МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

(РУТ (МИИТ)

Кафедра «Цифровые технологии управления транспортными процессами»

**КУРСОВАЯ РАБОТА**

**По дисциплине «Технологии программирования»**

**По теме: «Система тестирования»**

Группа: УИС-212

Студенты:

Загинайченко Д.Д

Антонов Е.М.

Филиппова О.С.

Преподаватель: Новиков А.И.

**Москва 2021**

**Задания по курсовой работе для дисциплины «Технологии программирования»**

Задание выдается на группу из 2-4 человек и выполняется на лабораторных занятиях и занятиях по курсовой работе.

Язык программирования – C#.

Описание проекта разрабатываемой информационной системы должно быть выполнено на языке UML в среде UML Designer.

Документация по заданию должна содержать:

1. Задание по курсовой работе.
2. Диаграммы на языке UML.
3. Краткое описание каждого класса и метода.
4. Руководство пользователя со скриншотами интерфейса программы.
5. Набор тестов и результатов для отладки отдельных методов, модулей и комплексной отладки задания.
6. Тексты программ с комментариями

Диаграммы языка UML:

* Диаграмма вариантов использования (Use Case Diagram)
* Диаграмма классов (Class Diagam)
* Диаграмма деятельности (Activity Diagram)
* Диаграмма состояний (State Machine Diagram)
* Диаграмма последовательности (Sequence Diagram)
* Диаграмма компонентов (Component Diagram)
* Диаграмма развертывания (размещения) (Deployment Diagram)

Необходимо разработать информационную систему в соответствии с заданием. В задании перечислены пользователи системы (актеры) и варианты использования системы актерами. Можно скорректировать список актеров и вариантов использования по согласованию с преподавателем.

Предлагаемый перечень заданий может быть дополнен любым другим заданием по согласованию с преподавателем. Приветствуется разработка проекта на основе реальных заданий.

Общие требования по выполнению задания:

* Каждый пользователь информационной системы должен регистрироваться в системе сам или это делает администратор
* При входе в систему каждый пользователь вводит свой логин и пароль
* Данные в системе хранятся в файлах
* При запуске системы все данные из файлов считываются в оперативную память и хранятся в форме таблиц.
* По завершению работы с системой данные из таблиц записываются обратно в файлы. Файлы при этом не обновляются, а создаются заново на основании данных из таблиц.

1. **Система тестирования**

Администратор

* Составление списков групп студентов, включает регистрацию студента
* Регистрация преподавателя, редактирование его данных
* Составление списка дисциплин и тестирований для них, предоставление доступа к ним преподавателям
* Возможность посмотреть результаты тестирования для одного студента

Преподаватель

* Подготовка списка вопросов для тестирования и вариантов ответов на каждый тест. Правильным может быть один или несколько ответов

Студент

* Прохождение тестирования

-Распределение работ:

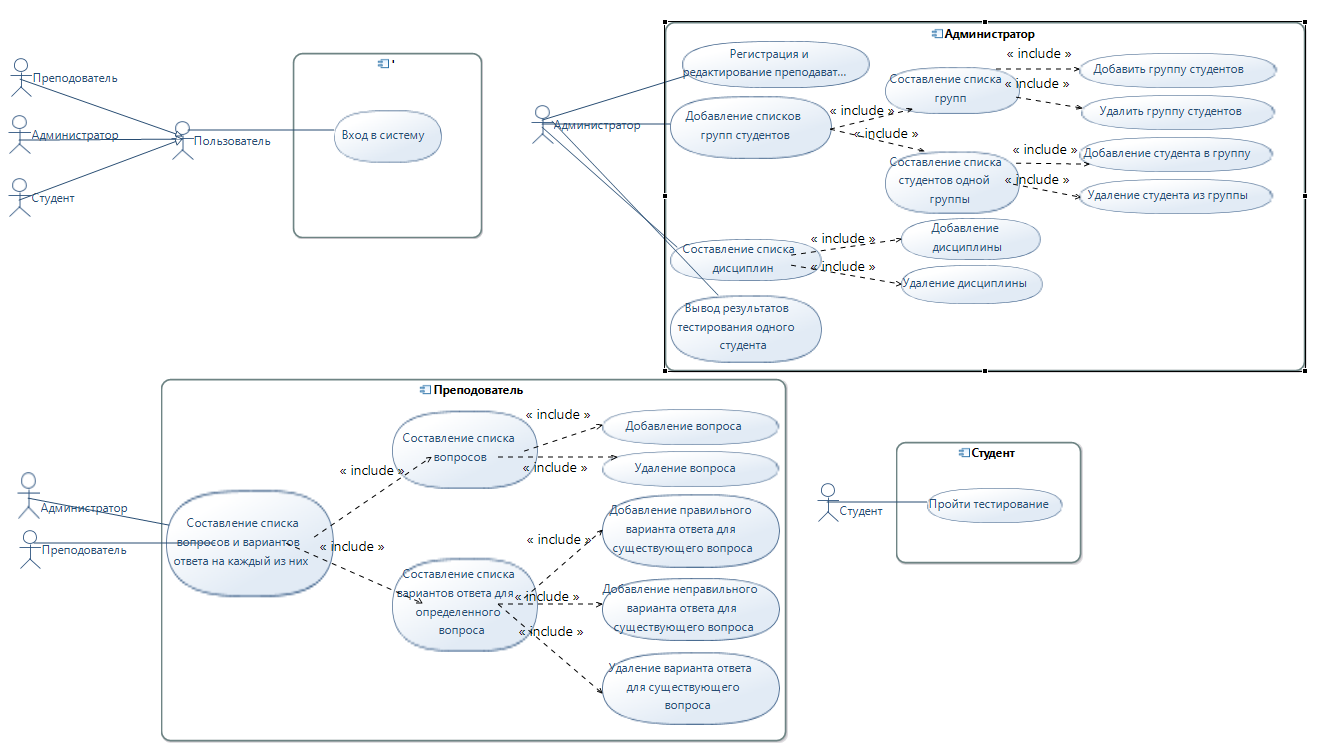
* Антонов Е.М. – Диаграммы
* Загинайченко Д.Д. – Программа, Диаграммы
* Филиппова О.С. – Диаграммы, Руководство пользователя

-Среды разработки

* UML Designer
* Visual Studio 2019
* Microsoft SQL Server Management Studio

**Диаграммы на языке UML**

1. **Диаграмма вариантов использования (Use Case Diagram)**

  
  
Диаграмма вариантов использования предназначена для описания взаимодействия пользователей с системой. На диаграмме показываются не конкретные пользователи, а их роли, которые они выполняют при взаимодействии с системой. С системой могут взаимодействовать как люди, так и некоторые приложения, а также некоторые устройства.  
Отношение включения используется для связи двух вариантов использования, чтобы показать, что при работе одного варианта использования выполняется другой вариант использования. Актер напрямую не соединяется со включаемым вариантом использования.  
Отношение расширения используется для связи двух вариантов использования, чтобы показать, что некоторый вариант использования расширяет возможности основного варианта использования. Стрелка в этом случае направлена на основной вариант использования  
Отношение обобщения связывает либо два варианта использования, либо двух актеров. В случае двух вариантов использования один вариант использования (обобщающий) содержит описание основных действий пользователя, а наследуемые от него варианты использования описывают действия пользователя в конкретных ситуациях.

1. **Поток событий системы**

|  |  |  |  |
| --- | --- | --- | --- |
| **Поток событий «Вход в систему»** | | | |
| 1 | Ввод логина/пароля | 2 | Ищет пользователя в списке введенной должности |
|  |  | 3.1 | Если пользователь не найден, выводит сообщение «Такого пользователя не существует», завершает под-поток |
|  |  | 3.2 | Определяет должность пользователя и открывает соответствующее окно рабочего места |

**Система «Пользователь»**

|  |  |  |  |
| --- | --- | --- | --- |
| **Подсистема «Администратор»** | | | |
| **1 Поток «Регистрация и редактирование преподавателя»** | | | |
| 1 | Заполнение информации о  пользователе, ввод логина, пароля и должности |  |  |
| 2 | Подтверждение ввода данных | 3 | Проверяет в списке пользователей введенной должности на наличие повторений данных |
|  |  | 4.1 | Если данные повторяются,  Выводит сообщение «Такой пользователь уже существует», завершает  под-поток |
|  |  | 4.2 | Вводит данные пользователя в список,  соответсвенный его должности |

|  |  |
| --- | --- |
| **3 Поток «Добавление списков групп студентов»** | |
| 1 | Составление списка групп |
| 2 | Составление списка студентов одной группы |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.1 Под-поток «Составление списка групп»** | | | |
| 1 | Добавляет группу студентов | | |
| 2 |  | 3 | Удаляет выбранную группу |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.2 Под-поток «Составление списка студентов одной группы»** | | | |
| 1 | Добавляет студента в выбранную группу | | |
| 2 | Выбирает студента в выбранной группе для удаления | 3 | Удаляет выбранного студента |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.2.1 Под-поток «Добавляет студента в выбранную группу»** | | | |
| 1 | Заполнение информации о  студенте, ввод логина и пароля |  |  |
| 2 | Подтверждение ввода данных | 3 | Проверяет в списке студентов данной группы на наличие повторений данных |
|  |  | 4.1 | Если данные повторяются,  Выводит сообщение «Такой студент в группе уже существует», завершает под-поток |
|  |  | 4.2 | Вводит данные студента в список данной группы и выводит обновленные данные на экране |

|  |  |  |  |
| --- | --- | --- | --- |
| **2 Поток «Составление списка дисциплин»** | | | |
| 1 | Добавление дисциплины | | |
| 2 | Выбирает дисциплину для удаления | 3 | Удаляет выбранную дисциплину |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.1 Под-поток «Добавляет дисциплину»** | | | |
| 1 | Заполнение информации о дисциплине |  |  |
| 2 | Подтверждение ввода данных | 3 | Проверяет в списке дисциплин на наличие повторений данных |
|  |  | 4.1 | Если данные повторяются,  Выводит сообщение «Такая дисциплина уже существует», завершает под-поток |
|  |  | 4.2 | Вводит данные в список дисциплин и выводит обновленные данные на экране |

|  |  |  |  |
| --- | --- | --- | --- |
| **4 Поток «Вывод результатов тестирования»** | | | |
| 1 | Администратор вводит номер группы и саму дисциплину | 2 | Ищет результаты по введенным данным |
|  |  | 3.1 | Выводит результаты на экран |
|  |  | 3.2 | Если не найдены,  Выводит сообщение «Результаты не найдены, проверьте правильность ввода» |

**Под-система «Преподаватель»**

|  |  |
| --- | --- |
| **Поток «Составление списка вопросов и вариантов ответа на каждый из них»** | |
| 1 | Составление списка вопросов |
| 2 | Составление списка вариантов ответа для определенного вопроса |

|  |  |  |  |
| --- | --- | --- | --- |
| **1.1 Под-поток «Составление списка вопросов»** | | | |
| 1 | Добавление вопроса в новый тест | | |
| 2 | Выбирает вопрос для удаления | 3 | Удаляет выбранный вопрос |

|  |  |  |  |
| --- | --- | --- | --- |
| **1.2 Под-поток «Составление списка вариантов ответа для определенного вопроса»** | | | |
| 1 | Добавление правильного варианта в вопрос | | |
| 2 | Выбирает вопрос для удаления | 3 | Удаляет выбранный вопрос |
| 4 | Добавление неправильного варианта вопроса |  |  |

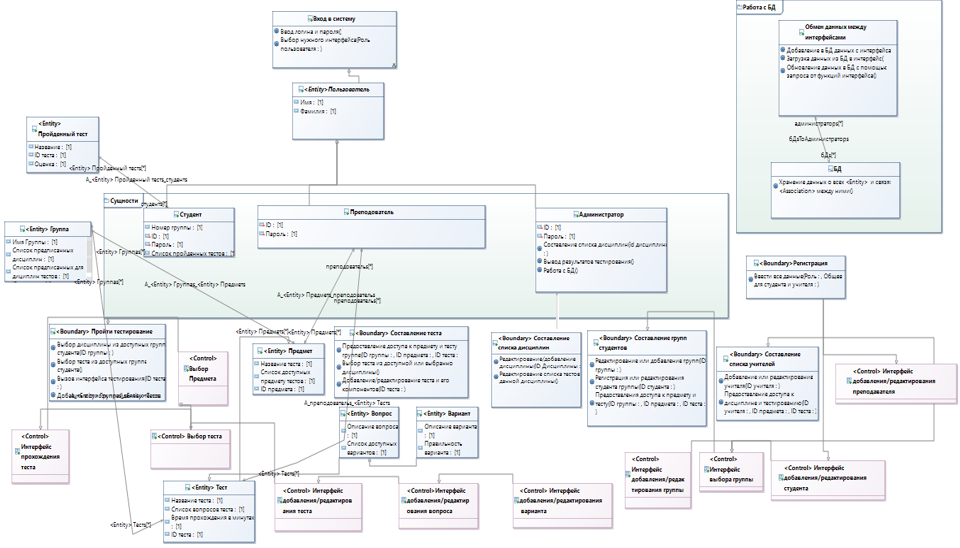
|  |  |  |  |
| --- | --- | --- | --- |
| **1.2.1(2) Под-поток «Добавление (не)правильного варианта ответа для существующего вопроса»** | | | |
| 1 | Ввод варианта |  |  |
| 2 | Подтверждение ввода данных | 3 | Проверяет в списке вариантов одного вопроса на наличие повторений данных |
|  |  | 4.1 | Если данные повторяются,  Выводит сообщение «Такой вопрос уже существует», завершает под-поток |
|  |  | 4.2 | Вводит данные в список ответов и выводит обновленные данные на экране |

|  |  |  |  |
| --- | --- | --- | --- |
| **1.1.1 Под-поток «Добавление вопроса»** | | | |
| 1 | Ввод вопроса |  |  |
| 2 | Подтверждение ввода данных | 3 | Проверяет в списке вопросов на наличие повторений данных |
|  |  | 4.1 | Если данные повторяются,  Выводит сообщение «Такой вопрос уже существует», завершает под-поток |
|  |  | 4.2 | Вводит данные в список вопросов и выводит обновленные данные на экране |

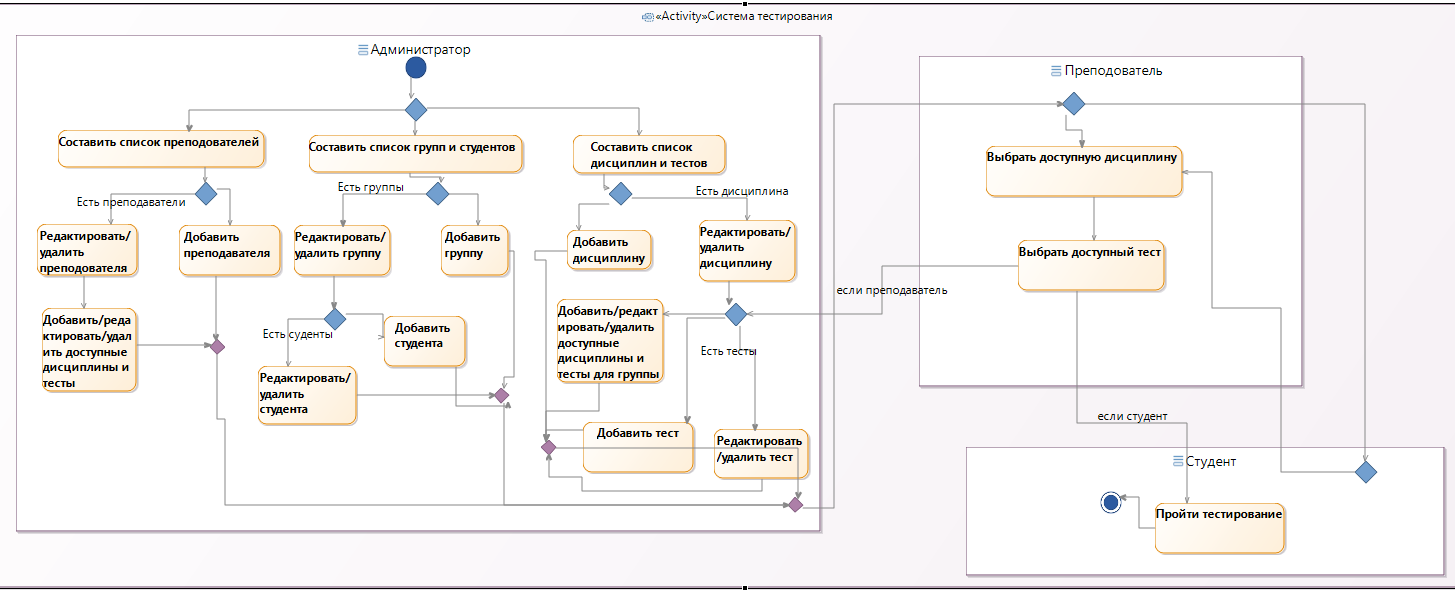
**Подсистема «Студент»**

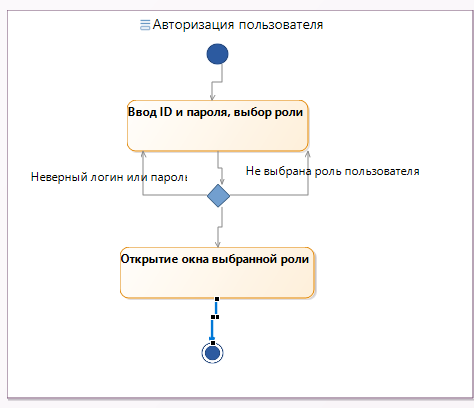
|  |  |  |  |
| --- | --- | --- | --- |
| **Поток**  **«Пройти тестирование»** | | | |
| 1 | Студент выбирает доступную дисциплину |  |  |
| 2 | Студент выбирает доступный тест по данной дисциплине | 3 | Загрузка теста на экран |
| 4 | Студент выбирает варианты ответа | 5 | Вариант сохраняется в базу |
|  |  | 6 | Определяет правильный ли введен вариант |
| 7 | Студент заканчивает тест | 8 | Выводит сообщение “Тест пройден” |
|  |  | 9 | Возвращает в меню списка тестов |

