МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**Кафедра «САПР»**

Пояснительная записка

**К курсовому проекту по дисциплине**

**«Объектно-ориентированное программирование»**

На тему: «Разработка информационно-аналитической системы водительских курсов с использованием технологии объектно-ориентированного программирования.»

Направление подготовки – 02.03.03 Математическое обеспечение и администрирование информационных систем

Выполнил: студент группы 21ВА1

Ишенбеков Н.М.

Принял: к.т.н., доцент Подмарькова Е.М.

**Пенза, 2024 г.**

**Аннотация**

В данной пояснительной записке приведено описание программы, реализующей информационно-аналитическую систему «Водительские курсы» и ее исходного кода.

Целью разработки курсового проекта является разработка программы для работы с системой «Водительские курсы».

В процессе выполнения курсового проекта должны быть выявлены действующие субъекты системы, определены и описаны варианты использования, определены списки имен существительных и глаголов, построены диаграммы вариантов использования, классов и последовательностей, а также создана, отлажена и протестирована программа. Программа должна быть написана на языке программирования С#, поддерживающем функционал ООП и быть разработана в среде Microsoft Visual Studio 2022.

Пояснительная записка состоит из: 46 страницы, 12 рисунков, 2 источников, 2 приложений.

**Содержание**

[**1.** **Описание программы** 4](#_Toc168970534)

[1.1. Общие сведения 4](#_Toc168970535)

[1.2. Функциональное значение 5](#_Toc168970536)

[1.3. Описание логической структуры 6](#_Toc168970537)

[**2.** **Процесс разработки приложения** 7](#_Toc168970538)

[2.1. Диаграмма вариантов использования 7](#_Toc168970539)

[2.2. Список имен существительных и глаголов 9](#_Toc168970540)

[2.3. Диаграмма классов 10](#_Toc168970541)

[2.4. Диаграммы последовательностей 11](#_Toc168970542)

[**3.** **Заключение** 12](#_Toc168970543)

[**4.** **Список использованной литературы** 13](#_Toc168970544)

[**Приложение А** 14](#_Toc168970545)

[**Приложение Б** 19](#_Toc168970546)

# **Описание программы**

# Общие сведения

Информационно-аналитическая система «Водительские курсы» поддерживает режимы поиска студентов и курсов, добавления и учета студентов, курсов, посещаемости, практических занятий, ошибок при вождении, тестировании, результатов экзаменов.

# Функциональное значение

Разработанная информационно-аналитическая система «Водительские курсы» предназначена для:

* Учета студентов
* Учета водительских курсов
* Учета посещаемости
* Учета тренировок вождения
* Учета тестирования
* Учета результатов сдачи экзаменов
* Анализа популярности курсов
* Анализа стоимости курсов
* Анализа качества курсов

# Описание логической структуры

Программа разбита на отдельные функциональные части – подпрограммы, которые распределяются по отдельным уровням иерархии. Каждая из подпрограмм решает только свою небольшую задачу по преобразованию данных, что позволяет упростить процесс написания и отладки программы в целом. Далее приводятся описания основных классов.

Student – класс с общей информацией о студентах курса

Course – класс для с общей информацией о курсах

CourseHasStudents – класс содержащий информацию какой курс записан студент

Attendance – класс для информации о посещяемости

DrivingPractice – класс с информацией о тренировках вождения

DrivingErrors – класс для информации об ошибках при вождении

TestResults – класс для информации о результатах тестов

ExamResults – класс для информации о результатах экзаменов

CourseAnalysis – класс для анализа информации о курсах

DB – класс для подключения к базе данных

# **Процесс разработки приложения**

# Диаграмма вариантов использования

Диаграмма вариантов использования представлена на рисунке 1.

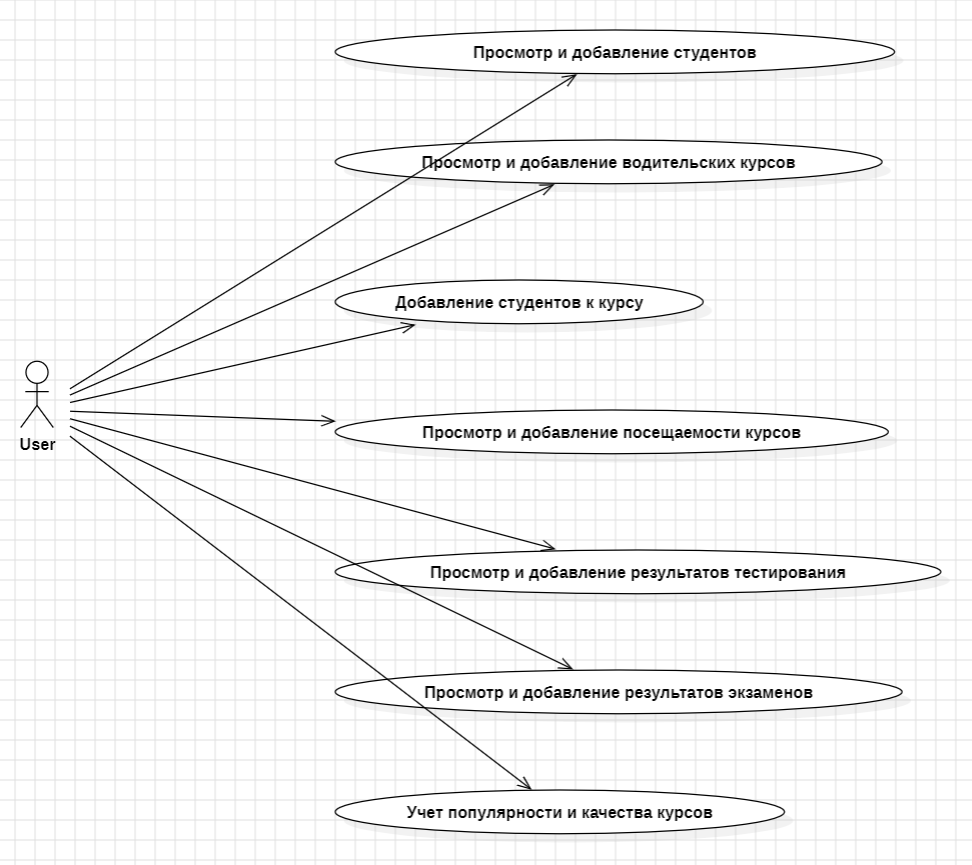


Рисунок 1. Диаграмма вариантов использования

Описание вариантов использования

1. Просмотр и добавление студентов

Администратор может получить список студентов из БД и добавлять в новых студентов

1. Просмотр и добавление водительских курсов

Администратор может получить информацию о курсах из БД, а также добавлять новые курсы.

1. Добавление студентов к курсу

Администратор может добавить студента в список студентов определенного курса.

1. Просмотр и добавление посещаемости курсов

Администратор может посмотреть данные о посещаемости курсов и добавить новые записи о посещаемости.

1. Просмотр и добавление результатов тестирования

Администратор может посмотреть данные о тестировании и добавить новые записи.

1. Просмотр и добавление результатов экзаменов

Администратор может посмотреть данные о экзаменах и добавить новые записи.

1. Учет популярности и качества курсов

Администратор может посмотреть данные о популярности курсов и качестве курсов.

# Список имен существительных и глаголов

**Существительные**

1. Студент;
2. Курс;
3. Курс-студент;
4. Посещаемость;
5. Тренировка вождения
6. Ошибки при вождении
7. Результат тестирования;
8. Результат экзамена;

**Глаголы**

1. Добавить студента/курс;

На интерфейсе должна отобразиться форма, в котором программа просит пользователя ввести данные о студенте/курсе. Эта информация должна заноситься в БД.

1. Загрузить данные;

Данные загружаются из БД.

1. Сохранить данные;

Сохраняются данные, во время работы приложения в БД.

1. Поиск по ФИО студента или по названию курса;

В списке курсов или студентов в поле ввода можно ввести имя. По введенным данным выводится список совпадений.

# Диаграмма классов

Диаграмма классов представлена на рисунке 2.



Рисунок 2. Диаграмма классов

Данная диаграмма классов отображает связи между классами и их основные атрибуты. Классы связаны отношениями. Например, класс Студенты категорий содержит агрегацию с классом Посещаемость.

# Диаграммы последовательностей

Диаграмма последовательности добавления нового курса и запись студента на этот курс на рисунке 3.

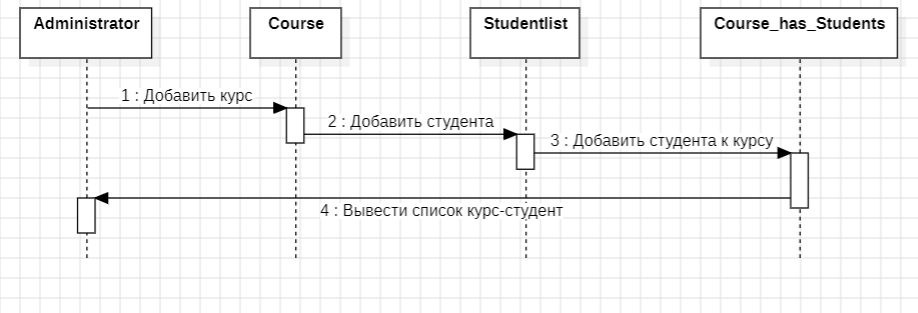


Рисунок 3. Диаграмма создания заказа

Администратор хочет добавить нового студента в новый курс, для этого ему необходимо id курса и студента.

# **Заключение**

В результате реализации проекта все требования технического задания были выполнены. Конечным результатом является приложение «Водительские курсы», которое может быть использовано для:

* Учета студентов курса
* Учета всех курсов
* Учета популярности и качества курсов
* Учета посещаемости
* Осуществления записи новых студентов

Программа написана на языке С# с использованием инструментов объектно-ориентированного программирования.

Данной системой могут пользоваться администраторы.

# **Список использованной литературы**

1. Уроки C# .NET Windows Forms + Базы данных SQL – YouTube: https://www.youtube.com/playlist?list=PL0lO\_mIqDDFWOMqSKFaLypANf1W7-o87q.
2. Руководство по MySql: <https://metanit.com/sql/mysql/>

(дата обращения: 25.03.2024)

# **Приложение А**

Скриншоты работы программы представлены на рисунках 4 – 13.

