Приложение П

Листинг кода

Листинг кода ИС “StudCheck”

Классы, представляющие классы приложения (из схемы данных):

Activ.cs:

namespace StudentGroups.Class.BaseClass

{

internal class Activ

{

public int id\_stud\_activ { get; set; }

public string hobby { get; set; }

public string hobby\_area { get; set; }

public string freetime\_interest { get; set; }

public string addit\_edu { get; set; }

public string socialLife\_part { get; set; }

}

}

CourseWorkCl.cs:

namespace StudentGroups.Class.BaseClass

{

internal class CourseWorkCl

{

public int id\_coursework { get; set; }

public string work\_theme { get; set; }

public DateTime work\_date { get; set; }

public string word\_grade { get; set; }

public int id\_stud { get; set; }

}

}

DecreesCl.cs:

namespace StudentGroups.Class.BaseClass

{

internal class DecreesCl

{

public int id\_decree { get; set; }

public string decree\_name { get; set; }

public DateTime decree\_date { get; set; }

public string decree\_type { get; set; }

public string decree\_years { get; set; }

public string decree\_endyear { get; set; }

public override string ToString() => this.decree\_name;

}

}

EduProcessCl.cs:

namespace StudentGroups.Class.BaseClass

{

internal class EduProcessCl

{

public int id\_edu\_process { get; set; }

public string edu\_rel { get; set; }

public string edu\_level { get; set; }

public string edu\_debt { get; set; }

public string edu\_interest { get; set; }

}

}

FamilyCl.cs:

using System;

namespace StudentGroups.Class.BaseClass

{

internal class FamilyCl

{

public int id\_family { get; set; }

public int id\_family\_suff { get; set; }

public string family\_status { get; set; }

public string success\_family\_check { get; set; }

public string parents\_check { get; set; }

public string mom\_FIO { get; set; }

public DateTime mom\_date { get; set; }

public string mom\_workplace { get; set; }

public string mom\_workpos { get; set; }

public string mom\_tel { get; set; }

public string dad\_FIO { get; set; }

public DateTime dad\_date { get; set; }

public string dad\_workplace { get; set; }

public string dad\_workpos { get; set; }

public string dad\_tel { get; set; }

public string caretaker\_check { get; set; }

public string caretaker\_FIO { get; set; }

public string caretaker\_degree { get; set; }

public DateTime caretaker\_date { get; set; }

public string caretaker\_workplace { get; set; }

public string caretaker\_workpos { get; set; }

public string caretaker\_tel { get; set; }

public string sis\_bro\_check { get; set; }

public string sis\_bro\_comment { get; set; }

}

}

FamilySuffCl.cs:

namespace StudentGroups.Class.BaseClass

{

internal class FamilySuffCl

{

public int id\_family\_suff { get; set; }

public string family\_suff { get; set; }

public string family\_house { get; set; }

public string family\_house\_condition { get; set; }

public string family\_influence { get; set; }

public string family\_rel { get; set; }

public string family\_help { get; set; }

public string family\_connection { get; set; }

public string college\_inform { get; set; }

}

}

GradCl.cs:

namespace StudentGroups.Class.BaseClass

{

internal class GradeCl

{

public int id\_grade { get; set; }

public string grade\_subject { get; set; }

public string grade\_hours { get; set; }

public string grade\_term { get; set; }

public string grade\_challenge { get; set; }

public DateTime grade\_date { get; set; }

public string grade\_value { get; set; }

}

}

GroupRel.cs:

namespace StudentGroups.Class.BaseClass

{

internal class GroupRelCl

{

public int id\_group\_rel { get; set; }

public string group\_position { get; set; }

public string rel\_opinion { get; set; }

public string socialTask\_opinion { get; set; }

public string group\_influence { get; set; }

public string college\_friends { get; set; }

}

}

GroupStudCl.cs:

namespace StudentGroups.Class.BaseClass

{

internal class GroupStudCl

{

public int groupID { get; set; }

public string groupName { get; set; }

public string groupCourse { get; set; }

public string groupSpec { get; set; }

public string groupFaculty { get; set; }

public override string ToString() => this.groupName;

}

}

ReportsCl.cs:

using System;

namespace StudentGroups.Class.BaseClass

{

internal class ReportsCl

{

public int id\_report { get; set; }

public string report\_name { get; set; }

public DateTime report\_date { get; set; }

public string is\_verified { get; set; }

public override string ToString() => this.report\_name;

}

}

PracticeCl.cs:

using System;

namespace StudentGroups.Class.BaseClass

{

internal class PracticeCl

{

public int id\_practice { get; set; }

public string practice\_name { get; set; }

public string practice\_term { get; set; }

public string practice\_dur { get; set; }

public string practice\_grade { get; set; }

public int id\_stud { get; set; }

public override string ToString() => this.practice\_name;

}

}

StudCard.cs:

using System;

namespace StudentGroups.Class.BaseClass

{

internal class StudCard

{

public int Stud\_id { get; set; }

public string stud\_Name { get; set; }

public string stud\_Family { get; set; }

public string stud\_Otche { get; set; }

public string gender { get; set; }

public DateTime birth\_date { get; set; }

public string stud\_tel { get; set; }

public string birth\_place { get; set; }

public string stud\_email { get; set; }

public int id\_group { get; set; }

public int id\_family { get; set; }

public int id\_studdata { get; set; }

public int id\_stud\_persona { get; set; }

public int id\_group\_rel { get; set; }

public int id\_stud\_activ { get; set; }

public int id\_edu\_process { get; set; }

public string stud\_tutor { get; set; }

public byte[] stud\_photo { get; set; }

public override string ToString()

{

return this.stud\_Name;

}

}

}

StudDataCl.cs:

using System;

namespace StudentGroups.Class.BaseClass

{

internal class StudDataCl

{

public int id\_studdata { get; set; }

public string stud\_building { get; set; }

public string reg\_address { get; set; }

public string fact\_address { get; set; }

public string home\_tel { get; set; }

public string prev\_edu { get; set; }

public string prev\_edu\_endyear { get; set; }

public string passport\_seria { get; set; }

public string passport\_num { get; set; }

public DateTime passport\_date { get; set; }

public string passport\_who { get; set; }

public string SNILS { get; set; }

public string police\_record\_check { get; set; }

public string police\_record\_who { get; set; }

public string police\_record\_reason { get; set; }

public DateTime police\_record\_date { get; set; }

public string internal\_record\_check { get; set; }

public string internal\_record\_reason { get; set; }

public DateTime internal\_record\_date { get; set; }

public DateTime discuss\_date { get; set; }

public DateTime police\_record\_end { get; set; }

public string police\_record\_endreason { get; set; }

}

}

StudPersona.cs:

namespace StudentGroups.Class.BaseClass

{

internal class StudPersonaCl

{

public int id\_stud\_persona { get; set; }

public string stud\_phys { get; set; }

public string stud\_subculture { get; set; }

public string stud\_socialhelp { get; set; }

public string stud\_college\_rel { get; set; }

public string stud\_envir { get; set; }

public string stud\_conflict { get; set; }

public string stud\_role { get; set; }

public string stud\_good\_behav { get; set; }

public string stud\_bad\_behav { get; set; }

public string stud\_crime { get; set; }

public string stud\_crime\_reason { get; set; }

public string stud\_bad\_habit { get; set; }

}

}

TutorStud.cs:

namespace StudentGroups.Class.BaseClass

{

internal class TutorStud

{

public int tutorID { get; set; }

public string tutorFIO { get; set; }

public string tutorTel { get; set; }

public string tutorEmail { get; set; }

public string tutorGroup { get; set; }

public override string ToString() => this.tutorFIO;

