | Разработка программной реализации | Написание непосредственно самой программы с использованием изученных теоретических знаний | 13.05.2019 |
| 5 | Тестирование | Проверка созданного приложения на предмет некорректного поведения при вводе ошибочных значений, определение уровня эргономики, устойчивости разработки | 23.05.2019 |
| 6 | Написание документации | Написание документации с описанием выполненной | 24.05.2017 |

Описание программы и её работы

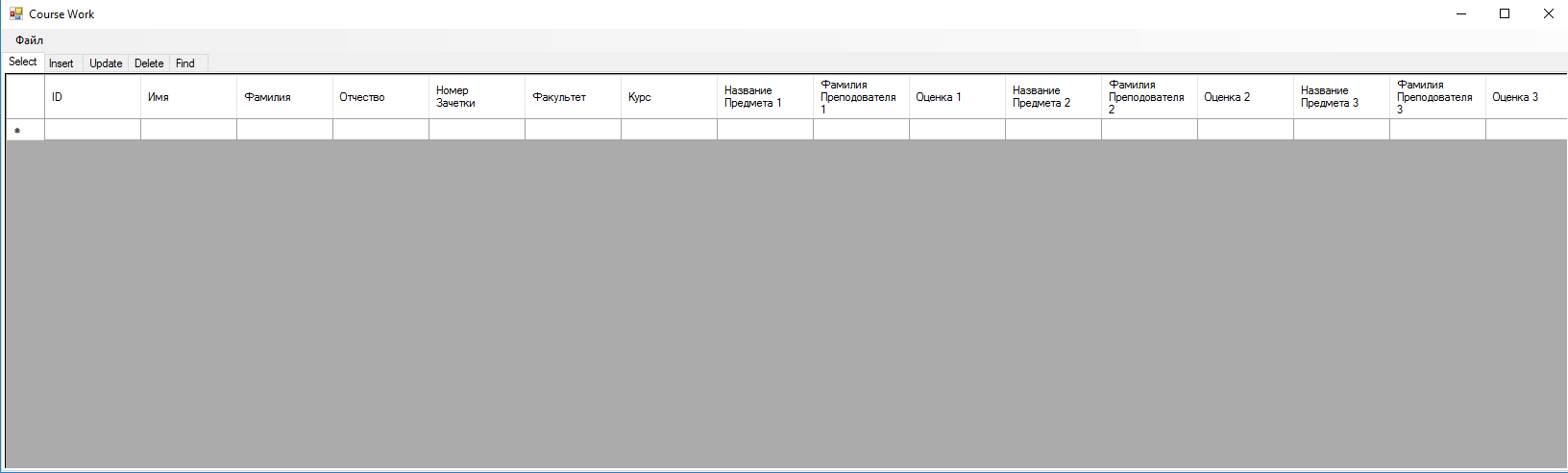
В данной курсовой работе мы изучали базу данных (в моем случае Sql). Для реализации моей программы был использован язык программирования c# и Windows Form в visual studio.

Первое что я создавал в программе был пользовательский интерфейс в нем я решил реализовать меню при помощи MenuStrip с его помощью я создал 5 основных вкладок для задания а именно Select, Insert, Update, Delete,Find