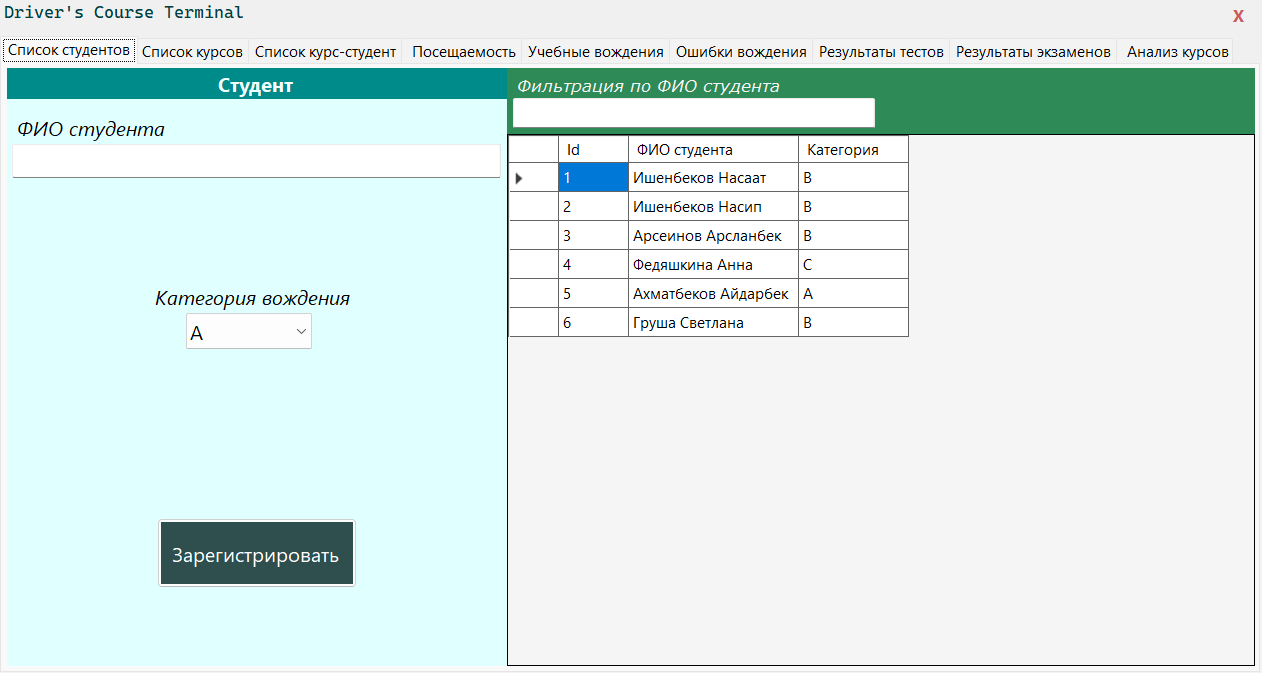


Рисунок 4. Вкладка список студентов

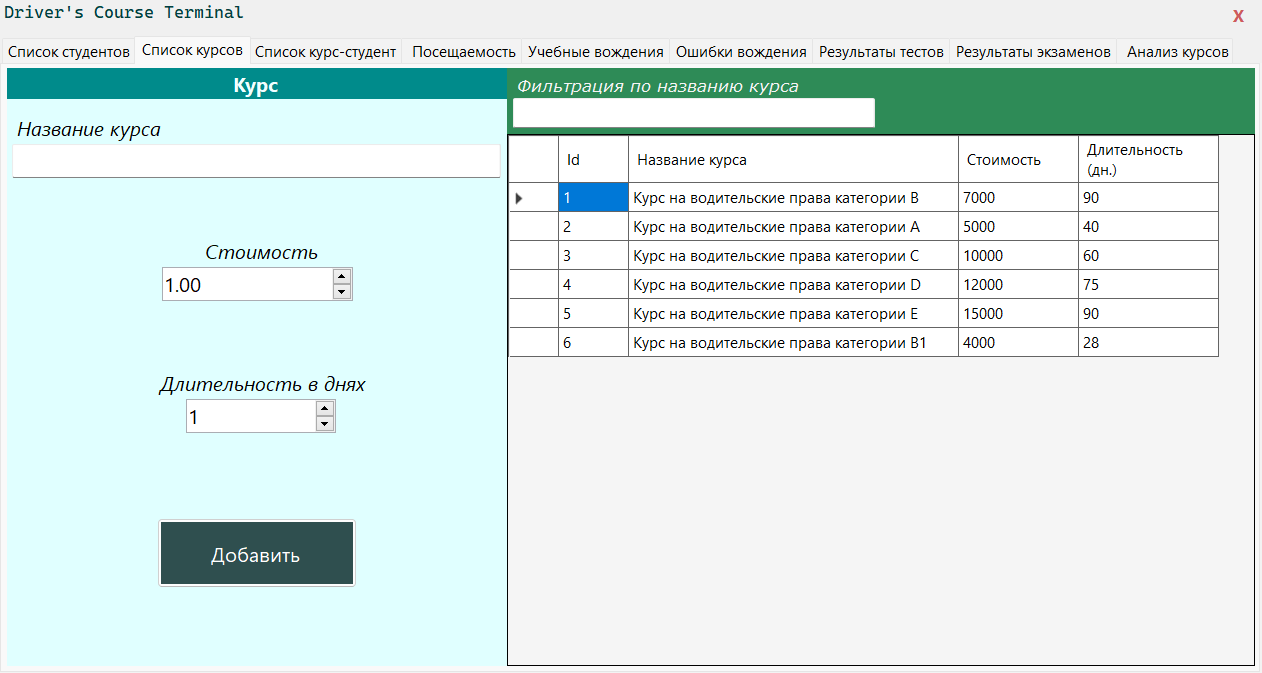


Рисунок 5. Вкладка список курсов

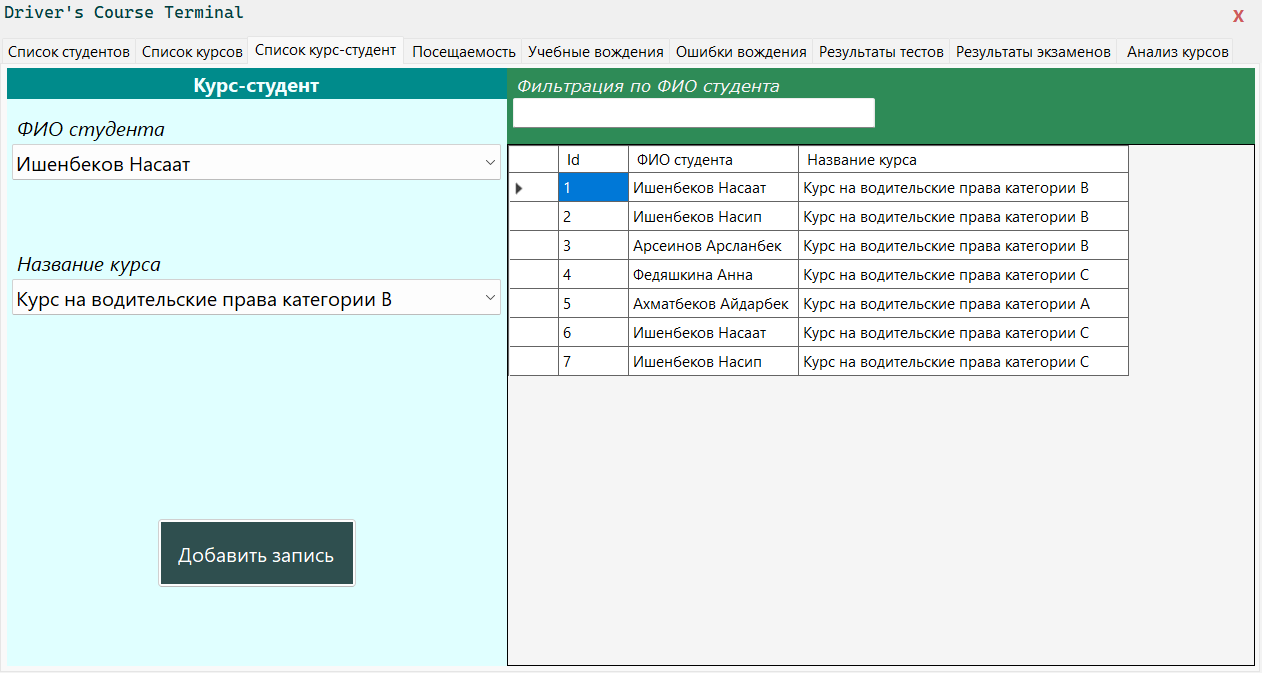


Рисунок 6. Вкладка список курс-студент

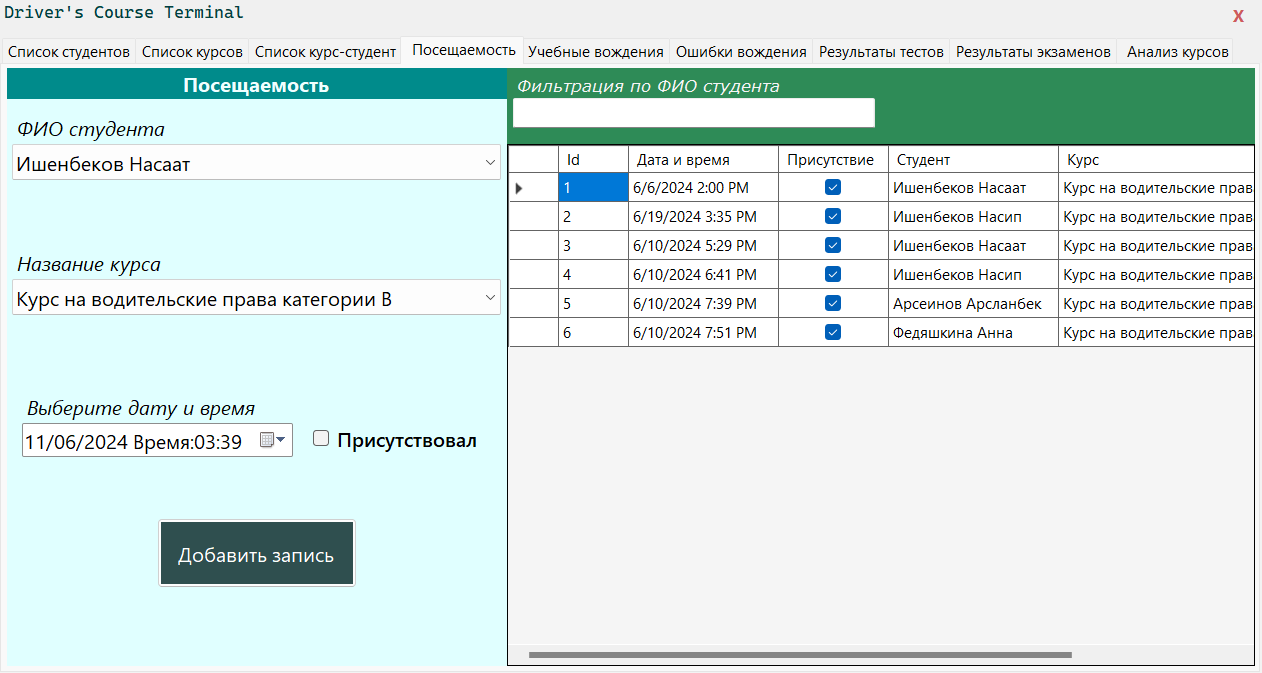


Рисунок 7. Вкладка посещаемость

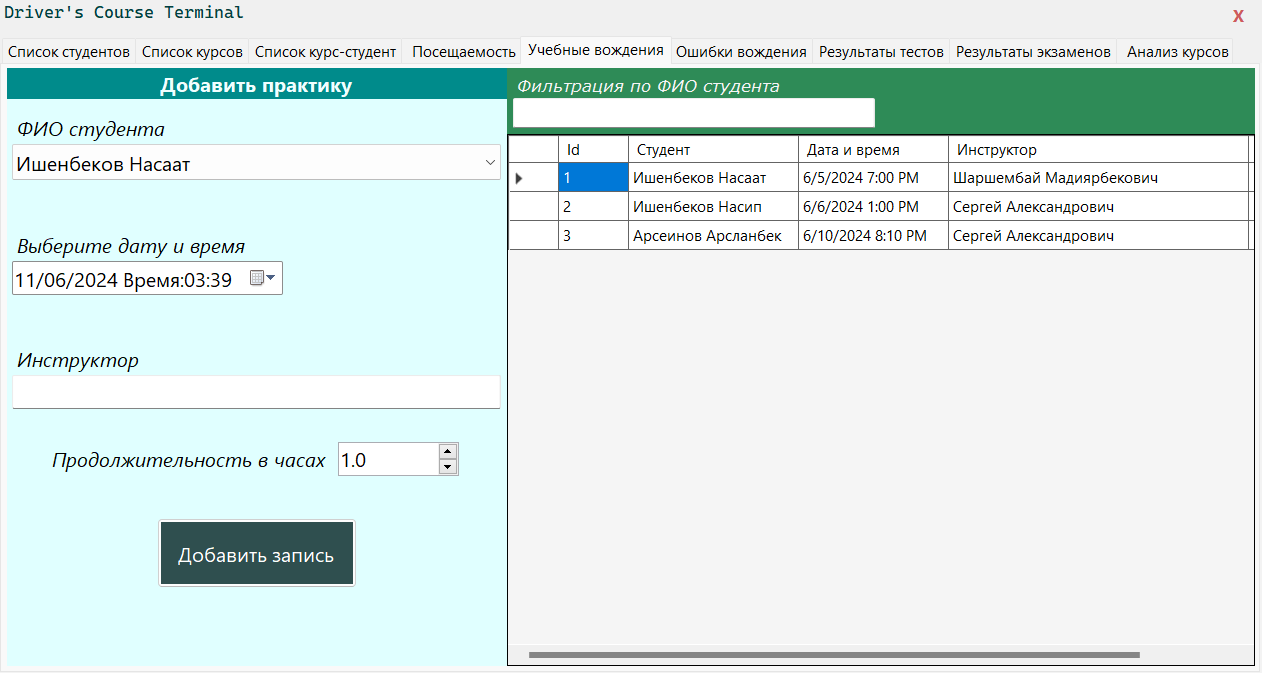


Рисунок 8. Вкладка учебные вождения

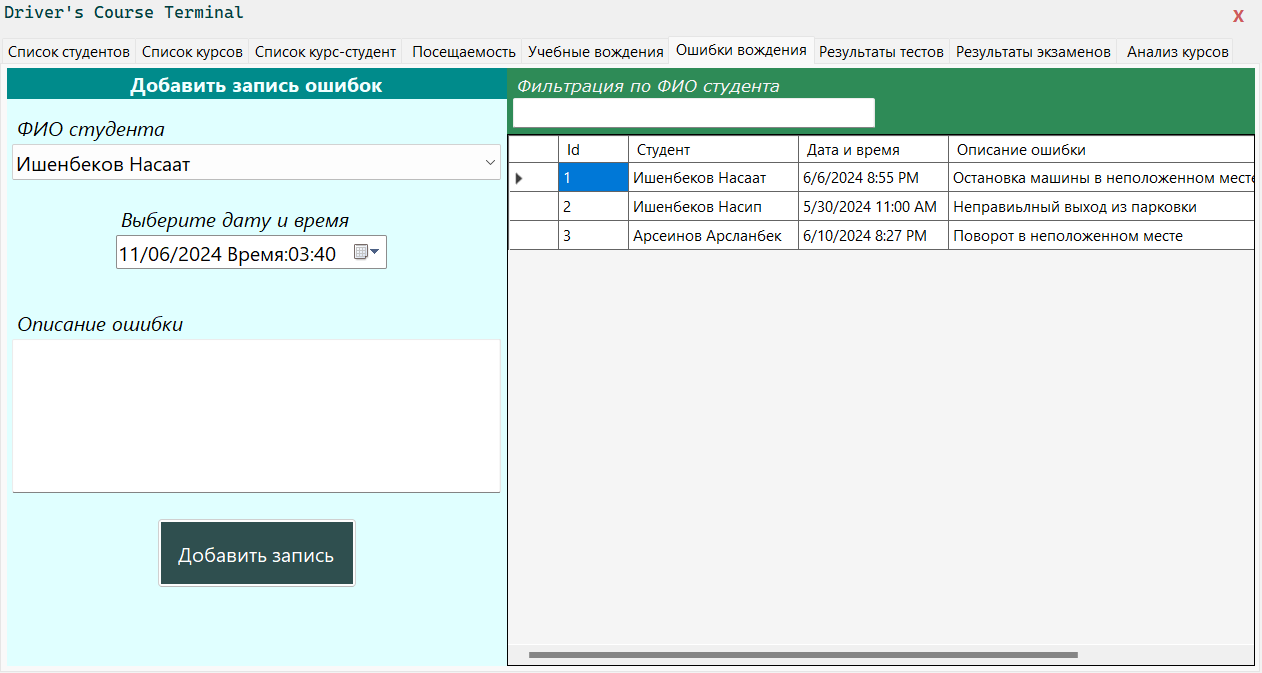


Рисунок 9. Вкладка ошибки вождения

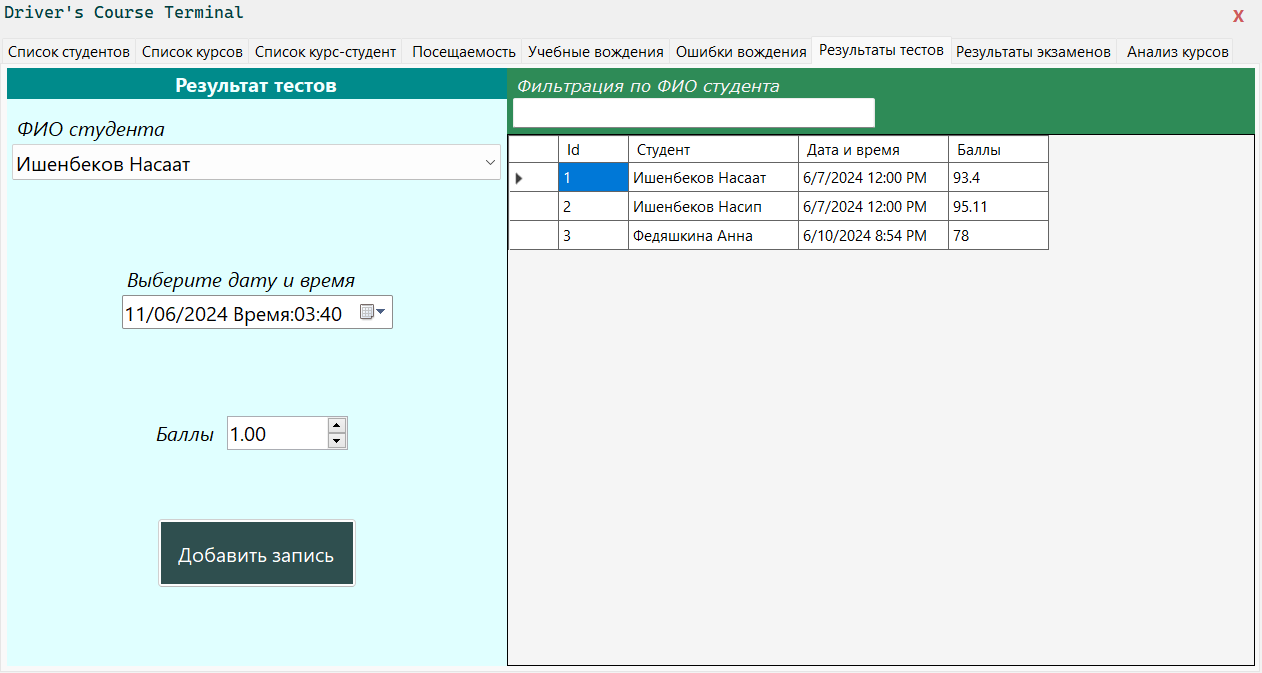


Рисунок 10. Вкладка результаты тестов

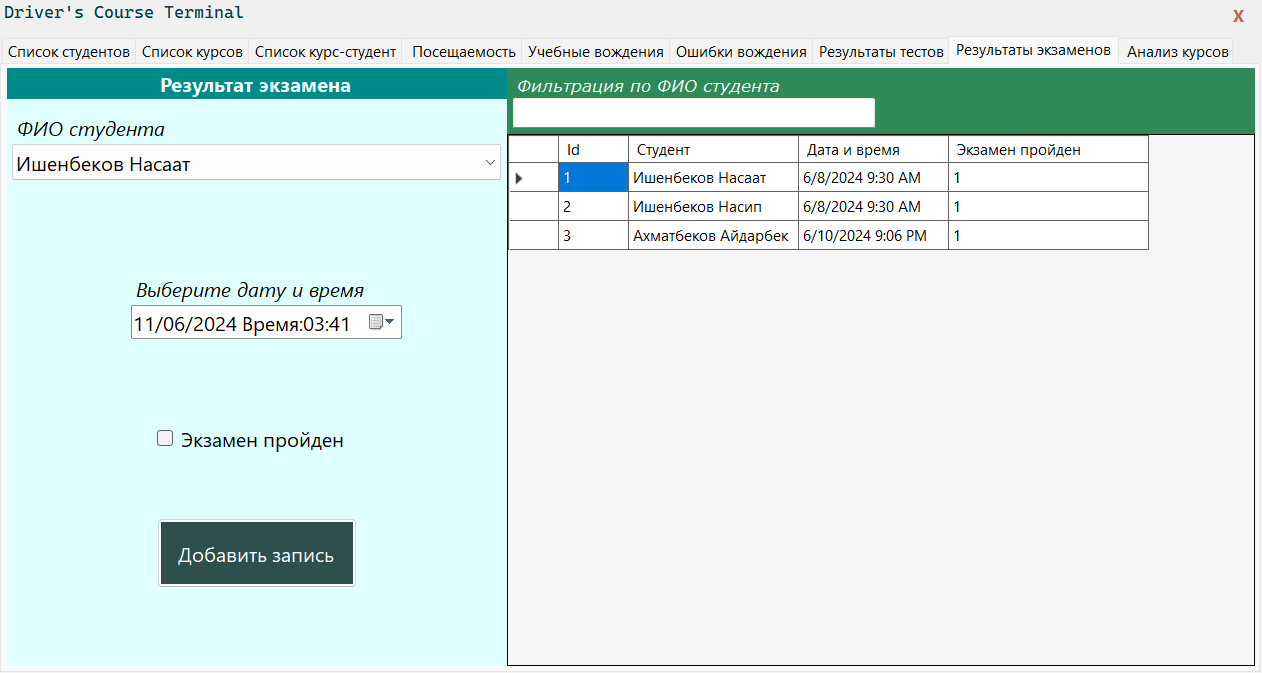


Рисунок 11. Вкладка результаты экзаменов

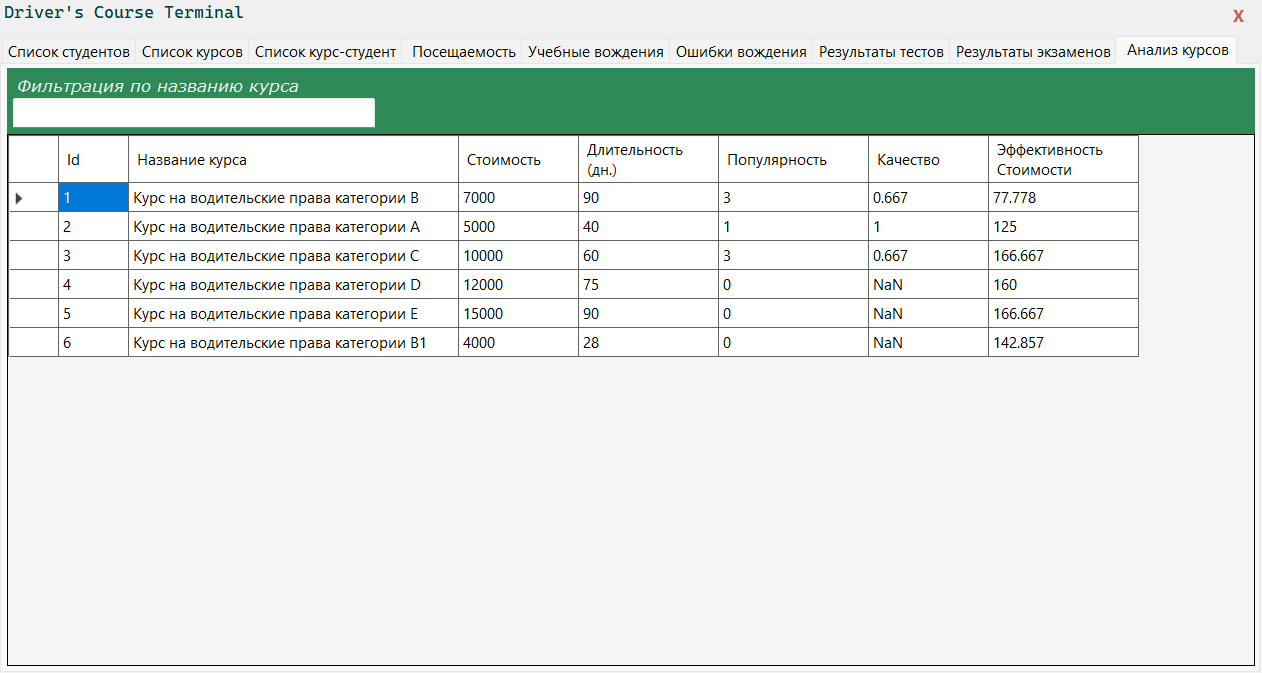


Рисунок 12. Вкладка анализ курсов

# **Приложение Б**

Исходный код программы.

**Form1.cs**

using MySql.Data.MySqlClient;

using System.Data;

namespace Driver\_s\_Course\_Terminal

{

public partial class Form1 : Form

{

// Create lists

CourseAnalysis courseAnalysis = new CourseAnalysis();

public Form1()

{

InitializeComponent();

labelApplicationName.Text = Application.CompanyName;

loadCourseAnalysis();

}

// Course analysis load

private void loadCourseAnalysis()

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlConnection conn = dB.GetConnection();

// Inizialize courses

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM `course`", conn);

adapter.SelectCommand = cmd;

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int idCourse = Convert.ToInt32(row["idCourse"]);

string name = row["Name"].ToString();

double cost = Convert.ToDouble(row["Cost"]);

int durationInDays = Convert.ToInt32(row["DurationInDays"]);

Course course = new Course(idCourse, name, cost, durationInDays);

// Inizialize course has students

cmd = new MySqlCommand("SELECT \* FROM `course\_has\_students` WHERE `Course\_idCourse` = @cId", conn);

cmd.Parameters.Add("@cId", MySqlDbType.VarChar).Value = idCourse;

adapter.SelectCommand = cmd;

DataTable course\_has\_students\_dt = new DataTable();

adapter.Fill(course\_has\_students\_dt);

foreach (DataRow c\_has\_s\_Row in course\_has\_students\_dt.Rows)

{

// Inizialize student

cmd = new MySqlCommand("SELECT \* FROM `students` WHERE `idStudents` = @sId", conn);