1. **Диаграмма классов (Class Diagam)**   
     
   Диаграмма классов служит для описания классов и их отношений. Диаграмма классов описывает, как данные, хранящиеся в программной системе и базе данных, так и классы, используемые для реализации программной системы.  
   Отношение зависимости также может показывать использование клиентом данных и операций поставщика. В этом случае используется частный случай отношения зависимости, которое называется использованием



1. **Диаграмма деятельности (Activity Diagram)**

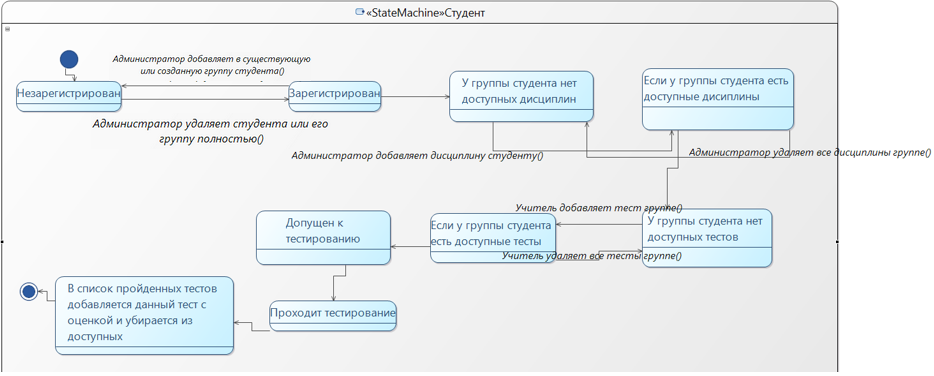




На диаграмме деятельности показывается последовательность выполнения определенных действий, передача данных и изменение состояния объектов данных.

Диаграмма деятельности используется для моделирования потоков управления деятельностью и потоков объектов деятельности.

1. **Диаграмма состояний (State Machine Diagram)**



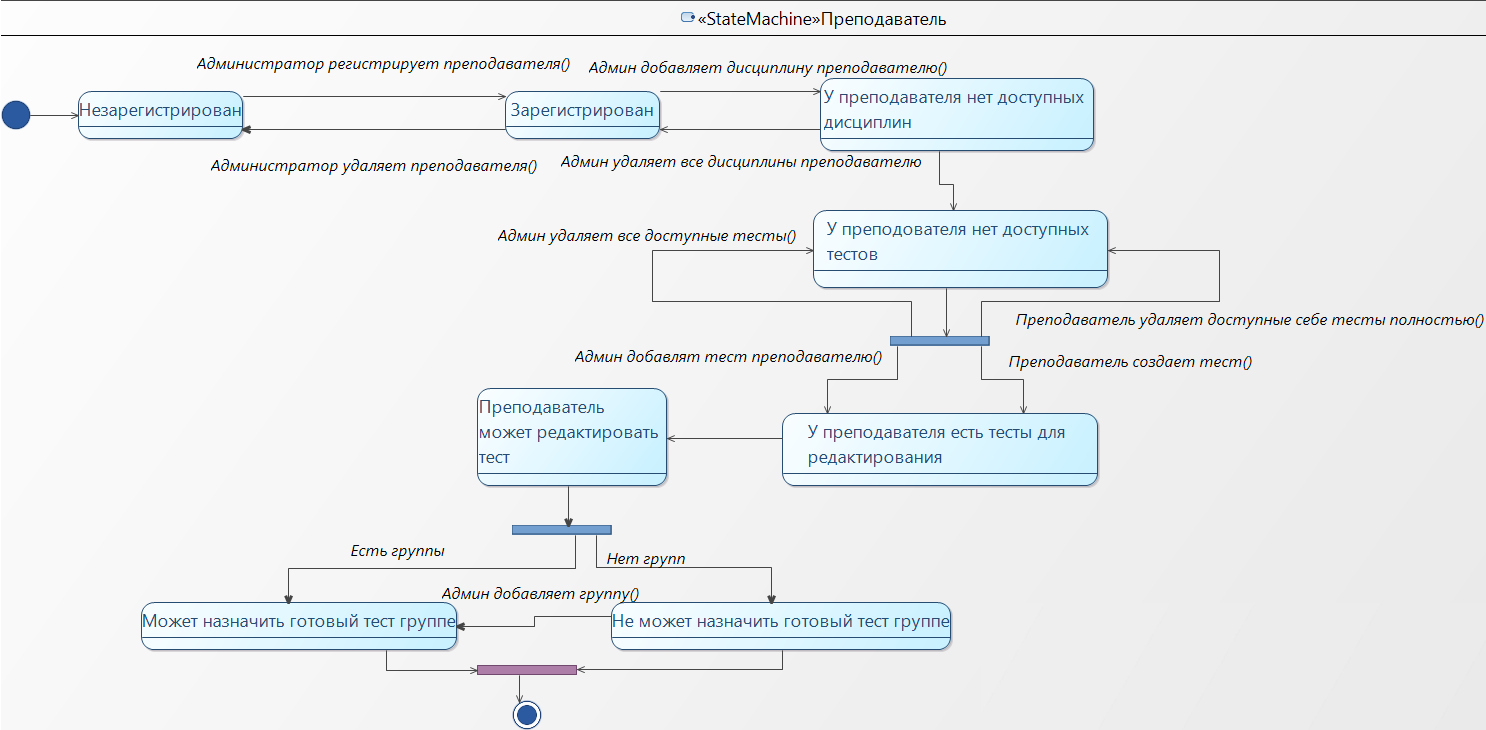


Диаграмма состояний служит для описания состояний, в которых могут находится объекты системы. Диаграмма состояний, как и диаграмма деятельности, относится к динамическим диаграммам, которые показывают, что происходит в системе со временем. На диаграмме изображаются состояния и переходы от одного состояния в другое состояние при возникновении событий.

1. **Диаграмма компонентов (Component Diagram)**

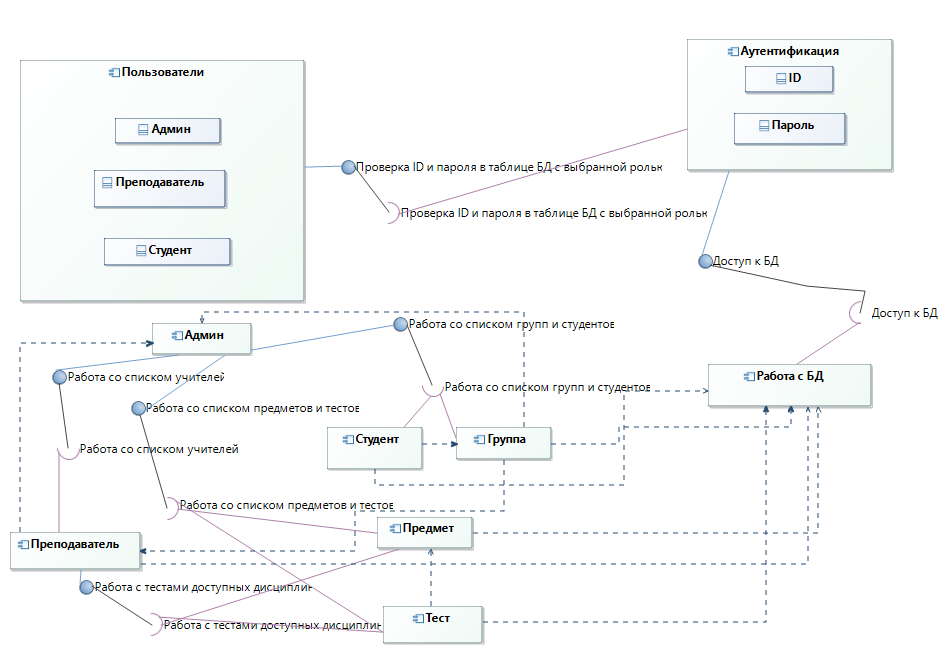


Диаграмма компонентов служит для описания структуры системы.

Отношение соединения (**Connector**) связывает два изображения одного и того же интерфейса, выступающего в роли предоставляемого и требуемого интерфейса, Отношение реализация интерфейса (**Interface Realization**) связывает интерфейс с компонентом или классом.

1. **Диаграмма развертывания (размещения) (Deployment Diagram)**

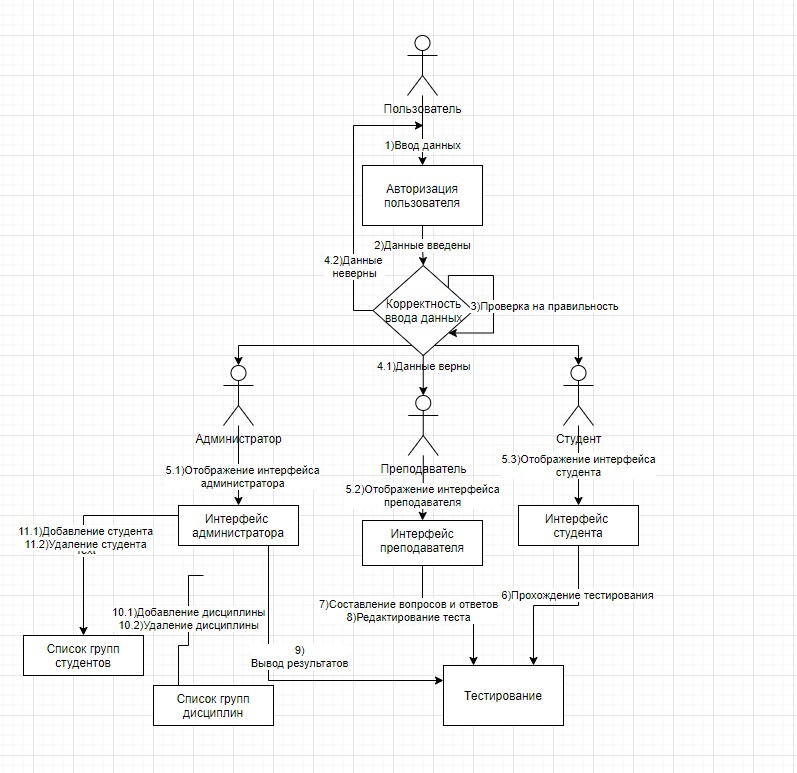
Изображение выглядит как текст

Автоматически созданное описание

Диаграмма развертывания показывает размещение артефактов (физических объектов, созданных человеком) по вычислительным узлам распределенной системы и взаимодействие узлов и артефактов.

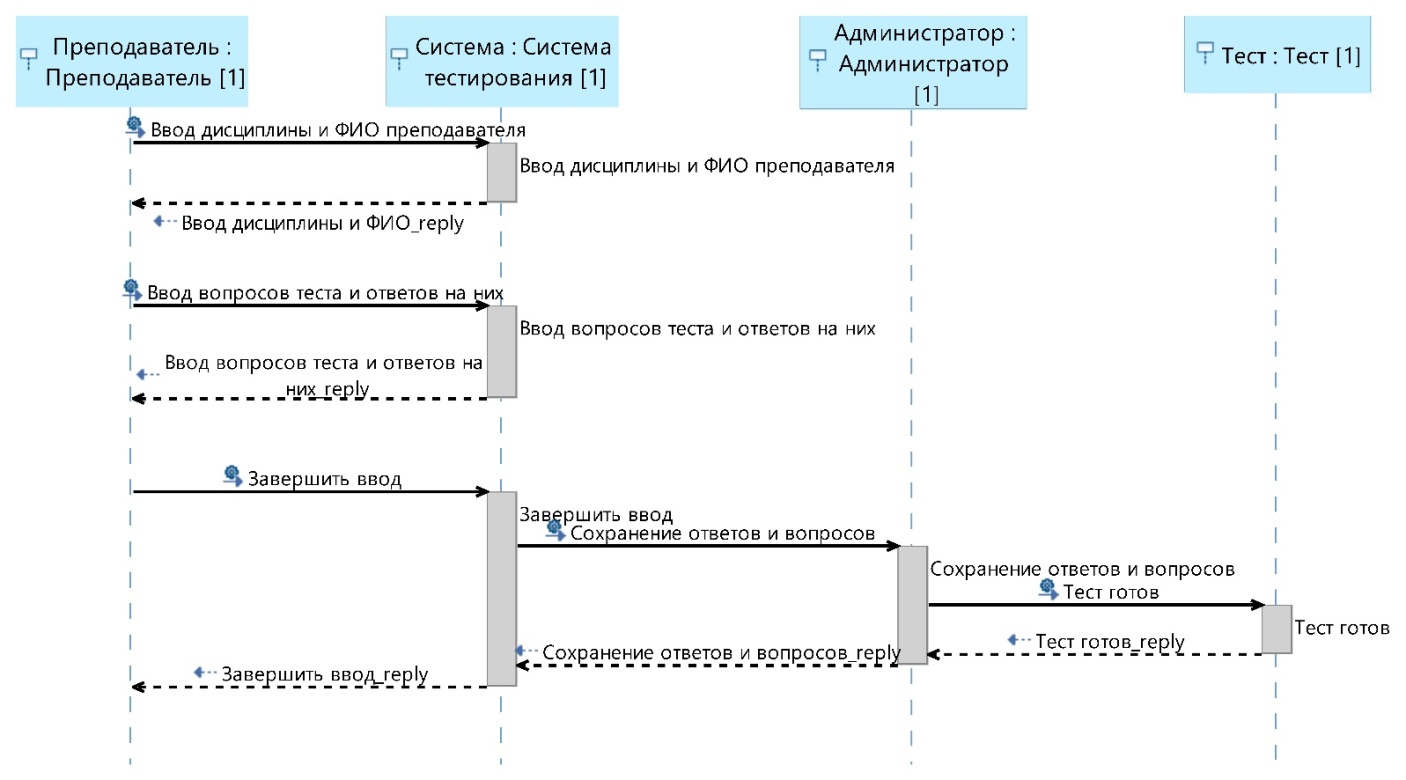
Элемент размещение (**Deployment**) используется для показа размещения артефакта на некотором узле.  
Элемент манифестации (**Manifestation**) устанавливает отношение между компонентом и артефактом, которое означает, что артефакт является физической реализацией компонента

1. **Диаграмма** **коммуникации (Communication Diagram)**

****

1. **Диаграмма** **последовательности (Sequence Diagram)**

Диаграмма последовательности служит для показа обмена сообщениями между актерами и объектами классов. На диаграмме показывается обмен сообщениями во время выполнения некоторого потока событий (сценария) варианта использования.



