ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ РОСТОВСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ (РИНХ)

Кафедра Информационных систем и прикладной информатики

Контрольная работа  
Разработка и сопровождение программных приложений

Выполнил студент:  
Группы 321-ПИZS  
Моисеенко Данил Александрович

Научный руководитель:  
Потапов Леонид Игоревич

Ростов-на-Дону   
2022г

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc87204685)

[1 Описание предметной области 4](#_Toc87204686)

[1.1 Описание задачи необходимых для поддержания в актуальном состоянии информации по студенческим группам 4](#_Toc87204687)

[2 Разработка моделей данных 5](#_Toc87204688)

[2.1 Создание модели данных 5](#_Toc87204689)

[3 Разработка представления данных по моделям 8](#_Toc87204690)

[3.1 Создание представления моделей и отображение данных 8](#_Toc87204691)

[4 Создание, редактирование и удаление данных форм 9](#_Toc87204692)

[4.1 Добавление данных 9](#_Toc87204693)

[4.2 Редактирование данных 10](#_Toc87204694)

[4.3 Удаление данных 11](#_Toc87204695)

[Заключение 12](#_Toc87204696)

[Список использованных источников 13](#_Toc87204697)

[Приложения](#_Toc87204698)

# **ВВЕДЕНИЕ**

В наше время существует множество программных приложений позволяющих обеспечивать качественное хранение и обработку информации. Так для хранения большого объема информации, касающейся определенной области очень удобно пользоваться системами управления базами данных (СУБД). Под базой данных (БД) будем понимать совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязь в конкретной предметной области. СУБД позволяет:

1. надежно хранить информацию;
2. изменять (добавлять, удалять, обновлять) информацию;
3. уменьшить время доступа к необходимой информации;
4. реализовать различные уровни доступа к информации, рассчитанные на различных пользователей.

Таким образом, СУБД очень хорошо подходят для хранения и систематизации любой информации на работе.

В последнее время базы данных находят всё более широкое применение в нашей жизни. Практически во всех отраслях экономики, промышленности, рыночных отношений используются базы данных, позволяющие хранить и обрабатывать информацию.

В рамках работы был произведен анализ существующих подсистем по поддержанию состояния информации студенческих групп. Было разработано приложение, которое помогает управлять состоянием информации студенческих групп, благодаря чему мы можем все время получать актуальную информацию студентов.

Объектом исследования является информационная подсистема формирования данных по студенческим группам.

Цель работы разработка приложения для поддержания информации студенческих групп с помощью языка программирования C#.

# **Описание предметной области**

Назначение подсистемы поддержание в актуальном состоянии информации по студенческим группам *(Рисунок 1)*.

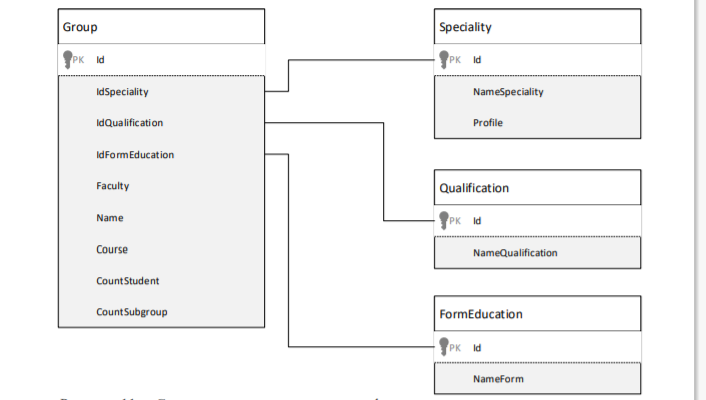


Рисунок 1 – Структура классов подсистемы формировании данных по студенческим группам

* 1. **Описание задачи необходимых для поддержания в актуальном состоянии информации по студенческим группам**
* Разработка и сопровождение программных систем. Технологии Microsoft.NET для разработки приложений;
* Разработка моделей данных;
* Разработка представления данных по моделям;
* Создание, удаление данных студенческих групп;
* Создание, редактирование и удаление данных о квалификации;
* Создание, редактирование и удаление данных о направлении подготовки;
* Создание, удаление данных формы обучения.

