СОДЕРЖАНИЕ

[Введение 4](#_Toc3233859)

[1 Экономическая характеристика объекта 5](#_Toc3233860)

[2 Постановка задачи 6](#_Toc3233861)

[2.1 Экономическое содержание задачи 6](#_Toc3233862)

[2.2 Входная информация 6](#_Toc3233863)

[3 Технические и инструментальные средства для решения задачи 7](#_Toc3233864)

[3.1 Обоснование выбора языка программирования 7](#_Toc3233865)

[3.2 Технические требования к компьютеру для решения задачи 9](#_Toc3233866)

[4 Информационное обеспечение задачи 10](#_Toc3233867)

[4.1 Классификаторы, используемые для решения задачи 10](#_Toc3233868)

[5 Программное обеспечение задачи 13](#_Toc3233869)

[5.1 Описание методов 13](#_Toc3233870)

[5.2 Алгоритм решения задачи 15](#_Toc3233871)

[5.3 Описание блок – схемы 15](#_Toc3233872)

[Заключение 19](#_Toc3233873)

[Список использованных источников 20](#_Toc3233874)

[Приложение А 21](#_Toc3233875)

[Приложение Б 22](#_Toc3233876)

# Введение

В рамках распоряжения Правительства Российской Федерации от 17 декабря 2009 года № 1993-р утверждён сводный перечень первоочередных государственных и муниципальных услуг, предоставляемых органами исполнительной власти субъектов Российской Федерации и органами местного самоуправления в электронном виде, а также услуг, предоставляемых в электронном виде учреждениями субъектов Российской Федерации и муниципальными учреждениями согласно приложению № 1 этого распоряжения.

Каждое образовательное учреждение имеет какой — либо оборот информации. В каждом образовательном учреждении есть какое — либо количество работников, которые обязаны работать с этой информацией. Чтобы упростить работу со этой информацией её хранят в цифровом формате. Как раз по этому и необходимы электронные журналы для работы с информацией об оценках, учениках(студентах), группах, дисциплинах и т.д.

Актуальность темы обусловлена тем, что она позволяет без проблем осуществлять некоторые действия над студентами, группами, оценками и дисциплинами.

Цель курсового проекта заключается в создании программного модуля оценки знаний студентов на базе ОС Windows.

В курсовом проекте решаются задачи:

* формирование базы данных;
* просмотра оценок по дисциплинам;
* просмотра оценок у определённых учеников(студентов);
* просмотра оценок по группам;
* изменение групп, студентов, дисциплин и оценок с помощью приложения на ПК.

# 1 Экономическая характеристика объекта

Образовательное учреждение Ростовской области «Волгодонский техникум информационных технологий, бизнеса и дизайна имени В. В. Самарского» основано в 1990 году в г. Волгодонске, как профессионально-техническое училище №105.  
В соответствии с приказом министерства общего и профессионального образования Ростовской области от 17 сентября 2014 года № 335-к «О переименовании учреждения и подтверждении работы в должности» техникум переименован в государственное бюджетное профессиональное образовательное учреждение Ростовской области «Волгодонский техникум информационных технологий, бизнеса и дизайна имени В.В. Самарского».

Информатизация образовательного процесса и управления образовательным учреждением реализуется по следующим направлениям:

* информатизация процесса обучения;
* информатизация профориентационной работы;
* информатизация методической работы;
* информатизация управленческой работы администрации и сотрудников.

Информационно-технологическая инфраструктура техникума соответствует современному уровню и включает в себя локальную сеть, объединяющую персональные компьютеры, копировально-множительную технику, презентационное оборудование.

Для подготовки обучающихся к занятиям с использованием сетевых учебных ресурсов и информационных интернет-ресурсов в техникуме имеется 16 компьютерных классов и читальный зал.

Парк компьютерной техники постоянно развивается и качественно и количественно. В настоящее время общий парк вычислительной техники в техникуме насчитывает 438 единиц IBM-совместимой вычислительной техники, из них стационарных персональных компьютеров – 317 ед., ноутбуков – 121 ед.

# 2 Постановка задачи

## 2.1 Экономическое содержание задачи

Цель курсового проекта – создание программного модуля оценки знаний студентов на базе ОС Windows для образовательных учреждений.

Разработанный модуль позволяет производить различные действия над студентами, оценками, группами и дисциплинами с помощью компьютера.

Программа создана на языке С#, графический интерфейс программы дружественный и понятный, сама программа проста в использовании и не требует специального образования или прохождение курсов.

В программе был использован шаблон Model-View-ViewModel. В шаблоне MVVM есть три основных компонента: модель, представление и модель представления. Каждый из них обслуживает отдельную цель. Кроме понимания обязанностей каждого компонента, важно понимать, как они взаимодействуют друг с другом. На высоком уровне в представлении «известно о модели представления и модели представления» известно о модели, но модель не знает модель представления, а модель представления не знает об этом представлении. Таким образом, модель представления изолирует представление от модели и позволяет модели развиваться независимо от представления.

Ниже приведены преимущества использования шаблона MVVM:

* Если реализована существующая реализация модели, которая инкапсулирует существующую бизнес-логику, она может быть сложной или рискованной для ее изменения. В этом сценарии модель представления выступает в качестве адаптера для классов модели и позволяет избежать внесения значительных изменений в код модели.
* Разработчики могут создавать модульные тесты для модели представления и модели без использования представления. Модульные тесты для модели представления могут выполнять точно те же функциональные возможности, которые используются в представлении.
* Пользовательский интерфейс приложения можно переконструировать, не затрагивая код, при условии, что представление полностью реализовано в XAML. Поэтому Новая версия представления должна работать с существующей моделью представления.
* Разработчики и разработчики могут одновременно работать с компонентами в процессе разработки. Дизайнеры могут сосредоточиться на представлении, тогда как разработчики могут работать над моделью представления и компонентами модели.

## 2.2 Входная информация

Входная информация представляет собой данные об оценках, студентах, дисциплинах и группах, хранящиеся в базе данных.

2.3 Выходная информация

Выходной информацией является уже обновленные данные об оценках, студентах, группах и дисциплинах над которыми были произведены действия доступные в программе.