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = c\_has\_s\_Row["Students\_idStudents"];

adapter.SelectCommand = cmd;

DataTable studentdt = new DataTable();

adapter.Fill(studentdt);

foreach (DataRow studentrow in studentdt.Rows)

{

int sStudentsId = Convert.ToInt32(studentrow["idStudents"]);

string sName = studentrow["Name"].ToString();

string sCategory = studentrow["Category"].ToString();

Student student = new Student(sStudentsId, sName, sCategory);

// Create temp datatable, adapter and command

DataTable tdataTable = new DataTable();

MySqlDataAdapter tadapter;

MySqlCommand tcmd;

// Inizialize attedance

tcmd = new MySqlCommand("SELECT \* FROM `attendance` WHERE `Students\_idStudents` = @sId", conn);

tcmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = sStudentsId;

tadapter = new MySqlDataAdapter(tcmd);

tadapter.Fill(tdataTable);

foreach (DataRow trow in tdataTable.Rows)

{

int aAttendanceId = Convert.ToInt32(trow["idAttendance"]);

int aStudentsId = Convert.ToInt32(trow["Students\_idStudents"]);

int aCourseId = Convert.ToInt32(trow["Course\_idCourse"]);

DateTime aDate = Convert.ToDateTime(trow["Date"]);

bool aIsPresent = Convert.ToInt32(trow["IsPresent"]) == 0 ? false : true;

Attendance attendance = new Attendance(aAttendanceId, aStudentsId, aCourseId, aDate, aIsPresent);

student.Attendances.Add(attendance);

}

// Inizialize driving practice

tcmd = new MySqlCommand("SELECT \* FROM `drivingpractice` WHERE `Students\_idStudents` = @sId", conn);

tcmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = sStudentsId;

tadapter = new MySqlDataAdapter(tcmd);

tdataTable.Reset();

tadapter.Fill(tdataTable);

foreach (DataRow trow in tdataTable.Rows)

{

int pPracticeId = Convert.ToInt32(trow["idDrivingPractice"]);

int pStudentsId = Convert.ToInt32(trow["Students\_idStudents"]);

DateTime pDate = Convert.ToDateTime(trow["Date"]);

string pInstructor = trow["Instructor"].ToString();

double pDurationInHours = Convert.ToDouble(trow["DurationInHours"]);

DrivingPractice practice = new DrivingPractice(pPracticeId, pStudentsId, pDate, pInstructor, pDurationInHours);

student.DrivingPractices.Add(practice);

}

// Inizialize driving errors