**Листинг программы на языке C#**

AddEditGroup.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AddEditGroup : Form

{

string Mode { get; set; }

int ID { get; set; }

public AddEditGroup(string mode, int GroupID=-1)

{

InitializeComponent();

Mode = mode;

if(Mode=="Edit")

{

ID = GroupID;

AddValues();

button1.Text = "Изменить";

}

}

private void AddValues()

{

string Text = "select GroupName from Groups where GroupID='" + ID + "'";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand queryID = new SqlCommand(Text, connection);

SqlDataReader dataReader = queryID.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

textBox1.Text = dataReader.GetString(0);

}

}

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

{

if (textBox1.Text != "")

{

Admin Dan = new Admin();

if (Mode == "Edit")

{

Dan.EditGroup(textBox1.Text,ID);

}

else

{

Dan.AddNewGroup(textBox1.Text);

}

Close();

}

else

{

MessageBox.Show("Не все значения введены");

}

}

}

}

AddEditStudent.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AddEditStudent : Form

{

string Mode { get; set; }

string GroupID { get; set; }

int ID { get; set; }

public AddEditStudent(string mode,string group,int StudentID=-1)

{

InitializeComponent();

Mode = mode;

GroupID = group;

ID = StudentID;

label4.Hide();

Look.Hide();

if(mode=="Edit")

{

Look.Show();

label4.Show();

AddValues();

button1.Text = "Изменить";

}

}

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

{

if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" )

{

Admin Dan = new Admin();

if (Mode == "Add")

{

Dan.AddNewStudent(textBox1.Text, textBox2.Text, textBox3.Text, GroupID);

}

else

{

Dan.EditStudent(ID.ToString(),textBox1.Text, textBox2.Text, textBox3.Text, GroupID);

}

Close();

}

else

{

MessageBox.Show("Не все значения введены");

}

}

private void AddValues()

{

string Text = "select StudentName,StudentSurname,Password from Student where ID='" +ID +"'";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand queryID = new SqlCommand(Text, connection);

SqlDataReader dataReader = queryID.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

textBox1.Text = dataReader.GetString(0);

textBox2.Text = dataReader.GetString(1);

textBox3.Text = dataReader.GetString(2);

}

}

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

{

StudentForm Look = new StudentForm(ID.ToString(),"Student","true");

Look.ShowDialog();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AddEditSubjects : Form

{

string Mode { get; set; }

int ID { get; set; }

public AddEditSubjects(string mode, int SubjectID=-1)

{

InitializeComponent();

Mode = mode;

if (Mode == "Edit")

{

ID = SubjectID;

AddValues();

button1.Text = "Изменить";

}

}

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

{

if (textBox1.Text != "")

{

Admin Dan = new Admin();

if (Mode == "Edit")

{

Dan.EditSubject(textBox1.Text);

}

else

{

Dan.AddNewSubject(textBox1.Text);

}

Close();

}

else

{

MessageBox.Show("Не все значения введены");

}

}

private void AddValues()

{

string Text = "select SubjectName from Subject where SubjectID='" + ID + "'";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand queryID = new SqlCommand(Text, connection);

SqlDataReader dataReader = queryID.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

textBox1.Text = dataReader.GetString(0);

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AddQuestionDialog : Form

{

string test { get; set; }

public AddQuestionDialog(string Test)

{

InitializeComponent();

test = Test;

}

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

{

if (QuestionDescription.Text != null )

{

Teacher MyTeacher = new Teacher();

MyTeacher.AddNewQuestion(QuestionDescription.Text,test);

Close();

}

}

}

}

AddTeacherDialog.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AddTeacherDialog : Form

{

public AddTeacherDialog()

{

InitializeComponent();

}

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

{

if (NameTeacher.Text != "" && Surname.Text != "" && Password.Text!="")

{

Admin Dan = new Admin();

Dan.AddNewTeacher(NameTeacher.Text, Surname.Text, Password.Text);

Close();

}

else

{

MessageBox.Show("Не все значения введены");

}

}

}

}

AddTestDialog.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AddTestDialog : Form

{

string Subject { get; set; }

string ID { get; set; }

string Mode { get; set; }

public AddTestDialog(string subject, string Id="",string mode="")

{

InitializeComponent();

Subject = subject;

Mode = mode;

ID = Id;

}

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

{

if(textBox1.Text!=null && textBox2.Text!=null)

{

bool Correct=false;

try

{

Correct=Int32.Parse(textBox2.Text)>0;

}

catch

{

MessageBox.Show("В минуты введено недопустимое значение");

textBox1.Text = "";

textBox2.Text = "";

}

if (Correct)

{

if (Mode== "")

{

Admin Dan = new Admin();

Dan.AddNewTest(textBox2.Text, textBox1.Text, Subject);

}

else

{

Teacher My = new Teacher();

My.AddNewTest(textBox2.Text, textBox1.Text, Subject,ID);

}

Close();

}

}

else

{

MessageBox.Show("Не все значения введены");

}

}

}

}

AddVariantDialog.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AddVariantDialog : Form

{

string Id { get; set; }

public AddVariantDialog(string ID)

{

InitializeComponent();

Id = ID;

}

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

{

if (textBox1.Text != null && comboBox1.SelectedItem!= null)

{

Teacher MyTeacher = new Teacher();

MyTeacher.AddVariant(textBox1.Text, comboBox1.SelectedItem.ToString(),Id);

Close();

}

else

{

MessageBox.Show("Не введено описание или не выбрана верность варианта");

}

}

}

}

Admin.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Text;

namespace Testing\_System

{

class Admin : User

{

public void DeleteTeacher(string TeacherID)

{

string Text = @"delete from TeacherSubjectTest where TeacherID='" + TeacherID + "'";

ExecuteQueryInDB(Text);

Text = @"delete from TeacherSubject where TeacherID='" + TeacherID + "'";

ExecuteQueryInDB(Text);

Text = @"delete from Teacher where ID='" + TeacherID + "'";

ExecuteQueryInDB(Text);

}

public void AddNewTeacher(string Name, string Surname, string Password)

{

string Text = @"insert into Teacher(TeacherName, TeacherSurname, Password) Values('" + Name + "', '" + Surname + "', '"+Password+"')";

ExecuteQueryInDB(Text);

}

public void EditTeacher(string Name, string Surname, string Password,string newName,string newSur,string newPass)

{

int ID = 0;

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

string Text = @"select ID from Teacher where TeacherName='" + Name + "' and TeacherSurname='" + Surname + "' and Password='" + Password + "'";

SqlCommand queryID = new SqlCommand(Text, connection);

SqlDataReader SelectID = queryID.ExecuteReader();

if(SelectID.HasRows)

{

SelectID.Read();

ID = SelectID.GetInt32(0);

}

SelectID.Close();

Text = @"update Teacher set TeacherName='"+ newName + "', TeacherSurname='" + newSur + "', Password='" + newPass + "' where ID='"+ID+"'";

SqlCommand query = new SqlCommand(Text, connection);

SqlDataReader \_ = query.ExecuteReader();

}

public void AddNewStudent(string Name, string Surname, string Password, string Group)

{

string Text = @"insert into Student(StudentName, StudentSurname, Password, GroupID) Values('"

+ Name + "', '" + Surname + "', '" + Password + "', '"+Group+"')";

ExecuteQueryInDB(Text);

}

public void EditStudent(string ID,string Name, string Surname, string Password, string Group)

{

string Text = @"Update Student set StudentName='"+ Name + "',StudentSurname='" + Surname + "',Password= '"

+ Password + "',GroupID= '" + Group + "' where ID='"+ID+"'";

ExecuteQueryInDB(Text);

}

public void AddNewGroup(string Name)

{

string Text = @"insert into Groups(GroupName) Values('"+ Name + "')";

ExecuteQueryInDB(Text);

}

public void EditGroup(string Name,int ID)

{

string Text = @"Update Groups set GroupName='" + Name + "' where GroupID='"+ID+"'";

ExecuteQueryInDB(Text);

}

public void AddNewSubject(string Name)

{

string Text = @"insert into Subject(SubjectName) Values('"+ Name + "')";

ExecuteQueryInDB(Text);

}

public void EditSubject(string Name)

{

string Text = @"Update Subject set SubjectName='" + Name + "'";

ExecuteQueryInDB(Text);

}

public void AddNewTest(string TestTime, string TestName,string ChoosedSubject)

{

string Text = @"insert into Test(TestName, TestMinutes) Values('" + TestName + "', '" + TestTime + "')";

ExecuteQueryInDB(Text);

string text = "select TestID from Test where TestName='" +TestName +"'";

string AddedTestID = FindInDB(text);

Text = @"insert into SubjectTest(SubjectID, TestID) Values('"+ ChoosedSubject + "', '" + AddedTestID + "')";

ExecuteQueryInDB(Text);

}

}

}

AdminForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class AdminForm : Form

{

public AdminForm()

{

InitializeComponent();

}

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

{

EditTeacher Edit = new EditTeacher();

Edit.Show();

Hide();

}

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

{

EditGroupStudent Edit = new EditGroupStudent();

Edit.Show();

Hide();

}

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

{

EditSubjectTest Edit = new EditSubjectTest();

Edit.Show();

Hide();

}

private void AdminForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

EditGroup.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditGroup : Form

{

string SubjectID { get; set; }

string TestId { get; set; }

public EditGroup(string TestID)

{

InitializeComponent();

TestId = TestID;

Teacher My = new Teacher();

SubjectID = My.FindInDB("select SubjectID from SubjectTest where TestID='"+TestID+"'");

string ToChoose = @"select GroupName from Groups where GroupID in (select GroupID from GroupSubjectTests where TestID='"

+ TestID + "' and SubjectID='"+SubjectID+"')";

string ToAdd = @"select GroupName from Groups";

AddToChooser(ToChoose);

AddToAdder(ToAdd);

}

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

{

if(Chooser.SelectedItem!=null)

{

Teacher My=new Teacher();

string command = @"delete from GroupSubjectTests where TestID='" + TestId + "' and GroupID in(select GroupID from Groups where GroupName='"

+ Chooser.SelectedItem.ToString()+"')";

My.ExecuteQueryInDB(command);

command = @"delete from GroupSubject where GroupID in(select GroupID from Groups where GroupName='"

+ Chooser.SelectedItem.ToString() + "')";

My.ExecuteQueryInDB(command);

Adder.Items.Add(Chooser.SelectedItem);

Chooser.Items.Remove(Chooser.SelectedItem);

Chooser.Text = "";

}

else

{

MessageBox.Show("Не выбрана группа для \nдля отмены теста");

}

}

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

{

if (Adder.SelectedItem != null)

{

int Id=0;

string name = @"select GroupID from Groups where GroupName='"+ Adder.SelectedItem.ToString()+"'";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(name, connection);

SqlDataReader dataReader = command.ExecuteReader();

if(dataReader.HasRows)

{

dataReader.Read();

Id = dataReader.GetInt32(0);

}

string commandinsert = @"insert into GroupSubjectTests values ('"+Id+"', '"+SubjectID+"','"+TestId+"')";

Teacher My = new Teacher();

My.ExecuteQueryInDB(commandinsert);

commandinsert = @"insert into GroupSubject values ('" + Id + "', '" + SubjectID + "')";

My.ExecuteQueryInDB(commandinsert);

Chooser.Items.Add(Adder.SelectedItem);

Adder.Items.Remove(Adder.SelectedItem);

Adder.Text = "";

}

else

{

MessageBox.Show("Не выбрана группа для \nдля назначения теста");

}

}

private void AddToChooser(string selectionCommandString)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

string Group = dataReader.GetString(0);

Chooser.Items.Add(Group);

}

}

}

private void AddToAdder(string selectionCommandString)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

string Group = dataReader.GetString(0);

if(!Chooser.Items.Contains(Group))

Adder.Items.Add(Group);

}

}

}

}

}

EditGroupStudent.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditGroupStudent : Form

{

List<string> Group = new List<string>();

List<int> GroupID = new List<int>();

List<string> Student = new List<string>();

List<int> StudentID = new List<int>();

public EditGroupStudent()

{

InitializeComponent();

GroupsFor();

}

private void GroupsFor()

{

Groups.Items.Clear();

Group.Clear();

GroupID.Clear();

string selectionCommandString = @"select GroupName,GroupID from Groups";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

Group.Add(dataReader.GetString(0));

GroupID.Add(dataReader.GetInt32(1));

Groups.Items.Add(dataReader.GetString(0));

}

}

}

private void StudentsFor()

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

Students.Items.Clear();

Student.Clear();

StudentID.Clear();

string ID = label3.Text;

string selectionCommandString = "select GroupID from Groups where GroupName='"+ID+"'";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

ID = dataReader.GetInt32(0).ToString();

}

dataReader.Close();

selectionCommandString = @"select StudentName,StudentSurname,ID from Student where GroupID='"

+ ID + "'";

SqlCommand commandselect = new SqlCommand(selectionCommandString, connection);

SqlDataReader DataReader = commandselect.ExecuteReader();

if (DataReader.HasRows)

{

while (DataReader.Read())

{

Student.Add(DataReader.GetString(0));

StudentID.Add(DataReader.GetInt32(2));

Students.Items.Add(DataReader.GetString(0)+" "+DataReader.GetString(1));

}

}

}

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

{

if (Groups.SelectedItem != null)

{

AddEditGroup My = new AddEditGroup("Edit",GroupID[Groups.SelectedIndex]);

My.ShowDialog();

GroupsFor();

Groups.Text = "";

}

else

{

MessageBox.Show("Не выбрана группа для редактирования");

}

}

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

{

if (Groups.SelectedItem != null)

{

Admin My = new Admin();

int Index = Groups.SelectedIndex;

string command = @"delete from AllPassedTests where StudentID in(select ID from Student where GroupID='" + GroupID[Index] + "')";

My.ExecuteQueryInDB(command);

command = @"delete from Student where GroupID='" + GroupID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from GroupSubjectTests where GroupID='" + GroupID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from GroupSubject where GroupID='" + GroupID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from Groups where GroupID='" + GroupID[Index] + "'";

My.ExecuteQueryInDB(command);

Groups.Items.Remove(Groups.SelectedItem.ToString());

Group.RemoveAt(Index);

GroupID.RemoveAt(Index);

Groups.Text = "";

Students.Items.Clear();

Student.Clear();

StudentID.Clear();

}

else

{

MessageBox.Show("Не выбрана группа для удаления");

}

}

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

{

if (Groups.SelectedItem != null)

{

label3.Text = Groups.SelectedItem.ToString();

StudentsFor();

Groups.Text = "";

}

else

{

MessageBox.Show("Не выбрана группа");

}

}

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

{

AddEditGroup My = new AddEditGroup("Add");

My.ShowDialog();

GroupsFor();

Groups.Text = "";

}

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

{

if (Groups.Items.Contains(label3.Text))

{

Hide();

AddEditStudent My = new AddEditStudent("Add",GroupID[Groups.SelectedIndex].ToString());

My.ShowDialog();

StudentsFor();

Show();

Students.Text = "";

}

else

{

MessageBox.Show("Не выбрана дисциплина для теста");

}

}

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

{

if (Students.SelectedItem != null)

{

Hide();

AddEditStudent My = new AddEditStudent("Edit", GroupID[Groups.SelectedIndex].ToString(), StudentID[Students.SelectedIndex]);

My.ShowDialog();

StudentsFor();

Show();

Students.Text = "";

}

else

{

MessageBox.Show("Не выбран студент для редактирования");

}

}

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

{

if (Students.SelectedItem != null)

{

Admin My = new Admin();

int Index = Students.SelectedIndex;

string command = @"delete from AllPassedTests where StudentID='" + StudentID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from Student where ID='" + StudentID[Index] + "'";

My.ExecuteQueryInDB(command);

Students.Items.Remove(Students.SelectedItem.ToString());

Student.RemoveAt(Index);

StudentID.RemoveAt(Index);

Students.Text = "";

}

else

{

MessageBox.Show("Не выбран студент \nдля удаления из группы");

}

}

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

{

AdminForm My = new AdminForm();

My.Show();

Hide();

}

private void EditGroupStudent\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

EditQuestions.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditQuestions : Form

{

string TestID { get; set; }

public EditQuestions(string TestId,string TestName)

{

TestID=TestId;

InitializeComponent();

this.Text = TestName;

AddQuestions();

}

private void AddQuestions()

{

string selectionCommandString= @"select QuestionDescription from Question where TestID='" + TestID + "'";

AllQuestions.Items.Clear();

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

string Question = dataReader.GetString(0);

AllQuestions.Items.Add(Question);

}

dataReader.Close();

}

}

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

{

if(AllQuestions.SelectedItem != null)

{

Teacher MyTeacher = new Teacher();

string selectionCommandString = @"select QuestionID from Question where QuestionDescription='" + AllQuestions.SelectedItem.ToString() + "'";

EditVariants MyVariants = new EditVariants(MyTeacher.FindInDB(selectionCommandString), AllQuestions.SelectedItem.ToString());

MyVariants.Show();

AllQuestions.Text = "";

AddQuestions();

}

else

{

MessageBox.Show("Не выбран вопрос для редактирования");

}

}

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

{

if (AllQuestions.SelectedItem != null)