# **Разработка моделей данных**

* 1. **Создание модели данных**

Общие сведения:

Создается WPF-приложение для обработки данных с студенческими группами *Рисунок 1*. Приложение должно отображать данные о студенческих групп, добавлять, редактировать и удалять данные.

Создаем проект приложения на основе шаблона классического приложения на C#. В созданный проект добавим три папки *(Рисунок 2):*

1. Model;
2. View;
3. ViewModel.

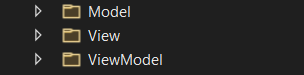


Рисунок 2 – Папки для проекта

В папке Model создаем классы, которые описывают предметную область *(Рисунок 3).*

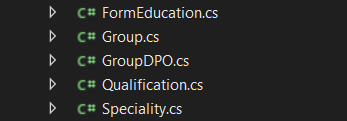


Рисунок 3 – Классы папки Model

В папке ViewModel создадим классы, в которых будут сформированы коллекции данных *(Рисунок 4).*

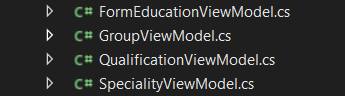


Рисунок 4 – Классы папки ViewModel

В папке View создадим окна для представления данных *(Рисунок 5).*

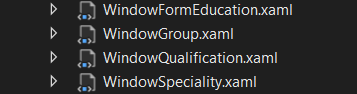


Рисунок 5 – Представления View

Для представления данных студенческих групп, шаблон экранной формы будет иметь вид, представленный на *(Рисунок 6).*

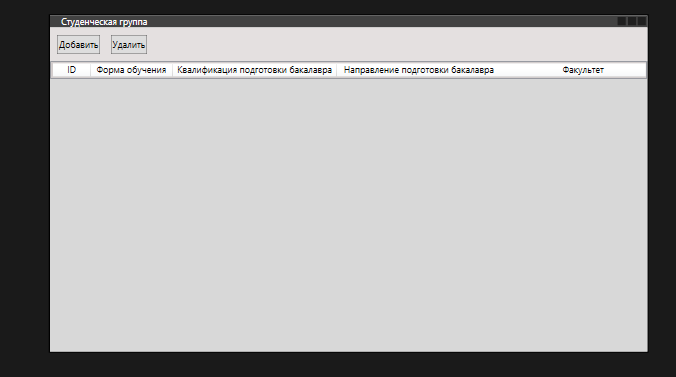


Рисунок 6 – Готовый шаблон для формы студенческая группа

На главном окне приложения создадим меню для вывода и обработки данных *(Рисунок 7)*.

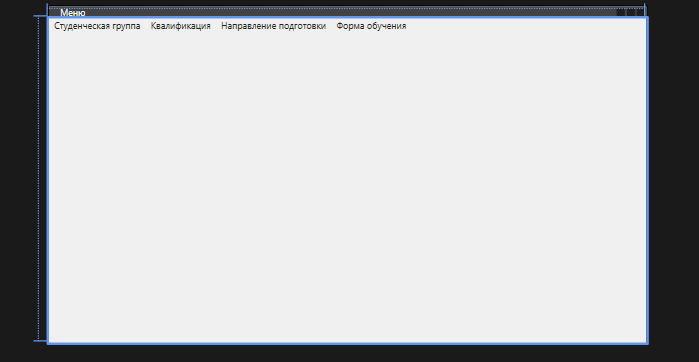


Рисунок 7 – Главное меню программы

XAML-разметка главной формы.