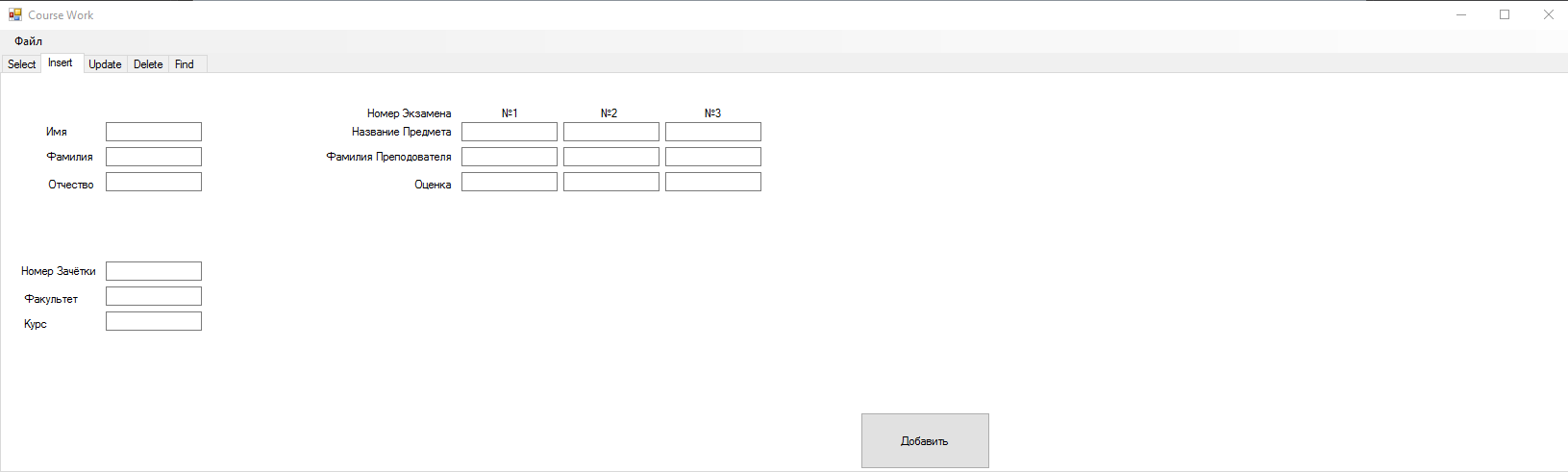


Сейчас я подробно функции каждой из вкладок.

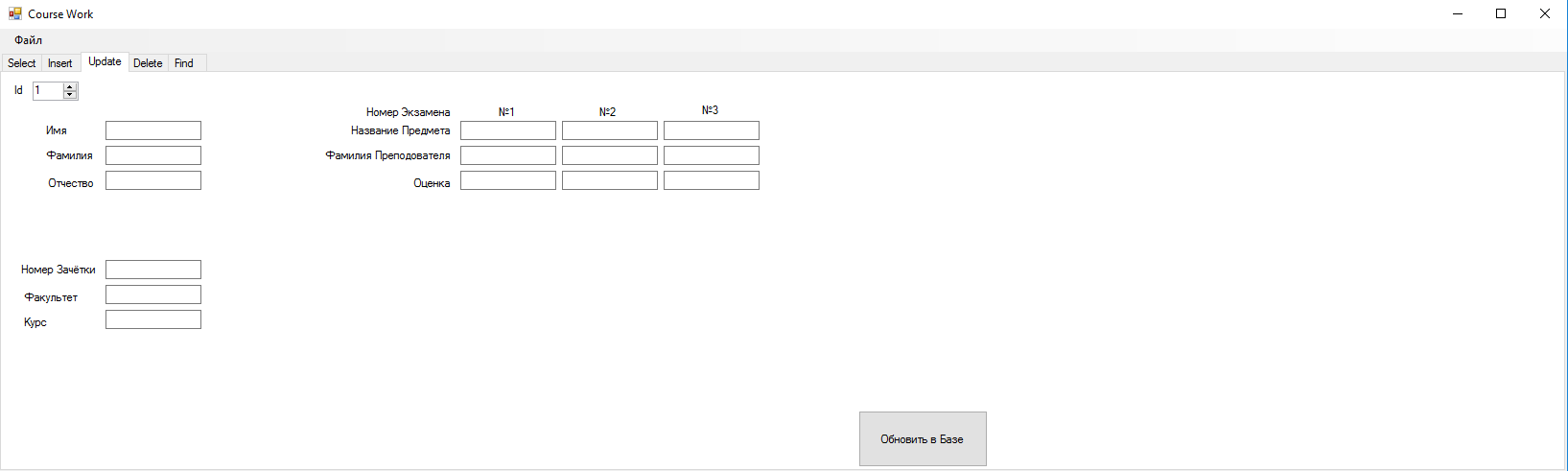
Select – Служит для отображения базы данных для это используется DataGridView.