{

int ID = GetQuestionID(@"select QuestionID from Question where QuestionDescription='" + AllQuestions.SelectedItem.ToString() + "'");

Teacher MyTeacher = new Teacher();

MyTeacher.DeleteQuestion(ID.ToString());

AllQuestions.Items.Remove(AllQuestions.SelectedItem);

AllQuestions.Text = "";

}

else

{

MessageBox.Show("Не выбран вопрос для удаления");

}

}

private int GetQuestionID(string selectionCommandString)

{

int QuestionID = 0;

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

QuestionID = dataReader.GetInt32(0);

}

return QuestionID;

}

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

{

AddQuestionDialog Dialog = new AddQuestionDialog(TestID);

Dialog.ShowDialog();

AddQuestions();

AllQuestions.Text = "";

}

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

{

EditGroup MyEdit = new EditGroup(TestID);

MyEdit.ShowDialog();

}

}

}

EditSubject.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditSubject : Form

{

List<string> TeacherSubjects = new List<string>();

List<int> TeacherSubjectsID = new List<int>();

List<string> SubjectsToAdd = new List<string>();

List<int> SubjectsToAddID = new List<int>();

List<string> TeacherTests = new List<string>();

List<int> TeacherTestsID = new List<int>();

List<string> TestsToAdd = new List<string>();

List<int> TestsToAddID = new List<int>();

string ID { get; set; }

public EditSubject(string Id)

{

InitializeComponent();

ID = Id;

SubjectsForDelete();

SubjectsForAdd();

TestsForDelete();

TestsForAdd();

}

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

{

if (SubjectChooser.SelectedItem != null)

{

Admin My = new Admin();

int Index = SubjectChooser.SelectedIndex;

string command = @"delete from TeacherSubject where TeacherID='"+ ID

+ "' and SubjectID='"+ TeacherSubjectsID[Index] + "'";

My.ExecuteQueryInDB(command);

TeacherSubjects.Remove(SubjectChooser.SelectedItem.ToString());

SubjectsToAddID.Add(TeacherSubjectsID[Index]);

TeacherSubjectsID.RemoveAt(Index);

SubjectsToAdd.Add(SubjectChooser.SelectedItem.ToString());

SubjectAdder.Items.Add(SubjectChooser.SelectedItem);

SubjectChooser.Items.Remove(SubjectChooser.SelectedItem);

SubjectChooser.Text = "";

TestAdder.Items.Clear();

TestChooser.Items.Clear();

TestsForDelete();

TestsForAdd();

}

else

{

MessageBox.Show("Не выбран предмет \nдля удаления для учителя");

}

}

private void SubjectsForDelete()

{

string selectionCommandString= @"select SubjectName,SubjectID from Subject where SubjectID in (select SubjectID from TeacherSubject where TeacherID='"

+ ID + "')";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

TeacherSubjects.Add(dataReader.GetString(0));

TeacherSubjectsID.Add(dataReader.GetInt32(1));

SubjectChooser.Items.Add(dataReader.GetString(0));

}

}

}

private void TestsForDelete()

{

string selectionCommandString= "select TestName,TestID from Test where TestID in (select TestID from TeacherSubjectTest where TeacherID='"

+ ID + "')";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

TeacherTests.Add(dataReader.GetString(0));

TeacherTestsID.Add(dataReader.GetInt32(1));

TestChooser.Items.Add(dataReader.GetString(0));

}

}

}

private void SubjectsForAdd()

{

string selectionCommandString= @"select SubjectName,SubjectID from Subject";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

if (!TeacherSubjects.Contains(dataReader.GetString(0)))

{

SubjectsToAdd.Add(dataReader.GetString(0));

SubjectsToAddID.Add(dataReader.GetInt32(1));

SubjectAdder.Items.Add(dataReader.GetString(0));

}

}

}

}

private void TestsForAdd()

{

string selectionCommandString= @"select TestName,TestID from Test";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

if (!TeacherTests.Contains(dataReader.GetString(0)))

{

if (TeacherSubjectsID.Contains(getSubjectID(dataReader.GetInt32(1))))

{

TestsToAdd.Add(dataReader.GetString(0));

TestsToAddID.Add(dataReader.GetInt32(1));

TestAdder.Items.Add(dataReader.GetString(0));

}

}

}

}

}

private int getSubjectID(int ID)

{

string Text = @"select SubjectID from SubjectTest where TestID='" + ID + "'";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(Text, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

return dataReader.GetInt32(0);

}

else return -1;

}

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

{

if (SubjectAdder.SelectedItem != null)

{

Admin My = new Admin();

int Index = SubjectAdder.SelectedIndex;

string command = @"insert into TeacherSubject(TeacherID,SubjectID) values ('"+ID+"', '"+SubjectsToAddID[Index]+"')";

My.ExecuteQueryInDB(command);

SubjectsToAdd.Remove(SubjectAdder.SelectedItem.ToString());

TeacherSubjectsID.Add(SubjectsToAddID[Index]);

SubjectsToAddID.RemoveAt(Index);

TeacherSubjects.Add(SubjectAdder.SelectedItem.ToString());

SubjectChooser.Items.Add(SubjectAdder.SelectedItem);

SubjectAdder.Items.Remove(SubjectAdder.SelectedItem);

SubjectAdder.Text = "";

TestAdder.Items.Clear();

TestChooser.Items.Clear();

TestsForDelete();

TestsForAdd();

}

else

{

MessageBox.Show("Не выбран предмет\nдля добавления для учителя");

}

}

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

{

if (TestChooser.SelectedItem != null)

{

Admin My = new Admin();

int Index = TestChooser.SelectedIndex;

string command = @"delete from TeacherSubjectTest where TeacherID='" + ID+ "' and SubjectID='"

+ My.FindInDB(@"select SubjectId from SubjectTest where TestID='"+ TeacherTestsID[Index]+"'")

+ "' and TestID='"+TeacherTestsID[Index]+"'";

My.ExecuteQueryInDB(command);

TeacherTests.Remove(TestChooser.SelectedItem.ToString());

TestsToAddID.Add(TeacherTestsID[Index]);

TeacherTestsID.RemoveAt(Index);

TestsToAdd.Add(TestChooser.SelectedItem.ToString());

TestAdder.Items.Add(TestChooser.SelectedItem);

TestChooser.Items.Remove(TestChooser.SelectedItem);

TestChooser.Text = "";

}

else

{

MessageBox.Show("Не выбран тест \nдля удаления для учителя");

}

}

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

{

if (TestAdder.SelectedItem != null)

{

Admin My = new Admin();

int Index = TestAdder.SelectedIndex;

string command = @"insert into TeacherSubjectTest(TeacherID,SubjectID,TestID) values ('" + ID + "', '"

+ My.FindInDB(@"select SubjectId from SubjectTest where TestID='" + TestsToAddID[Index] + "'") + "', '"

+TestsToAddID[Index]+"')";

My.ExecuteQueryInDB(command);

TestsToAdd.Remove(TestAdder.SelectedItem.ToString());

TeacherTestsID.Add(TestsToAddID[Index]);

TestsToAddID.RemoveAt(Index);

TeacherTests.Add(TestAdder.SelectedItem.ToString());

TestChooser.Items.Add(TestAdder.SelectedItem);

TestAdder.Items.Remove(TestAdder.SelectedItem);

TestAdder.Text = "";

}

else

{

MessageBox.Show("Не выбран тест\nдля добавления для учителя");

}

}

}

}

EditSubjectTest.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditSubjectTest : Form

{

List<string> Subject = new List<string>();

List<int> SubjectID = new List<int>();

List<string> Test = new List<string>();

List<int> TestID = new List<int>();

public EditSubjectTest()

{

InitializeComponent();

SubjectsFor();

}

private void SubjectsFor()

{

Subjects.Items.Clear();

Subject.Clear();

SubjectID.Clear();

string selectionCommandString = @"select SubjectName,SubjectID from Subject";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

Subject.Add(dataReader.GetString(0));

SubjectID.Add(dataReader.GetInt32(1));

Subjects.Items.Add(dataReader.GetString(0));

}

}

}

private void TestsFor()

{

Tests.Items.Clear();

Test.Clear();

TestID.Clear();

int ID = Subjects.SelectedIndex;

label3.Text = Subject[ID].ToString();

string selectionCommandString = @"select TestName,TestID from Test where TestID in(select TestID from SubjectTest where SubjectID='"

+ SubjectID[ID] + "')";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

Test.Add(dataReader.GetString(0));

TestID.Add(dataReader.GetInt32(1));

Tests.Items.Add(dataReader.GetString(0));

}

}

}

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

{

if (Subjects.SelectedItem != null)

{

Hide();

AddEditSubjects My = new AddEditSubjects("Edit", SubjectID[Subjects.SelectedIndex]);

My.ShowDialog();

SubjectsFor();

Show();

Subjects.Text = "";

}

else

{

MessageBox.Show("Не выбран предмет для редактирования");

}

}

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

{

if (Subjects.SelectedItem != null)

{

Admin My = new Admin();

int Index = Subjects.SelectedIndex;

string command = @"delete from GroupSubject where SubjectID in(select SubjectID from Subject where SubjectID='" + SubjectID[Index] + "')";

My.ExecuteQueryInDB(command);

command = @"delete from GroupSubjectTests where SubjectID='" + SubjectID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from SubjectTest where SubjectID='" + SubjectID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from TeacherSubject where SubjectID='" + SubjectID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from TeacherSubjectTest where SubjectID='" + SubjectID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from Subject where SubjectID='" + SubjectID[Index] + "'";

My.ExecuteQueryInDB(command);

Subjects.Items.Remove(Subjects.SelectedItem.ToString());

Subject.RemoveAt(Index);

SubjectID.RemoveAt(Index);

Subjects.Text = "";

Tests.Items.Clear();

Test.Clear();

TestID.Clear();

}

else

{

MessageBox.Show("Не выбран предмет для удаления");

}

}

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

{

if (Subjects.SelectedItem != null)

{

label3.Text = Subjects.SelectedItem.ToString();

TestsFor();

Subjects.Text = "";

}

else

{

MessageBox.Show("Не выбран предмет");

}

}

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

{

if (Tests.SelectedItem != null)

{

EditQuestions My = new EditQuestions(TestID[Tests.SelectedIndex].ToString(),Test[Tests.SelectedIndex]);

My.ShowDialog();

TestsFor();

Tests.Text = "";

}

else

{

MessageBox.Show("Не выбран тест для редактирования");

}

}

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

{

if (Tests.SelectedItem != null)

{

Admin My = new Admin();

int Index = Tests.SelectedIndex;

string command = @"delete from AllPassedTests where StudentID='" + TestID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from GroupSubjectTests where TestID='" + TestID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from SubjectTest where TestID='" + TestID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from TeacherSubjectTest where TestID='" + TestID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from Variant where QuestionID in(select QuestionID from Question where TestID='" + TestID[Index] + "')";

My.ExecuteQueryInDB(command);

command = @"delete from Question where TestID='" + TestID[Index] + "'";

My.ExecuteQueryInDB(command);

command = @"delete from Test where TestID='" + TestID[Index] + "'";

My.ExecuteQueryInDB(command);

Tests.Items.Remove(Tests.SelectedItem.ToString());

Test.RemoveAt(Index);

TestID.RemoveAt(Index);

Tests.Text = "";

}

else

{

MessageBox.Show("Не выбран тест \nдля удаления из предмета");

}

}

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

{

if (Subjects.Items.Contains(label3.Text))

{

Hide();

AddTestDialog My = new AddTestDialog(SubjectID[Subjects.Items.IndexOf(label3.Text)].ToString());

My.ShowDialog();

TestsFor();

Show();

Tests.Text = "";

}

else

{

MessageBox.Show("Не выбрана дисциплина для теста");

}

}

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

{

Hide();

AddEditSubjects My = new AddEditSubjects("Add");

My.ShowDialog();

SubjectsFor();

Show();

Subjects.Text = "";

}

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

{

AdminForm My = new AdminForm();

My.Show();

Hide();

}

private void EditSubjectTest\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

EditTeacher.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditTeacher : Form

{

List<int> TeacherID = new List<int>();

List<string> TeacherName = new List<string>();

List<string> TeacherSurname = new List<string>();

List<string> TeacherPassword = new List<string>();

public EditTeacher()

{

InitializeComponent();

AddTeachers();

}

private void AddTeachers()

{

Chooser.Items.Clear();

TeacherName.Clear();

TeacherSurname.Clear();

TeacherPassword.Clear();

TeacherID.Clear();

string selectionCommandString = @"select \* from Teacher";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

TeacherName.Add(dataReader.GetString(1));

TeacherSurname.Add(dataReader.GetString(3));

TeacherPassword.Add(dataReader.GetString(2));

Chooser.Items.Add(dataReader.GetString(1)+" "+dataReader.GetString(3));

TeacherID.Add(dataReader.GetInt32(0));

}

}

}

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

{

if (Chooser.SelectedItem != null)

{

EditTeacherDialog dialog = new EditTeacherDialog(TeacherName[Chooser.SelectedIndex], TeacherSurname[Chooser.SelectedIndex],TeacherPassword[Chooser.SelectedIndex]);

dialog.ShowDialog();

AddTeachers();

Chooser.Text = "";

}

else

{

MessageBox.Show("Не выбран учитель для редактирования");

}

}

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

{

if (Chooser.SelectedItem != null)

{

int ID = GetTeacherID(@"select ID from Teacher where TeacherName='"

+ TeacherName[Chooser.SelectedIndex] + "' and TeacherSurname='"+TeacherSurname[Chooser.SelectedIndex]+"'");

Admin Dan = new Admin();

Dan.DeleteTeacher(ID.ToString());

Chooser.Text = "";

Chooser.Items.Clear();

AddTeachers();

}

else

{

MessageBox.Show("Не выбран учитель для удаления");

}

}

private int GetTeacherID(string selectionCommandString)

{

int TeacherID = 0;

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

TeacherID = dataReader.GetInt32(0);

}

return TeacherID;

}

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

{

AddTeacherDialog Dialog = new AddTeacherDialog();

Dialog.ShowDialog();

Chooser.Text = "";

AddTeachers();

}

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

{

AdminForm My = new AdminForm();

My.Show();

Hide();

}

private void EditTeacher\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

EditTeacherDialog.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditTeacherDialog : Form

{

string Namee { get; set; }

string Surname { get; set; }

string Password { get; set; }

public EditTeacherDialog(string name, string surname, string pass)

{

InitializeComponent();

Namee=textBox1.Text = name;

Surname=textBox2.Text = surname;

Password=textBox3.Text = pass;

}

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

{

Teacher My = new Teacher();

EditSubject MyEdit = new EditSubject(My.FindInDB("select ID from Teacher where TeacherName='"+Namee+"' and Password='"+Password+"'"));

MyEdit.Show();

}

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

{

if (textBox3.Text != "" || textBox1.Text != "" || textBox2.Text != "")

{

Admin Dan = new Admin();

Dan.EditTeacher(Namee, Surname,Password,textBox1.Text, textBox2.Text, textBox3.Text);

Close();

}

else

{

MessageBox.Show("Данные не могут быть пустыми");

}

}

}

}

EditVariantDialog.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditVariantDialog : Form

{

public EditVariantDialog(string Variant)

{

InitializeComponent();

textBox1.Text = Variant;

User My = new User();

comboBox1.Text = My.FindInDB("select VariantCorrectness from Variant where VariantDescription='" + Variant + "'");

}

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

{

if (textBox1.Text != null && (comboBox1.Text=="false"||comboBox1.Text=="true"))

{

Teacher MyTeacher = new Teacher();

MyTeacher.EditVariant(textBox1.Text, comboBox1.Text);

Close();

}

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class LoginForm : Form

{

User MyUser = new User();

public LoginForm()

{

InitializeComponent();

}

private void ReadyToEnterButtonClick(object sender, EventArgs e)

{

if (RoleChooser.SelectedItem != null)

{

MyUser.Role = RoleChooser.SelectedItem.ToString();

MyUser.ID = LoginBox.Text.ToString();

string selectionCommandString = "select ID from " + MyUser.Role +

" R where R.ID='" + MyUser.ID + "' and R.Password='" + PasswordBox.Text.ToString()+ "'";

if (MyUser.CheckInDataBase(selectionCommandString))

{

if (MyUser.Role == "Student"|| MyUser.Role == "Teacher")

{

StudentForm MyStudent = new StudentForm(MyUser.ID, MyUser.Role);

MyStudent.Show();

}

if (MyUser.Role == "Admin")

{

AdminForm MyAdmin = new AdminForm();

MyAdmin.Show();