}

}

StudList.cs:

using System;

using System.ComponentModel.DataAnnotations;

namespace StudentGroups.Class

{

public class StudList

{

public int Stud\_id { get; set; }

public string stud\_Name { get; set; }

public string stud\_Family { get; set; }

public string stud\_Otche { get; set; }

public string gender { get; set; }

public DateTime birth\_date { get; set; }

public string stud\_group { get; set; }

public string stud\_tel { get; set; }

public string stud\_email { get; set; }

public string stud\_tutor { get; set; }

public byte[] stud\_photo { get; set; }

}

}

PracticeTable.cs:

namespace StudentGroups.Class

{

internal class PracticeTable

{

public int ID\_practice { get; set; }

public int ID\_studdent { get; set; }

public string Stud\_Name { get; set; }

public string Stud\_Family { get; set; }

public string Practice\_name { get; set; }

public string Practice\_term { get; set; }

public string Practice\_dur { get; set; }

public string Practice\_grade { get; set; }

}

}  
GradeTable.cs:

using System;

namespace StudentGroups.Class

{

internal class GradeTable

{

public int id\_grade { get; set; }

public int id\_studdent { get; set; }

public string stud\_family { get; set; }

public string stud\_name { get; set; }

public string grade\_subject { get; set; }

public string grade\_hours { get; set; }

public string grade\_term { get; set; }

public string grade\_challenge { get; set; }

public DateTime grade\_date { get; set; }

public string grade\_value { get; set; }

}

}

GradeClass.cs:

using System;

namespace StudentGroups.Class

{

internal class GradeClass

{

public int Id\_group { get; set; }

public string Group\_Name { get; set; }

public string Tutor\_FIO { get; set; }

public string Stud\_number { get; set; }

public string Average\_grade { get; set; }

}

}

CourseTable.cs:

namespace StudentGroups.Class

{

internal class CourseTable

{

public int id\_coursework { get; set; }

public int id\_studdent { get; set; }

public string stud\_family { get; set; }

public string stud\_name { get; set; }

public string work\_theme { get; set; }

public DateTime work\_date { get; set; }

public string word\_grade { get; set; }

public int id\_stud { get; set; }

}

}

Дополнительные классы:

Author.cs:

using System.ComponentModel.DataAnnotations;

namespace StudentGroups.Class

{

public class Author

{

[Key]

public int ID { get; set; }

public string login { get; set; }

public string user\_pass { get; set; }

public int ID\_role { get; set; }

public Author() { }

public Author(string login, string user\_pass, int ID\_role)

{

this.login = login;

this.user\_pass = user\_pass;

this.ID\_role = ID\_role;

}

}

}

AuthRole.cs:

namespace StudentGroups.Class

{

internal class AuthRole

{

public static int RoleChooser { get; set; }

}

}

DBContext.cs:

using System.Data.Entity;

namespace StudentGroups.Class

{

class AppContext : DbContext

{

public static AppContext \_context;

public AppContext() : base("StudCheckConnectionString")

{

}

public static AppContext GetContext()

{

if (\_context is null) \_context = new AppContext();

return \_context;

}

public DbSet<StudList> Students { get; set; }

public DbSet<Author> Authorize { get; set; }

}

}

MailService.cs:

using MimeKit;

using MailKit.Net.Smtp;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentGroups.Class

{

internal class MailService

{

private readonly string from = "gorun24021999@gmail.com";

private readonly string password = "ehhjdjtuobsbfosb";

public void Send(string to, string subject, string BodyText)

{

MimeMessage message = new MimeMessage();

message.From.Add(new MailboxAddress("Sergey", from));

message.To.Add(new MailboxAddress("Commision", to));

BodyBuilder builder = new BodyBuilder();

builder.TextBody = BodyText;

message.Body = builder.ToMessageBody();

message.Subject = subject;

using (var client = new SmtpClient())

{

client.ServerCertificateValidationCallback = (s, c, h, e) => true;

client.Connect("smtp.gmail.com", 465);

client.Authenticate(from, password);

client.Send(message);

client.Disconnect(true);

}

}

}

}

PageChangeClass.cs

using System.Windows.Controls;

using System.Windows.Input;

using CommunityToolkit.Mvvm.Input;

using StudentGroups.Windows.MainPages;

namespace StudentGroups.Class

{

internal class PageChangeClass : PageClass

{

private Page GlavPage = new GlavPage();

private Page StudPage = new StudPage();

private Page GradePage = new JournalPage();

private Page TutorPage = new TutorPage();

private Page \_glavnayaPage = new GlavPage();

public Page glavnayaPage

{

get => \_glavnayaPage;

set => Set(ref \_glavnayaPage, value);

}

public ICommand OpenStudentPage => new RelayCommand(() => glavnayaPage = StudPage);

public ICommand OpenGlavPage => new RelayCommand(() => glavnayaPage = GlavPage);

public ICommand OpenGradePage => new RelayCommand(() => glavnayaPage = GradePage);

public ICommand OpenTutorPage => new RelayCommand(() => glavnayaPage = TutorPage);

}

}

PageClass.cs:

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace StudentGroups.Class

{

internal abstract class PageClass : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void onPropertyChanged([CallerMemberName] string PropertyName = null)

{

PropertyChanged.Invoke(this, new PropertyChangedEventArgs(PropertyName));

}

protected virtual bool Set<T>(ref T field, T value, [CallerMemberName] string PropertyName = null)

{

if (Equals(field, value)) return false;

field = value;

onPropertyChanged(PropertyName);

return true;

}

}

}

Speciality.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentGroups.Class

{

internal class Speciality

{

public int groupID { get; set; }

public string groupName { get; set; }

public string groupSpec { get; set; }

public string groupFaculty { get; set; }

public override string ToString() => this.groupFaculty;

}

}

StudEditCheck.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentGroups.Class

{

internal class StudEditCheck

{

public static int StudIDCheck { get; set; }

}

}

Окна:

AcadHub.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace StudentGroups.Windows.MainPages.StudentsWind

{

/// <summary>

/// Логика взаимодействия для AddHub.xaml

/// </summary>

public partial class AcadHub : Window

{

public AcadHub()

{

InitializeComponent();

}

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

{

CourseHub a = new CourseHub();

a.Show();

Close();

}

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

{

PracticeHub a = new PracticeHub();

a.Show();

Close();

}

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

{

GradeHub a = new GradeHub();

a.Show();

Close();

}

private void Image\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

}

}

GradeHub. xaml.cs

using StudentGroups.Class.BaseClass;

using StudentGroups.Class;

using StudentGroups.Data;

using System;

using System.Collections.Generic;

using System.Data;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Microsoft.Office.Interop.Excel;

using excel = Microsoft.Office.Interop.Excel;

using TextBox = System.Windows.Controls.TextBox;

using Window = System.Windows.Window;

namespace StudentGroups.Windows.MainPages.StudentsWind

{

/// <summary>

/// Логика взаимодействия для AddAcademy.xaml

/// </summary>

public partial class GradeHub : Window

{

StudCheckDBEntities db = new StudCheckDBEntities();

List<GradeCl> gradeList = new List<GradeCl>();

List<StudCard> stdList = new List<StudCard>();

List<GradeTable> gradeTable = new List<GradeTable>();

public GradeHub()

{

InitializeComponent();

Load();

}

private void Load()

{

GradeDG.ItemsSource = null;

foreach (var q in db.Students.ToList())

{

foreach (var x in db.AcademicPerfomance.ToList())

{

GradeTable tmp = new GradeTable

{

id\_studdent = q.ID\_stud,

stud\_family = q.stud\_family,

stud\_name = q.stud\_name,

id\_grade = x.ID\_acad\_perfom,

grade\_subject = x.Subject,

grade\_date = x.Challenge\_date,

grade\_hours = x.Hours,

grade\_challenge = x.Challenge\_type,

grade\_term = x.Term,

grade\_value = x.Grade

};

gradeTable.Add(tmp);

}

}

GradeDG.ItemsSource = gradeTable;

}

public void ExportToExcel()

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

excel.Visible = true;

excel.Interactive = false;

excel.ScreenUpdating = false;

excel.UserControl = false;

excel.DisplayAlerts = false;

for (int j = 2; j < GradeDG.Columns.Count + 1; j++) //Столбцы

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[1, j + 2];

sheet1.Columns[j].NumberFormat = "@";

myRange.Value2 = GradeDG.Columns[j - 1].Header;

myRange.Font.Name = "Times New Roman";

myRange.Cells.Font.Size = 16;

myRange.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.Style.WrapText = false;

myRange.Columns.ColumnWidth = 200;

}

Microsoft.Office.Interop.Excel.Range myRang2 = sheet1.get\_Range("A1", "C1");

myRang2.Value = "ФИО";

myRang2.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.Merge(Type.Missing);

myRang2.Font.Name = "Times New Roman";

myRang2.Font.Bold = true;

myRang2.Cells.Font.Size = 16;

sheet1.Columns.AutoFit();

sheet1.Rows.AutoFit();

excel.Interactive = true;

excel.ScreenUpdating = true;

excel.UserControl = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

WindowState = WindowState.Minimized;

}

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

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

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

{

ExportToExcel();

}

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

{

AcademyWind.AddGrade a = new AcademyWind.AddGrade();

a.Show();

Close();

}

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

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

private void SearchBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var SearchBox = sender as TextBox;

if (SearchVar.Text == "ID оценки")

{

if (SearchBox.Text != "")

{

var filList = gradeTable.Where(x => x.id\_grade.ToString().Contains(SearchBox.Text)).ToList();

GradeDG.ItemsSource = null;

GradeDG.ItemsSource = filList;