# 3 Технические и инструментальные средства для решения задачи

## 3.1 Обоснование выбора языка программирования

Данное приложение сделано с помощью языка C# – объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

В качестве среды для разработки приложения была выбрана программа Microsoft Visual Studio, линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight. Существует множество сред разработки, но данная была выбрана в ввиду ее удобного графического интерфейса и средств отладки. Microsoft Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Microsoft Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Главные возможности среды разработки Microsoft Visual Studio:

* включены все «интеллектуальные» возможности по редактированию кода;
* есть возможность визуального просмотра будущего приложения;
* сборка проекта работает быстро;
* удобный конструктор интерфейсов;
* удобное и интуитивно понятное логирование проекта;
* указанные цвета и рисунки, использованные в layout'e отображаются на границе в виде небольших превью, которые легко помогают понять какой конкретно ресурс используется;
* среда разработки является технологиями компании Microsoft;
* при выборе ресурса, его содержимое отображается во всплывающих окнах;
* возможность создания одним кликом новых окон и страниц;
* мониторинг используемой памяти;
* возможность подключать сторонние дополнения(плагины);
* возможность добавления новых инструментов;

## 3.2 Технические требования к компьютеру для решения задачи

Рекомендуемые системные требования к компьютеру на которой будет установлена Visual Studio следующие:

* Операционная система – Microsoft® Windows® 7/8/10 (32- или 64-bit);
* 2 ГБ ОЗУ. рекомендуется 8 ГБ ОЗУ (минимум 2,5 ГБ при выполнении на виртуальной машине) ;
* Место на жестком диске: до 210 ГБ (минимум 800 МБ) свободного места в зависимости от установленных компонентов; обычно для установки требуется от 20 до 50 ГБ свободного места;
* Видеоадаптер с минимальным разрешением 720p (1280 на 720 пикселей); для оптимальной работы Visual Studio рекомендуется разрешение WXGA (1366 на 768 пикселей) или более высокое;

# 4 Информационное обеспечение задачи

## 4.1 Классификаторы, используемые для решения задачи

Классификатор представляет собой набор таблиц (справочников).

Объект – некоторая сущность в цифровом пространстве, обладающая определённым состоянием и поведением в данной предметной области.

Атрибут – элемент, описывающий любую характеристику объекта, важную для данной предметной области. Объектом в данной базе данных будет: информация о группах приведенная в таблице 4.1, студентах показанных на таблице 4.2, оценках на таблице 4.3 и дисциплинах на таблице 4.4.

Таблица 4.1 – Атрибуты группы

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Группа | Идентификационный код  Название  Курс |

Таблица 4.2 – Атрибуты студента

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Студент | Идентификационный код  Ф.И.О Студента  Идентификационный код группы |

Таблица 4.3 – Атрибуты оценки

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Оценка | Идентификационный код  Идентификационный код студента  Идентификационный код дисциплины  Оценка  Дата выставления оценки |

Таблица 4.4 – Атрибуты дисциплины

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Дисциплина | Идентификационный код  Название |

Для выполнения проекта была выбрана СУБД SQL Server. Схема БД представлена на рисунке 4.1.

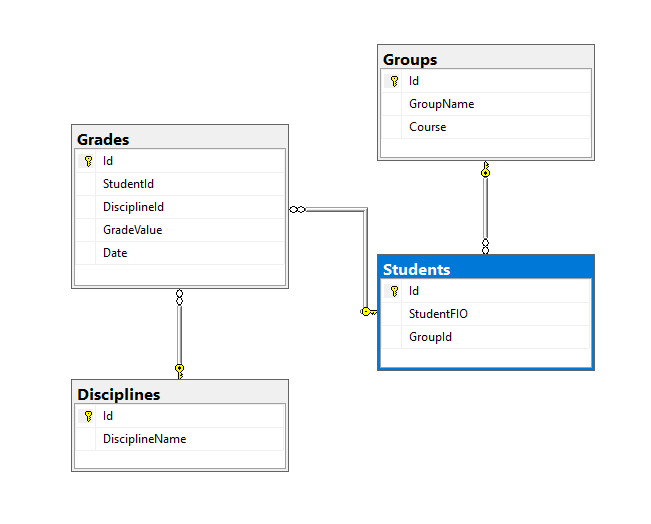


Рисунок 4.1 – Связи таблиц в БД

Структура таблиц группы, студенты, оценки и дисциплины базы данных представлена на рисунках 4.2 – 4.5 соответственно.

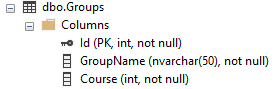


Рисунок 4.2 – Таблица «Группы»

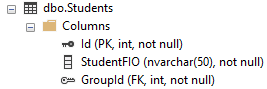


Рисунок 4.3 – Таблица «Студенты»

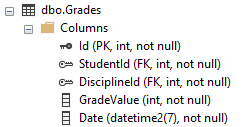


Рисунок 4.4 – Таблица «Оценки»

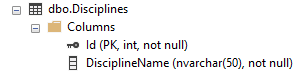


Рисунок 4.5 – Таблица «Дисциплины»

# 5 Программное обеспечение задачи

## 5.1 Описание методов

В примера взята часть кода для входа в приложение под своей учетной записью. При нажатии на кнопку вызывается основное окно, для дальнейшей работы с приложением.

using System.Windows;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Views;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.ViewModels

{

public class LoginViewModel:DataManageVM

{

private RelayCommand checkAuthUser;

public RelayCommand CheckAuthUser

{

get

{

return checkAuthUser ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (Login == null || Login.Replace(" ", "").Length == 0)

{

ShowMessageToUser("Некорректный логин");

}

if (Password == null || Password.Replace(" ", "").Length == 0)

{

ShowMessageToUser("Некорректный пароль");

}

else

{

resultStr = "Неправильный логин или пароль";

if (DataWorker.CheckAuthUser(Login, Password))

{

resultStr = "Успешный вход";

ShowMessageToUser(resultStr);

OpenWorkWindowMethod();

SetNullValuesToProperties();

wnd.Close();

}

else

{

ShowMessageToUser(resultStr);

}

SetNullValuesToProperties();

}

}

);

}

}

private RelayCommand openRegisterWindow;

public RelayCommand OpenRegisterWindow

{

get

{

return openRegisterWindow ?? new RelayCommand(obj =>

{

OpenRegisterWindowMethod();

}

);

}

}

private RelayCommand openWorkWindow;

public RelayCommand OpenWorkWindow

{

get

{

return openWorkWindow ?? new RelayCommand(obj =>

{

OpenWorkWindowMethod();

SetNullValuesToProperties();

}

);

}

}

private void OpenRegisterWindowMethod()

{

RegistrationWindow newRegisterWindow = new RegistrationWindow();

SetCenterPositionAndOpen(newRegisterWindow);

}

protected void OpenWorkWindowMethod()

{

WorkWindow newWorkWindow = new WorkWindow();

SetCenterPositionAndOpen(newWorkWindow);

}

}

## }

## 5.2 Алгоритм решения задачи

Для проведения каких-либо действий с оценками, студентами, группами или дисциплинами необходимо выбрать действие: удалить, редактировать или добавить.