}

Hide();

}

else

{

MessageBox.Show("Неправильный логин или пароль");

}

}

else

{

MessageBox.Show("Не выбрана роль пользователя");

}

}

}

}

Form2.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class StudentForm : Form

{

User MyUser = new User();

String ifAdmin { get; set; }

public StudentForm(string ID,string Role, string IfAdmin="")

{

string selectionCommandString = "";

MyUser.Role = Role;

MyUser.ID=ID;

ifAdmin = IfAdmin;

InitializeComponent();

this.Text = Role + " " + ID;

if (MyUser.Role == "Student")

{

selectionCommandString = "select SubjectName from Subject where SubjectID in (select SubjectID from GroupSubject where GroupID in(select GroupID from Student where ID='" +

MyUser.ID + "'))";

}

if (MyUser.Role == "Teacher")

{

selectionCommandString = "select SubjectName from Subject S where S.SubjectID in (select SubjectID from TeacherSubject TS where TS.TeacherID ='"+

MyUser.ID + "')";

}

AddSubjects(selectionCommandString);

}

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

{

if (SubjectChooser.SelectedItem != null)

{

string selectionCommandString = "select SubjectID from Subject where SubjectName='" + SubjectChooser.SelectedItem + "'";

if (ifAdmin == "")

{

SubjectForm MySubject = new SubjectForm(MyUser.ID, MyUser.Role, SubjectChooser.SelectedItem.ToString(), GetSubjectID(selectionCommandString));

MySubject.Show();

}

else

{

SubjectForm MySubject = new SubjectForm(MyUser.ID, MyUser.Role, SubjectChooser.SelectedItem.ToString(), GetSubjectID(selectionCommandString),"true");

MySubject.Show();

}

this.Hide();

}

else

{

MessageBox.Show("Предмет не выбран");

}

}

private void AddSubjects(string selectionCommandString)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

string Subject = dataReader.GetString(0);

SubjectChooser.Items.Add(Subject);

}

dataReader.Close();

}

}

private int GetSubjectID(string selectionCommandString)

{

int SubjectID = 0;

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

SubjectID = dataReader.GetInt32(0);

}

return SubjectID;

}

private void StudentForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

Form3.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class EditVariants : Form

{

string ID { get; set; }

public EditVariants(string QuestionID, string Description)

{

InitializeComponent();

this.Text = Description;

ID = QuestionID;

AddVariants();

}

private void AddVariants()

{

AllVariants.Items.Clear();

string selectionCommandString = @"select VariantDescription, VariantCorrectness from Variant where QuestionID='" + ID + "'";

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

string Question = dataReader.GetString(0);

string Correctnes = dataReader.GetString(1);

AllVariants.Items.Add(Question+" is "+Correctnes);

}

}

}

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

{

if (AllVariants.SelectedItem != null)

{

string text= AllVariants.SelectedItem.ToString();

if(text.Contains(" is true"))

{

text=text.Replace(" is true", "");

}

if (text.Contains(" is false"))

{

text=text.Replace(" is false", "");

}

EditVariantDialog MyVariants = new EditVariantDialog(text);

MyVariants.Show();

AllVariants.Text = "";

AddVariants();

}

else

{

MessageBox.Show("Не выбран вариант для редактирования");

}

}

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

{

if (AllVariants.SelectedItem != null)

{

string Name = AllVariants.SelectedItem.ToString();

if (Name.Contains(" is true"))

{

Name = Name.Replace(" is true","");

}

if (Name.Contains(" is false"))

{

Name = Name.Replace(" is false", "");

}

Teacher MyTeacher = new Teacher();

string ID = MyTeacher.FindInDB(@"select VariantID from Variant where VariantDescription='" + Name + "'");

MyTeacher.DeleteVariant(ID);

AddVariants();

AllVariants.Text = "";

}

else

{

MessageBox.Show("Не выбран вариант для удаления");

}

}

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

{

AddVariantDialog Dialog = new AddVariantDialog(ID);

Dialog.ShowDialog();

AddVariants();

AllVariants.Text = "";

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Testing\_System

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.SetHighDpiMode(HighDpiMode.SystemAware);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LoginForm());

Application.ApplicationExit += delegate

{

foreach (Form Hidden in Application.OpenForms)

Hidden.Close();

};

}

}

}

Student.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Text;

namespace Testing\_System

{

class Student : User

{

public List <int> IDofQuestions=new List<int>();

public List<String> QuestionDesacriptions = new List<string>();

public List<List<int>> IDofVariants = new List<List<int>>();

public List<List<string>> VariantDescriptions = new List<List<string>>();

public List<string> CorrectVariantDescriptions = new List<string>();

public List<string> ListOfTestsAvailable = new List<string>();

public List<int> PassedTestID = new List<int>();

public List<int> Marks = new List<int>();

public int TestTime { get; set; }

public void GetAllTestInfo(string TestID)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(@"select QuestionID,QuestionDescription from Question where TestID='" + TestID + "'", connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

IDofQuestions.Add(dataReader.GetInt32(0));

QuestionDesacriptions.Add(dataReader.GetString(1));

}

}

dataReader.Close();

SqlCommand Command = new SqlCommand(@"select VariantID, VariantDescription, VariantCorrectness from Variant where QuestionID in(select QuestionID from Question where TestID='" + TestID + "')");

Command.Connection = connection;

SqlConnection Connection = new SqlConnection(connectionString);

Connection.Open();

SqlDataReader DataReader = Command.ExecuteReader();

if (DataReader.HasRows)

{

if (IDofQuestions.Count != 0)

{

for(int QuestionIndex=0;QuestionIndex<IDofQuestions.Count;QuestionIndex++)

{

IDofVariants.Add(new List<int>());

VariantDescriptions.Add(new List<string>());

string Text = @"select COUNT(VariantID) from Variant where QuestionID='" + IDofQuestions[QuestionIndex] + "';";

SqlCommand query = new SqlCommand(Text);

query.Connection = Connection;

SqlDataReader CountReader = query.ExecuteReader();

CountReader.Read();

var VariantCount = CountReader.GetInt32(0);

for (int VariantIndex=0;VariantIndex<VariantCount;VariantIndex++)

{

DataReader.Read();

IDofVariants[QuestionIndex].Add(DataReader.GetInt32(0));

VariantDescriptions[QuestionIndex].Add(DataReader.GetString(1));

if(DataReader.GetString(2)=="true")

{

CorrectVariantDescriptions.Add(VariantDescriptions[QuestionIndex][VariantIndex]);

}

}

CountReader.Close();

}

}

DataReader.Close();

string TextMin = @"select TestMinutes from Test T where T.TestID='" + TestID + "'";

SqlCommand queryMin = new SqlCommand(TextMin, connection);

SqlDataReader Reader = queryMin.ExecuteReader();

if (Reader.HasRows)

{

Reader.Read();

TestTime = Reader.GetInt32(0);

}

}

}

public void UpdatePassedTest(int Mark, string TestID, string ID)

{

string Text = @"insert into AllPassedTests Values('" + TestID + "', '" + ID + "', '" + Mark + "')";

ExecuteQueryInDB(Text);

}

public void GetPassedTestID(string ID)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

string Text = @"select PassedTestID, Mark from AllPassedTests where StudentID='"+ID+"'";

SqlCommand query = new SqlCommand(Text, connection);

SqlDataReader dataReader = query.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

PassedTestID.Add(dataReader.GetInt32(0));

Marks.Add(dataReader.GetInt32(1));

}

}

}

}

}

SubjectForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class SubjectForm : Form

{

User MyUser = new User();

public List<int> PassedTestID;

public List<int> Marks;

public string role="";

string selectionCommandString { get; set; }

string ifAdmin { get; set; }

public SubjectForm(string ID, string Role, String Subject, int SubjectID, string IfAdmin="")

{

InitializeComponent();

MyUser.ID = ID;

ifAdmin = IfAdmin;

string selectionCommandString = "";

this.Text = Subject;

MyUser.Role=role = Role;

if (Role == "Student")

{

if(IfAdmin!="")

{

label1.Location=new Point(185,26);

label1.Text = "Непройденные тесты";

TestChooser.Location=new Point(185, 86);

GoToTest.Hide();

back.Hide();

label2.Location = new Point(170, 146);

PassedTestBox.Location = new Point(154,214);

}

AddTestButton.Hide();

DeleteTestButton.Hide();

GoToTest.Location = new Point(89,241);

Student MyStudent = new Student();

MyStudent.GetPassedTestID(ID);

PassedTestID = new List<int>(MyStudent.PassedTestID);

Marks = new List<int>(MyStudent.Marks);

selectionCommandString = "select TestName,TestID from Test where TestID in (select TestID from GroupSubjectTests where GroupID in (select GroupID from Student where ID='"

+ ID + "') and SubjectID='" + SubjectID + "')";

}

if (Role == "Teacher")

{

PassedTestBox.Hide();

label2.Hide();

TestChooser.Width = 400;

TestChooser.Location=new Point(70, 164);

GoToTest.Location = new Point(70,241);

DeleteTestButton.Location = new Point(220, 241);

AddTestButton.Location = new Point(370,241);

label1.Location = new Point(210, 96);

selectionCommandString = @"select TestName,TestID from Test where TestID in (select TestID from TeacherSubjectTest where TeacherID='"

+ ID + "' and SubjectID='" + SubjectID + "')";

}

AddTests(selectionCommandString);

}

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

{

if(TestChooser.SelectedItem != null)

{

if (role == "Student")

{

string selectionCommand = @"select TestID from Test where TestName='" + TestChooser.SelectedItem.ToString() + "'";

Test MyTest = new Test(MyUser.FindInDB(selectionCommand), TestChooser.SelectedItem.ToString(),MyUser.ID);

MyTest.ShowDialog();

Student MyStudent = new Student();

MyStudent.GetPassedTestID(MyUser.ID);

PassedTestID = new List<int>(MyStudent.PassedTestID);

Marks = new List<int>(MyStudent.Marks);

string select = MyStudent.FindInDB("select TestName from Test where TestID='" + PassedTestID[^1] + "'");

TestChooser.Items.Remove(select);

PassedTestBox.Items.Add(select + " с оценкой " + Marks[^1]);

TestChooser.Text = "";

}

if(role=="Teacher")

{

string selectionCommand = @"select TestID from Test where TestName='" + TestChooser.SelectedItem.ToString() + "'";

string TestID = MyUser.FindInDB(selectionCommand);

EditQuestions MyQuestions = new EditQuestions(TestID, TestChooser.SelectedItem.ToString());

MyQuestions.ShowDialog();

}

}

else

{

if (role == "Teacher")

{

MessageBox.Show("Тест для редактирования не выбран");

}

else

{

MessageBox.Show("Тест для прохождения не выбран");

}

}

}

private void AddTests(string selectionCommandString)

{

TestChooser.Items.Clear();

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

string Subject = dataReader.GetString(0);

if (role == "Student")

{

int Passed=dataReader.GetInt32(1);

if (PassedTestID.Contains(Passed) == false)

{

TestChooser.Items.Add(Subject);

}

else

{

int Position = PassedTestID.IndexOf(Passed);

PassedTestBox.Items.Add(Subject+" с оценкой "+Marks[Position]);

}

}

if (role=="Teacher")

{

TestChooser.Items.Add(Subject);

}

}

}

}

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

{

AddTestDialog Dialog = new AddTestDialog(this.Text,MyUser.ID,"Student");

Dialog.ShowDialog();

string Test = MyUser.FindInDB("select Name from AddedTest where ID='" + MyUser.ID + "'");

if (Test != "")

{

TestChooser.Items.Add(Test);

}

}

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

{

if (TestChooser.SelectedItem != null)

{

string Item = TestChooser.SelectedItem.ToString();

string ID = MyUser.FindInDB(@"select TestID from Test where TestName='" + Item + "'");

Teacher MyTeacher = new Teacher();

MyTeacher.DeleteTest(ID);

TestChooser.Items.Remove(TestChooser.SelectedItem);

TestChooser.Text = "";

}

else

{

MessageBox.Show("Тест для удаления не выбран");

}

}

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

{

StudentForm My = new StudentForm(MyUser.ID, MyUser.Role);

My.Show();

Hide();

}

private void SubjectForm\_FormClosed(object sender, FormClosedEventArgs e)

{

if (ifAdmin == "")

{

Application.Exit();

}

}

}

}

Teacher.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Text;

namespace Testing\_System

{

class Teacher : User

{

public void AddNewTest(string TestTime, string TestName, string Subject, string ID)

{

string Text = @"insert into Test(TestName, TestMinutes) Values('" + TestName + "', '" + TestTime + "')";

ExecuteQueryInDB(Text);

Text = @"select TestID from Test where TestName='" + TestName+"'";

string AddedTestID = FindInDB(Text);

Text = @"select SubjectID from Subject where SubjectName='" + Subject + "'";

string SubID = FindInDB(Text);

Text = @"insert into TeacherSubjectTest Values('" + ID + "', '" + SubID + "', '"+AddedTestID+"')";

ExecuteQueryInDB(Text);

Text = @"insert into SubjectTest(SubjectID,TestID) Values('" + SubID + "', '" +AddedTestID+ "')";

ExecuteQueryInDB(Text);

Text = @"insert into AddedTest Values('" + TestName + "', '" + ID + "')";

ExecuteQueryInDB(Text);

}

public void DeleteTest(string TestID)

{

string Text = @"delete from TeacherSubjectTest where TestID='" + TestID + "'";

ExecuteQueryInDB(Text);

Text = @"delete from GroupSubjectTests where TestID='" + TestID + "'";

ExecuteQueryInDB(Text);

Text = @"delete from Variant where QuestionID in(select QuestionID from Question where TestId='" + TestID + "')";

ExecuteQueryInDB(Text);

Text = @"delete from Question where TestID='" + TestID + "'";

ExecuteQueryInDB(Text);

Text = @"delete from SubjectTest where TestID='" + TestID + "'";

ExecuteQueryInDB(Text);

Text = @"delete from Test where TestID='"+TestID+"'";

ExecuteQueryInDB(Text);

}

public void DeleteQuestion(string QuestionID)

{

string Text = @"delete from Variant where QuestionID='" + QuestionID + "'";

ExecuteQueryInDB(Text);

Text = @"delete from Question where QuestionID='" + QuestionID + "'";

ExecuteQueryInDB(Text);

}

public void AddNewQuestion(string Question,string TestID)

{

string Text = @"insert into Question(TestID,QuestionDescription) Values('" +TestID + "', '" + Question +"')";

ExecuteQueryInDB(Text);

}

public void EditVariant(string Variant, string Correctness)

{

if(Variant!=null && Correctness!=null)

{

string Text = @"update Variant set VariantDescription='"+Variant+"', VariantCorrectness='"+Correctness+"'";

ExecuteQueryInDB(Text);

}

}

public void DeleteVariant(string ID)

{

string Text = @"delete from Variant where VariantID='" +ID+ "'";

ExecuteQueryInDB(Text);

}

public void AddVariant(string Variant, string Correctness, string QuestionID)

{

if (Variant != null && Correctness != null)

{

string Text = @"insert into Variant (VariantDescription, QuestionID, VariantCorrectness) values ('"+Variant+"', '"+QuestionID+"', '"+Correctness+"')";

ExecuteQueryInDB(Text);

}

}

public string selectReadiness(string query)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader Select = command.ExecuteReader();

if (Select.HasRows)

{

Select.Read();

return Select.GetString(0);

}

else return "";

}

}

}

Test.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Testing\_System

{

public partial class Test : Form

{

int IndexOfCurrentQuestion;

double NumberOfCorrectAnswers;

double Mark=-1;

string VariantChooserText;

int TestSeconds;

public List<int> IDofQuestions;

public List<String> QuestionDesacriptions;

public List<List<int>> IDofVariants;

public List<List<string>> VariantDescriptions;

public List<string> CorrectVariantDescriptions;

public int TestMinutes;

string TestId;

string UserID;

public Test(string TestID,string TestName,string UserId="")

{

IndexOfCurrentQuestion = 0;

UserID = UserId;

TestId = TestID;

this.Text = TestName;

NumberOfCorrectAnswers = 0;

TestSeconds = 60;

Student MyStudent = new Student();

InitializeComponent();

MyStudent.GetAllTestInfo(TestID);

TestMinutes = MyStudent.TestTime - 1;

IDofQuestions = new List<int>(MyStudent.IDofQuestions);

QuestionDesacriptions = new List<string>(MyStudent.QuestionDesacriptions);

IDofVariants = new List<List<int>>(MyStudent.IDofVariants);

VariantDescriptions = new List<List<string>>(MyStudent.VariantDescriptions);