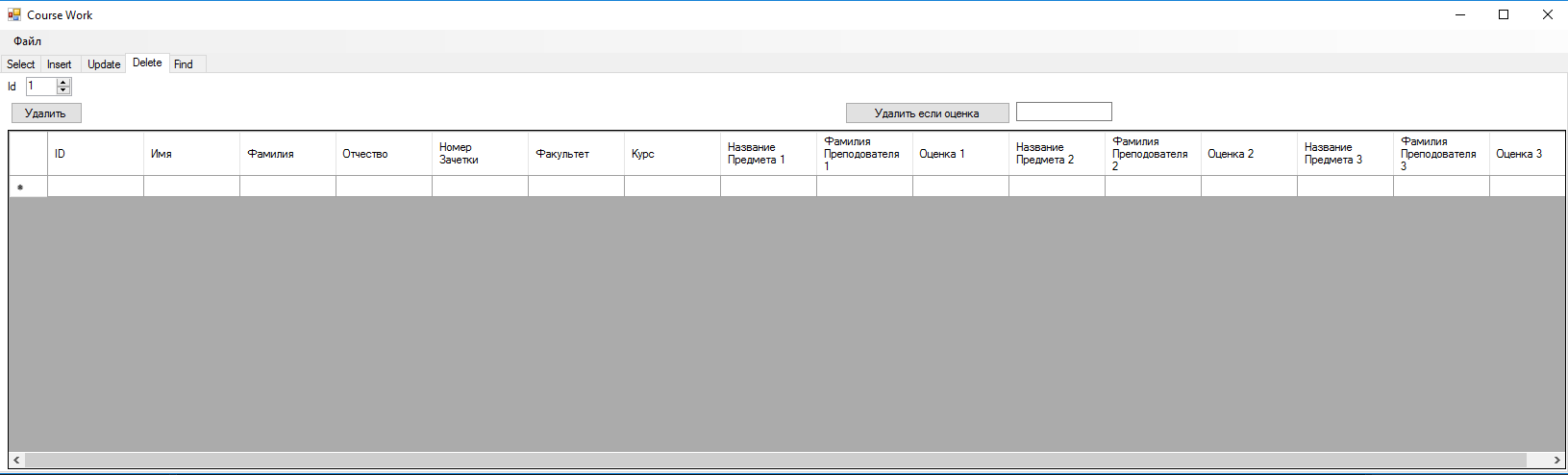
Insert – Служит для добавления в бд новых элементов.



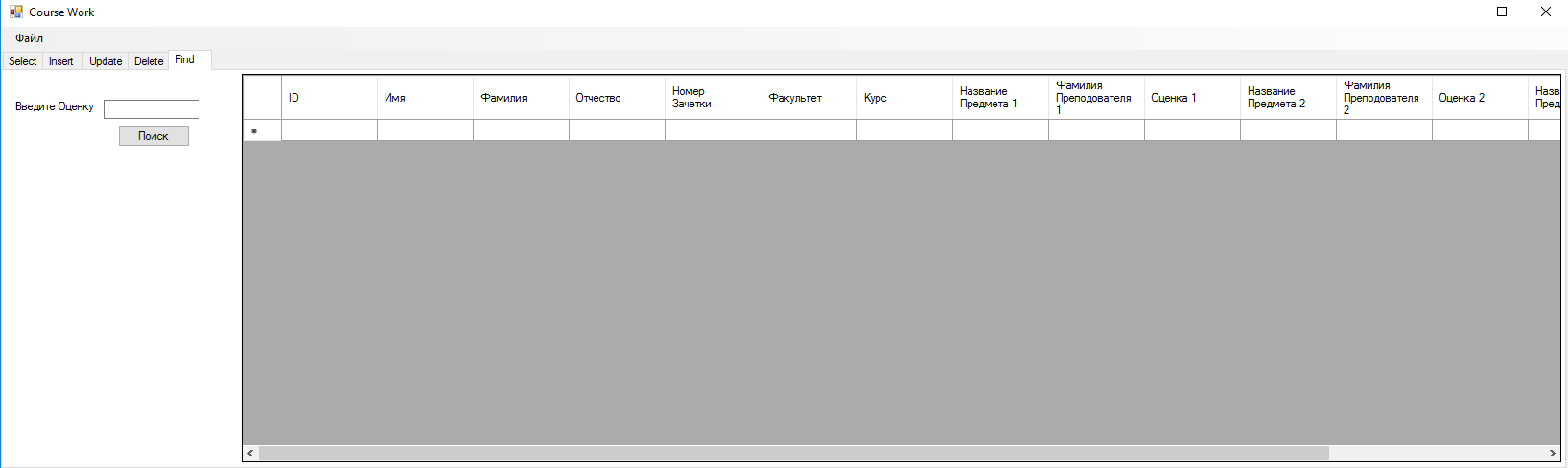
Update – Служит для обновления информации об элементу по его ID