<Window x:Class="Variant\_11.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Variant\_11"

mc:Ignorable="d"

Title="Меню" Height="450" Width="800">

<Grid>

<Menu>

<MenuItem x:Name="Group" Header="Студенческая группа" Click="Group\_Click"></MenuItem>

<MenuItem x:Name="Qualification" Header="Квалификация" Click="Qualification\_Click"></MenuItem>

<MenuItem x:Name="Speciality" Header="Направление подготовки" Click="Speciality\_Click"></MenuItem>

<MenuItem x:Name="FormEducation" Header="Форма обучения" Click="FormEducation\_Click"></MenuItem>

</Menu>

</Grid>

</Window>

# **Разработка представления данных по моделям**

* 1. **Создание представления моделей и отображение данных**

Для отображения списка будем использовать класс ListView, который предназначен для отображения данных спискового типа, состоящих из множества столбцов. Для класса ListView используется свойство View (ListView.View), которое обеспечивает стили и форматирование данных. Для создания в отображении списка столбцов будем использовать класс GridView, который предоставляет списковое представление со множеством столбцов. Столбцы определяются в коллекции GridView.Columns путем задания объектов GridViewColumn. На данном этапе проектирования приложения для объектов GridViewColumn зададим только заголовок столбца Header.

После сборки и запуска программы при вызове окна квалификации экранная форма будет иметь вид *(Рисунок 8).*

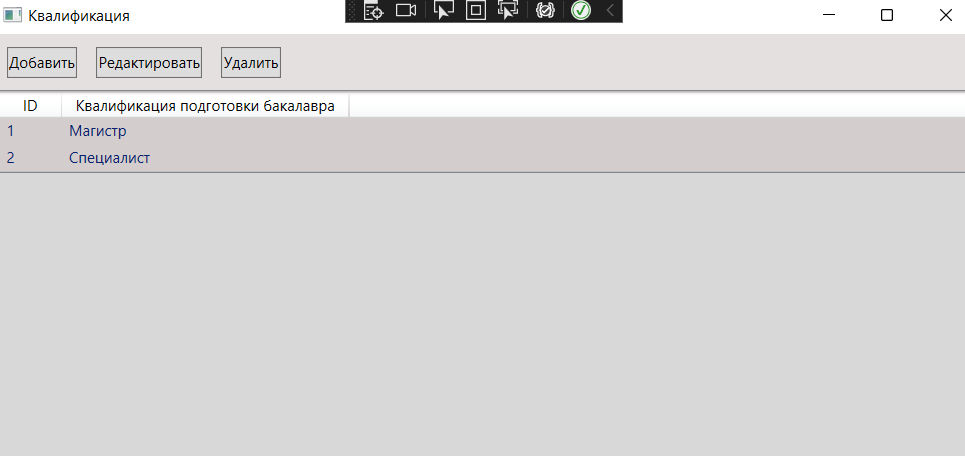


Рисунок 8 – Форма квалифиация

# **Создание, редактирование и удаление данных форм**

* 1. **Добавление данных**

Для обеспечения функциональности обработки данных добавим в окно формы кнопки добавить, удалить и редактировать *(Рисунок 9)*.

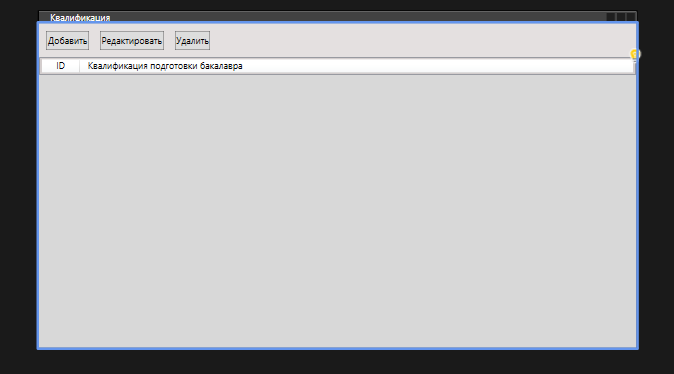


Рисунок 9 – Пример кнопок на форме квалификации

При создании новой квалификации её код должен вычисляться как максимальный код квалификации в имеющихся данных, т.е. в списке квалификации, увеличенный на единицу. Для определения максимального значения кода в списке добавим в класс QualificationViewModel метод MaxId().