## 5.3 Описание блок – схемы

Последовательное описание блок – схемы приложения приведено ниже:

Шаг 1. Запуск программы.

Шаг 2. Вопрос: Есть ли у вас аккаунт? Если нет, то переходим к шагу 3. Если есть то переходим к шагу 4.

Шаг 3. Регистрируемся. Возвращаемя к авторизации.

Шаг 4. Вход в приложение под своей учетной записью.

Шаг 5. Вопрос: Выбрано ли действие? Если да, то переходим к шагу 6, если нет, то переходим к 8 шагу.

Шаг 6. Выполнение

Шаг 7. Возвращаемся к 4 шагу.

Шаг 8. Закрытие приложения.

Блок-схема приведена в приложении А.

Листинг программы приведен в приложении Б.

5.4 Описание интерфейса программы

В окне входа в приложение, представленном на рисунке 5.1, находятся два текстовых поля логин и пароль, кнопка авторизации и регистрации. Для дальнейшей работы нужно верно указать логин и пароль. Если логин и пароль верны, то откроется основное окно, представленное на рисунке 5.2 с возможностью выбора действий. Кнопки с изображением лупы используются для сортировки данных, а при нажатии на кнопку таблицы откроется окно, с редактирование данных в БД, показанное на рисунке 5.3. Чтобы добавить, удалить или редактировать данные нужно выбрать таблицу и пункт для действия, как на рисунке 5.4.

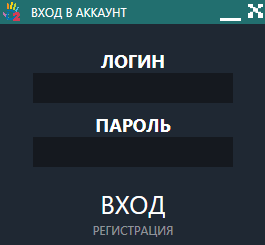


Рисунок 5.1 – Окно входа

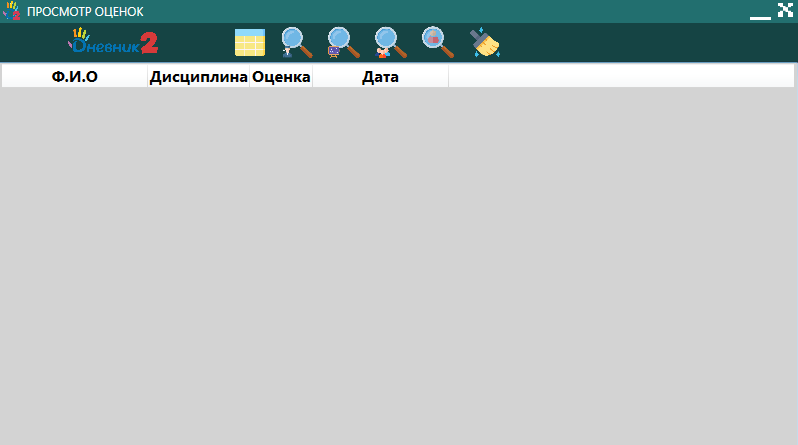


Рисунок 5.2 – Основное окно

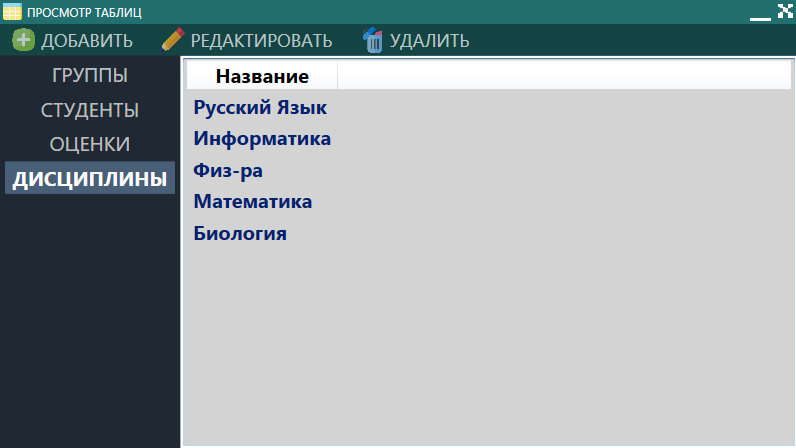


Рисунок 5.3 – Окно редактирования данных

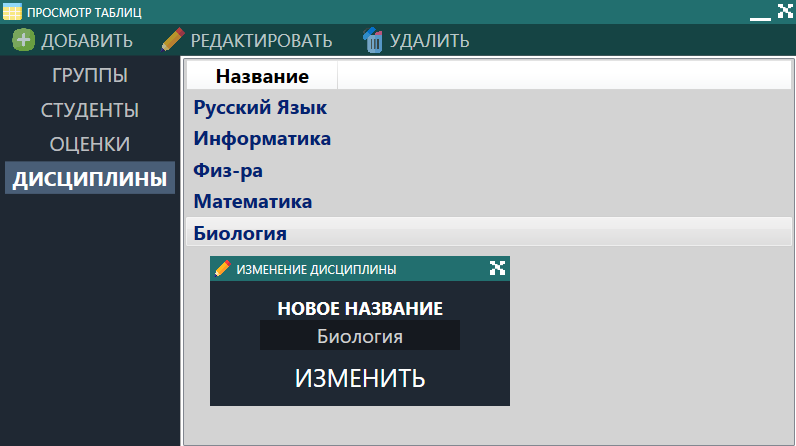


Рисунок 5.4 – Окно действия

Для успешной работы с программой необходимо:

1. ОС Windows.
2. Соеденение с БД.
3. Учетная запись.

# Заключение

Разработанный программный модуль оценки знаний студентов для образовательных учреждений представляет собой простое приложение.

Программа разработана на языке С# в интегрированной среде разработки Microsoft Visual Studio с использованием СУБД SQL Server. Приложение интуитивно понятно даже неопытному пользователю и не затрудняет его работу с оценками, группами, дисциплинами и студентами.

В дальнейшем в приложение будет обновляться и будут добавляться новые функции необходимые для работы с данными.

В процессе написания кода для данной программы были приобретены практические навыки в работе в СУБД SQL Server и языке программирования С#.