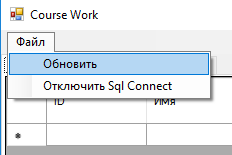


Delete –Служит для удаления Элементов бд по id или при наличии определенной оценки для удобства используется DataGridView

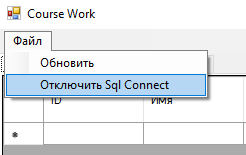


Find – Служит для поиска студентов по определенной оценки и выдачи справки о них для удобства используется DataGridView

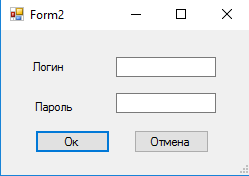


Так же после любого действия с программой требуется обновить базу данных для получения обновленной информации

На случай возникновения ошибок из-за не закрытого подключения есть функция Отключить Sql Connect



Так же при запуске программы потребуется ввести Логин и Пароль для входа в программу ими являются слово admin т.е. Логин admin и Пароль admin



Данная программа предназначена для удобного управления и работой с базой данных для преподавателей Университета.

Функциональная Часть программы

В данном пункте я рассмотрю свою работу с точки зрения кода и его реализации и начну с вкладки Select

SqlConnection sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\Курсовая\CourseWork\CourseWork\DatabaseCourseDAta.mdf;Integrated Security=True");

public string reload = "SELECT \* FROM [Students]";

public Form1()

{

InitializeComponent();

LoadData(reload);

Form2 PF = new Form2();

if (PF.ShowDialog() == DialogResult.Cancel)

Application.Exit();

}

private async void LoadData(string reload)

{

await sqlConnection.OpenAsync();

string reloading = reload;

SqlCommand command = new SqlCommand(reloading, sqlConnection);

SqlDataReader reader = await command.ExecuteReaderAsync();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[16]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

data[data.Count - 1][6] = reader[6].ToString();

data[data.Count - 1][7] = reader[7].ToString();

data[data.Count - 1][8] = reader[8].ToString();

data[data.Count - 1][9] = reader[9].ToString();

data[data.Count - 1][10] = reader[10].ToString();

data[data.Count - 1][11] = reader[11].ToString();

data[data.Count - 1][12] = reader[12].ToString();

data[data.Count - 1][13] = reader[13].ToString();

data[data.Count - 1][14] = reader[14].ToString();

data[data.Count - 1][15] = reader[15].ToString();

}

reader.Close();

sqlConnection.Close();

foreach (string[] s in data)

{

dataGridView1.Rows.Add(s);

dataGridView2.Rows.Add(s);

dataGridView3.Rows.Add(s);

}

}

В данном отрезке кода мы создаем подключение к нашей базе данных через SqlConnection , после чего используем функцию загрузки программы которая вызывает Form2 ,а т.е Логин и Пароль нашей программы. После чего мы создаем функцию LoadData которая обновляет данные во всех DataGridView при запуске программы .

public Form2()

{

InitializeComponent();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

if (textBox1.Text == "admin" && textBox2.Text == "admin")

{

Form2 s = new Form2();

s.Show();

this.Hide();

}

else

{

MessageBox.Show("Пароль или логин не верены!");

textBox1.Clear();

textBox2.Clear();

}

}

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

{

Application.Exit();

}

Данных фрагмент кода являет собой Form2, в нем написаны методы при нажатии на кнопку ОК и ОТМЕНА которые или проверяют введеные данные или же закрывают программу.