}

else

{

GradeDG.ItemsSource = gradeTable;

}

}

else if (SearchVar.Text == "Фамилия")

{

if (SearchBox.Text != "")

{

var filList = gradeTable.Where(x => x.stud\_family.ToString().Contains(SearchBox.Text)).ToList();

GradeDG.ItemsSource = null;

GradeDG.ItemsSource = filList;

}

else

{

GradeDG.ItemsSource = gradeTable;

}

}

else if (SearchVar.Text == "Предмет")

{

if (SearchBox.Text != "")

{

var filList = gradeTable.Where(x => x.grade\_subject.ToString().Contains(SearchBox.Text)).ToList();

GradeDG.ItemsSource = null;

GradeDG.ItemsSource = filList;

}

else

{

GradeDG.ItemsSource = gradeTable;

}

}

else if (SearchVar.Text == "Часы")

{

if (SearchBox.Text != "")

{

var filList = gradeTable.Where(x => x.grade\_hours.ToString().Contains(SearchBox.Text)).ToList();

GradeDG.ItemsSource = null;

GradeDG.ItemsSource = filList;

}

else

{

GradeDG.ItemsSource = gradeTable;

}

}

else if (SearchVar.Text == "Тип испытания")

{

if (SearchBox.Text != "")

{

var filList = gradeTable.Where(x => x.grade\_challenge.ToString().Contains(SearchBox.Text)).ToList();

GradeDG.ItemsSource = null;

GradeDG.ItemsSource = filList;

}

else

{

GradeDG.ItemsSource = gradeTable;

}

}

else if (SearchVar.Text == "Дата испытания")

{

if (SearchBox.Text != "")

{

var filList = gradeTable.Where(x => x.grade\_date.ToString().Contains(SearchBox.Text)).ToList();

GradeDG.ItemsSource = null;

GradeDG.ItemsSource = filList;

}

else

{

GradeDG.ItemsSource = gradeTable;

}

}

else if (SearchVar.Text == "Семестр")

{

if (SearchBox.Text != "")

{

var filList = gradeTable.Where(x => x.grade\_term.ToString().Contains(SearchBox.Text)).ToList();

GradeDG.ItemsSource = null;

GradeDG.ItemsSource = filList;

}

else

{

GradeDG.ItemsSource = gradeTable;

}

}

}

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

{

var gr = GradeDG.SelectedItem as AcademicPerfomance;

var b = db.AcademicPerfomance.Where(q => q.ID\_acad\_perfom == gr.ID\_acad\_perfom).FirstOrDefault();

MessageBoxResult ad = MessageBox.Show("Вы уверены в удалении оценки?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.AcademicPerfomance.Remove(b);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

}

}

Practicehub.xaml.cs

using Microsoft.Office.Interop.Word;

using StudentGroups.Class;

using StudentGroups.Class.BaseClass;

using StudentGroups.Data;

using StudentGroups.Windows.MainPages.AcademyWind;

using System;

using System.Collections.Generic;

using System.Data;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Microsoft.Office.Interop.Excel;

using excel = Microsoft.Office.Interop.Excel;

using TextBox = System.Windows.Controls.TextBox;

using Window = System.Windows.Window;

namespace StudentGroups.Windows.MainPages.StudentsWind

{

/// <summary>

/// Логика взаимодействия для AddPractice.xaml

/// </summary>

public partial class PracticeHub : Window

{

StudCheckDBEntities db = new StudCheckDBEntities();

List<PracticeCl> practList = new List<PracticeCl>();

List<StudCard> stdList = new List<StudCard>();

List<PracticeTable> practTable = new List<PracticeTable>();

public PracticeHub()

{

InitializeComponent();

Load();

}

private void Load()

{

PracticeDG.ItemsSource = null;

foreach (var q in db.Students.ToList())

{

foreach (var x in db.Practice.ToList())

{

PracticeTable tmp = new PracticeTable

{

ID\_studdent = q.ID\_stud,

Stud\_Family = q.stud\_family,

Stud\_Name = q.stud\_name,

ID\_practice = x.ID\_practice,

Practice\_name = x.Practice\_name,

Practice\_dur = x.Practice\_dur,

Practice\_term = x.Practice\_term,

Practice\_grade = x.Practice\_grade

};

practTable.Add(tmp);

}

}

PracticeDG.ItemsSource = practTable;

}

private void DelCourse()

{

var pract = PracticeDG.SelectedItem as Practice;

var b = db.Practice.Where(q => q.ID\_practice == pract.ID\_practice).FirstOrDefault();

MessageBoxResult ad = MessageBox.Show("Вы уверены в удалении практики?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.Practice.Remove(b);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

public void ExportToExcel()

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

excel.Visible = true;

excel.Interactive = false;

excel.ScreenUpdating = false;

excel.UserControl = false;

excel.DisplayAlerts = false;

for (int j = 2; j < PracticeDG.Columns.Count + 1; j++) //Столбцы

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[1, j + 2];

sheet1.Columns[j].NumberFormat = "@";

myRange.Value2 = PracticeDG.Columns[j - 1].Header;

myRange.Font.Name = "Times New Roman";

myRange.Cells.Font.Size = 16;

myRange.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.Style.WrapText = false;

myRange.Columns.ColumnWidth = 200;

}

Microsoft.Office.Interop.Excel.Range myRang2 = sheet1.get\_Range("A1", "C1");

myRang2.Value = "ФИО";

myRang2.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.Merge(Type.Missing);

myRang2.Font.Name = "Times New Roman";

myRang2.Font.Bold = true;

myRang2.Cells.Font.Size = 16;

sheet1.Columns.AutoFit();

sheet1.Rows.AutoFit();

excel.Interactive = true;

excel.ScreenUpdating = true;

excel.UserControl = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

WindowState = WindowState.Minimized;

}

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

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

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

{

MessageBoxResult ad = MessageBox.Show("Создать отчет по практическим работам?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

ExportToExcel();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

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

{

AddPractice a = new AddPractice();

a.Show();

Close();

}

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

{

DelCourse();

}

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

{

MainWindow main = new MainWindow();

main.Show();

Close();

}

private void SearchBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var SearchBox = sender as TextBox;

if (SearchVar.Text == "ID работы")

{

if (SearchBox.Text != "")

{

var filList = practTable.Where(x => x.ID\_practice.ToString().Contains(SearchBox.Text)).ToList();

PracticeDG.ItemsSource = null;

PracticeDG.ItemsSource = filList;

}

else

{

PracticeDG.ItemsSource = practTable;

}

}

else if (SearchVar.Text == "Наименование практики")

{

if (SearchBox.Text != "")

{

var filList = practTable.Where(x => x.Practice\_name.ToString().Contains(SearchBox.Text)).ToList();

PracticeDG.ItemsSource = null;

PracticeDG.ItemsSource = filList;

}

else

{

PracticeDG.ItemsSource = practTable;

}

}

else if (SearchVar.Text == "Фамилия")

{

if (SearchBox.Text != "")

{

var filList = practTable.Where(x => x.Stud\_Family.ToString().Contains(SearchBox.Text)).ToList();

PracticeDG.ItemsSource = null;

PracticeDG.ItemsSource = filList;

}

else

{

PracticeDG.ItemsSource = practTable;

}

}

else if (SearchVar.Text == "Часы практики")

{

if (SearchBox.Text != "")

{

var filList = practTable.Where(x => x.Practice\_dur.ToString().Contains(SearchBox.Text)).ToList();

PracticeDG.ItemsSource = null;

PracticeDG.ItemsSource = filList;

}

else

{

PracticeDG.ItemsSource = practTable;

}

}

else if (SearchVar.Text == "Семестр")

{

if (SearchBox.Text != "")

{

var filList = practTable.Where(x => x.Practice\_term.ToString().Contains(SearchBox.Text)).ToList();

PracticeDG.ItemsSource = null;

PracticeDG.ItemsSource = filList;

}

else

{

PracticeDG.ItemsSource = practTable;

}

}

}

}

}

CourseHub.xaml.cs

using Microsoft.Office.Interop.Word;

using StudentGroups.Class.BaseClass;

using StudentGroups.Class;

using StudentGroups.Data;

using System;

using System.Collections.Generic;

using System.Data;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Microsoft.Office.Interop.Excel;

using excel = Microsoft.Office.Interop.Excel;

using TextBox = System.Windows.Controls.TextBox;

using Window = System.Windows.Window;

namespace StudentGroups.Windows.MainPages.StudentsWind

{

/// <summary>

/// Логика взаимодействия для AddCourse.xaml

/// </summary>

public partial class CourseHub : Window

{

StudCheckDBEntities db = new StudCheckDBEntities();

List<CourseWork> courseList = new List<CourseWork>();