Цель курсового проекта достигнута. Задачи выполнены.

# Список использованных источников

Интернет ресурсы:

1. Документация по Entity Framework. [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/ef/
2. Википедия. Язык программирования C# [Электронный ресурс] –Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp
3. Шаблон Model-View-ViewModel. [Электронный ресурс] – https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

# Приложение А

(обязательное)

Блок – схема программы

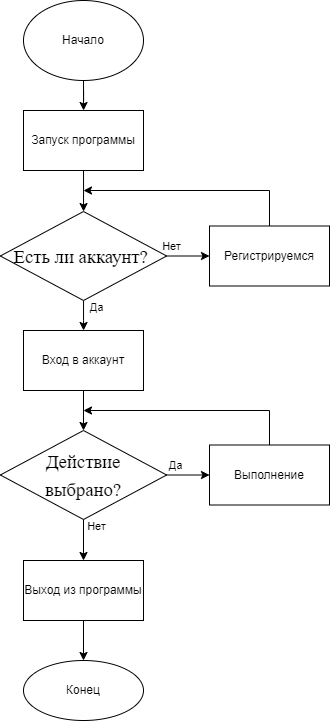


Рисунок А.1–Блок схема программы

# Приложение Б

(обязательное)

Листинг программы

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Views;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.ViewModels

{

public class WorkWindowViewModel:DataManageVM

{

private RelayCommand standartWorkWindowView;

public RelayCommand StandartWorkWindowView

{

get

{

return standartWorkWindowView ?? new RelayCommand(obj =>

{

UpdateGradesWWInfo();

StudentInf = null;

GroupInf = null;

}

);

}

}

private RelayCommand openDeleteEditWindow;

public RelayCommand OpenDeleteEditWindow

{

get

{

return openDeleteEditWindow ?? new RelayCommand(obj =>

{

OpenDeleteEditWindowMethod();

UpdateInfoView();

}

);

}

}

private RelayCommand openSearchStudentWindow;

public RelayCommand OpenSearchStudentWindow

{

get

{

return openSearchStudentWindow ?? new RelayCommand(obj =>

{

OpenSearchStudentWindowMethod();

}

);

}

}

private RelayCommand openSearchDisciplineWindow;

public RelayCommand OpenSearchDisciplineWindow

{

get

{

return openSearchDisciplineWindow ?? new RelayCommand(obj =>

{

OpenSearchDisciplineWindowMethod();

}

);

}

}

private RelayCommand openSearchGroupWindow;

public RelayCommand OpenSearchGroupWindow

{

get

{

return openSearchGroupWindow ?? new RelayCommand(obj =>

{

OpenSearchGroupWindowMethod();

}

);

}

}

private RelayCommand openSearchStudentDisciplineWindow;

public RelayCommand OpenSearchStudentDisciplineWindow

{

get

{

return openSearchStudentDisciplineWindow ?? new RelayCommand(obj =>

{

OpenSearchStudentDisciplineWindowMethod();

}

);

}

}

private void OpenDeleteEditWindowMethod()

{

DeleteEditWindow newDeleteEditWindow = new DeleteEditWindow();

SetCenterPositionAndOpen(newDeleteEditWindow);

}

private void OpenSearchStudentWindowMethod()

{

SearchStudentWindow newSearchStudentWindow = new SearchStudentWindow();

SetCenterPositionAndOpen(newSearchStudentWindow);

}

private void OpenSearchDisciplineWindowMethod()

{

SearchDisciplineWindow newSearchDisciplineWindow = new SearchDisciplineWindow();

SetCenterPositionAndOpen(newSearchDisciplineWindow);

}

private void OpenSearchGroupWindowMethod()

{

SearchGroupWindow newSearchGroupWindow = new SearchGroupWindow();

SetCenterPositionAndOpen(newSearchGroupWindow);

}

private void OpenSearchStudentDisciplineWindowMethod()

{

SearchStudentDisciplineWindow newSearchStudentDisciplineWindow = new SearchStudentDisciplineWindow();

SetCenterPositionAndOpen(newSearchStudentDisciplineWindow);

}

}

}

using System.Windows;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Views;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.ViewModels

{

public class SearchViewModel:DataManageVM

{

private RelayCommand sortGroupWorkWindowView;

public RelayCommand SortGroupWorkWindowView

{

get

{

return sortGroupWorkWindowView ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

if (GroupInf == null)

{

ShowMessageToUser("Выберите группу!");

}

else

{

UpdateGradesWWInfoSortedByGroup();

wnd.Close();

}

}

);

}

}

private RelayCommand sortStudentDisciplineWorkWindowView;

public RelayCommand SortStudentDisciplineWorkWindowView

{

get

{

return sortStudentDisciplineWorkWindowView ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

if (StudentInf == null)

{

ShowMessageToUser("Выберите студента!");

}

if (DisciplineInf == null)

{

ShowMessageToUser("Выберите студента!");

}

else

{

UpdateGradesWWInfoSortedByStudentDiscipline();

wnd.Close();

}

}

);

}

}

private RelayCommand sortDisciplineWorkWindowView;

public RelayCommand SortDisciplineWorkWindowView

{

get

{

return sortDisciplineWorkWindowView ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

if (DisciplineInf == null)

{

ShowMessageToUser("Выберите дисциплину!");

}

else

{

UpdateGradesWWInfoSortedByDiscipline();

wnd.Close();

}

}

);

}

}

private RelayCommand sortStudentWorkWindowView;

public RelayCommand SortStudentWorkWindowView

{

get

{

return sortStudentWorkWindowView ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

if (StudentInf == null)

{

ShowMessageToUser("Выберите студента!");

}

else

{

UpdateGradesWWInfoSortedByStudent();

wnd.Close();

}

}

);

}

}

private RelayCommand sortStudentComboBox;

public RelayCommand SortStudentComboBox

{

get

{

return sortStudentComboBox ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

if (GroupInf == null)

{

ShowMessageToUser("Выберите группу!");

}

else

{

UpdateStudentComboBoxInfo();

}

}

);

}

}

private RelayCommand sortStudentDisciplineComboBox;

public RelayCommand SortStudentDisciplineComboBox

{

get

{

return sortStudentDisciplineComboBox ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

if (GroupInf == null)

{

ShowMessageToUser("Выберите группу!");

}

else

{

UpdateStudentDisciplineComboBoxInfo();

}

}

);