Таким Образом при запуске программы мы видим окно входа и только после него мы попадаем в программу на вкладку Select в которой уже загружена бд.

Следующей описанной функцией будет Insert

private void botton\_insert\_Click(object sender, EventArgs e)

{

sqlConnection.Open();

if (labelInsertError.Visible)

{

labelInsertError.Visible = false;

}

if (!string.IsNullOrEmpty(textBoxName.Text) && !string.IsNullOrWhiteSpace(textBoxName.Text) &&

!string.IsNullOrEmpty(textBoxSecondName.Text) && !string.IsNullOrWhiteSpace(textBoxSecondName.Text) &&

!string.IsNullOrEmpty(textBoxThridName.Text) && !string.IsNullOrWhiteSpace(textBoxThridName.Text) &&

!string.IsNullOrEmpty(textBoxRecordBook.Text) && !string.IsNullOrWhiteSpace(textBoxRecordBook.Text) &&

!string.IsNullOrEmpty(textBoxFaculty.Text) && !string.IsNullOrWhiteSpace(textBoxFaculty.Text) &&

!string.IsNullOrEmpty(textBoxCourse.Text) && !string.IsNullOrWhiteSpace(textBoxCourse.Text))

{

SqlCommand command = new SqlCommand("INSERT INTO [Students] (Name, SName,TName, RecordBookNumber, Faculty, Course,SubjectName1,TheacherSurName1,Mark1,SubjectName2,TheacherSurName2,Mark2,SubjectName3,TheacherSurName3,Mark3)" +

"VALUES(@Name, @SName, @TName, @RecordBookNumber, @Faculty, @Course ,@SubjectName1,@TheacherSurName1,@Mark1,@SubjectName2,@TheacherSurName2,@Mark2,@SubjectName3,@TheacherSurName3,@Mark3)", sqlConnection);

command.Parameters.AddWithValue("SubjectName1", textBoxSubject1.Text);

command.Parameters.AddWithValue("TheacherSurName1", textBoxTeacherSurname1.Text);

command.Parameters.AddWithValue("Mark1", textBoxMark1.Text);

command.Parameters.AddWithValue("SubjectName2", textBoxSubject2.Text);

command.Parameters.AddWithValue("TheacherSurName2", textBoxTeacherSurname2.Text);

command.Parameters.AddWithValue("Mark2", textBoxMark2.Text);

command.Parameters.AddWithValue("SubjectName3", textBoxSubject3.Text);

command.Parameters.AddWithValue("TheacherSurName3", textBoxTeacherSurname3.Text);

command.Parameters.AddWithValue("Mark3", textBoxMark3.Text);

command.Parameters.AddWithValue("Name", textBoxName.Text);

command.Parameters.AddWithValue("SName", textBoxSecondName.Text);

command.Parameters.AddWithValue("TName", textBoxThridName.Text);

command.Parameters.AddWithValue("RecordBookNumber", textBoxRecordBook.Text);

command.Parameters.AddWithValue("Faculty", textBoxFaculty.Text);

command.Parameters.AddWithValue("Course", textBoxCourse.Text);

try

{

command.ExecuteNonQuery();

sqlConnection.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);

sqlConnection.Close();

}

}

else

{

labelInsertError.Visible = true;

labelInsertError.Text = "Ошибка Вы не правильно заполнили поля";

sqlConnection.Close();

}}

В данном фрагменте мы видим код кнопки добавить на вкладке Insert. Для начала мы подключаемcя к sql после мы делаем невидимой строку ошибки. После чего делаем проверку пустых полей в textbox из которых берем данные если они пустые тогда меняем текст в строке на ошибку и делаем её видимой. Иначе создаем SqlCommand в котором описано добавление элеметов в бд. После используем данную команду считывая данные из нужных нам textbox . После этого методом try{},catch{} проверяем можем ли мы выполнить sql-выражение и вернуть строки в таблицу. Если нет то мы получим ошибку указывающую на проблему при исполнении .