tcmd = new MySqlCommand("SELECT \* FROM `drivingerrors` WHERE `Students\_idStudents` = @sId", conn);

tcmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = sStudentsId;

tadapter = new MySqlDataAdapter(tcmd);

tdataTable.Reset();

tadapter.Fill(tdataTable);

foreach (DataRow trow in tdataTable.Rows)

{

int eErrorsId = Convert.ToInt32(trow["idDrivingErrors"]);

int eStudentsId = Convert.ToInt32(trow["Students\_idStudents"]);

DateTime eDate = Convert.ToDateTime(trow["Date"]);

string eDescription = trow["ErrorDescription"].ToString();

DrivingErrors error = new DrivingErrors(eErrorsId, eStudentsId, eDate, eDescription);

student.DrivingErrors.Add(error);

}

// Inizialize test results

tcmd = new MySqlCommand("SELECT \* FROM `testresults` WHERE `Students\_idStudents` = @sId", conn);

tcmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = sStudentsId;

tadapter = new MySqlDataAdapter(tcmd);

tdataTable.Reset();

tadapter.Fill(tdataTable);

foreach (DataRow trow in tdataTable.Rows)

{

int tsTestId = Convert.ToInt32(trow["idTestResults"]);

int tsStudentId = Convert.ToInt32(trow["Students\_idStudents"]);

DateTime tsDate = Convert.ToDateTime(trow["Date"]);

double tsScore = Convert.ToDouble(trow["Score"]);

TestResults test = new TestResults(tsTestId, tsStudentId, tsDate, tsScore);

student.TestResults.Add(test);

}

// Inizialize exam results

tcmd = new MySqlCommand("SELECT \* FROM `examresults` WHERE `Students\_idStudents` = @sId", conn);

tcmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = sStudentsId;

tadapter = new MySqlDataAdapter(tcmd);

tdataTable.Reset();

tadapter.Fill(tdataTable);

foreach (DataRow trow in tdataTable.Rows)

{

int exExamId = Convert.ToInt32(trow["idExamResults"]);

int exStudentId = Convert.ToInt32(trow["Students\_idStudents"]);

DateTime exDate = Convert.ToDateTime(trow["Date"]);

bool exIsPassed = Convert.ToInt32(trow["IsPassed"]) == 0 ? false : true;

ExamResults exam = new ExamResults(exExamId, exStudentId, exDate, exIsPassed);

student.ExamResults.Add(exam);

}

course.Students.Add(student);

}

}

courseAnalysis.AddCourse(course);

}

}

// Application

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

{

Application.Exit();