}

}

private void UpdateGradesWWInfoSortedByGroup()

{

AllGradesByGroupId = DataWorker.GetAllGradesByGroupId(GroupInf.Id);

WorkWindow.AllGradeInfoListView.ItemsSource = null;

WorkWindow.AllGradeInfoListView.Items.Clear();

WorkWindow.AllGradeInfoListView.ItemsSource = AllGradesByGroupId;

WorkWindow.AllGradeInfoListView.Items.Refresh();

}

private void UpdateGradesWWInfoSortedByStudentDiscipline()

{

AllGradesByStudentDisciplineId = DataWorker.GetAllGradesByStudentDisciplineId(StudentInf.Id,DisciplineInf.Id);

WorkWindow.AllGradeInfoListView.ItemsSource = null;

WorkWindow.AllGradeInfoListView.Items.Clear();

WorkWindow.AllGradeInfoListView.ItemsSource = AllGradesByStudentDisciplineId;

WorkWindow.AllGradeInfoListView.Items.Refresh();

}

private void UpdateGradesWWInfoSortedByDiscipline()

{

AllGradesByDisciplineId = DataWorker.GetAllGradesByDisciplineId(DisciplineInf.Id);

WorkWindow.AllGradeInfoListView.ItemsSource = null;

WorkWindow.AllGradeInfoListView.Items.Clear();

WorkWindow.AllGradeInfoListView.ItemsSource = AllGradesByDisciplineId;

WorkWindow.AllGradeInfoListView.Items.Refresh();

}

private void UpdateGradesWWInfoSortedByStudent()

{

AllGradesByStudentId = DataWorker.GetAllGradesByStudentId(StudentInf.Id);

WorkWindow.AllGradeInfoListView.ItemsSource = null;

WorkWindow.AllGradeInfoListView.Items.Clear();

WorkWindow.AllGradeInfoListView.ItemsSource = AllGradesByStudentId;

WorkWindow.AllGradeInfoListView.Items.Refresh();

}

private void UpdateStudentComboBoxInfo()

{

AllStudentsByGroupId = DataWorker.GetAllStudentsByGroupId(GroupInf.Id);

SearchStudentWindow.SortStudentComboBox.ItemsSource = null;

SearchStudentWindow.SortStudentComboBox.Items.Clear();

SearchStudentWindow.SortStudentComboBox.ItemsSource = AllStudentsByGroupId;

SearchStudentWindow.SortStudentComboBox.Items.Refresh();

}

private void UpdateStudentDisciplineComboBoxInfo()

{

AllStudentsByGroupId = DataWorker.GetAllStudentsByGroupId(GroupInf.Id);

SearchStudentDisciplineWindow.SortStudentComboBox.ItemsSource = null;

SearchStudentDisciplineWindow.SortStudentComboBox.Items.Clear();

SearchStudentDisciplineWindow.SortStudentComboBox.ItemsSource = AllStudentsByGroupId;

SearchStudentDisciplineWindow.SortStudentComboBox.Items.Refresh();

}

}

}

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Views;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.ViewModels

{

public class DataManageVM : INotifyPropertyChanged

{

//все Студенты

private List<Student> allStudents = DataWorker.GetAllStudents();

public List<Student> AllStudents

{

get { return allStudents; }

set

{

allStudents = value;

NotifyPropertyChanged("AllStudents");

}

}

//все Дисциплины

private List<Discipline> allDisciplines = DataWorker.GetAllDisciplines();

public List<Discipline> AllDisciplines

{

get { return allDisciplines; }

set

{

allDisciplines = value;

NotifyPropertyChanged("AllDisciplines");

}

}

//все Группы

private List<Group> allGroups = DataWorker.GetAllGroups();

public List<Group> AllGroups

{

get { return allGroups; }

set

{

allGroups = value;

NotifyPropertyChanged("AllGroups");

}

}

//все Оценки

private List<Grade> allGrades = DataWorker.GetAllGrades();

public List<Grade> AllGrades

{

get { return allGrades; }

set

{

allGrades = value;

NotifyPropertyChanged("AllGrades");

}

}

//все Оценки по студенту

private List<Grade> allGradesByStudentId;

public List<Grade> AllGradesByStudentId

{

get { return allGradesByStudentId; }

set

{

allGradesByStudentId = value;

NotifyPropertyChanged("AllGradesByStudentId");

}

}

//все Студенты по группе

private List<Student> allStudentsByGroupId;

public List<Student> AllStudentsByGroupId

{

get { return allStudentsByGroupId; }

set

{

allStudentsByGroupId = value;

NotifyPropertyChanged("AllStudentsByGroupId");

}

}

//все Оценки по дисциплине

private List<Grade> allGradesByDisciplineId;

public List<Grade> AllGradesByDisciplineId

{

get { return allGradesByDisciplineId; }

set

{

allGradesByDisciplineId = value;

NotifyPropertyChanged("AllGradesByDisciplineId");

}

}

//все Оценки по группе

private List<Grade> allGradesByGroupId;

public List<Grade> AllGradesByGroupId

{

get { return allGradesByGroupId; }

set

{

allGradesByGroupId = value;

NotifyPropertyChanged("AllGradesByGroupId");

}

}

//все Оценки по студенту и дисциплине

private List<Grade> allGradesByStudentDisciplineId;

public List<Grade> AllGradesByStudentDisciplineId

{

get { return allGradesByStudentDisciplineId; }

set

{

allGradesByStudentDisciplineId = value;

NotifyPropertyChanged("AllGradesByStudentDisciplineId");

}

}

//свойства для групп

public static Group GroupInf { get; set; }

public static string GroupName { get; set; }

public static int Course { get; set; }

//свойства для дисциплин

public static Discipline DisciplineInf { get; set; }

public static string DisciplineName { get; set; }

//свойства для пользователя

public static string Login { get; set; }

public static string Password { get; set; }

//свойства для Студента

public static Student StudentInf { get; set; }

public static string StudentFIO { get; set; }

public static Group StudentGroup { get; set; }

//свойства для Оценки

public static Student GradeStudent { get; set; }

public static Discipline GradeDiscipline { get; set; }

public static int GradeValue { get; set; }

public static DateTime Date { get; set; }

//свойства для выделенных элементов

public TabItem SelectedTabItem { get; set; }

public static Student SelectedStudent { get; set; }

public static Group SelectedGroup { get; set; }

public static Grade SelectedGrade { get; set; }

public static Discipline SelectedDiscipline { get; set; }

protected void SetCenterPositionAndOpen(Window window)