public int MaxId()

{

int max = 0;

foreach (var p in this.ListQualification)

{

if (max < p.ID)

{

max = p.ID;

};

}

return max;

}

* 1. **Редактирование данных**

При редактировании данных создадим для кнопки редактировать окна WindowsRole код обработчика btnEdit\_Click.

private void btnEdit\_Click(object sender, RoutedEventArgs e)

{

WindowNewQualification wnqual = new WindowNewQualification

{

Title = "Редактирование",

Owner = this

};

Qualification qualification = WNPQualification.SelectedItem as Qualification;

if (qualification != null)

{

Qualification qqq = qualification.ShallowCopy();

wnqual.DataContext = qqq;

if (wnqual.ShowDialog() == true)

{

qualification.NameQualification = qqq.NameQualification;

WNPQualification.ItemsSource = null;

WNPQualification.ItemsSource = vmQual.ListQualification;

}

}

else

{

MessageBox.Show("Необходимо выбрать поле для редактированния",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

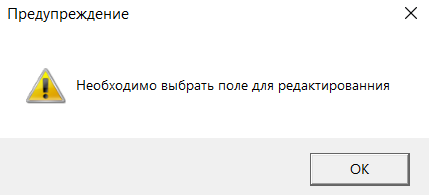
} 

Рисунок 10 – Предупреждение редактирования

* 1. **Удаление данных**

При удалении создадим для кнопки удалить окна WindowsRole код обработчика btnDelete\_Click.

private void btnDelete\_Click(object sender, RoutedEventArgs e)

{

Qualification qual = (Qualification)WNPQualification.SelectedItem;

if (qual != null)

{

MessageBoxResult result = MessageBox.Show("Удалить " +

qual.NameQualification, "Предупреждение", MessageBoxButton.OKCancel,

MessageBoxImage.Warning);

if (result == MessageBoxResult.OK)

{

vmQual.ListQualification.Remove(qual);

}

}

else

{

MessageBox.Show("Необходимо выбрать поле для удаления",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

Если выбор для удаления произведен, то выводим предупреждение *(Рисунок 11)*.