}

private void labelCloseApp\_MouseEnter(object sender, EventArgs e)

{

labelCloseApp.BackColor = Color.IndianRed;

labelCloseApp.ForeColor = SystemColors.Menu;

}

private void labelCloseApp\_MouseLeave(object sender, EventArgs e)

{

labelCloseApp.BackColor = SystemColors.Menu;

labelCloseApp.ForeColor = Color.IndianRed;

}

Point lastPoint;

private void panelMenuApplication\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panelMenuApplication\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

// Students

private void tabPageStudents\_Enter(object sender, EventArgs e)

{

textBoxStudentsFilter\_TextChanged(sender, e);

}

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

{

string name = textBoxStudentsStudentName.Text;

string category = comboBoxStudentsCategory.Text;

// Checking for an empty string

if (textBoxStudentsStudentName.Text.Length < 1)

{

comboBoxStudentsCategory.SelectedIndex = 0;

MessageBox.Show("Пожалуйста введите имя!");

return;

}

DB dB = new DB();

MySqlCommand cmd = new MySqlCommand("INSERT INTO `students` (`Name`, `Category`) VALUES (@sN, @sct);", dB.GetConnection());

cmd.Parameters.Add("@sN", MySqlDbType.VarChar).Value = name;

cmd.Parameters.Add("@sct", MySqlDbType.VarChar).Value = category;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

MessageBox.Show("Студент успешно добавлен!");

else

MessageBox.Show("Ошибка при регистрации!");

textBoxStudentsStudentName.Text = "";

comboBoxStudentsCategory.SelectedIndex = 0;

dB.closeConnection();

tabPageStudents\_Enter(sender, e);

}

private void textBoxStudentsFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxStudentsFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT `idStudents`, `Name`, `Category` FROM `students`\r\n" +

"WHERE `Name` LIKE @fN;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewStudents.DataSource = null;

dataGridViewStudents.Columns.Clear();

comboBoxStudentsCategory.SelectedIndex = 0;

// Set the DataSource of the DataGridView to the DataTable

dataGridViewStudents.DataSource = dt;

// Adjust column headers if necessary

dataGridViewStudents.Columns["idStudents"].HeaderText = "Id";

dataGridViewStudents.Columns["Name"].HeaderText = "ФИО студента";

dataGridViewStudents.Columns["Category"].HeaderText = "Категория";

dataGridViewStudents.Columns["idStudents"].Width = 70;

dataGridViewStudents.Columns["Name"].Width = 170;

dataGridViewStudents.Columns["Category"].Width = 110;

}

int idStudents;

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

{

buttonStudentsAddList.Visible = false;

buttonStudentsEdit.Visible = true;

buttonStudentsCancel.Visible = true;

idStudents = int.Parse(dataGridViewStudents[0, dataGridViewStudents.SelectedCells[0].RowIndex].Value.ToString());

textBoxStudentsStudentName.Text = dataGridViewStudents[1, dataGridViewStudents.SelectedCells[0].RowIndex].Value.ToString();

comboBoxStudentsCategory.SelectedItem = dataGridViewStudents[2, dataGridViewStudents.SelectedCells[0].RowIndex].Value;

}

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

{

// Checking for an empty string

if (textBoxStudentsStudentName.Text.Length < 1 ||

comboBoxStudentsCategory.Text.Length < 1)

{

MessageBox.Show("Пожалуйста заполенте все поля!");

return;

}

DB dB = new DB();

MySqlCommand cmd = new MySqlCommand("UPDATE `students` SET `Name` = @sN, `Category` = @ct WHERE `students`.`idStudents` = @sId", dB.GetConnection());

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = idStudents;

cmd.Parameters.Add("@sN", MySqlDbType.VarChar).Value = textBoxStudentsStudentName.Text;

cmd.Parameters.Add("@ct", MySqlDbType.VarChar).Value = comboBoxStudentsCategory.Text;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

MessageBox.Show("Запись изменена!");

else

MessageBox.Show("Ошибка при редактировании!");

textBoxStudentsStudentName.Text = "";

comboBoxStudentsCategory.SelectedIndex = 0;

// Restore buttons visible

buttonStudentsCancel.Visible = false;

buttonStudentsEdit.Visible = false;

buttonStudentsAddList.Visible = true;

dB.closeConnection();

tabPageStudents\_Enter(sender, e);

}

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

{

textBoxStudentsStudentName.Text = "";

comboBoxStudentsCategory.SelectedIndex = 0;

// Restore buttons visible

buttonStudentsCancel.Visible = false;

buttonStudentsEdit.Visible = false;

buttonStudentsAddList.Visible = true;

}

// Courses

private void tabPageCourses\_Enter(object sender, EventArgs e)

{

textBoxCourseFilter\_TextChanged(sender, e);

}

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

{

// Checking for an empty string

if (textBoxCourseName.Text.Length < 1)

{

MessageBox.Show("Пожалуйста введите имя!");

return;

}

string name = textBoxCourseName.Text;

double cost = (double)numericUpDownCourseCost.Value;

int durationInDays = (int)numericUpDownCourseDurationInDays.Value;

DB dB = new DB();

MySqlCommand cmd = new MySqlCommand("INSERT INTO `course` (`Name`, `Cost`, `DurationInDays`) VALUES (@cN, @cC, @cD);", dB.GetConnection());

cmd.Parameters.Add("@cN", MySqlDbType.VarChar).Value = name;

cmd.Parameters.Add("@cC", MySqlDbType.VarChar).Value = cost;

cmd.Parameters.Add("@cD", MySqlDbType.VarChar).Value = durationInDays;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

{

DataTable dt = new DataTable();

cmd = new MySqlCommand("SELECT \* FROM `course` WHERE `idCourse` = (SELECT MAX(`idCourse`) FROM `course`);", dB.GetConnection());

MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int courseId = Convert.ToInt32(row["idCourse"]);

Course course = new Course(courseId, name, cost, durationInDays);

courseAnalysis.AddCourse(course);

}

MessageBox.Show("Курс успешно добавлен!");

}

else

MessageBox.Show("Ошибка при добавлении!");

textBoxCourseName.Text = "";

numericUpDownCourseCost.Value = 1;

numericUpDownCourseDurationInDays.Value = 1;

dB.closeConnection();

textBoxCourseFilter\_TextChanged(sender, e);

}

private void textBoxCourseFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxCourseFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM `course`\r\n" +

"WHERE `Name` LIKE @fN;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewCourse.DataSource = null;

dataGridViewCourse.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewCourse.DataSource = dt;

// Adjust column headers if necessary

dataGridViewCourse.Columns["idCourse"].HeaderText = "Id";

dataGridViewCourse.Columns["Name"].HeaderText = "Название курса";

dataGridViewCourse.Columns["Cost"].HeaderText = "Стоимость";

dataGridViewCourse.Columns["DurationInDays"].HeaderText = "Длительность\n(дн.)";

dataGridViewCourse.Columns["idCourse"].Width = 70;

dataGridViewCourse.Columns["Name"].Width = 330;

dataGridViewCourse.Columns["Cost"].Width = 120;

dataGridViewCourse.Columns["DurationInDays"].Width = 140;

}

// Course has students

private void tabPageCourseHasStudents\_Enter(object sender, EventArgs e)

{

// Refresh dataTable

textBoxCourseHasStudents\_TextChanged(sender, e);

// Initialize database connection and data adapter

DB dB = new DB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

// DataTable for the combobox

DataTable dtStudensNames = new DataTable();

DataTable dtCoursesNames = new DataTable();

MySqlCommand cmdStudensNames = new MySqlCommand("SELECT `idStudents`, `Name` FROM `students`", dB.GetConnection());

MySqlCommand cmdCoursesNames = new MySqlCommand("SELECT `idCourse`, `Name` FROM `course`", dB.GetConnection());

dB.openConnection();

adapter.SelectCommand = cmdCoursesNames;

adapter.Fill(dtCoursesNames);

comboBoxCourseHasStudentsCourseName.DataSource = dtCoursesNames;

comboBoxCourseHasStudentsCourseName.DisplayMember = "Name";

comboBoxCourseHasStudentsCourseName.ValueMember = "idCourse";

adapter.SelectCommand = cmdStudensNames;

adapter.Fill(dtStudensNames);

comboBoxCourseHasStudentsStudentName.DataSource = dtStudensNames;

comboBoxCourseHasStudentsStudentName.DisplayMember = "Name";

comboBoxCourseHasStudentsStudentName.ValueMember = "idStudents";

dB.closeConnection();

}

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

{

// Checking for an empty string

if (comboBoxCourseHasStudentsStudentName.Text.Length < 1 ||

comboBoxCourseHasStudentsCourseName.Text.Length < 1)

{

MessageBox.Show("Пожалуйста заполенте все поля!");

return;

}

DB dB = new DB();

int studentId = Convert.ToInt32(comboBoxCourseHasStudentsStudentName.SelectedValue);

int courseId = Convert.ToInt32(comboBoxCourseHasStudentsCourseName.SelectedValue);

MySqlCommand cmd = new MySqlCommand("INSERT INTO `course\_has\_students` (`Students\_idStudents`, `Course\_idCourse`)VALUES (@sId, @cId);", dB.GetConnection());

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = studentId;

cmd.Parameters.Add("@cId", MySqlDbType.VarChar).Value = courseId;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

{

cmd = new MySqlCommand("SELECT \* FROM `students` WHERE `idStudents` = @sId;", dB.GetConnection());

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = studentId;

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

string name = Convert.ToString(row["Name"]);

string category = Convert.ToString(row["Category"]);

Student student = new Student(studentId, name, category);

courseAnalysis.AddCoursehasStudent(courseId, student);

}

MessageBox.Show("Запись добавлена!");

}

else

MessageBox.Show("Ошибка при добавлении!");

comboBoxCourseHasStudentsStudentName.SelectedIndex = 0;

comboBoxCourseHasStudentsCourseName.SelectedIndex = 0;

dB.closeConnection();

tabPageCourseHasStudents\_Enter(sender, e);

}

private void textBoxCourseHasStudents\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxCourseHasStudentsFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT `idCourse\_has\_Students`, `students`.`Name` AS `studentName`, `course`.`Name` AS `courseName` FROM `course\_has\_students`\r\n" +

"LEFT JOIN `students` ON `Students\_idStudents` = `students`.`idStudents`\r\n" +

"LEFT JOIN `course` on `Course\_idCourse` = `course`.`idCourse`\r\n" +

"WHERE `students`.`Name` LIKE @fN\r\n" +

"ORDER BY `idCourse\_has\_Students`;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewCourseHasStudents.DataSource = null;

dataGridViewCourseHasStudents.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewCourseHasStudents.DataSource = dt;