QuestionDescription.Text = QuestionDesacriptions[IndexOfCurrentQuestion];

CorrectVariantDescriptions = new List<string>(MyStudent.CorrectVariantDescriptions);

for (int VariantIndex = 0; VariantIndex < IDofVariants[IndexOfCurrentQuestion].Count; VariantIndex++)

{

VariantChooser.Items.Add(VariantDescriptions[IndexOfCurrentQuestion][VariantIndex]);

}

timer1.Start();

}

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

{

VariantChooserText = VariantChooser.Text;

}

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

{

if (VariantChooserText == CorrectVariantDescriptions[IndexOfCurrentQuestion])

{

NumberOfCorrectAnswers++;

}

if (IndexOfCurrentQuestion < IDofQuestions.Count-1)

{

IndexOfCurrentQuestion++;

QuestionDescription.Text = QuestionDesacriptions[IndexOfCurrentQuestion];

VariantChooser.Items.Clear();

VariantChooser.Text = "";

for (int VariantIndex = 0; VariantIndex < IDofVariants[IndexOfCurrentQuestion].Count; VariantIndex++)

{

VariantChooser.Items.Add(VariantDescriptions[IndexOfCurrentQuestion][VariantIndex]);

}

}

else

{

Mark = NumberOfCorrectAnswers / IDofQuestions.Count\*100;

MessageBox.Show("Поздравляем, вы сдали тест на " + Mark + " процентов");

Student MyStudent = new Student();

MyStudent.UpdatePassedTest((int)Mark, TestId, UserID);

this.Close();

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (TestMinutes > 0 && TestSeconds>0)

{

TestSeconds--;

Time.Text = TestMinutes + " minutes " + TestSeconds + " seconds";

if (TestSeconds==0)

{

TestMinutes--;

TestSeconds = 60;

}

}

else

{

timer1.Stop();

Time.Text = "Time's up!";

MessageBox.Show("You didn't finish in time.", "Sorry!");

}

}

}

}

User.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Text;

namespace Testing\_System

{

class User

{

public string ID { get; set; }

public string Role { get; set; }

public bool CheckInDataBase(string selectionCommandString)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

bool Checker = dataReader.HasRows;

dataReader.Close();

connection.Close();

return Checker;

}

public void ExecuteQueryInDB(string selectionCommandString)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

command.ExecuteNonQuery();

connection.Close();

}

public string FindInDB(string selectionCommandString)

{

string connectionString = "Server=DanDan;Database=TestingSystem;Trusted\_Connection=True";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

SqlCommand command = new SqlCommand(selectionCommandString, connection);

SqlDataReader dataReader = command.ExecuteReader();

if (dataReader.HasRows)

{

dataReader.Read();

if (dataReader.GetDataTypeName(0) == "int")

{

return dataReader.GetInt32(0).ToString();

}

if (dataReader.GetDataTypeName(0) == "varchar")

{

return dataReader.GetString(0);

}

return "";

}

else return "";

}

}

}

# Руководство пользователя

1. **Введение.** 
   1. Область применения

Требования настоящего документа применяются при:

* + - промышленной эксплуатации;
    - приемочных испытаниях;
    - опытной эксплуатации;
    - предварительных комплексных испытаниях;
  1. Краткое описание возможностей

Система предназначена для возможности проведения студентам тестирования дистанционно с помощью десктопного приложения.

* 1. Уровень подготовки пользователя

Пользователь системы должен иметь опыт работы с OC Windows (7/8/10/11).

* 1. Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю - Руководство пользователя OC Windows.

1. **Назначения и условия применения.**
2. Система предназначена для работы со списками.

**3. Подготовка к работе.**

3.1 Состав и содержание дистрибутивного носителя

1. OC Windows (7/8/10/11).

3.2 Порядок загрузки данных и программ

* + - 1. Запустить программу. (При условии, что сервер MSSQL

Постоянно включен)

3.3 Порядок проверки работоспособности

* + 1. Запустить программу.
    2. Пройти авторизацию/регистрацию.
    3. Убедиться, что открылось окно.
    4. Убедиться в работоспособности открывшегося окна

**4.Описание операций. (Администратор)**

4.1. Выполняемые функции и задачи Функции: Составление списков преподавателей, студентов, групп, дисциплин и тестов.

Задачи:

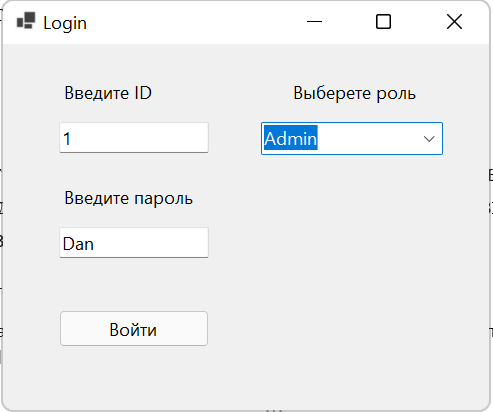
- Вводит новых пользователей и редактирует их данные, добавляет дисциплины и тесты к ним, а также имеет возможность редактировать их, может в этом заменить учителя.

Операция 1: «Авторизация Админа»

Для выполнения авторизации необходимо знать ID и пароль совпадающие с данными в БД ввести их и в поля программы и выбрать роль «Admin»

Изображение выглядит как стол

Автоматически созданное описание



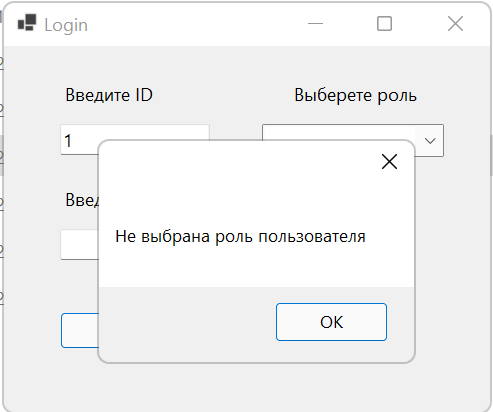


Тогда нажав на обведенную кнопку мы перейдем в окно Админа

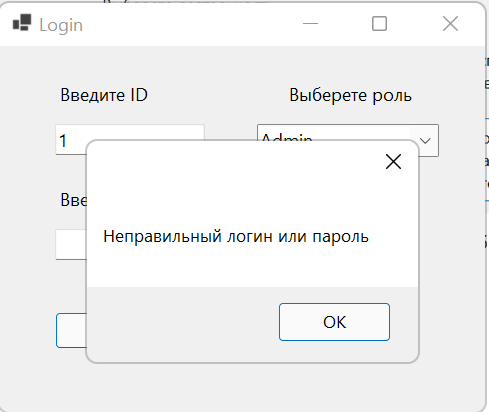
Изображение выглядит как текст

Автоматически созданное описание

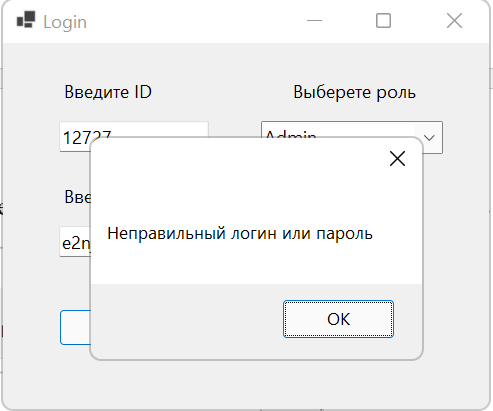
Если мы нарушим хотя бы одно из условий появится уведомление об ошибке



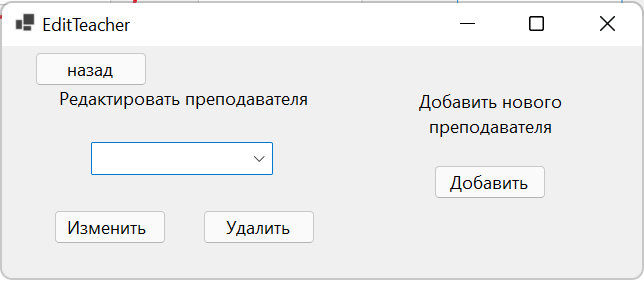
Или такая если выбрали роль но не ввели ID или пароль



Это же произойдет если мы введем случайное неправильное значение



Следующие 3 операций находятся в форме



Для перехода в которую необходимо нажать на 1 кнопку в форме после авторизации

Изображение выглядит как текст

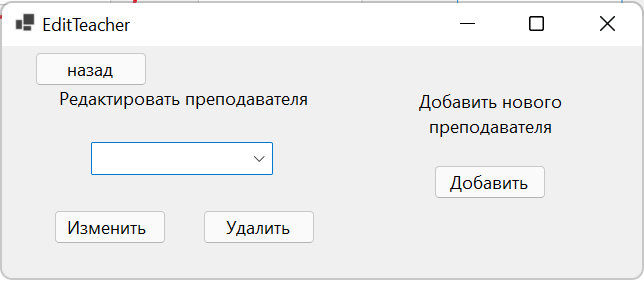
Автоматически созданное описание



Операция 2: «Добавить преподователя»

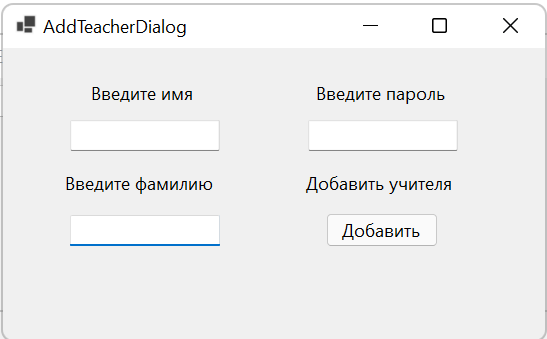


На форме нажмите на обведенную кнопку для дальнейших действий.





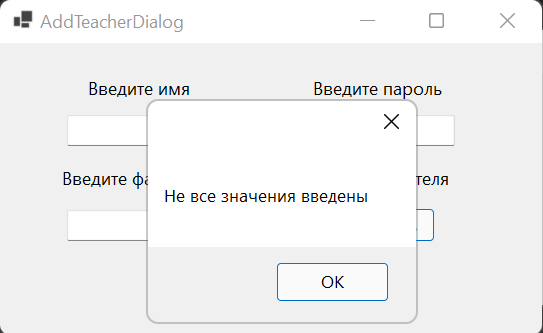
В появившемся диалоге заполните информацию о преподавателе и нажмите на обведенную кнопку

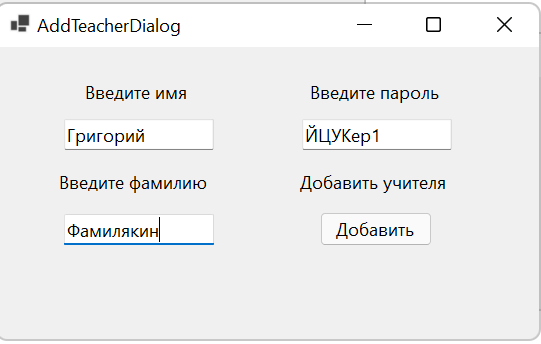




При закрытии окна диалога операция не будет выполнена

Также есть проверка на то что не все данные введены







Удостовериться что операция прошла успешно является отображение

Нового преподавателя в списке, а также в БД

Изображение выглядит как текст

Автоматически созданное описание



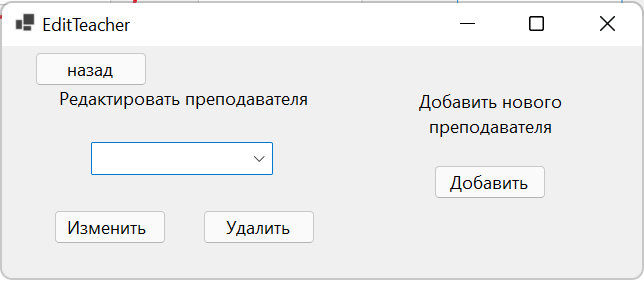
Изображение выглядит как стол

Автоматически созданное описание



Операция 3: «Удаление преподавателя»

В той же форме необходимо выбрать преподавателя для удаления, нажав на выпадающий список





Выберем только что добавленного преподавателя

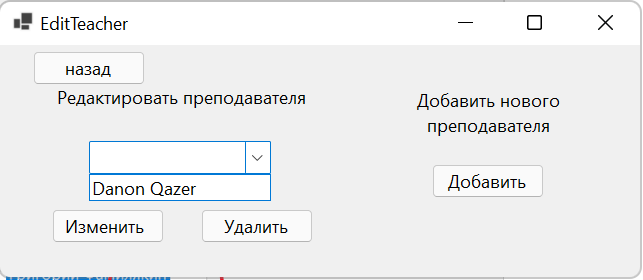
Изображение выглядит как текст

Автоматически созданное описание



Необходимо нажать на обведенную кнопку

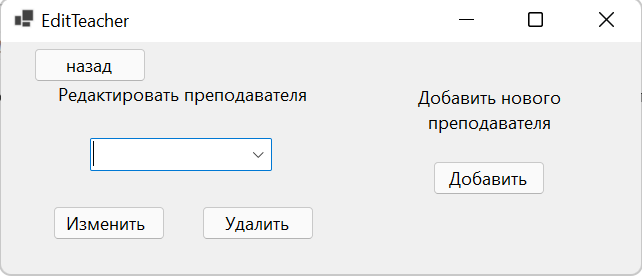
Результат – это: отсутствие удаленного преподавателя в списке и в БД



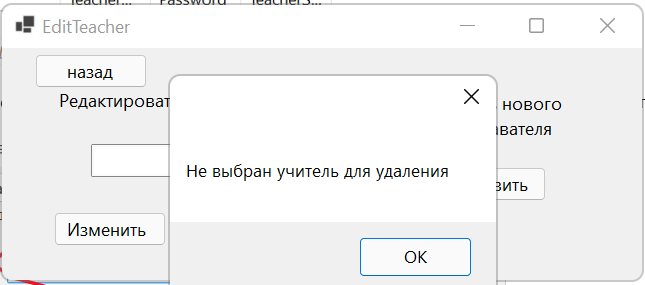
Изображение выглядит как стол

Автоматически созданное описание

Но удостоверьтесь что вы выбрали в списке преподавателя иначе вас уведомит программа







Операция 4: «Редактирование преподавателя»

Для этой операции так же как и для удаления выполняется проверка на отсутствие выбора преподавателя

Изображение выглядит как текст

Автоматически созданное описание

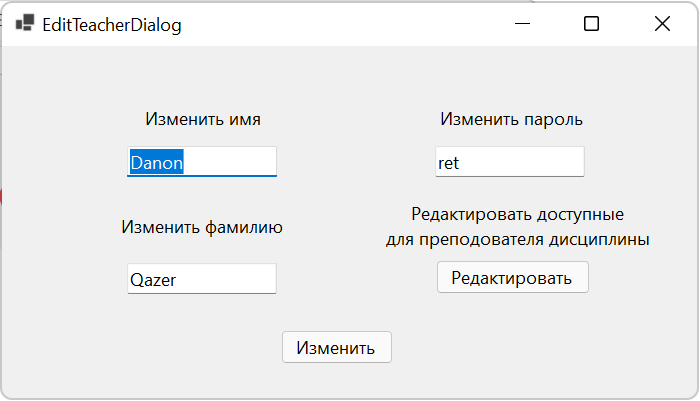
Для успешного выполнения операции необходимо выбрать преподавателя в списке и нажать на обведенную кнопку

Изображение выглядит как текст

Автоматически созданное описание



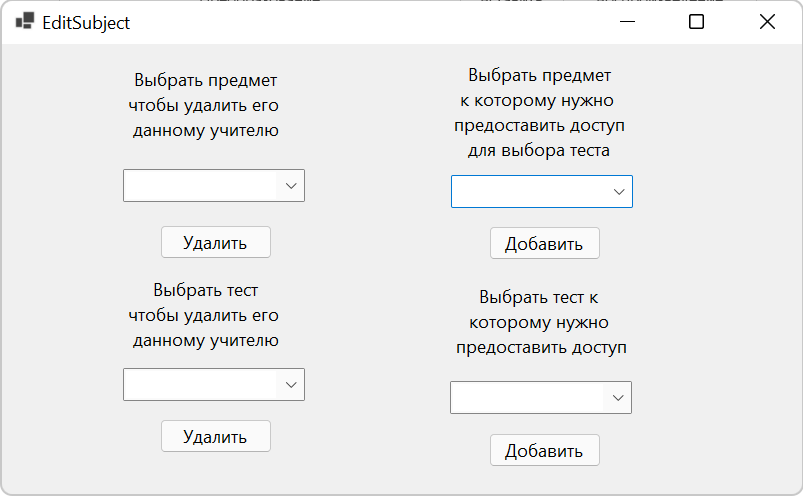
Откроется похожий при добавлении диалог с той разницей, что мы редактируем уже существующие данные а также можем добавить преподавателю доступные дисциплины