{

window.WindowStartupLocation = WindowStartupLocation.CenterScreen;

window.ShowDialog();

}

public static void SetNullValuesToProperties()

{

//для Студента

StudentFIO = null;

StudentGroup = null;

//для Оценки

GradeStudent = null;

GradeDiscipline = null;

GradeValue = 0;

Date = DateTime.Today;

//для Группы

GroupName = null;

Course = 0;

//для Пользоваетля

Login = null;

Password = null;

}

protected void UpdateInfoView()

{

UpdateWWInfoView();

UpdateDisciplinesInfo();

UpdateGroupsInfo();

UpdateStudentsInfo();

UpdateGradesEDWInfo();

}

protected void UpdateWWInfoView()

{

UpdateGradesWWInfo();

}

protected void UpdateGradesWWInfo()

{

AllGrades = DataWorker.GetAllGrades();

WorkWindow.AllGradeInfoListView.ItemsSource = null;

WorkWindow.AllGradeInfoListView.Items.Clear();

WorkWindow.AllGradeInfoListView.ItemsSource = AllGrades;

WorkWindow.AllGradeInfoListView.Items.Refresh();

}

protected void UpdateGradesEDWInfo()

{

AllGrades = DataWorker.GetAllGrades();

DeleteEditWindow.AllGradeInfoListView.ItemsSource = null;

DeleteEditWindow.AllGradeInfoListView.Items.Clear();

DeleteEditWindow.AllGradeInfoListView.ItemsSource = AllGrades;

DeleteEditWindow.AllGradeInfoListView.Items.Refresh();

}

protected void UpdateGroupsInfo()

{

AllGroups = DataWorker.GetAllGroups();

DeleteEditWindow.AllGroupInfoListView.ItemsSource = null;

DeleteEditWindow.AllGroupInfoListView.Items.Clear();

DeleteEditWindow.AllGroupInfoListView.ItemsSource = AllGroups;

DeleteEditWindow.AllGroupInfoListView.Items.Refresh();

}

protected void UpdateStudentsInfo()

{

AllStudents = DataWorker.GetAllStudents();

DeleteEditWindow.AllStudentInfoListView.ItemsSource = null;

DeleteEditWindow.AllStudentInfoListView.Items.Clear();

DeleteEditWindow.AllStudentInfoListView.ItemsSource = AllStudents;

DeleteEditWindow.AllStudentInfoListView.Items.Refresh();

}

protected void UpdateDisciplinesInfo()

{

AllDisciplines = DataWorker.GetAllDisciplines();

DeleteEditWindow.AllDisciplineInfoListView.ItemsSource = null;

DeleteEditWindow.AllDisciplineInfoListView.Items.Clear();

DeleteEditWindow.AllDisciplineInfoListView.ItemsSource = AllDisciplines;

DeleteEditWindow.AllDisciplineInfoListView.Items.Refresh();

}

public static void ShowMessageToUser(string message)

{

MessageWindow newMessageWindow = new MessageWindow(message);

newMessageWindow.WindowStartupLocation = WindowStartupLocation.CenterScreen;

newMessageWindow.ShowDialog();

}

public event PropertyChangedEventHandler PropertyChanged;

private void NotifyPropertyChanged(String propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

}

}

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Views;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.ViewModels

{

public class DeleteEditViewModel:DataManageVM

{

private RelayCommand deleteItem;

public RelayCommand DeleteItem

{

get

{

return deleteItem ?? new RelayCommand(obj =>

{

string resultStr = "Ничего не выбрано";

//если группа

if (SelectedTabItem.Name == "GroupsTab" && SelectedGroup != null)

{

resultStr = DataWorker.DeleteGroup(SelectedGroup);

UpdateInfoView();

}

//если студент

if (SelectedTabItem.Name == "StudentsTab" && SelectedStudent != null)

{

resultStr = DataWorker.DeleteStudent(SelectedStudent);

UpdateInfoView();

}

//если дисциплина

if (SelectedTabItem.Name == "DisciplinesTab" && SelectedDiscipline != null)

{

resultStr = DataWorker.DeleteDiscipline(SelectedDiscipline);

UpdateInfoView();

}

//если отдел

if (SelectedTabItem.Name == "GradesTab" && SelectedGrade != null)

{

resultStr = DataWorker.DeleteGrade(SelectedGrade);

UpdateInfoView();

}

//обновление

SetNullValuesToProperties();

ShowMessageToUser(resultStr);

}

);

}

}

private RelayCommand openEditItemWnd;

public RelayCommand OpenEditItemWnd

{

get

{

return openEditItemWnd ?? new RelayCommand(obj =>

{

string resultStr = "Ничего не выбрано";

//если группа

if (SelectedTabItem.Name == "GroupsTab" && SelectedGroup != null)

{

OpenEditGroupWindowMethod(SelectedGroup);

}

//если оценка

if (SelectedTabItem.Name == "GradesTab" && SelectedGrade != null)

{

OpenEditGradeWindowMethod(SelectedGrade);

}

//если дисциплина

if (SelectedTabItem.Name == "DisciplinesTab" && SelectedDiscipline != null)

{

OpenEditDisciplineWindowMethod(SelectedDiscipline);

}

//если студент

if (SelectedTabItem.Name == "StudentsTab" && SelectedStudent != null)

{

OpenEditStudentWindowMethod(SelectedStudent);

}

}

);

}

}

private RelayCommand openAddItemWnd;

public RelayCommand OpenAddItemWnd

{

get

{

return openAddItemWnd ?? new RelayCommand(obj =>

{

string resultStr = "Ничего не выбрано";

//если группа

if (SelectedTabItem.Name == "GroupsTab")

{

OpenAddGroupWindowMethod();

}

//если оценка

if (SelectedTabItem.Name == "GradesTab")

{

OpenAddGradeWindowMethod();

}

//если дисциплина

if (SelectedTabItem.Name == "DisciplinesTab")

{

OpenAddDisciplineWindowMethod();

}

//если студент

if (SelectedTabItem.Name == "StudentsTab")

{

OpenAddStudentWindowMethod();

}

}

);

}

}

//методы открытия окон

private void OpenEditGroupWindowMethod(Group group)

{

EditGroupWindow newEditGroupWindow = new EditGroupWindow(group);

SetCenterPositionAndOpen(newEditGroupWindow);

}

private void OpenEditGradeWindowMethod(Grade grade)