List<StudCard> stdList = new List<StudCard>();

List<CourseTable> courseTable = new List<CourseTable>();

public CourseHub()

{

InitializeComponent();

Load();

}

private void Load()

{

CourseDG.ItemsSource = null;

foreach (var q in db.Students.ToList())

{

foreach (var x in db.CourseWork.ToList())

{

CourseTable tmp = new CourseTable

{

id\_studdent = q.ID\_stud,

stud\_family = q.stud\_family,

stud\_name = q.stud\_name,

id\_coursework = x.ID\_coursework,

work\_theme = x.Work\_theme,

word\_grade = x.Word\_grade,

work\_date = x.Work\_date

};

courseTable.Add(tmp);

}

}

CourseDG.ItemsSource = courseTable;

}

public void ExportToExcel()

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

excel.Visible = true;

excel.Interactive = false;

excel.ScreenUpdating = false;

excel.UserControl = false;

excel.DisplayAlerts = false;

for (int j = 2; j < CourseDG.Columns.Count + 1; j++) //Столбцы

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[1, j + 2];

sheet1.Columns[j].NumberFormat = "@";

myRange.Value2 = CourseDG.Columns[j - 1].Header;

myRange.Font.Name = "Times New Roman";

myRange.Cells.Font.Size = 16;

myRange.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.Style.WrapText = false;

myRange.Columns.ColumnWidth = 200;

}

Microsoft.Office.Interop.Excel.Range myRang2 = sheet1.get\_Range("A1", "C1");

myRang2.Value = "ФИО";

myRang2.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.Merge(Type.Missing);

myRang2.Font.Name = "Times New Roman";

myRang2.Font.Bold = true;

myRang2.Cells.Font.Size = 16;

sheet1.Columns.AutoFit();

sheet1.Rows.AutoFit();

excel.Interactive = true;

excel.ScreenUpdating = true;

excel.UserControl = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

WindowState = WindowState.Minimized;

}

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

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

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

{

AcademyWind.AddCourse a = new AcademyWind.AddCourse();

a.Show();

Close();

}

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

{

ExportToExcel();

}

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

{

var cour = CourseDG.SelectedItem as CourseWork;

var b = db.CourseWork.Where(q => q.ID\_coursework == cour.ID\_coursework).FirstOrDefault();

MessageBoxResult ad = MessageBox.Show("Вы уверены в удалении работы?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.CourseWork.Remove(b);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

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

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

private void SearchBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var SearchBox = sender as TextBox;

if (SearchVar.Text == "ID работы")

{

if (SearchBox.Text != "")

{

var filList = courseTable.Where(x => x.id\_coursework.ToString().Contains(SearchBox.Text)).ToList();

CourseDG.ItemsSource = null;

CourseDG.ItemsSource = filList;

}

else

{

CourseDG.ItemsSource = courseTable;

}

}

else if (SearchVar.Text == "Наименование")

{

if (SearchBox.Text != "")

{

var filList = courseTable.Where(x => x.work\_theme.ToString().Contains(SearchBox.Text)).ToList();

CourseDG.ItemsSource = null;

CourseDG.ItemsSource = filList;

}

else

{

CourseDG.ItemsSource = courseTable;

}

}

else if (SearchVar.Text == "Фамилия")

{

if (SearchBox.Text != "")

{

var filList = courseTable.Where(x => x.stud\_family.ToString().Contains(SearchBox.Text)).ToList();

CourseDG.ItemsSource = null;

CourseDG.ItemsSource = filList;

}

else

{

CourseDG.ItemsSource = courseTable;

}

}

else if (SearchVar.Text == "Дата сдачи")

{

if (SearchBox.Text != "")

{

var filList = courseTable.Where(x => x.work\_date.ToString().Contains(SearchBox.Text)).ToList();

CourseDG.ItemsSource = null;

CourseDG.ItemsSource = filList;

}

else

{

CourseDG.ItemsSource = courseTable;

}

}

else if (SearchVar.Text == "Оценка")

{

if (SearchBox.Text != "")

{

var filList = courseTable.Where(x => x.word\_grade.ToString().Contains(SearchBox.Text)).ToList();

CourseDG.ItemsSource = null;

CourseDG.ItemsSource = filList;

}

else

{

CourseDG.ItemsSource = courseTable;

}

}

}

}

}

AddGrade.xaml.cs

using System;

using System.Collections.Generic;

using System.Data.Entity.Validation;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using StudentGroups.Class;

using StudentGroups.Class.BaseClass;

using StudentGroups.Data;

using StudentGroups.Windows.MainPages.StudentsWind;

namespace StudentGroups.Windows.MainPages.AcademyWind

{

/// <summary>

/// Логика взаимодействия для AddGrade.xaml

/// </summary>

public partial class AddGrade : Window

{

StudCheckDBEntities db = new StudCheckDBEntities();

List<StudCard> stdCard = new List<StudCard>();

public AddGrade()

{

InitializeComponent();

StudChooseCmb.ItemsSource = null;

StudLoad();

}

private void StudLoad()

{

var Tutorcheck = db.Tutors.Where(s => s.ID\_user == AuthRole.RoleChooser).FirstOrDefault();

foreach (var x in db.Students.Where(q => q.ID\_group == Tutorcheck.ID\_group).ToList())

{

StudCard tmp = new StudCard

{

Stud\_id = x.ID\_stud,

stud\_Name = x.stud\_family + " " + x.stud\_name,

stud\_Family = x.stud\_family

};

stdCard.Add(tmp);

StudChooseCmb.ItemsSource = stdCard;

}

}

private void AddAcad()

{

try

{

AcademicPerfomance acad = new AcademicPerfomance();

acad.Subject = SubjectTB.Text;

acad.Term = TermTB.Text;

acad.Challenge\_type = ChallengeTB.Text;

acad.Challenge\_date = (DateTime)ChallengeDate.SelectedDate;

acad.Grade = GradeTB.Text;

acad.Hours = HoursTB.Text;

var per = stdCard.Where(q => q.stud\_Name == StudChooseCmb.Text).FirstOrDefault().Stud\_id;

acad.Id\_stud = db.Students.Where(q=>q.ID\_stud == per).FirstOrDefault().ID\_stud;

MessageBoxResult ad = MessageBox.Show("Вы уверены в добавлении студента?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.AcademicPerfomance.Add(acad);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

//catch (Exception Ex)

//{

// MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

//}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Ошибка!!! " + validationError.Entry.Entity.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

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

{

AddAcad();

GradeHub a = new GradeHub();

a.Show();

Close();

}

private void Image\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

}

}

AddCourse.xaml.cs

using Org.BouncyCastle.Asn1.Cmp;

using StudentGroups.Class.BaseClass;

using StudentGroups.Class;

using StudentGroups.Data;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using StudentGroups.Windows.MainPages.StudentsWind;

namespace StudentGroups.Windows.MainPages.AcademyWind

{

/// <summary>

/// Логика взаимодействия для AddCourse.xaml

/// </summary>

public partial class AddCourse : Window

{

StudCheckDBEntities db = new StudCheckDBEntities();

List<StudCard> stdCard = new List<StudCard>();

public AddCourse()

{

InitializeComponent();

StudChooseCmb.ItemsSource = null;

StudLoad();

}

private void StudLoad()

{

var Tutorcheck = db.Tutors.Where(s => s.ID\_user == AuthRole.RoleChooser).FirstOrDefault();

foreach (var x in db.Students.Where(q => q.ID\_group == Tutorcheck.ID\_group).ToList())

{

StudCard tmp = new StudCard

{

Stud\_id = x.ID\_stud,

stud\_Name = x.stud\_family + " " + x.stud\_name

};

stdCard.Add(tmp);

}

StudChooseCmb.ItemsSource = stdCard;

}

private void AddCourseWork()

{

try

{

CourseWork work = new CourseWork();

work.Work\_theme = CourseNameTB.Text;

work.Work\_date = (DateTime)CourseDate.SelectedDate;

work.Word\_grade = CourseGradeTB.Text;

work.Id\_stud = (StudChooseCmb.SelectedItem as StudCard).Stud\_id;

MessageBoxResult ad = MessageBox.Show("Вы уверены в добавлении студента?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.CourseWork.Add(work);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

catch (Exception Ex)

{

MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

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

{

AddCourseWork();

CourseHub a = new CourseHub();

a.Show();

Close();

}

private void Image\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

}

}

AddPractice.xaml.cs

using StudentGroups.Data;

using StudentGroups.Class.BaseClass;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Text.RegularExpressions;

using StudentGroups.Windows.MainPages.StudentsWind;

using StudentGroups.Class;

namespace StudentGroups.Windows.MainPages.AcademyWind

{

/// <summary>

/// Логика взаимодействия для AddPractice.xaml

/// </summary>

public partial class AddPractice : Window

{

StudCheckDBEntities db = new StudCheckDBEntities();