// Adjust column headers if necessary

dataGridViewCourseHasStudents.Columns["idCourse\_has\_Students"].HeaderText = "Id";

dataGridViewCourseHasStudents.Columns["studentName"].HeaderText = "ФИО студента";

dataGridViewCourseHasStudents.Columns["courseName"].HeaderText = "Название курса";

dataGridViewCourseHasStudents.Columns["idCourse\_has\_Students"].Width = 70;

dataGridViewCourseHasStudents.Columns["studentName"].Width = 170;

dataGridViewCourseHasStudents.Columns["courseName"].Width = 330;

}

// Attendances

private void tabPageAttendance\_Enter(object sender, EventArgs e)

{

// Refresh dataTable

textBoxStudentAttedanceFilter\_TextChanged(sender, e);

// Initialize database connection and data adapter

DB dB = new DB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

DataTable dtStudensNames = new DataTable();

MySqlCommand cmdStudensNames = new MySqlCommand("SELECT `idStudents`, `Name` FROM `students`", dB.GetConnection());

dB.openConnection();

adapter.SelectCommand = cmdStudensNames;

adapter.Fill(dtStudensNames);

comboBoxAttendanceStudentName.DataSource = dtStudensNames;

comboBoxAttendanceStudentName.DisplayMember = "Name";

comboBoxAttendanceStudentName.ValueMember = "idStudents";

LoadCoursesForStudent(Convert.ToInt32(comboBoxAttendanceStudentName.SelectedValue));

dB.closeConnection();

}

private void LoadCoursesForStudent(int studentId)

{

// Initialize database connection and data adapter

DB dB = new DB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

DataTable dtCoursesNames = new DataTable();

// Get courses for the selected student

MySqlCommand cmd\_course\_has\_students = new MySqlCommand(

"SELECT `course`.`idCourse`, `course`.`Name` FROM `course` " +

"INNER JOIN `course\_has\_students` ON `course`.`idCourse` = `course\_has\_students`.`Course\_idCourse` " +

"WHERE `course\_has\_students`.`Students\_idStudents` = @studentId", dB.GetConnection());

cmd\_course\_has\_students.Parameters.Add("@studentId", MySqlDbType.Int32).Value = studentId;

dB.openConnection();

adapter.SelectCommand = cmd\_course\_has\_students;

adapter.Fill(dtCoursesNames);

dB.closeConnection();

// Set the DataSource of the ComboBox to the DataTable

comboBoxAttendanceCourseName.DataSource = dtCoursesNames;

comboBoxAttendanceCourseName.DisplayMember = "Name";

comboBoxAttendanceCourseName.ValueMember = "idCourse";

}

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

{

if (comboBoxAttendanceStudentName.SelectedValue is int studentId)

{

LoadCoursesForStudent(studentId);

}

}

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

{

// Checking for an empty string

if (comboBoxAttendanceStudentName.Text.Length < 1 ||

comboBoxAttendanceCourseName.Text.Length < 1 ||

dateTimePickerAttendanceDateTime.Text.Length < 1)

{

MessageBox.Show("Пожалуйста заполенте все поля!");

return;

}

DB dB = new DB();

int studentId = Convert.ToInt32(comboBoxAttendanceStudentName.SelectedValue);

int courseId = Convert.ToInt32(comboBoxAttendanceCourseName.SelectedValue);

DateTime date = Convert.ToDateTime(dateTimePickerAttendanceDateTime.Value);

bool isPresent = checkBoxAttendaceIsPresent.Checked;

MySqlCommand cmd = new MySqlCommand("INSERT INTO `attendance` (`Date`, `IsPresent`, `Students\_idStudents`, `Course\_idCourse`) VALUES (@tD, @iP, @sId, @cId);", dB.GetConnection());

cmd.Parameters.AddWithValue("@tD", date);

cmd.Parameters.Add("@iP", MySqlDbType.VarChar).Value = isPresent == true ? 1 : 0;

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = studentId;

cmd.Parameters.Add("@cId", MySqlDbType.VarChar).Value = courseId;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

{

DataTable dt = new DataTable();

cmd = new MySqlCommand("SELECT \* FROM `attendance` WHERE `idAttendance` = (SELECT MAX(`idAttendance`) FROM `attendance`);", dB.GetConnection());

MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int attendanceId = Convert.ToInt32(row["idAttendance"]);

Attendance attendance = new Attendance(attendanceId, studentId, courseId, date, isPresent);

courseAnalysis.AddAttendance(courseId, studentId, attendance);

}

MessageBox.Show("Запись добавлена!");

}

else

MessageBox.Show("Ошибка при добавлении!");

comboBoxAttendanceStudentName.SelectedIndex = 0;

dB.closeConnection();

tabPageAttendance\_Enter(sender, e);

}

private void textBoxStudentAttedanceFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxStudentAttedanceFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT `attendance`.`idAttendance`, `attendance`.`Date`, `attendance`.`IsPresent`, `students`.`Name` AS `studentsName`, `course`.`Name` AS `courseName` FROM `students`\r\n" +

"INNER JOIN `attendance` ON `attendance`.`Students\_idStudents` = `students`.`idStudents`\r\n" +

"LEFT JOIN `course` ON `attendance`.`Course\_idCourse` = `course`.`idCourse`\r\n" +

"WHERE `students`.`Name` LIKE @fN\r\n" +

"ORDER BY `attendance`.`idAttendance`;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewAttendance.DataSource = null;

dataGridViewAttendance.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewAttendance.DataSource = dt;

// Adjust column headers if necessary

dataGridViewAttendance.Columns["idAttendance"].HeaderText = "Id";

dataGridViewAttendance.Columns["Date"].HeaderText = "Дата и время";

dataGridViewAttendance.Columns["IsPresent"].HeaderText = "Присутствие";

dataGridViewAttendance.Columns["studentsName"].HeaderText = "Студент";

dataGridViewAttendance.Columns["courseName"].HeaderText = "Курс";

dataGridViewAttendance.Columns["idAttendance"].Width = 70;

dataGridViewAttendance.Columns["Date"].Width = 150;

dataGridViewAttendance.Columns["IsPresent"].Width = 110;

dataGridViewAttendance.Columns["studentsName"].Width = 170;

dataGridViewAttendance.Columns["courseName"].Width = 400;

}

// Driving practices

private void tabPageDrivingPractice\_Enter(object sender, EventArgs e)

{

// Refresh dataTable

textBoxDrivingPracticeFilter\_TextChanged(sender, e);

// Initialize database connection and data adapter

DB dB = new DB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

// DataTable for the combobox

DataTable dtStudensNames = new DataTable();

MySqlCommand cmdStudensNames = new MySqlCommand("SELECT `idStudents`, `Name` FROM `students`", dB.GetConnection());

dB.openConnection();

adapter.SelectCommand = cmdStudensNames;

adapter.Fill(dtStudensNames);

comboBoxDrivingPracticeStudentName.DataSource = dtStudensNames;

comboBoxDrivingPracticeStudentName.DisplayMember = "Name";

comboBoxDrivingPracticeStudentName.ValueMember = "idStudents";

dB.closeConnection();

}

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

{

// Checking for an empty string

if (comboBoxDrivingPracticeStudentName.Text.Length < 1 ||

dateTimePickerDrivingPracticeDateTime.Text.Length < 1)

{

MessageBox.Show("Пожалуйста заполенте все поля!");

return;

}

DB dB = new DB();

int studentId = Convert.ToInt32(comboBoxDrivingPracticeStudentName.SelectedValue);

DateTime date = Convert.ToDateTime(dateTimePickerDrivingPracticeDateTime.Value);

string instructor = textBoxDrivingPracticeInstructorName.Text;

double durationInHours = Convert.ToDouble(numericUpDownDrivingPracticeDurationInHours.Value);

MySqlCommand cmd = new MySqlCommand("INSERT INTO `drivingpractice` (`Students\_idStudents`, `Date`, `Instructor`, `DurationInHours`) VALUES (@sId, @Dt, @ins, @DiH)", dB.GetConnection());

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = studentId;

cmd.Parameters.AddWithValue("@Dt", date);

cmd.Parameters.Add("@ins", MySqlDbType.VarChar).Value = instructor;

cmd.Parameters.Add("@DiH", MySqlDbType.VarChar).Value = durationInHours;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

{

DataTable dt = new DataTable();

cmd = new MySqlCommand("SELECT \* FROM `drivingpractice` WHERE `idDrivingPractice` = (SELECT MAX(`idDrivingPractice`) FROM `drivingpractice`);", dB.GetConnection());

MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int drivingPracticeId = Convert.ToInt32(row["idDrivingPractice"]);

DrivingPractice practice = new DrivingPractice(drivingPracticeId, studentId, date, instructor, durationInHours);

courseAnalysis.AddDrivingPractice(studentId, practice);

}

MessageBox.Show("Запись добавлена!");

}

else

MessageBox.Show("Ошибка при добавлении!");

comboBoxDrivingPracticeStudentName.SelectedIndex = 0;

textBoxDrivingPracticeInstructorName.Text = "";

numericUpDownDrivingPracticeDurationInHours.Value = 1;

dB.closeConnection();

tabPageDrivingPractice\_Enter(sender, e);

}

private void textBoxDrivingPracticeFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxDrivingPracticeFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT `idDrivingPractice`, `students`.`Name` AS `studentName`, `Date`, `Instructor`, `DurationInHours` FROM `DrivingPractice`\r\n" +

"LEFT JOIN `students` ON `DrivingPractice`.`Students\_idStudents` = `students`.`idStudents`\r\n" +

"WHERE `students`.`Name` LIKE @fN " +

"ORDER BY `idDrivingPractice`;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewDrivingPractice.DataSource = null;

dataGridViewDrivingPractice.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewDrivingPractice.DataSource = dt;

// Adjust column headers if necessary

dataGridViewDrivingPractice.Columns["idDrivingPractice"].HeaderText = "Id";

dataGridViewDrivingPractice.Columns["studentName"].HeaderText = "Студент";

dataGridViewDrivingPractice.Columns["Date"].HeaderText = "Дата и время";

dataGridViewDrivingPractice.Columns["Instructor"].HeaderText = "Инструктор";

dataGridViewDrivingPractice.Columns["DurationInHours"].HeaderText = "Длит.(час)";

dataGridViewDrivingPractice.Columns["idDrivingPractice"].Width = 70;

dataGridViewDrivingPractice.Columns["studentName"].Width = 170;

dataGridViewDrivingPractice.Columns["Date"].Width = 150;

dataGridViewDrivingPractice.Columns["Instructor"].Width = 300;

dataGridViewDrivingPractice.Columns["DurationInHours"].Width = 110;

}

// Driving errors

private void tabPageDrivingErrors\_Enter(object sender, EventArgs e)