{

EditGradeWindow newEditGradeWindow = new EditGradeWindow(grade);

SetCenterPositionAndOpen(newEditGradeWindow);

}

private void OpenEditStudentWindowMethod(Student student)

{

EditStudentWindow newEditStudentWindow = new EditStudentWindow(student);

SetCenterPositionAndOpen(newEditStudentWindow);

}

private void OpenEditDisciplineWindowMethod(Discipline discipline)

{

EditDisciplineWindow newEditDisciplineWindow = new EditDisciplineWindow(discipline);

SetCenterPositionAndOpen(newEditDisciplineWindow);

}

//окно добавления

private void OpenAddGradeWindowMethod()

{

AddGradeWindow newAddGradeWindow = new AddGradeWindow();

SetCenterPositionAndOpen(newAddGradeWindow);

}

private void OpenAddDisciplineWindowMethod()

{

AddDisciplineWindow newAddDisciplineWindow = new AddDisciplineWindow();

SetCenterPositionAndOpen(newAddDisciplineWindow);

}

private void OpenAddStudentWindowMethod()

{

AddStudentWindow newAddStudentWindow = new AddStudentWindow();

SetCenterPositionAndOpen(newAddStudentWindow);

}

private void OpenAddGroupWindowMethod()

{

AddGroupWindow newAddGroupWindow = new AddGroupWindow();

SetCenterPositionAndOpen(newAddGroupWindow);

}

}

}

using System.Windows;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Views;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.ViewModels

{

public class AddDisciplineViewModel:DataManageVM

{

private RelayCommand addNewDiscipline;

public RelayCommand AddNewDiscipline

{

get

{

return addNewDiscipline ?? new RelayCommand(obj =>

{

Window wnd = obj as Window;

string resultStr = "";

if (DisciplineName == null || DisciplineName.Replace(" ", "").Length == 0)

{

DataManageVM.ShowMessageToUser("Некорректное название");

}

else

{

resultStr = DataWorker.CreateDiscipline(DisciplineName);

UpdateInfoView();

ShowMessageToUser(resultStr);

SetNullValuesToProperties();

wnd.Close();

}

}

);

}

}

}

}

using System.Windows;

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.ViewModels

{

public class EditDisciplineViewModel:DataManageVM

{

private RelayCommand editDiscipline;

public RelayCommand EditDiscipline

{

get

{

return editDiscipline ?? new RelayCommand(obj =>

{

Window window = obj as Window;

string resultStr = "Не выбрана дисциплина";

if (SelectedDiscipline != null)

{

resultStr = DataWorker.EditDiscipline(SelectedDiscipline, DisciplineName);

UpdateInfoView();

SetNullValuesToProperties();

ShowMessageToUser(resultStr);

window.Close();

}

else ShowMessageToUser(resultStr);

}

);

}

}

}

}

using WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models.Data;

using System.Collections.Generic;

using System.Linq;

using System;

namespace WPF\_EF\_MVVM\_SA\_Proj.Resources.MVVM.Models

{

public static class DataWorker

{

//получить всех студентов

public static List<Student> GetAllStudents()

{

using (ApplicationContext db = new ApplicationContext())

{

var result = db.Students.ToList();

return result;

}

}

//получить все оценки

public static List<Grade> GetAllGrades()

{

using (ApplicationContext db = new ApplicationContext())

{

var result = db.Grades.ToList();

return result;

}

}

//получить все Дисциплины

public static List<Discipline> GetAllDisciplines()

{

using (ApplicationContext db = new ApplicationContext())

{

var result = db.Disciplines.ToList();

return result;

}

}

//получить все группы

public static List<Group> GetAllGroups()

{

using (ApplicationContext db = new ApplicationContext())

{

var result = db.Groups.ToList();

return result;

}

}

//создать Студента

public static string CreateStudent(string firstName,Group group)

{

string result = "Уже существует";

using (ApplicationContext db = new ApplicationContext())

{

//проверяем сущесвует ли студент

bool checkIsExist = db.Students.Any(el => el.StudentFIO == firstName);

if (!checkIsExist)

{

Student newStudent = new Student

{

StudentFIO = firstName,

GroupId=group.Id

};

db.Students.Add(newStudent);

db.SaveChanges();

result = "Студент добавлен!";

}

return result;

}

}

//создать Группу

public static string CreateGroup(string groupName,int course)

{

string result = "Уже существует";

using (ApplicationContext db = new ApplicationContext())

{

//проверяем сущесвует ли группа

bool checkIsExist = db.Groups.Any(el => el.GroupName == groupName);

if (!checkIsExist)

{

Group newGroup = new Group

{

GroupName = groupName,

Course = course

};

db.Groups.Add(newGroup);

db.SaveChanges();

result = "Группа добавлена!";

}

return result;

}

}

//содать Оценку

public static string CreateGrade(int grade,DateTime date, Discipline discipline,Student student)

{

string result = "Уже существует";

using (ApplicationContext db = new ApplicationContext())

{

Grade newGrade = new Grade

{

StudentId = student.Id,

DisciplineId = discipline.Id,

GradeValue = grade,

Date = date

};

db.Grades.Add(newGrade);

db.SaveChanges();

result = "Оценка добавлена!";

return result;

}

}

//создать Дисчиплину

public static string CreateDiscipline(string disciplineName)

{

string result = "Уже существует";

using (ApplicationContext db = new ApplicationContext())

{

//check the user is exist

bool checkIsExist = db.Disciplines.Any(el => el.DisciplineName==disciplineName);

if (!checkIsExist)

{

Discipline newDiscipline = new Discipline

{

DisciplineName = disciplineName,

};

db.Disciplines.Add(newDiscipline);

db.SaveChanges();

result = "Сделано!";

}

return result;

}

}

//создать пользователя

public static bool CheckAuthUser(string login,string password)

{

using (ApplicationContext db = new ApplicationContext())

{

return db.Registers.Any(el => el.Password == password && el.Login == login);

}

}

public static string CreateUser(string login,string password)

{

string result = "Уже существует";

using (ApplicationContext db = new ApplicationContext())

{

//check the user is exist

bool checkIsExist = db.Registers.Any(el => el.Password == password && el.Login==login );

if (!checkIsExist)

{

Register newUser = new Register

{

Login = login,

Password = password,

};

db.Registers.Add(newUser);

db.SaveChanges();

result = "Пользователь добавлен!";

}

return result;

}

}

//удаление Оценку

public static string DeleteGrade(Grade grade)