При закрытии окна диалога операция не будет выполнена

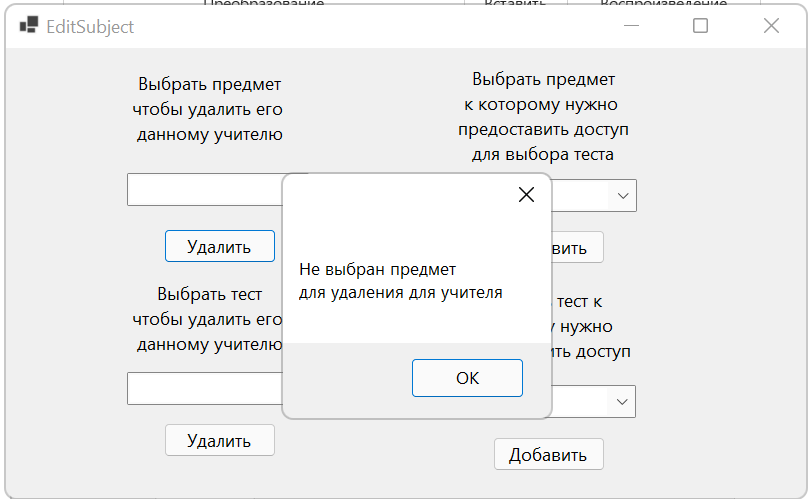
Нажав на обведенную кнопку откроется еще один диалог в котором все предельно объясненно надписями



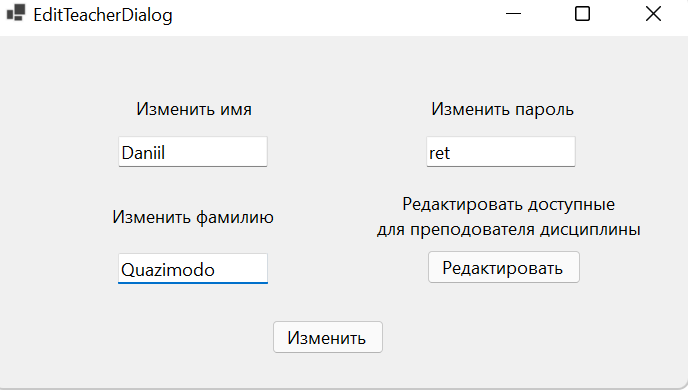


Редактирование доступа совершается сразу после нажатия на кнопки контроля

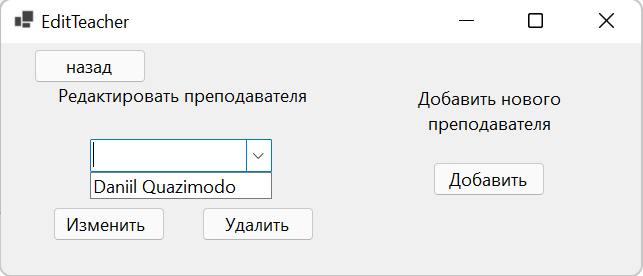
При условии что выбран какой-либо элемент иначе высветится знакомое уведомление, например



Изменение данных происходит после нажатия на обведенную кнопки, изменим же данные и проверим отклик программы и БД







Изображение выглядит как стол

Автоматически созданное описание

Данные действительно обновились.

Следующие 6 операций находятся в форме

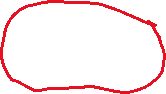
Изображение выглядит как текст

Автоматически созданное описание

Для перехода в которую необходимо нажать на 2 кнопку в форме после авторизации

Изображение выглядит как текст

Автоматически созданное описание



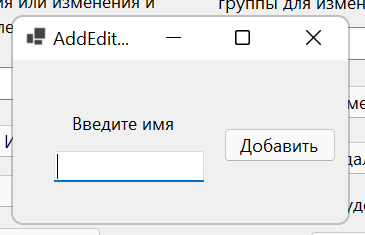
Операция 10: «Добавление группы»

Операция начнется после нажатия на обведенную кнопку

Изображение выглядит как текст

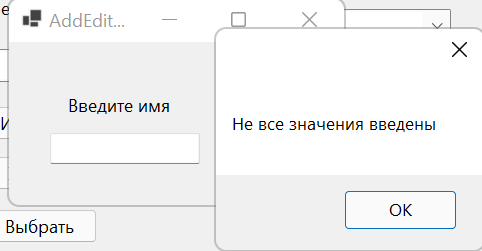
Автоматически созданное описание



Появится форма диалога при закрытии которой операция будет отменена, 



При добавлении пустого значении появится уведомление об ошибке



На форме введем имя группы и нажмем на обведенную кнопку

Изображение выглядит как текст

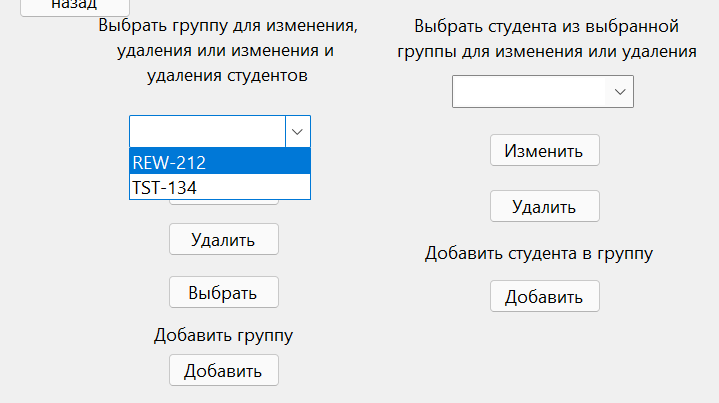
Автоматически созданное описание



Результат это: добавление новой группы в БД и в список существующих групп

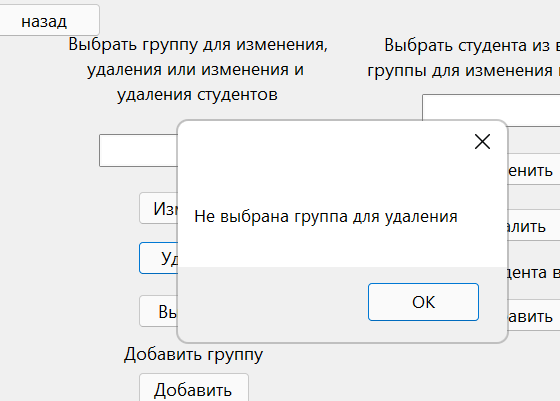
Изображение выглядит как стол

Автоматически созданное описание



Операция 6: «Удаление группы»

Для выполнения операции необходимо выбрать группу в списке групп иначе появится уведомление об ошибке



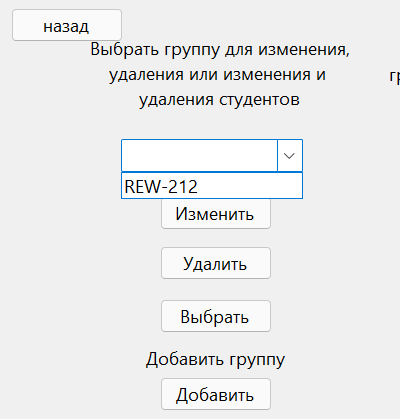
Выбрав группу нажмем на обведенную кнопку

Изображение выглядит как текст

Автоматически созданное описание



Результат успешного выполнения – удаление из списка групп, а также из БД

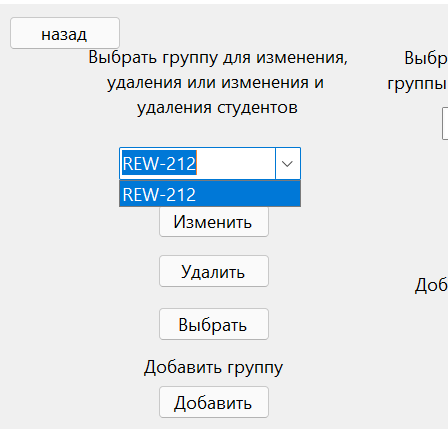
 Изображение выглядит как стол

Автоматически созданное описание



Операция 7: «Изменение названия группы»

Для запуска операции нужно выбрать группу в списке групп и нажать на обведенную кнопку



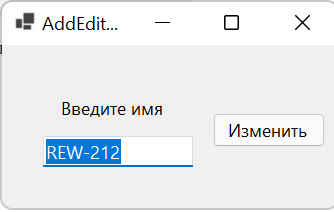


Если не выбрать группу то появится уведомление об ошибке

Изображение выглядит как текст

Автоматически созданное описание

Далее откроется диалог идентичный диалогу ввода только с существующим именем группы



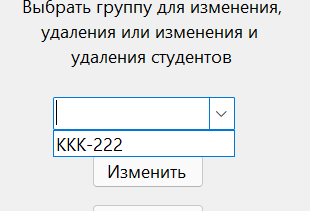
При вводе нулевого значения высветится ошибка

Изображение выглядит как текст

Автоматически созданное описание

Результатом нажатия на кнопку «Изменить» является обновление данных в списке и в БД

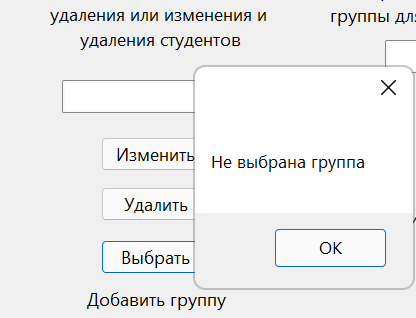
Изображение выглядит как текст

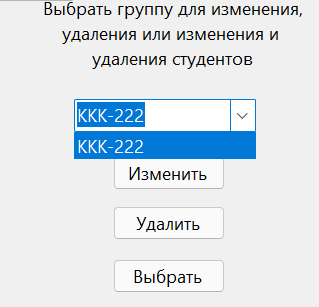
Автоматически созданное описание  Изображение выглядит как стол

Автоматически созданное описание

Следующие 3 операции возможны только при выборе предмета в списке и нажатии кнопки «Выбрать»

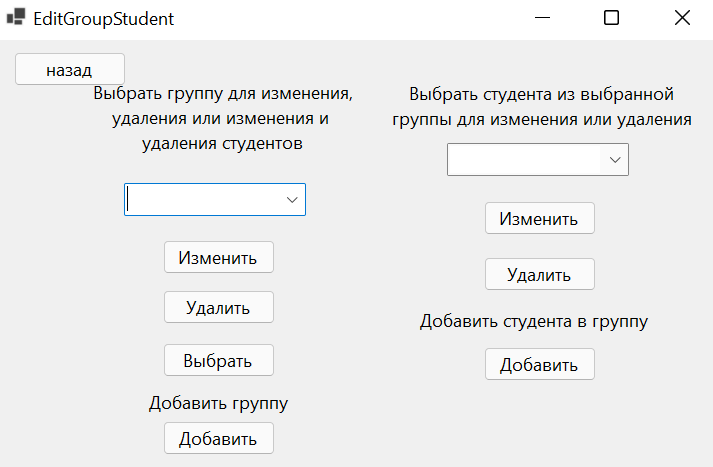
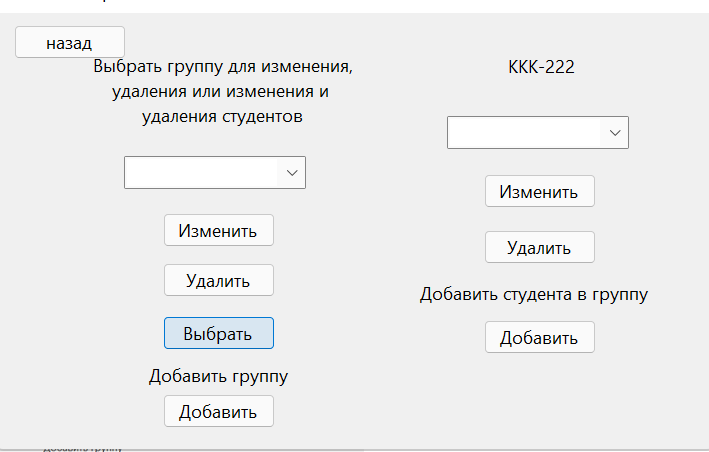
Не выбрав в списке предмет высветится ошибка







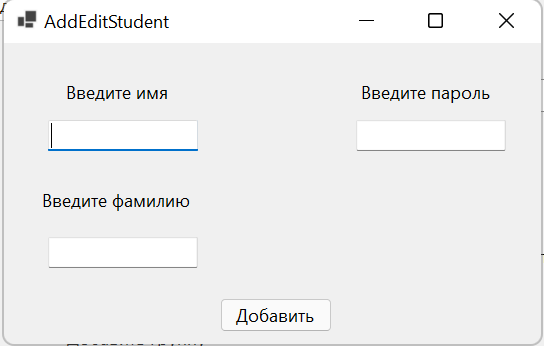
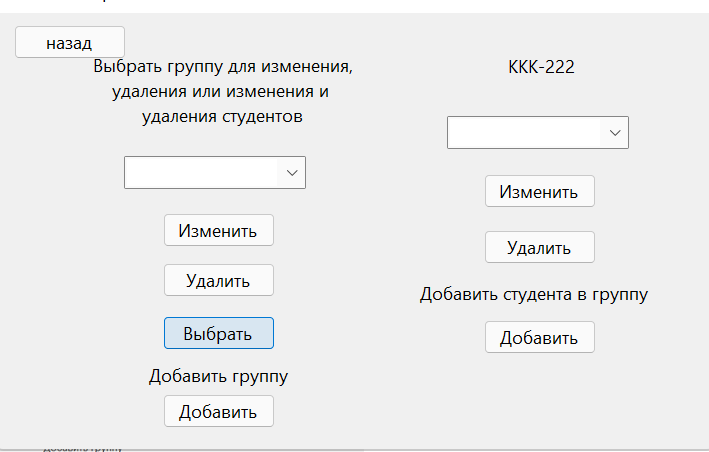
После нажатия текст над выбором студента изменится на название группы

Что означает что группа выбрана и можно выбирать из нее студентов для редактирования или удаления, добавляться они также будут только в эту группу

Операция 8: «Добавление студента в группу»

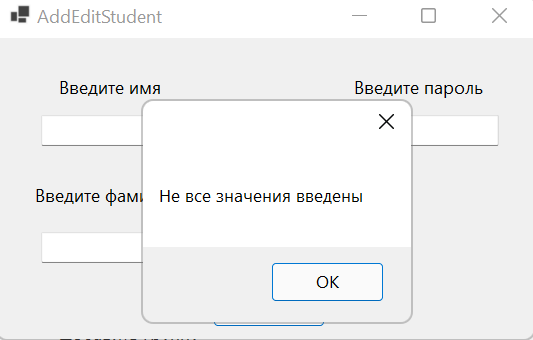
Для выполнения данной операции необходимо нажать на кнопку «Добавить»





После этого откроется форма справа, если мы оставим пустым хотя бы одно значение высветится

Ошибка

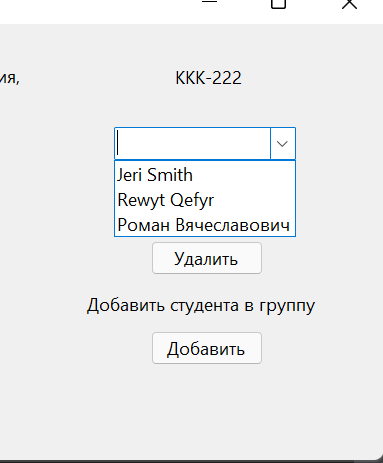


После ввода всех данных и нажатия на кнопку «Добавить» операция завершится

Изображение выглядит как текст

Автоматически созданное описание

И результатом будет добавление студента в список студентов группы и в БД

Изображение выглядит как стол

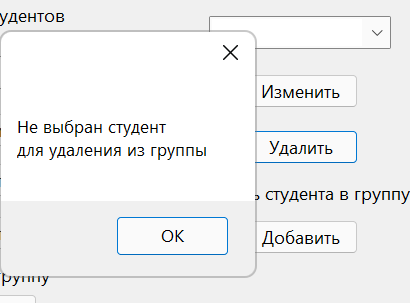
Автоматически созданное описание

Операция 9: «Удалить студента из группы»

Операция запустится при выборе студента из списка и нажатии на кнопку «Удалить»