List<StudCard> stdCard = new List<StudCard>();

public AddPractice()

{

InitializeComponent();

StudChooseCmb.ItemsSource = null;

StudLoad();

}

private void StudLoad()

{

var Tutorcheck = db.Tutors.Where(s => s.ID\_user == AuthRole.RoleChooser).FirstOrDefault();

foreach (var x in db.Students.Where(q => q.ID\_group == Tutorcheck.ID\_group).ToList())

{

StudCard tmp = new StudCard

{

Stud\_id = x.ID\_stud,

stud\_Name = x.stud\_family + " " + x.stud\_name

};

stdCard.Add(tmp);

StudChooseCmb.ItemsSource = stdCard;

}

}

private void AddCourse()

{

try

{

Practice practice = new Practice();

practice.Practice\_name = PraNameTB.Text;

practice.Practice\_term = TermTB.Text;

practice.Practice\_dur = HoursTB.Text;

practice.Practice\_grade = PraGradeTB.Text;

practice.Id\_stud = (StudChooseCmb.SelectedItem as StudCard).Stud\_id;

MessageBoxResult ad = MessageBox.Show("Вы уверены в добавлении студента?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.Practice.Add(practice);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

catch (Exception Ex)

{

MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void Image\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

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

{

AddCourse();

PracticeHub a = new PracticeHub();

a.Show();

Close();

}

}

}

AddStudBook.xaml.cs

using StudentGroups.Data;

using StudentGroups.Class;

using StudentGroups.Class.BaseClass;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using Microsoft.Win32;

using System.IO;

using Microsoft.Reporting.Map.WebForms.BingMaps;

using System.Data.Entity.Validation;

using TextBox = System.Windows.Controls.TextBox;

namespace StudentGroups.Windows.MainPages.StudentsWind

{

/// <summary>

/// Логика взаимодействия для AddStudBook.xaml

/// </summary>

///

public partial class AddStudBook : Window

{

Data.StudCheckDBEntities db = new StudCheckDBEntities();

List<GroupStudCl> groupList = new List<GroupStudCl>();

List<Speciality> specList = new List<Speciality>();

List<TutorStud> tutorList = new List<TutorStud>();

byte[] Photo\_byte = null;

int checkBase, checkEdu, checkRel, checkRecord, checkFamily, checkActiv, checkData = 0;

public AddStudBook()

{

InitializeComponent();

GroupCmb.ItemsSource = null;

TutorCmb.ItemsSource = null;

ComboAddData();

}

private void ComboAddData()

{

foreach (var s in db.Tutors.ToList())

{

TutorStud tmp2 = new TutorStud()

{

tutorID = s.ID\_tutor,

tutorFIO = s.tutor\_family + " " + s.tutor\_name + " " + s.tutor\_otche

};

tutorList.Add(tmp2);

}

TutorCmb.ItemsSource = tutorList;

foreach (var x in db.Groups.ToList())

{

GroupStudCl tmp = new GroupStudCl

{

groupID = x.ID\_group,

groupName = x.group\_name,

groupCourse = x.course,

groupFaculty = x.faculty

};

groupList.Add(tmp);

Speciality tmp2 = new Speciality

{

groupID = x.ID\_group,

groupName = x.group\_name,

groupFaculty = x.faculty + " " + x.speciality

};

specList.Add(tmp2);

}

GroupCmb.ItemsSource = groupList;

}

private void cmbAdd()

{

int groupChoose = (GroupCmb.SelectedItem as GroupStudCl).groupID;

var spec = specList.Where(q => q.groupID == groupChoose);

SpecialCmb.ItemsSource = spec;

}

//public void TextValidation()

//{

// try

// {

// checkBase = 0;

// if (String.IsNullOrEmpty(SurnameTB.Text) || String.IsNullOrEmpty(NameTB.Text) || String.IsNullOrEmpty(OtcheTB.Text) ||

// String.IsNullOrEmpty(SeriaPassTB.Text) || String.IsNullOrEmpty(NumberPassTB.Text) ||

// String.IsNullOrEmpty(WhoPassTB.Text) || String.IsNullOrEmpty(RegAddres.Text) || String.IsNullOrEmpty(FactAdress.Text)

// || String.IsNullOrEmpty(EmailBox.Text))

// {

// MessageBox.Show("Заполните информацию в вкладке: Основная информация", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (SeriaPassTB.Text.Length != 4)

// {

// MessageBox.Show("В серии паспорта должно быть 4 цифры", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (NumberPassTB.Text.Length != 6)

// {

// MessageBox.Show("В номере паспорта должно быть 6 цифры", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (SNILSTB.Text.Length != 11)

// {

// MessageBox.Show("В номере СНИЛСа должно быть 11 цифр", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (MobPhoneBox.Text.Length != 11)

// {

// MessageBox.Show("В номер телефона должно быть 11 цифр", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (HomePhoneBox.Text.Length != 11)

// {

// MessageBox.Show("В номер телефона должно быть 11 цифр", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else

// {

// }

// checkEdu = 0;

// if (String.IsNullOrEmpty(PrevEduTB.Text)|| BeginDate.SelectedDate.Value == null || String.IsNullOrEmpty(EndDate.Text) || String.IsNullOrEmpty(DecreeName.Text))

// {

// MessageBox.Show("Заполните информацию в вкладке: Учебная информация", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkEdu = 1;

// }

// checkActiv = 0;

// if (String.IsNullOrEmpty(HobbyTB.Text))

// {

// MessageBox.Show("Заполните информацию в вкладке: Активность", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkActiv = 1;

// }

// checkRecord = 0;

// if (PoliceRecCheckBox.IsChecked == false)

// {

// MessageBox.Show("Заполните информацию в вкладке: Постановка на учет", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkFamily = 1;

// }

// }

// catch (Exception Ex)

// {

// MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

// }

//}

private void SaveStudent()

{

try {

//

Students stud = new Students();

stud.stud\_family = SurnameTB.Text;

stud.stud\_name = NameTB.Text;

if (OtcheCheck.IsChecked == true)

{

stud.stud\_otche = "Отсутствует";

}

else

{

stud.stud\_otche = OtcheTB.Text;

}

stud.gender = GenderCmb.Text;

stud.date\_birth = (DateTime)DateBirthTB.SelectedDate;

stud.place\_birth = BirthPlaceTB.Text;

stud.stud\_tel = MobPhoneBox.Text;

stud.stud\_email = EmailBox.Text;

stud.stud\_image = Photo\_byte;

var tutList = tutorList.Where(q => q.tutorFIO == TutorCmb.Text).First().tutorID;

stud.ID\_tutor = tutList;

var grList = groupList.Where(q => q.groupName == GroupCmb.Text).First().groupID;

stud.ID\_group = grList;

StudActivity activ = new StudActivity();

activ.Hobby = HobbyTB.Text;

activ.Hobby\_area = HobbyAreaTB.Text;

activ.Freetime\_interest = InterestTB.Text;

activ.Addit\_edu = AddEduTB.Text;

activ.SocialLife\_part = SocialLifeTB.Text;

activ.Freetime\_interest = FreeTimeTB.Text;

EduProcess edu = new EduProcess();

edu.Edu\_rel = EduRelCmb.Text;

edu.Edu\_level = EduLevelCmb.Text;

edu.Edu\_interest = InterestTB.Text;

edu.Edu\_debt = DebtTB.Text;

StudentData data = new StudentData();

data.stud\_building = StudBuildTB.Text;

data.reg\_address = RegAddres.Text;

data.fact\_address = FactAdress.Text;

data.home\_tel = HomePhoneBox.Text;

data.prev\_edu = PrevEduTB.Text;

data.prev\_edu\_endyear = PrevEndYearTBB.Text;

data.passport\_seria = SeriaPassTB.Text;

data.passport\_number = NumberPassTB.Text;

data.passport\_date = (DateTime)DatePass.SelectedDate;

data.passport\_who = WhoPassTB.Text;

data.SNILS = SNILSTB.Text;

if (PoliceRecCheckBox.IsChecked == false)

{

data.police\_record\_check = "Состоит";

data.police\_record\_reason = RecReasonTB.Text;

data.police\_record\_date = RecDate.SelectedDate;

data.police\_record\_who = PolieRecPlaceTB.Text;

}

else

{

data.police\_record\_check = "Не состоит";

}

if (InternalCheckBox.IsChecked == true)