{

// Refresh dataTable

textBoxDrivingErrorsFilter\_TextChanged(sender, e);

// Initialize database connection and data adapter

DB dB = new DB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

// DataTable for the combobox

DataTable dtStudensNames = new DataTable();

MySqlCommand cmdStudensNames = new MySqlCommand("SELECT `idStudents`, `Name` FROM `students`", dB.GetConnection());

dB.openConnection();

adapter.SelectCommand = cmdStudensNames;

adapter.Fill(dtStudensNames);

comboBoxDrivingErrorsStudentName.DataSource = dtStudensNames;

comboBoxDrivingErrorsStudentName.DisplayMember = "Name";

comboBoxDrivingErrorsStudentName.ValueMember = "idStudents";

dB.closeConnection();

}

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

{

// Checking for an empty string

if (comboBoxDrivingErrorsStudentName.Text.Length < 1 ||

dateTimePickerDrivingErrorsDateTime.Text.Length < 1)

{

MessageBox.Show("Пожалуйста заполенте все поля!");

return;

}

DB dB = new DB();

int studentId = Convert.ToInt32(comboBoxDrivingErrorsStudentName.SelectedValue);

DateTime date = Convert.ToDateTime(dateTimePickerDrivingErrorsDateTime.Value);

string description = textBoxDrivingErrorsDescription.Text;

MySqlCommand cmd = new MySqlCommand("INSERT INTO `drivingerrors` (`Students\_idStudents`, `Date`, `ErrorDescription`) VALUES (@sId, @dT, @erD);", dB.GetConnection());

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = studentId;

cmd.Parameters.AddWithValue("@Dt", date);

cmd.Parameters.Add("@erD", MySqlDbType.VarChar).Value = description;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

{

DataTable dt = new DataTable();

cmd = new MySqlCommand("SELECT \* FROM `drivingerrors` WHERE `idDrivingErrors` = (SELECT MAX(`idDrivingErrors`) FROM `drivingerrors`);", dB.GetConnection());

MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int errorsId = Convert.ToInt32(row["idDrivingErrors"]);

DrivingErrors errors = new DrivingErrors(errorsId, studentId, date, description);

courseAnalysis.AddDrivingErrors(studentId, errors);

}

MessageBox.Show("Запись добавлена!");

}

else

MessageBox.Show("Ошибка при добавлении!");

comboBoxDrivingErrorsStudentName.SelectedIndex = 0;

textBoxDrivingErrorsDescription.Text = "";

dB.closeConnection();

tabPageDrivingErrors\_Enter(sender, e);

}

private void textBoxDrivingErrorsFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxDrivingErrorsFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT `idDrivingErrors`, `students`.`Name` AS `studentName`, `Date`, `ErrorDescription` FROM `drivingerrors`\r\n" +

"LEFT JOIN `students` ON `drivingerrors`.`Students\_idStudents` = `students`.`idStudents`\r\n" +

"WHERE `students`.`Name` LIKE @fN\r\n" +

"ORDER BY `idDrivingErrors`;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewDrivingErrors.DataSource = null;

dataGridViewDrivingErrors.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewDrivingErrors.DataSource = dt;

// Adjust column headers if necessary

dataGridViewDrivingErrors.Columns["idDrivingErrors"].HeaderText = "Id";

dataGridViewDrivingErrors.Columns["studentName"].HeaderText = "Студент";

dataGridViewDrivingErrors.Columns["Date"].HeaderText = "Дата и время";

dataGridViewDrivingErrors.Columns["ErrorDescription"].HeaderText = "Описание ошибки";

dataGridViewDrivingErrors.Columns["idDrivingErrors"].Width = 70;

dataGridViewDrivingErrors.Columns["studentName"].Width = 170;

dataGridViewDrivingErrors.Columns["Date"].Width = 150;

dataGridViewDrivingErrors.Columns["ErrorDescription"].Width = 500;

}

// Test results

private void tabPageTestResults\_Enter(object sender, EventArgs e)

{

// Refresh dataTable

textBoxTestResultsFilter\_TextChanged(sender, e);

// Initialize database connection and data adapter

DB dB = new DB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

// DataTable for the combobox

DataTable dtStudensNames = new DataTable();

MySqlCommand cmdStudensNames = new MySqlCommand("SELECT `idStudents`, `Name` FROM `students`", dB.GetConnection());

dB.openConnection();

adapter.SelectCommand = cmdStudensNames;

adapter.Fill(dtStudensNames);

comboBoxTestResultsStudentName.DataSource = dtStudensNames;

comboBoxTestResultsStudentName.DisplayMember = "Name";

comboBoxTestResultsStudentName.ValueMember = "idStudents";

dB.closeConnection();

}

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

{

// Checking for an empty string

if (comboBoxTestResultsStudentName.Text.Length < 1 ||

dateTimePickerTestResultsDateTime.Text.Length < 1)

{

MessageBox.Show("Пожалуйста заполенте все поля!");

return;

}

DB dB = new DB();

int studentId = Convert.ToInt32(comboBoxTestResultsStudentName.SelectedValue);

DateTime date = Convert.ToDateTime(dateTimePickerTestResultsDateTime.Value);

double score = Convert.ToDouble(numericUpDownTestResultsScore.Value);

MySqlCommand cmd = new MySqlCommand("INSERT INTO `testresults` (`Students\_idStudents`, `Date`, `Score`) VALUES (@sId, @dT, @sc);", dB.GetConnection());

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = studentId;

cmd.Parameters.AddWithValue("@Dt", date);

cmd.Parameters.Add("@sc", MySqlDbType.VarChar).Value = score;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

{

DataTable dt = new DataTable();

cmd = new MySqlCommand("SELECT \* FROM `testresults` WHERE `idTestResults` = (SELECT MAX(`idTestResults`) FROM `testresults`);", dB.GetConnection());

MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int testId = Convert.ToInt32(row["idTestResults"]);

TestResults test = new TestResults(testId, studentId, date, score);

courseAnalysis.AddTestResults(studentId, test);

}

MessageBox.Show("Запись добавлена!");

}

else

MessageBox.Show("Ошибка при добавлении!");

comboBoxTestResultsStudentName.SelectedIndex = 0;

numericUpDownTestResultsScore.Value = 0;

dB.closeConnection();

tabPageTestResults\_Enter(sender, e);

}

private void textBoxTestResultsFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxTestResultsFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT `idTestResults`, `students`.`Name` AS `studentName`, `Date`, `Score` FROM `testresults`\r\n" +

"LEFT JOIN `students` ON `testresults`.`Students\_idStudents` = `students`.`idStudents`\r\n" +

"WHERE `students`.`Name` LIKE @fN\r\n" +

"ORDER BY `idTestResults`;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewTestResults.DataSource = null;

dataGridViewTestResults.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewTestResults.DataSource = dt;

// Adjust column headers if necessary

dataGridViewTestResults.Columns["idTestResults"].HeaderText = "Id";

dataGridViewTestResults.Columns["studentName"].HeaderText = "Студент";

dataGridViewTestResults.Columns["Date"].HeaderText = "Дата и время";

dataGridViewTestResults.Columns["Score"].HeaderText = "Баллы";

dataGridViewTestResults.Columns["idTestResults"].Width = 70;

dataGridViewTestResults.Columns["studentName"].Width = 170;

dataGridViewTestResults.Columns["Date"].Width = 150;

dataGridViewTestResults.Columns["Score"].Width = 100;

}

// Exam results

private void tabPageExamResults\_Enter(object sender, EventArgs e)

{

// Refresh dataTable

textBoxExamResultsFilter\_TextChanged(sender, e);

// Initialize database connection and data adapter

DB dB = new DB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

// DataTable for the combobox

DataTable dtStudensNames = new DataTable();

MySqlCommand cmdStudensNames = new MySqlCommand("SELECT `idStudents`, `Name` FROM `students`", dB.GetConnection());

dB.openConnection();

adapter.SelectCommand = cmdStudensNames;

adapter.Fill(dtStudensNames);

comboBoxExamResultsStudentName.DataSource = dtStudensNames;

comboBoxExamResultsStudentName.DisplayMember = "Name";

comboBoxExamResultsStudentName.ValueMember = "idStudents";

dB.closeConnection();

}

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

{

// Checking for an empty string

if (comboBoxExamResultsStudentName.Text.Length < 1 ||

dateTimePickerExamResultsDateTime.Text.Length < 1)

{

MessageBox.Show("Пожалуйста заполенте все поля!");

return;

}

DB dB = new DB();

int studentId = Convert.ToInt32(comboBoxExamResultsStudentName.SelectedValue);

DateTime date = Convert.ToDateTime(dateTimePickerExamResultsDateTime.Value);

bool isPassed = checkBoxExamResultsIsPassed.Checked;

MySqlCommand cmd = new MySqlCommand("INSERT INTO `examresults` (`Students\_idStudents`, `Date`, `IsPassed`) VALUES (@sId, @dT, @iP);", dB.GetConnection());

cmd.Parameters.Add("@sId", MySqlDbType.VarChar).Value = studentId;

cmd.Parameters.AddWithValue("@Dt", date);

cmd.Parameters.Add("@iP", MySqlDbType.VarChar).Value = isPassed == true ? 1 : 0;

dB.openConnection();

if (cmd.ExecuteNonQuery() == 1)

{

DataTable dt = new DataTable();

cmd = new MySqlCommand("SELECT \* FROM `examresults` WHERE `idExamResults` = (SELECT MAX(`idExamResults`) FROM `examresults`);", dB.GetConnection());

MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);

adapter.Fill(dt);

foreach (DataRow row in dt.Rows)

{

int examId = Convert.ToInt32(row["idExamResults"]);

ExamResults exam = new ExamResults(examId, studentId, date, isPassed);

courseAnalysis.AddExamResults(studentId, exam);

}

MessageBox.Show("Запись добавлена!");

}

else

MessageBox.Show("Ошибка при добавлении!");

comboBoxExamResultsStudentName.SelectedIndex = 0;

checkBoxExamResultsIsPassed.Checked = false;

dB.closeConnection();

tabPageExamResults\_Enter(sender, e);

}

private void textBoxExamResultsFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxExamResultsFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT `idExamResults`, `students`.`Name` AS `studentName`, `Date`, `IsPassed` FROM `examresults`\r\n" +

"LEFT JOIN `students` ON `examresults`.`Students\_idStudents` = `students`.`idStudents`\r\n" +

"WHERE `students`.`Name` LIKE @fN\r\n" +

"ORDER BY `idExamResults`;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Clear the DataGridView before adding new items

dataGridViewExamResults.DataSource = null;

dataGridViewExamResults.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewExamResults.DataSource = dt;