Дальше опишем Update

sqlConnection.Open();

if (labelUpdError.Visible)

{

labelUpdError.Visible = false;

}

if (!string.IsNullOrEmpty(textBoxUpdName.Text) && !string.IsNullOrWhiteSpace(textBoxUpdName.Text) &&

!string.IsNullOrEmpty(textBoxUpdSecondName.Text) && !string.IsNullOrWhiteSpace(textBoxUpdSecondName.Text) &&

!string.IsNullOrEmpty(textBoxUpdThridName.Text) && !string.IsNullOrWhiteSpace(textBoxUpdThridName.Text) &&

!string.IsNullOrEmpty(textBoxUpdRecordBook.Text) && !string.IsNullOrWhiteSpace(textBoxUpdRecordBook.Text) &&

!string.IsNullOrEmpty(textBoxUpdFaculty.Text) && !string.IsNullOrWhiteSpace(textBoxUpdFaculty.Text) &&

!string.IsNullOrEmpty(textBoxUpdCourse.Text) && !string.IsNullOrWhiteSpace(textBoxUpdCourse.Text))

{

SqlCommand command = new SqlCommand("UPDATE [Students] SET [Name] = @Name,[SName] = @SName,[TName] = @TName,[RecordBookNumber] = @RecordBookNumber,[Faculty] = @Faculty,[Course] = @Course WHERE [Id] = @Id"

+ "[SubjectName1] = SubjectName1,[TheacherSurName1] = TheacherSurName1,[Mark1] = Mark1 ,[SubjectName2] = SubjectName2,[TheacherSurName2] = TheacherSurName2,[Mark2] = Mark2 ,[SubjectName3] = SubjectName3,[TheacherSurName3] = TheacherSurName3,[Mark3] = Mark3 , ", sqlConnection);

command.Parameters.AddWithValue("Id", numericUpDownId.Value);

command.Parameters.AddWithValue("SubjectName1", textBoxUpdSubjectName1.Text);

command.Parameters.AddWithValue("TheacherSurName1", textBoxUpdTeacherSurname1.Text);

command.Parameters.AddWithValue("Mark1", textBoxUpdMark1.Text);

command.Parameters.AddWithValue("SubjectName2", textBoxUpdSubjectName2.Text);

command.Parameters.AddWithValue("TheacherSurName2", textBoxUpdTeacherSurname2.Text);

command.Parameters.AddWithValue("Mark2", textBoxUpdMark3.Text);

command.Parameters.AddWithValue("SubjectName3", textBoxUpdSubjectName3.Text);

command.Parameters.AddWithValue("TheacherSurName3", textBoxUpdTeacherSurname3.Text);

command.Parameters.AddWithValue("Mark3", textBoxUpdMark3.Text);

command.Parameters.AddWithValue("Name", textBoxUpdName.Text);

command.Parameters.AddWithValue("SName", textBoxUpdSecondName.Text);

command.Parameters.AddWithValue("TName", textBoxUpdThridName.Text);

command.Parameters.AddWithValue("RecordBookNumber", textBoxUpdRecordBook.Text);

command.Parameters.AddWithValue("Faculty", textBoxUpdFaculty.Text);

command.Parameters.AddWithValue("Course", textBoxUpdCourse.Text);

try

{

command.ExecuteNonQuery();

sqlConnection.Close();

}

catch

{

MessageBox.Show("Не удалось обновить запись");

sqlConnection.Close();

}

}

else

{

labelUpdError.Visible = true;

labelUpdError.Text = "Ошибка Вы не правильно заполнили поля";

sqlConnection.Close();

}

Принцип работы данной функции почти идентичен Insert, но в отличии от Insert данный метод обновляет данные в базе данных по указанному ,в остальном очень похожа на Insert как по получению информации так и по обмену с бд.