{

data.internal\_record\_check = "Состоит";

data.internal\_record\_reason = InternalReasTB.Text;

data.internal\_record\_date = InternalDate.SelectedDate;

data.discuss\_date = InternalTalkDate.SelectedDate;

data.police\_record\_endreason = RemoveRecReason.Text;

data.police\_record\_end = RemoveRecDate.SelectedDate;

}

StudGroupRel groupRel = new StudGroupRel();

groupRel.Group\_position = StudPosCmb.Text;

groupRel.Rel\_opinion = OpinionRelCmb.Text;

groupRel.SocialTask\_opinion = SocialTaskCmb.Text;

groupRel.Group\_influence = GroupInflCmb.Text;

groupRel.College\_friends = FriendTB.Text;

StudPersona pers = new StudPersona();

pers.Stud\_phys = PhysCmb.Text;

pers.Stud\_subculture = SubcultureTB.Text;

pers.Stud\_socialhelp = SpecHelpTB.Text;

pers.Stud\_college\_rel = CollRelCmb.Text;

pers.Stud\_envir = EnvirRelTB.Text;

pers.Stud\_conflict = ConflictTB.Text;

pers.Stud\_role = StudRoleCmb.Text;

pers.Stud\_good\_behav = GoodTB.Text;

pers.Stud\_bad\_behav = BadTB.Text;

pers.Stud\_crime = CrimeTB.Text;

pers.Stud\_crime\_reason = ReasonCrimeTB.Text;

pers.Stud\_bad\_habit = BadHabitTB.Text;

Family family = new Family();

family.Family\_status = FamilyStatusCmb.Text;

if (SisBroCheckBox.IsChecked == true)

{

family.Sis\_bro\_check = "Да";

family.Sis\_bro\_comment = SisBroListTB.Text;

}

else

{

family.Sis\_bro\_check = "Нет";

family.Sis\_bro\_comment = "-";

}

string stat = FamilyStatusCmb.Text;

if (stat == "Полная")

{

family.Parents\_check = "Полная";

family.Mom\_FIO = MomSurnameTB.Text + " " + MomNameTB.Text + " " + MomOtcheTB.Text;

family.Mom\_tel = MomPhoneTB.Text;

family.Mom\_date = MomDate.SelectedDate;

family.Mom\_workplace = MomWorkTB.Text;

family.Mom\_workpos = MomWorkPosTB.Text;

family.Dad\_FIO = DadSurnameTB.Text + " " + DadNameTB.Text + " " + DadOtcheTB.Text;

family.Dad\_tel = MomPhoneTB.Text;

family.Dad\_date = MomDate.SelectedDate;

family.Dad\_workplace = MomWorkTB.Text;

family.Dad\_workpos = MomWorkPosTB.Text;

family.Caretaker\_check = "Нет";

family.Caretaker\_date = DateTime.MinValue;

family.Caretaker\_degree = "-";

family.Caretaker\_FIO = "-";

family.Caretaker\_tel = "-";

family.Caretaker\_workplace = "-";

family.Caretaker\_workpos = "-";

}

else if (stat == "Только мать")

{

family.Mom\_FIO = MomSurnameTB.Text + " " + MomNameTB.Text + " " + MomOtcheTB.Text;

family.Mom\_tel = MomPhoneTB.Text;

family.Mom\_date = MomDate.SelectedDate;

family.Mom\_workplace = MomWorkTB.Text;

family.Mom\_workpos = MomWorkPosTB.Text;

family.Caretaker\_check = "Нет";

family.Caretaker\_date = DateTime.MinValue;

family.Caretaker\_degree = "-";

family.Caretaker\_FIO = "-";

family.Caretaker\_tel = "-";

family.Caretaker\_workplace = "-";

family.Caretaker\_workpos = "-";

family.Dad\_date = DateTime.MinValue;

family.Dad\_FIO = "-";

family.Dad\_tel = "-";

family.Dad\_workplace = "-";

family.Dad\_workpos = "-";

}

else if (stat == "Только отец")

{

family.Dad\_FIO = DadSurnameTB.Text + " " + DadNameTB.Text + " " + DadOtcheTB.Text;

family.Dad\_tel = DadPhoneTB.Text;

family.Dad\_date = DadDate.SelectedDate;

family.Dad\_workplace = DadWorkTB.Text;

family.Dad\_workpos = DadWorkPosTB.Text;

family.Caretaker\_check = "Нет";

family.Caretaker\_date = DateTime.MinValue;

family.Caretaker\_degree = "-";

family.Caretaker\_FIO = "-";

family.Caretaker\_tel = "-";

family.Caretaker\_workplace = "-";

family.Caretaker\_workpos = "-";

family.Mom\_date = DateTime.MinValue;

family.Mom\_FIO = "-";

family.Mom\_tel = "-";

family.Mom\_workplace = "-";

family.Mom\_workpos = "-";

}

else if (stat == "Опекун")

{

family.Caretaker\_check = "Да";

family.Caretaker\_FIO = CaretSurname + " " + CaretName.Text + " " + CaretOtche.Text;

family.Caretaker\_tel = CaretPhoneTB.Text;

family.Caretaker\_date = CareDate.SelectedDate;

family.Caretaker\_workplace = CaretWorkTB.Text;

family.Caretaker\_workpos = CaretWorkPosTB.Text;

family.Caretaker\_degree = CaretakerLevl.Text;

family.Mom\_date = DateTime.MinValue;

family.Mom\_FIO = "-";

family.Mom\_tel = "-";

family.Mom\_workplace = "-";

family.Mom\_workpos = "-";

family.Dad\_date = DateTime.MinValue;

family.Dad\_FIO = "-";

family.Dad\_tel = "-";

family.Dad\_workplace = "-";

family.Dad\_workpos = "-";

}

else

{

family.Dad\_date = DateTime.MinValue;

family.Dad\_FIO = "-";

family.Dad\_tel = "-";

family.Dad\_workplace = "-";

family.Dad\_workpos = "-";

family.Mom\_date = DateTime.MinValue;

family.Mom\_FIO = "-";

family.Mom\_tel = "-";

family.Mom\_workplace = "-";

family.Mom\_workpos = "-";

family.Caretaker\_check = "Нет";

family.Caretaker\_date = DateTime.MinValue;

family.Caretaker\_degree = "-";

family.Caretaker\_FIO = "-";

family.Caretaker\_tel = "-";

family.Caretaker\_workplace = "-";

family.Caretaker\_workpos = "-";

}

FamilySufficiency famSuf = new FamilySufficiency();

famSuf.family\_house = HouseTB.Text;

famSuf.family\_house\_condition = HousingTB.Text;

famSuf.family\_influence = FamilyInfTB.Text;

famSuf.family\_rel = FamilyRelTB.Text;

famSuf.family\_help = FamlyHelpTB.Text;

famSuf.family\_connection = FamilyConnectTB.Text;

famSuf.college\_inform = CollegeInformTB.Text;

//Decrees dec = new Decrees();

//dec.Decree\_name = "Зачисление" + " " + stud.stud\_name + " " + stud.stud\_family;

//dec.Decree\_type = "Зачисление";

//dec.Decree\_date = (DateTime)BeginDate.SelectedDate;

//dec.Decree\_endyear = EndDate.Text;

//dec.Decree\_years = "2022-2023";

//dec.Id\_stud =

MessageBoxResult ad = MessageBox.Show("Вы уверены в добавлении студента?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.Students.Add(stud);

db.EduProcess.Add(edu);

db.Family.Add(family);

db.FamilySufficiency.Add(famSuf);

db.StudentData.Add(data);

db.StudActivity.Add(activ);

db.StudGroupRel.Add(groupRel);

db.StudPersona.Add(pers);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Ошибка!!! " + validationError.Entry.Entity.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

catch (Exception Ex)

{

MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

try

{

this.DragMove();

}

catch

{

}

}

public void TextValidationTextBox(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space) e.Handled = true;

}

private void NumberValidationTextBox(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void NumberValidationNumberDate(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9.]+");

e.Handled = regex.IsMatch(e.Text);

}

private void NumberValidationNumberPassport(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void TextValidationTextBox(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^a-zA-ZА-яА-я]");

e.Handled = regex.IsMatch(e.Text);

}

private void NumberValidationNumberClass(object sender, TextCompositionEventArgs e)

{

TextBox box = (TextBox)sender;

Regex regex = new Regex("[^91]+");

e.Handled = regex.IsMatch(e.Text);

}

public void AddPhoto()

{

OpenFileDialog open = new OpenFileDialog();

open.Title = "Select a picture";

open.Filter = "All supported graphics|\*.jpg;\*.jpeg;\*.png|" +

"JPEG (\*.jpg;\*.jpeg)|\*.jpg;\*.jpeg|" +

"Portable Network Graphic (\*.png)|\*.png";

if (open.ShowDialog() == true)