Изображение выглядит как текст

Автоматически созданное описание Если студент не будет выбран высветится ошибка



При выборе студента и нажатии на кнопку результат будет следующий:

студент удалится из списка студентов группы а также из БД

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как стол

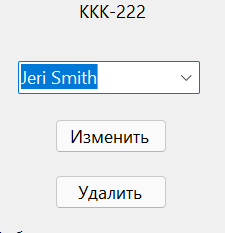
Автоматически созданное описание

Операция 10: «Изменение данных студента выбранной группы»

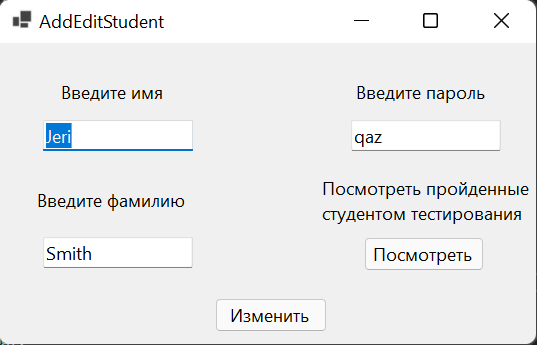
Если не выбрать студента из списка появится уведомление об ошибке

Изображение выглядит как текст

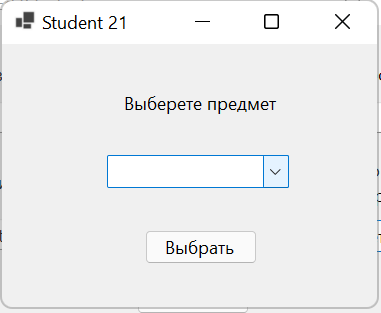
Автоматически созданное описание

Выбрав же студента из списка и нажав на кнопку «Изменить» откроется

форма редактирования с данными студента а также с кнопкой перехода к просмотру пройденных тестов



Перейдем в нее нажав на кнопку «Посмотреть»

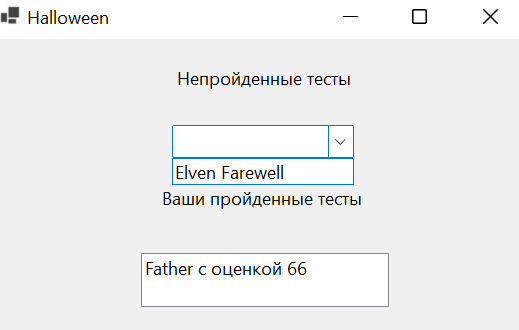


Откроется окно выбора предмета для просмотра оценки за тест

Изображение выглядит как текст

Автоматически созданное описание

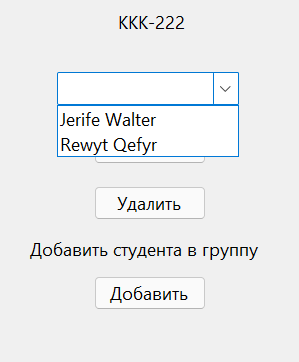
Выбрав доступное нажимаем на кнопку «Выбрать»



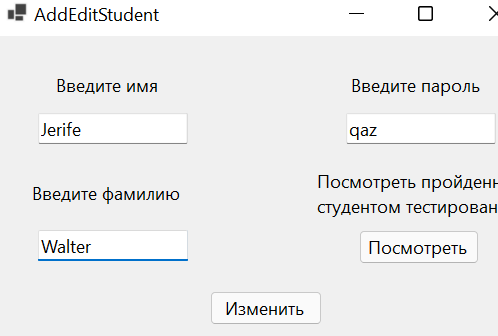
Открывается окно с пройденными и не пройденными тестами если они конечно были доступны группе студента иначе ничего не высветится

Изменение данных происходит аналогично с добавлением данных студента

Изменим данные и нажмем на кнопку «Изменить» Результатом будет обновление данных в списке студентов и в БД

Изображение выглядит как стол

Автоматически созданное описание



Следующие 6 операций находятся в форме

Изображение выглядит как текст

Автоматически созданное описание

Для перехода в которую необходимо нажать на 3 кнопку в форме после авторизации

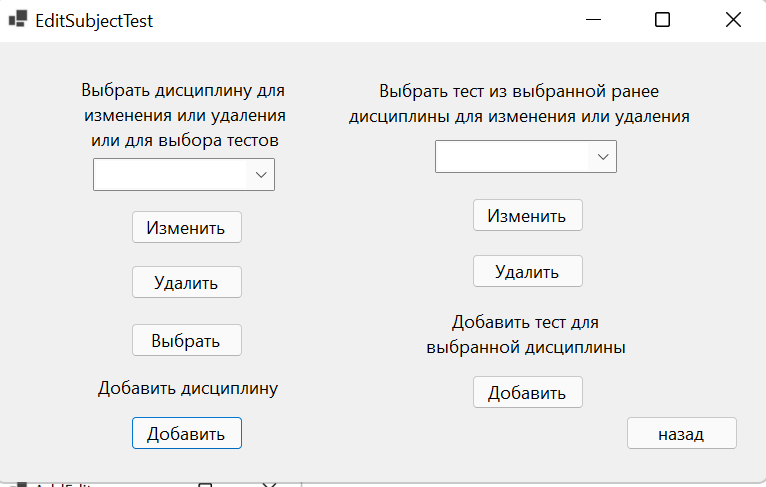
Изображение выглядит как текст

Автоматически созданное описание



Операция 11: «Добавление предмета»

Операция начнется после нажатия на обведенную кнопку





Появится форма диалога при закрытии которой операция будет отменена,

Изображение выглядит как текст

Автоматически созданное описание



При добавлении пустого значении появится уведомление об ошибке

Изображение выглядит как текст

Автоматически созданное описание

На форме введем имя предмета и нажмем на обведенную кнопку

Изображение выглядит как текст

Автоматически созданное описание



Результат это: добавление нового предмета в БД и в список существующих предметов

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как стол

Автоматически созданное описание

Операция 12: «Удаление предмета»

Для выполнения операции необходимо выбрать предмет в списке предметов иначе появится уведомление об ошибке

Изображение выглядит как текст

Автоматически созданное описание

Выбрав предмет нажмем на обведенную кнопку

Изображение выглядит как текст

Автоматически созданное описание



Результат успешного выполнения – удаление из списка предметов, а также из БД

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как стол

Автоматически созданное описание



Операция 13: «Изменение названия предмета»

Для запуска операции нужно выбрать предмет в списке предметов и нажать на обведенную кнопку

Изображение выглядит как текст

Автоматически созданное описание



Если не выбрать группу то появится уведомление об ошибке

Изображение выглядит как текст

Автоматически созданное описание

Далее откроется диалог идентичный диалогу ввода только с существующим именем предмета

Изображение выглядит как текст

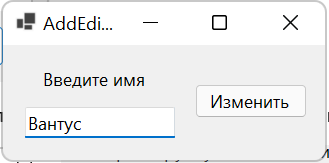
Автоматически созданное описание

При вводе нулевого значения высветится ошибка

Изображение выглядит как текст

Автоматически созданное описание

Результатом нажатия на кнопку «Изменить» является обновление данных в списке и в БД

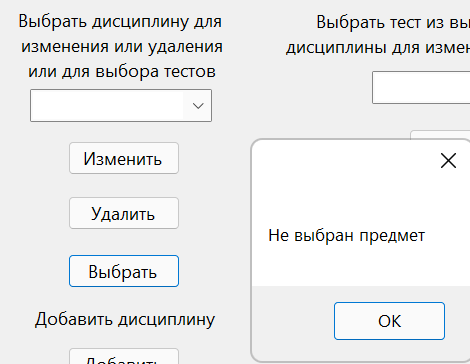
 Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как стол

Автоматически созданное описание

Следующие 3 операции возможны только при выборе предмета в списке и нажатии кнопки «Выбрать»

Не выбрав в списке предмет высветится ошибка

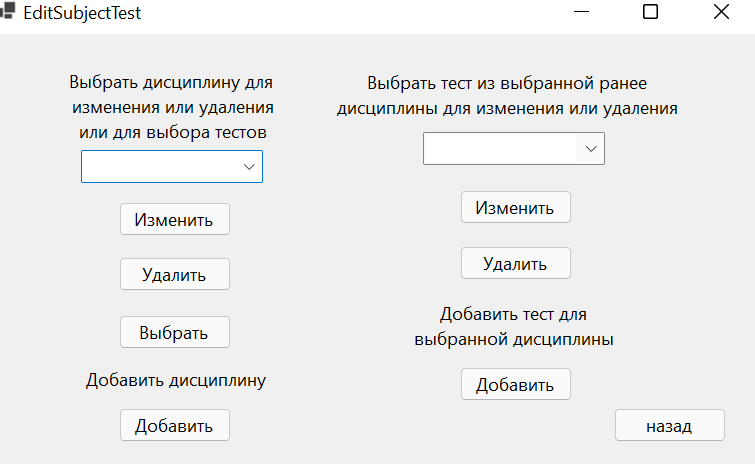
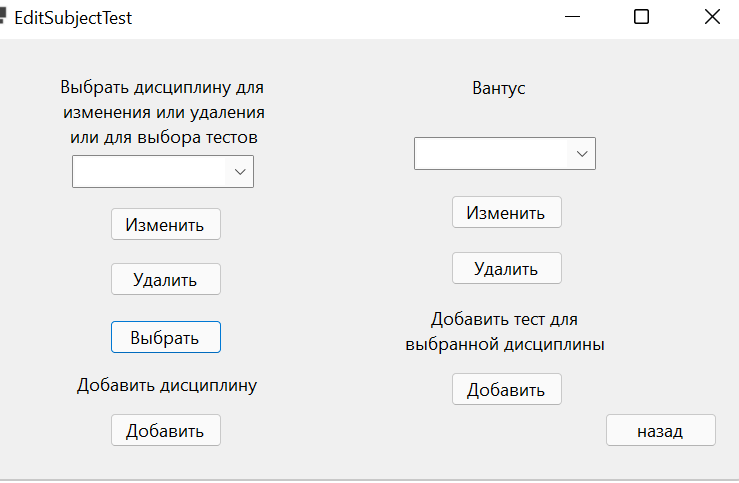


Изображение выглядит как текст

Автоматически созданное описание



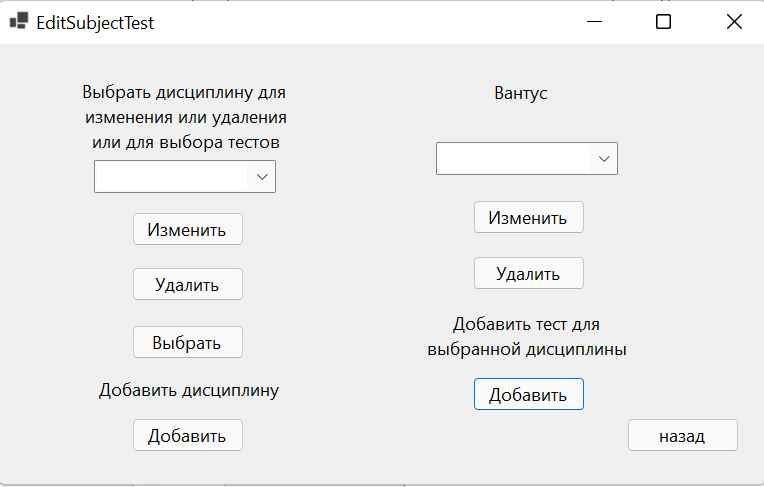
После нажатия текст над выбором студента изменится на название группы

Что означает что предмет выбран и можно выбирать из него студентов тесты для редактирования или удаления, добавляться они также будут только в этот предмет

Операция 14: «Добавление теста в предмет»

Для выполнения данной операции необходимо нажать на кнопку «Добавить»

Изображение выглядит как текст

Автоматически созданное описаниеПосле этого откроется форма справа, если мы оставим пустым хотя бы одно значение высветится



Ошибка

Изображение выглядит как текст

Автоматически созданное описание

После ввода всех данных и нажатия на кнопку «Добавить» операция завершится

Изображение выглядит как текст

Автоматически созданное описание

И результатом будет добавление студента в список студентов группы и в БД

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как стол

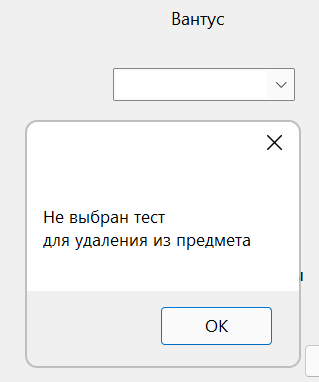
Автоматически созданное описание

Операция 15: «Удалить тест из предмета»

Операция запустится при выборе теста из списка и нажатии на кнопку «Удалить»

Изображение выглядит как текст

Автоматически созданное описаниеЕсли студент не будет выбран высветится ошибка



При выборе студента и нажатии на кнопку результат будет следующий:

студент удалится из списка студентов группы а также из БД

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описание

Операция 16: «Редактирование теста в дисциплине»

Данная операция начнет выполнение при выборе теста из списка тестов и нажатии кнопки «Изменить»

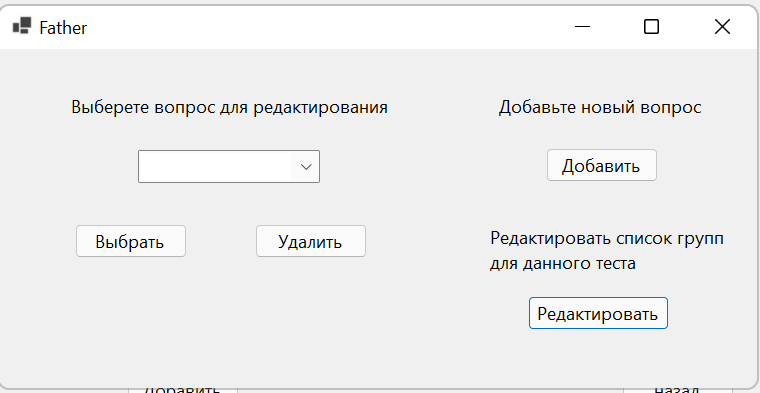
Не выбрав тест из списка высветится ошибка

Изображение выглядит как текст

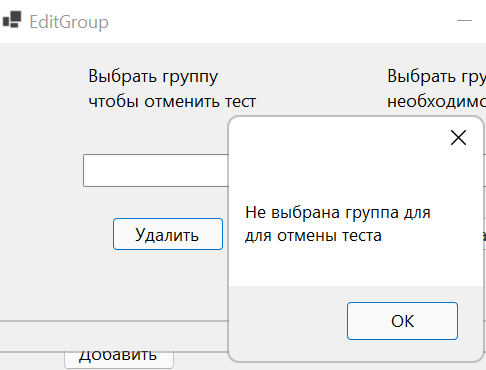
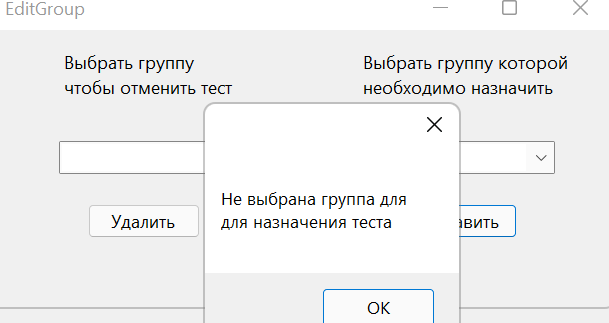
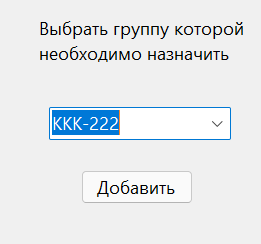
Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

После выбора теста откроется диалог редактирования данного теста где можно редактировать вопросы и назначать тесты определенной группе при нажатии кнопки «Редактировать»

* 



* Откроется диалог добавление и удаление теста группам
* Не выбрав группу высветится ошибка
* 
* Результат добавления
* Изображение выглядит как стол

  Автоматически созданное описаниеИзображение выглядит как стол

  Автоматически созданное описаниеИзображение выглядит как стол

  Автоматически созданное описание
* Добавление в таблицу связи ID группы, предмета и теста и в список доступных групп
* Изображение выглядит как текст

  Автоматически созданное описание
* При удалении происходит удаление из таблицы связи и из списка
* Изображение выглядит как текст

  Автоматически созданное описаниеИзображение выглядит как текст

  Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, белый

  Автоматически созданное описание
* Редактирование вопросов и вариантов ответа происходит также
* Как и других вещей в программе.

s