Теперь опишу вкладку Delete

sqlConnection.Open();

SqlCommand command = new SqlCommand("DELETE FROM [Students] WHERE [Id] = @Id", sqlConnection);

command.Parameters.AddWithValue("Id", numericUpDownDelId.Value);

try

{

command.ExecuteNonQuery();

sqlConnection.Close();

}

catch

{

MessageBox.Show("Не удалось удалить запись");

sqlConnection.Close();

}

Она открывает sql подключениеб после чего создаем SqlCommand который удаляет по Id строки . Также возвращает измененные строки через ExecuteNonQuery и закрывает подключение.

Помимо удалить по Id так же есть кнопка “Удалить если оценка”

sqlConnection.Open();

if (labelDelError.Visible)

{

labelDelError.Visible = false;

}

SqlCommand command = new SqlCommand("DELETE FROM [Students] WHERE [Mark1] = @Mark1 AND [Mark2] = @Mark2 AND [Mark3] = @Mark3", sqlConnection);

command.Parameters.AddWithValue("Mark1", textBoxMarks.Text);

command.Parameters.AddWithValue("Mark2", textBoxMarks.Text);

command.Parameters.AddWithValue("Mark3", textBoxMarks.Text);

try

{

command.ExecuteNonQuery();

sqlConnection.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);

}

Она работает по принципу удаления по Id, вместо этого удаляет элемент если хоть одно поле оценки имеет оценку указанную в textBox .

Следующая и последняя описанная функция будет Find.

dataGridView3.Rows.Clear();

sqlConnection.Open();

string reloading = reload;

SqlCommand command = new SqlCommand "SELECT \* FROM [Students] WHERE (Mark1 LIKE '%' + @search + '%') AND (Mark2 LIKE '%' + @search + '%') AND (Mark1 LIKE '%' + @search + '%')", sqlConnection);

command.Parameters.AddWithValue("search", textBoxFind.Text);

SqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[16]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

data[data.Count - 1][6] = reader[6].ToString();

data[data.Count - 1][7] = reader[7].ToString();

data[data.Count - 1][8] = reader[8].ToString();

data[data.Count - 1][9] = reader[9].ToString();

data[data.Count - 1][10] = reader[10].ToString();

data[data.Count - 1][11] = reader[11].ToString();

data[data.Count - 1][12] = reader[12].ToString();

data[data.Count - 1][13] = reader[13].ToString();

data[data.Count - 1][14] = reader[14].ToString();

data[data.Count - 1][15] = reader[15].ToString();

}

reader.Close();

sqlConnection.Close();

foreach (string[] s in data)

{

dataGridView3.Rows.Add(s);

}

Она осуществляется при помощи SqlComand , которая выбирает строки где в поле оценка есть значения как в textBox. И отображает в своем отдельном DataGridView такие строки при этом просто не отбражая те которые не подходят по поиску.

Последнее что хотелось бы затронуть в данной работе это функции вкладки файл а именно “Обновить” и “Отключить Sql Connect”

dataGridView1.Rows.Clear();

dataGridView2.Rows.Clear();

dataGridView3.Rows.Clear();

SqlDataReader sqlReader = null;

reload = "SELECT \* FROM [Students]";

SqlCommand command = new SqlCommand(reload, sqlConnection);

try

{

LoadData(reload);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

if (sqlReader != null)

sqlReader.Close();

}

Просто вызывает метод LoadData(он описан выше) бд, а “Отключить Sql Connect” просто закрывает подключение

sqlConnection.Close();

Итог

Таким образом в данной программе есть все основные функции работы с бд в Sql. Данная программа является наглядным примером по работе с Sql в visual studio. Во время выполнения данной работы я столкнулся с множеством проблем решения которых давалось с трудом, в итоге созданная программа отлично работает и может помочь например преподавателю или деканату найти нужных студентов не проводя проверок вручную.