{

Photo\_byte = File.ReadAllBytes(open.FileName); // получаем байты выбранного файла

PhotoStud.Source = new BitmapImage(new Uri(open.FileName));

}

}

private void PoliceRecCheckBox\_Checked(object sender, RoutedEventArgs e)

{

RecDate.IsEnabled = true;

RecReasonTB.IsEnabled = true;

PolieRecPlaceTB.IsEnabled = true;

}

private void PoliceRecCheckBox\_Unchecked(object sender, RoutedEventArgs e)

{

RecDate.IsEnabled = false;

RecReasonTB.IsEnabled = false;

PolieRecPlaceTB.IsEnabled = false;

}

private void MomExpand(object sender, RoutedEventArgs e)

{

DadState.IsExpanded = false;

CaretakerState.IsExpanded = false;

Sis\_broState.IsExpanded = false;

}

private void DadExpend(object sender, RoutedEventArgs e)

{

MomState.IsExpanded = false;

CaretakerState.IsExpanded = false;

Sis\_broState.IsExpanded = false;

}

private void CaretakerExp(object sender, RoutedEventArgs e)

{

DadState.IsExpanded = false;

MomState.IsExpanded = false;

Sis\_broState.IsExpanded = false;

}

private void SisBroExp(object sender, RoutedEventArgs e)

{

DadState.IsExpanded = false;

CaretakerState.IsExpanded = false;

MomState.IsExpanded = false;

}

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

{

try

{

SaveStudent();

}

catch(Exception Ex)

{

MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

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

{

var uriSource = new Uri(@"/Resources/stud\_icon1.png", UriKind.Relative);

PhotoStud.Source = new BitmapImage(uriSource);

}

private void OtcheCheck\_Checked(object sender, RoutedEventArgs e)

{

OtcheTB.IsEnabled = false;

}

private void OtcheCheck\_Unchecked(object sender, RoutedEventArgs e)

{

OtcheTB.IsEnabled = true;

}

private void PoliceRecCheckBox\_Checked\_1(object sender, RoutedEventArgs e)

{

PolieRecPlaceTB.IsEnabled = false;

RecDate.IsEnabled = false;

RecReasonTB.IsEnabled = false;

}

private void GroupCmb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cmbAdd();

}

private void FamilyStatusCmb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (FamilyStatusCmb.SelectedIndex == 0)

{

MomState.Visibility = Visibility.Visible;

DadState.Visibility = Visibility.Visible;

CaretakerState.Visibility = Visibility.Hidden;

}

if (FamilyStatusCmb.SelectedIndex == 1)

{

MomState.Visibility = Visibility.Visible;

MomState.IsExpanded = true;

DadState.Visibility = Visibility.Hidden;

CaretakerState.Visibility = Visibility.Hidden;

}

if (FamilyStatusCmb.SelectedIndex == 2)

{

MomState.Visibility = Visibility.Hidden;

DadState.Visibility = Visibility.Visible;

MomState.IsExpanded = true;

CaretakerState.Visibility = Visibility.Hidden;

}

if (FamilyStatusCmb.SelectedIndex == 3)

{

MomState.Visibility = Visibility.Hidden;

DadState.Visibility = Visibility.Hidden;

CaretakerState.Visibility = Visibility.Visible;

CaretakerState.IsExpanded = true;

}

else

{

MomState.IsEnabled = false;

DadState.IsEnabled = false;

CaretakerState.IsEnabled = false;

}

}

private void InternalCheckBox\_Checked(object sender, RoutedEventArgs e)

{

InternalDate.IsEnabled = true;

InternalReasTB.IsEnabled = true;

InternalTalkDate.IsEnabled = true;

RemoveRecDate.IsEnabled = true;

RemoveRecReason.IsEnabled = true;

}

private void InternalCheckBox\_Unchecked(object sender, RoutedEventArgs e)

{

InternalDate.IsEnabled = false;

InternalReasTB.IsEnabled = false;

InternalTalkDate.IsEnabled = false;

RemoveRecDate.IsEnabled = false;

RemoveRecReason.IsEnabled = false;

}

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

{

WindowState = WindowState.Minimized;

}

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

{

StudHub a = new StudHub();

a.Show();

Close();

}

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

{

AddPhoto();

}

}

}

EditStudBook.xaml.cs

using StudentGroups.Data;

using StudentGroups.Class;

using StudentGroups.Class.BaseClass;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using Microsoft.Win32;

using System.IO;

using Microsoft.Reporting.Map.WebForms.BingMaps;

using System.Data.Entity.Validation;

using TextBox = System.Windows.Controls.TextBox;

using System.Windows.Markup;

using Microsoft.Office.Interop.Word;

using System.Data.Common;

using System.Data.SQLite;

using System.Data;

using Window = System.Windows.Window;

using System.Data.Entity.Migrations;

namespace StudentGroups.Windows.MainPages.StudentsWind

{

/// <summary>

/// Логика взаимодействия для EditStudBook.xaml

/// </summary>

public partial class EditStudBook : Window

{

Data.StudCheckDBEntities db = new StudCheckDBEntities();

List<GroupStudCl> groupList = new List<GroupStudCl>();

List<Speciality> specList = new List<Speciality>();

List<TutorStud> tutorList = new List<TutorStud>();

List<StudCard> card = new List<StudCard>();

List<StudDataCl> studData = new List<StudDataCl>();

List<Activ> act = new List<Activ>();

List<FamilyCl> fam = new List<FamilyCl>();

List<FamilySuffCl> suff = new List<FamilySuffCl>();

List<GroupRelCl> rel = new List<GroupRelCl>();

List<StudPersonaCl> pers = new List<StudPersonaCl>();

List<EduProcessCl> edu = new List<EduProcessCl>();

byte[] Photo\_byte = null;

int checkBase, checkEdu, checkRel, checkRecord, checkFamily, checkActiv, checkData = 0;

public EditStudBook()

{

InitializeComponent();

LoadStud();

LoadStudent();

}

private void ComboAddData()

{

foreach (var s in db.Tutors.ToList())

{

TutorStud tmp2 = new TutorStud()

{

tutorID = s.ID\_tutor,

tutorFIO = s.tutor\_family + " " + s.tutor\_name + " " + s.tutor\_otche

};

tutorList.Add(tmp2);

}

TutorCmb.ItemsSource = tutorList;

foreach (var x in db.Groups.ToList())

{

GroupStudCl tmp = new GroupStudCl

{

groupID = x.ID\_group,

groupName = x.group\_name,

groupCourse = x.course,

groupFaculty = x.faculty

};

groupList.Add(tmp);

Speciality tmp2 = new Speciality

{

groupID = x.ID\_group,

groupName = x.group\_name,

groupFaculty = x.faculty + " " + x.speciality

};

specList.Add(tmp2);

}

GroupCmb.ItemsSource = groupList;

}

private void cmbAdd()

{

int groupChoose = (GroupCmb.SelectedItem as GroupStudCl).groupID;

var spec = specList.Where(q => q.groupID == groupChoose);

SpecialCmb.ItemsSource = spec;

}

private void LoadStud()

{

int StudNum = StudEditCheck.StudIDCheck;

foreach (var x in db.Students.Where(x => x.ID\_stud == StudNum).ToList())

{

StudCard tmp = new StudCard

{

Stud\_id = x.ID\_stud,

stud\_Family = x.stud\_family,

stud\_Name = x.stud\_name,

stud\_Otche = x.stud\_otche,

stud\_email = x.stud\_email,

stud\_tel = x.stud\_tel,

stud\_photo = x.stud\_image,

stud\_tutor = db.Tutors.Where(q => q.ID\_tutor == x.ID\_tutor).SingleOrDefault().tutor\_family,

id\_group = db.Groups.Where(q => q.ID\_group == x.ID\_group).SingleOrDefault().ID\_group,

//id\_studdata = db.StudentData.Where(q => q.ID\_studdata == x.ID\_studdata).SingleOrDefault().ID\_studdata,

//id\_family = db.Family.Where(q => q.ID\_family == x.ID\_family).SingleOrDefault().ID\_family,

//id\_stud\_activ = db.StudActivity.Where(q => q.ID\_stud\_activ == x.ID\_stud\_activ).SingleOrDefault().ID\_stud\_activ,

//id\_edu\_process = db.EduProcess.Where(q => q.ID\_edu\_process == x.ID\_edu\_process).SingleOrDefault().ID\_edu\_process,

//id\_group\_rel = db.StudGroupRel.Where(q => q.ID\_group\_rel == x.ID\_group\_rel).SingleOrDefault().ID\_group\_rel,