{

string result = "Такой оценки не существует";

using (ApplicationContext db = new ApplicationContext())

{

db.Grades.Remove(grade);

db.SaveChanges();

result = $"Сделано! Оценка {grade.GradeValue} студента {GetStudentByGradeId((int)grade.StudentId)} удалена!";

}

return result;

}

//удаление Студента

public static string DeleteStudent(Student student)

{

string result = "Такого студента не существует";

using (ApplicationContext db = new ApplicationContext())

{

db.Students.Remove(student);

db.SaveChanges();

result = $"Сделано! Студент {student.StudentFIO} из группы {GetGroupByStudentId((int)student.GroupId)} удален!";

}

return result;

}

//удаление Дисциплины

public static string DeleteDiscipline(Discipline discipline)

{

string result = "Такой дисциплины не существует";

using (ApplicationContext db = new ApplicationContext())

{

//check position is exist

db.Disciplines.Remove(discipline);

db.SaveChanges();

result = $"Сделано! Дисциплина {discipline.DisciplineName} удалена!";

}

return result;

}

//удаление Группы

public static string DeleteGroup(Group group)

{

string result = "Такой группы не существует";

using (ApplicationContext db = new ApplicationContext())

{

//check position is exist

db.Groups.Remove(group);

db.SaveChanges();

result = $"Сделано! Группа {group.GroupName} удалена!";

}

return result;

}

//редактирование Группы

public static string EditGroup(Group oldGroup, string newName, int newCourse)

{

string result = "Такой группы не существует";

using (ApplicationContext db = new ApplicationContext())

{

Group group = db.Groups.FirstOrDefault(group => group.Id == oldGroup.Id);

group.GroupName = newName;

group.Course = newCourse;

db.SaveChanges();

result = $"Сделано! Группа {group.GroupName} изменена!";

}

return result;

}

//редактирование Студента

public static string EditStudent(Student oldStudent, string newFIO,Group newGroup)

{

string result = "Такого студента не существует";

using (ApplicationContext db = new ApplicationContext())

{

Student student = db.Students.FirstOrDefault(student => student.Id == oldStudent.Id);

student.StudentFIO = newFIO;

student.GroupId = newGroup.Id;

db.SaveChanges();

result = $"Сделано! Студент {student.StudentFIO} из группы {GetGroupByStudentId((int)student.GroupId)} изменен!";

}

return result;

}

//редактирование Дисиплины

public static string EditDiscipline(Discipline oldDiscipline, string newDisciplineName)

{

string result = "Такой дисциплины не существует";

using (ApplicationContext db = new ApplicationContext())

{

Discipline discipline = db.Disciplines.FirstOrDefault(discipline => discipline.Id == oldDiscipline.Id);

discipline.DisciplineName = newDisciplineName;

db.SaveChanges();

result = $"Сделано! Дисциплина {discipline.DisciplineName} изменена!";

}

return result;

}

//редактирование Оценки

public static string EditGrade(Grade oldGrade, Student newStudent, Discipline newDiscipline, int newGradeValue,DateTime newDate)

{

string result = "Такой оценки не существует";

using (ApplicationContext db = new ApplicationContext())

{

//check user is exist

Grade grade = db.Grades.FirstOrDefault(grade => grade.Id == oldGrade.Id);

if (grade != null)

{

grade.StudentId = newStudent.Id;

grade.DisciplineId = newDiscipline.Id;

grade.GradeValue = newGradeValue;

grade.Date = newDate;

db.SaveChanges();

result = $"Сделано! Оценка {grade.GradeValue} студента {GetStudentByGradeId((int)grade.StudentId)} изменена!";

}

}

return result;

}

//получение студента по id студента

public static Student GetStudentById(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

Student student = db.Students.FirstOrDefault(st => st.Id == id);

return student;

}

}

//получение дисциплины по id дисциплины

public static Discipline GetDisciplineById(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

Discipline discipline = db.Disciplines.FirstOrDefault(dis => dis.Id == id);

return discipline;

}

}

//получение оценки по id оценки

public static Grade GetGradeById(int id)

{

using(ApplicationContext db = new ApplicationContext())

{

Grade grade = db.Grades.FirstOrDefault(grd => grd.Id == id);

return grade;

}

}

//получение группы по id группы

public static Group GetGroupById(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

Group group = db.Groups.FirstOrDefault(grp => grp.Id == id);

return group;

}

}

//получение всех студентов по id группы

public static List<Student> GetAllStudentsByGroupId(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

List<Student> students = (from student in GetAllStudents() where student.GroupId == id select student).ToList();

return students;

}

}

//получение всех оценок по id дисциплины

public static List<Grade> GetAllGradesByDisciplineId(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

List<Grade> grades = (from grade in GetAllGrades() where grade.DisciplineId == id select grade).ToList();

return grades;

}

}

//получение всех оценок по id студента

public static List<Grade> GetAllGradesByGroupId(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

List<Grade> grades = (from grade in GetAllGrades() where grade.GradeStudent.GroupId == id select grade).ToList();

return grades;

}

}

//получение всех оценок по id студента и дсициплины

public static List<Grade> GetAllGradesByStudentDisciplineId(int studentId, int disciplineId)

{

using (ApplicationContext db = new ApplicationContext())

{

List<Grade> grades = (from grade in GetAllGrades() where grade.StudentId == studentId && grade.DisciplineId == disciplineId select grade).ToList();

return grades;

}

}

//получение всех оценок по id студента

public static List<Grade> GetAllGradesByStudentId(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

List<Grade> grades = (from grade in GetAllGrades() where grade.StudentId == id select grade).ToList();

return grades;

}

}

//получение всех оценок по id студента

public static List<Grade> GetAllGradesByStudentAndDisciplineId(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

List<Grade> grades = (from grade in GetAllGrades() where grade.StudentId == id && grade.DisciplineId == id select grade).ToList();

return grades;

}

}

//получение студента по id оценки

public static string GetStudentByGradeId(int gradeId)

{

using (ApplicationContext db = new ApplicationContext())

{

Student student = db.Students.FirstOrDefault(student => student.Id == gradeId);

return student.StudentFIO;

}

}

//получение группы по id студента

public static string GetGroupByStudentId(int studentGroupId)

{

using (ApplicationContext db = new ApplicationContext())

{

Group group = db.Groups.FirstOrDefault(group => group.Id == studentGroupId);

return group.GroupName;

}

}

}

}