// Adjust column headers if necessary

dataGridViewExamResults.Columns["idExamResults"].HeaderText = "Id";

dataGridViewExamResults.Columns["studentName"].HeaderText = "Студент";

dataGridViewExamResults.Columns["Date"].HeaderText = "Дата и время";

dataGridViewExamResults.Columns["IsPassed"].HeaderText = "Экзамен пройден";

dataGridViewExamResults.Columns["idExamResults"].Width = 70;

dataGridViewExamResults.Columns["studentName"].Width = 170;

dataGridViewExamResults.Columns["Date"].Width = 150;

dataGridViewExamResults.Columns["IsPassed"].Width = 200;

}

// Course analysis

private void tabPageCourseAnalysis\_Enter(object sender, EventArgs e)

{

textBoxCourseAnalysisFilter\_TextChanged(sender, e);

}

private void textBoxCourseAnalysisFilter\_TextChanged(object sender, EventArgs e)

{

// Initialize database connection and data adapter

DB dB = new DB();

DataTable dt = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

string filter = textBoxCourseAnalysisFilter.Text;

// Create and configure the SQL command

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM `course` WHERE `Name` LIKE @fN;", dB.GetConnection());

cmd.Parameters.Add("@fN", MySqlDbType.VarChar).Value = filter + "%";

// Assign the command to the adapter

adapter.SelectCommand = cmd;

// Fill the DataTable with data from the database

adapter.Fill(dt);

// Add new columns to the DataTable

dt.Columns.Add("Popularity", typeof(double));

dt.Columns.Add("Quality", typeof(double));

dt.Columns.Add("CostEffectiveness", typeof(double));

// Clear the DataGridView before adding new items

dataGridViewCourseAnalysis.DataSource = null;

dataGridViewCourseAnalysis.Columns.Clear();

// Set the DataSource of the DataGridView to the DataTable

dataGridViewCourseAnalysis.DataSource = dt;

// Adjust column headers if necessary

dataGridViewCourseAnalysis.Columns["idCourse"].HeaderText = "Id";

dataGridViewCourseAnalysis.Columns["Name"].HeaderText = "Название курса";

dataGridViewCourseAnalysis.Columns["Cost"].HeaderText = "Стоимость";

dataGridViewCourseAnalysis.Columns["DurationInDays"].HeaderText = "Длительность\n(дн.)";

dataGridViewCourseAnalysis.Columns["Popularity"].HeaderText = "Популярность";

dataGridViewCourseAnalysis.Columns["Quality"].HeaderText = "Качество";

dataGridViewCourseAnalysis.Columns["CostEffectiveness"].HeaderText = "Эффективность Стоимости";

dataGridViewCourseAnalysis.Columns["idCourse"].Width = 70;

dataGridViewCourseAnalysis.Columns["Name"].Width = 330;

dataGridViewCourseAnalysis.Columns["Cost"].Width = 120;

dataGridViewCourseAnalysis.Columns["DurationInDays"].Width = 140;

dataGridViewCourseAnalysis.Columns["Popularity"].Width = 150;

dataGridViewCourseAnalysis.Columns["Quality"].Width = 120;

dataGridViewCourseAnalysis.Columns["CostEffectiveness"].Width = 150;

// Создаем объект анализа курсов

foreach (DataRow row in dt.Rows)

{

int courseId = Convert.ToInt32(row["idCourse"]);

string courseName = row["Name"].ToString();

double courseCost = Convert.ToDouble(row["Cost"]);

int courseDuration = Convert.ToInt32(row["DurationInDays"]);

double popularity = courseAnalysis.GetCoursePopularity(courseId);

double quality = courseAnalysis.GetCourseQuality(courseId);

double costEffectiveness = courseAnalysis.GetCourseCostEffectiveness(courseId);

row["Popularity"] = popularity;

row["Quality"] = Math.Round(quality, 3);

row["CostEffectiveness"] = Math.Round(costEffectiveness, 3);

}

// Обновляем DataGridView

dataGridViewCourseAnalysis.DataSource = dt;

}

}

}

DB.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Driver\_s\_Course\_Terminal

{

internal class DB

{

MySqlConnection connection = new MySqlConnection(

"server=localhost;port=3306;username=root;password=root;database=DriverCourses");

public void openConnection()

{

if (connection.State == System.Data.ConnectionState.Closed)

connection.Open();

}

public void closeConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

connection.Close();

}

public MySqlConnection GetConnection()

{

return connection;

}

}

}

DrivingCourses.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Driver\_s\_Course\_Terminal

{

public class Student

{

public int StudentId { get; set; }

public string Name { get; set; }

public string LicenseCategory { get; set; } // Категории прав (A, B, C, D, E)

public List<Attendance> Attendances { get; set; }

public List<DrivingPractice> DrivingPractices { get; set; }

public List<DrivingErrors> DrivingErrors { get; set; }

public List<TestResults> TestResults { get; set; }

public List<ExamResults> ExamResults { get; set; }

public Student(int studentId, string name, string category)

{

StudentId = studentId;

Name = name;

LicenseCategory = category;

Attendances = new List<Attendance>();

DrivingPractices = new List<DrivingPractice>();

DrivingErrors = new List<DrivingErrors>();

TestResults = new List<TestResults>();

ExamResults = new List<ExamResults>();

}

}

public class Course

{

public int CourseId { get; set; }

public string Name { get; set; }

public double Cost { get; set; }

public int DurationInDays { get; set; }

public List<Student> Students { get; set; }

public Course(int courseId, string name, double cost, int durationInDays)

{

CourseId = courseId;

Name = name;

Cost = cost;

DurationInDays = durationInDays;

Students = new List<Student>();

}

// Виртуальные методы, которые могут быть переопределены в наследниках

public virtual double GetPopularity()

{

return Students.Count;

}

public virtual double GetQuality()

{

double passedCount = Students

.SelectMany(s => s.ExamResults)

.Count(er => er.IsPassed);

return Students.Count > 0 ? passedCount / Students.Count : 0;

}

public virtual double GetCostEffectiveness()

{

return DurationInDays > 0 ? Cost / DurationInDays : 0;

}

}

public class CourseHasStudents

{

public int CourseHasStudentsId { get; set; }

public int StudentId { get; set; }

public int CourseId { get; set; }

public CourseHasStudents(int courseHasStudentsId, int studentId, int courseId)

{

CourseHasStudentsId = courseHasStudentsId;

StudentId = studentId;

CourseId = courseId;

}

}

public class Attendance

{

public int AttendanceId { get; set; }

public int StudentId { get; set; }

public int CourseId { get; set; }

public DateTime Date { get; set; }

public bool IsPresent { get; set; }

public Attendance(int attendanceId, int studentId, int courseId, DateTime date, bool isPresent)

{

AttendanceId = attendanceId;

StudentId = studentId;

CourseId = courseId;

Date = date;

IsPresent = isPresent;

}

}

public class DrivingPractice

{

public int PracticeId { get; set; }

public int StudentId { get; set; }

public DateTime Date { get; set; }

public string Instructor { get; set; }

public double DurationInHours { get; set; }

public DrivingPractice(int practiceId, int studentId, DateTime date, string instructor, double durationInHours)

{

PracticeId = practiceId;

StudentId = studentId;

Date = date;

Instructor = instructor;

DurationInHours = durationInHours;

}

}

public class DrivingErrors

{

public int ErrorId { get; set; }

public int StudentId { get; set; }

public DateTime Date { get; set; }

public string ErrorDescription { get; set; }

public DrivingErrors(int errorId, int studentId, DateTime date, string errorDescription)

{

ErrorId = errorId;

StudentId = studentId;

Date = date;

ErrorDescription = errorDescription;

}

}

public class TestResults

{

public int TestId { get; set; }

public int StudentId { get; set; }

public DateTime Date { get; set; }

public double Score { get; set; }

public TestResults(int testId, int studentId, DateTime date, double score)

{

TestId = testId;

StudentId = studentId;

Date = date;

Score = score;

}

}

public class ExamResults

{

public int ExamId { get; set; }

public int StudentId { get; set; }

public DateTime Date { get; set; }

public bool IsPassed { get; set; }

public ExamResults(int examId, int studentId, DateTime date, bool isPassed)

{

ExamId = examId;

StudentId = studentId;

Date = date;

IsPassed = isPassed;

}

}

public class CourseAnalysis

{

public static List<Course> Courses { get; set; }

public CourseAnalysis()

{

Courses = new List<Course>();

}

public void AddCourse(Course course)

{

Courses.Add(course);

}

public void AddCoursehasStudent(int courseId, Student student)

{

foreach (Course course in Courses)

{

if (course.CourseId == courseId)

{

course.Students.Add(student);

break;

}

}

}

public void AddAttendance(int courseId, int studentId, Attendance attendance)

{

foreach (Course course in Courses)

{

if (course.CourseId == courseId)

{

foreach (Student student in course.Students)

{

if (student.StudentId == studentId)

{

student.Attendances.Add(attendance);

break;

}

}

break;

}

}

}

public void AddDrivingPractice(int studentId, DrivingPractice practice)

{

bool added = false;

foreach (Course course in Courses)

{

if (added)

break;

foreach (Student student in course.Students)

{

if (student.StudentId == studentId)

{

student.DrivingPractices.Add(practice);

added = true;

break;

}

}

}

}

public void AddDrivingErrors(int studentId, DrivingErrors errors)

{

bool added = false;

foreach (Course course in Courses)

{

if (added)

break;

foreach (Student student in course.Students)

{

if (student.StudentId == studentId)

{

student.DrivingErrors.Add(errors);

added = true;

break;

}

}

}

}

public void AddTestResults(int studentId, TestResults test)

{

bool added = false;

foreach (Course course in Courses)

{

if (added)

break;

foreach (Student student in course.Students)

{

if (student.StudentId == studentId)

{

student.TestResults.Add(test);

added = true;

break;

}

}

}

}

public void AddExamResults(int studentId, ExamResults exam)

{

bool added = false;

foreach (Course course in Courses)

{

if (added)

break;

foreach (Student student in course.Students)

{

if (student.StudentId == studentId)

{

student.ExamResults.Add(exam);

added = true;

break;

}

}

}

}

public double GetCoursePopularity(int courseId)

{

Course course = Courses.FirstOrDefault(c => c.CourseId == courseId);

return course != null ? course.GetPopularity() : 0;

}

public double GetCourseQuality(int courseId)

{

Course course = Courses.FirstOrDefault(c => c.CourseId == courseId);

return course != null ? course.GetQuality() : 0;

}

public double GetCourseCostEffectiveness(int courseId)

{

Course course = Courses.FirstOrDefault(c => c.CourseId == courseId);

return course != null ? course.GetCostEffectiveness() : 0;

}

}

}