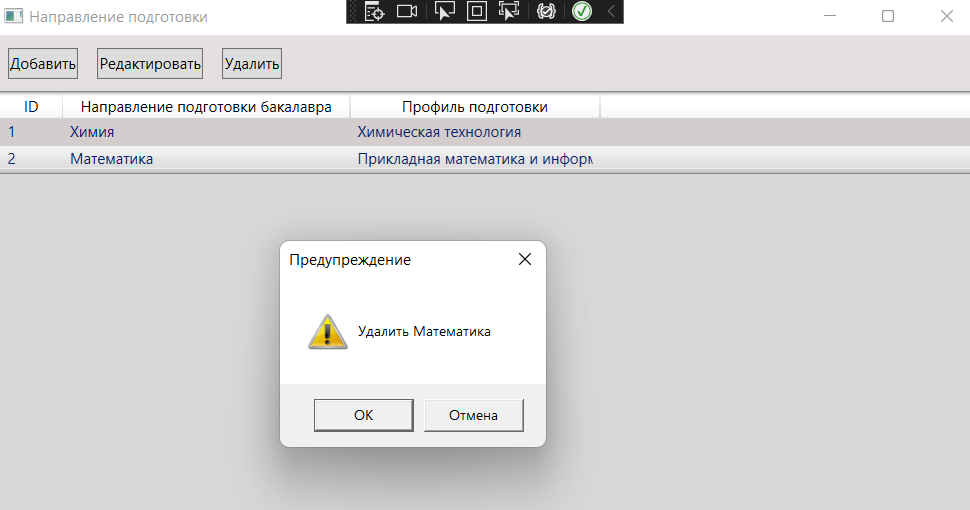


Рисунок 11 – Предупреждение об удалении данных

# **ЗАКЛЮЧЕНИЕ**

В ходе данной контрольной работы было создано приложение для управления состояния студенческих групп, которое позволяет мониторить информацию о группах в реальный момент времени. Приложение было разработано на платформе С#, WPF. Данная платформа предназначена для создания приложений на базе.NET. Была использована актуальная на сегодняшний день платформа компании Microsoft Visual Studio.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Основы WPF // <http://msdn.microsoft.com/ruru/library/ms746927.aspx>.
2. Мак-Дональд, М. WPF:Windows Presentation Foundation в .NET 3.0 для профессионалов.: Пер. с англ. –М.: ООО “И.Д. Вильямс”, 2008.
3. Нейгел, К. C# 4.0 и платформа .NET 4 для профессионалов. / К. Нейгел, Б. Ивьен, Дж. Глинн, К. Уотсон.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2011.
4. Петцольд Ч. Microsoft Windows Presentation Foundation. Базовый курс.: - М.: Изд. «Русская Редакция» СПб.: Питер, 2008.
5. Троелсен, Э. Язык программирования C# 2010 и платформа .NET 4.0, 5-е изд.: Пер. с англ. –М.: ООО “И.Д. Вильямс”, 2011.

# **ПРИЛОЖЕНИЕ A**

**Код программы С#**

1. Group

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Variant\_11.ViewModel;

namespace Variant\_11.Model

{

class Group

{

public int ID { get; set; }

public int SpecialityID { get; set; }

public int QualificationID { get; set; }

public int FormEducationID { get; set; }

public string Faculty { get; set; }

public string Name { get; set; }

public int Course { get; set; }

public int CountStudent { get; set; }

public int CountSubGroup { get; set; }

public Group() { }

public Group(int id, int specialityID, int qualificationID, int formEducationID, string faculty, string name, int course, int countStudent, int countSubGroup)

{

this.ID = id;

this.SpecialityID = specialityID;

this.QualificationID = qualificationID;

this.FormEducationID = formEducationID;

this.Faculty = faculty;

this.Name = name;

this.Course = course;

this.CountStudent = countStudent;

this.CountSubGroup = countSubGroup;

}

public Group CopyFromGroupDPO(GroupDPO groupdpo)

{

FormEducationViewModel vmform = new FormEducationViewModel();

int Formid = 0;

foreach (var it in vmform.ListFormEducation)

{

if (it.NameForm == groupdpo.FormEducationID)

{

Formid = it.ID;

break;

}

}

QualificationViewModel vmQual = new QualificationViewModel();

int qualid = 0;

foreach (var qq in vmQual.ListQualification)

{

if (qq.NameQualification == groupdpo.QualificationID)

{

qualid = qq.ID;

break;

}

}

SpecialityViewModel vmspec = new SpecialityViewModel();

int specid = 0;

foreach (var ww in vmspec.ListSpeciality)

{

if (ww.NameSpeciality == groupdpo.SpecialityID)

{

specid = ww.ID;

break;

}

}

if ((Formid != 0) && (qualid != 0) && (specid != 0))

{

this.ID = groupdpo.ID;

this.FormEducationID = Formid;

this.QualificationID = qualid;

this.SpecialityID = specid;

this.Faculty = groupdpo.Faculty;

this.Name = groupdpo.Name;

this.Course = groupdpo.Course;

this.CountStudent = groupdpo.CountStudent;

this.CountSubGroup = groupdpo.CountSubGroup;

}

return this;

}

}

}

1. Speciality

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Variant\_11.Model

{

class Speciality

{

public int ID { get; set; }

public string NameSpeciality { get; set; }

public string Profile { get; set; }

public Speciality() { }

public Speciality(int id, string nameSpeciality, string profile)

{

this.ID = id;

this.NameSpeciality = nameSpeciality;

this.Profile = profile;

}

public Speciality ShallowCopy()

{

return (Speciality)this.MemberwiseClone();

}

}

}