//id\_stud\_persona = db.StudPersona.Where(q => q.ID\_stud\_persona == x.ID\_stud\_persona).SingleOrDefault().ID\_stud\_persona,

id\_studdata = StudEditCheck.StudIDCheck,

id\_family = StudEditCheck.StudIDCheck,

id\_stud\_activ = StudEditCheck.StudIDCheck,

id\_edu\_process = StudEditCheck.StudIDCheck,

id\_group\_rel = StudEditCheck.StudIDCheck,

id\_stud\_persona = StudEditCheck.StudIDCheck,

birth\_date = x.date\_birth,

gender = x.gender,

birth\_place = x.place\_birth

};

card.Add(tmp);

}

foreach (var q in db.StudentData.Where(q => q.ID\_studdata == StudEditCheck.StudIDCheck).ToList())

{

StudDataCl tmp = new StudDataCl

{

id\_studdata = q.ID\_studdata,

stud\_building = q.stud\_building,

fact\_address = q.fact\_address,

reg\_address = q.reg\_address,

SNILS = q.SNILS,

home\_tel = q.home\_tel,

passport\_seria = q.passport\_seria,

passport\_num = q.passport\_number,

passport\_date = q.passport\_date,

passport\_who = q.passport\_who,

prev\_edu = q.prev\_edu,

prev\_edu\_endyear = q.prev\_edu\_endyear,

police\_record\_check = q.police\_record\_check,

internal\_record\_check = q.internal\_record\_check,

police\_record\_date = (DateTime)q.police\_record\_date,

police\_record\_end = (DateTime)q.police\_record\_end,

police\_record\_who = q.police\_record\_who,

police\_record\_endreason = q.police\_record\_endreason,

police\_record\_reason = q.police\_record\_reason,

internal\_record\_date = (DateTime)q.internal\_record\_date,

internal\_record\_reason = q.internal\_record\_reason,

discuss\_date = (DateTime)q.discuss\_date,

};

studData.Add(tmp);

}

foreach (var q in db.StudActivity.Where(q => q.ID\_stud\_activ == StudNum).ToList())

{

Activ tmp2 = new Activ

{

id\_stud\_activ = q.ID\_stud\_activ,

addit\_edu = q.Addit\_edu,

hobby\_area = q.Hobby\_area,

hobby = q.Hobby,

freetime\_interest = q.Freetime\_interest,

socialLife\_part = q.SocialLife\_part

};

act.Add(tmp2);

}

foreach (var q in db.Family.Where(q => q.ID\_family == StudNum).ToList())

{

FamilyCl tmp4 = new FamilyCl

{

id\_family = q.ID\_family,

family\_status = q.Family\_status,

dad\_date = (DateTime)q.Dad\_date,

dad\_FIO = q.Dad\_FIO,

dad\_tel = q.Dad\_tel,

dad\_workplace = q.Dad\_workplace,

dad\_workpos = q.Dad\_workpos,

mom\_date = (DateTime)q.Mom\_date,

mom\_FIO = q.Mom\_FIO,

mom\_workplace = q.Mom\_workplace,

mom\_workpos = q.Mom\_workpos,

mom\_tel = q.Mom\_tel,

caretaker\_check = q.Caretaker\_check,

caretaker\_date = (DateTime)q.Caretaker\_date,

caretaker\_degree = q.Caretaker\_degree,

caretaker\_FIO = q.Caretaker\_FIO,

caretaker\_tel = q.Caretaker\_tel,

caretaker\_workplace = q.Caretaker\_workplace,

caretaker\_workpos = q.Caretaker\_workpos,

sis\_bro\_check = q.Sis\_bro\_check,

sis\_bro\_comment = q.Sis\_bro\_comment

};

foreach (var s in db.FamilySufficiency.Where(s => s.ID\_family\_suff == StudNum).ToList())

{

FamilySuffCl tmp5 = new FamilySuffCl

{

id\_family\_suff = s.ID\_family\_suff,

college\_inform = s.college\_inform,

family\_suff = s.family\_suff,

family\_connection = s.family\_influence,

family\_help = s.family\_help,

family\_house = s.family\_house,

family\_house\_condition = s.family\_house\_condition,

family\_influence = s.family\_influence,

family\_rel = s.family\_rel

};

suff.Add(tmp5);

}

fam.Add(tmp4);

}

foreach (var q in db.StudGroupRel.Where(q => q.ID\_group\_rel == StudNum).ToList())

{

GroupRelCl tmp6 = new GroupRelCl

{

id\_group\_rel = q.ID\_group\_rel,

group\_influence = q.Group\_influence,

group\_position = q.Group\_position,

college\_friends = q.College\_friends,

socialTask\_opinion = q.SocialTask\_opinion,

rel\_opinion = q.Rel\_opinion

};

rel.Add(tmp6);

}

foreach (var q in db.StudPersona.Where(q => q.ID\_stud\_persona == StudNum).ToList())

{

StudPersonaCl tmp7 = new StudPersonaCl

{

id\_stud\_persona = q.ID\_stud\_persona,

stud\_bad\_behav = q.Stud\_bad\_behav,

stud\_bad\_habit = q.Stud\_bad\_habit,

stud\_college\_rel = q.Stud\_college\_rel,

stud\_conflict = q.Stud\_conflict,

stud\_crime = q.Stud\_crime,

stud\_crime\_reason = q.Stud\_crime\_reason,

stud\_envir = q.Stud\_envir,

stud\_good\_behav = q.Stud\_good\_behav,

stud\_phys = q.Stud\_phys,

stud\_role = q.Stud\_role,

stud\_socialhelp = q.Stud\_socialhelp,

stud\_subculture = q.Stud\_subculture,

};

pers.Add(tmp7);

}

foreach (var q in db.EduProcess.Where(q => q.ID\_edu\_process == StudNum).ToList())

{

EduProcessCl tmp8 = new EduProcessCl

{

id\_edu\_process = q.ID\_edu\_process,

edu\_debt = q.Edu\_debt,

edu\_interest = q.Edu\_interest,

edu\_level = q.Edu\_level,

edu\_rel = q.Edu\_rel,

};

edu.Add(tmp8);

}

}

//public void TextValidation()

//{

// try

// {

// checkBase = 0;

// if (String.IsNullOrEmpty(SurnameTB.Text) || String.IsNullOrEmpty(NameTB.Text) || String.IsNullOrEmpty(OtcheTB.Text) ||

// String.IsNullOrEmpty(SeriaPassTB.Text) || String.IsNullOrEmpty(NumberPassTB.Text) ||

// String.IsNullOrEmpty(WhoPassTB.Text) || String.IsNullOrEmpty(RegAddres.Text) || String.IsNullOrEmpty(FactAdress.Text)

// || String.IsNullOrEmpty(EmailBox.Text))

// {

// MessageBox.Show("Заполните информацию в вкладке: Основная информация", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (SeriaPassTB.Text.Length != 4)

// {

// MessageBox.Show("В серии паспорта должно быть 4 цифры", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (NumberPassTB.Text.Length != 6)

// {

// MessageBox.Show("В номере паспорта должно быть 6 цифры", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (SNILSTB.Text.Length != 11)

// {

// MessageBox.Show("В номере СНИЛСа должно быть 11 цифр", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (MobPhoneBox.Text.Length != 11)

// {

// MessageBox.Show("В номер телефона должно быть 11 цифр", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else if (HomePhoneBox.Text.Length != 11)

// {

// MessageBox.Show("В номер телефона должно быть 11 цифр", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkBase = 1;

// }

// else

// {

// }

// checkEdu = 0;

// if (String.IsNullOrEmpty(PrevEduTB.Text) || String.IsNullOrEmpty(PrevEndYearTBB.Text)

// || BeginDate.SelectedDate.Value == null || String.IsNullOrEmpty(EndDate.Text) || String.IsNullOrEmpty(DecreeName.Text)

// || String.IsNullOrEmpty((string)((ComboBoxItem)CourseCmb.SelectedItem).Content))

// {

// MessageBox.Show("Заполните информацию в вкладке: Учебная информация", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkEdu = 1;

// }

// checkActiv = 0;

// if (String.IsNullOrEmpty(HobbyTB.Text))

// {

// MessageBox.Show("Заполните информацию в вкладке: Активность", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkActiv = 1;

// }

// checkRecord = 0;

// if (PoliceRecCheckBox.IsChecked == false)

// {

// MessageBox.Show("Заполните информацию в вкладке: Постановка на учет", "Сообщение", MessageBoxButton.OK, MessageBoxImage.Error);

// checkFamily = 1;

// }

// }

// catch (Exception Ex)

// {

// MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

// }

//}

private void LoadStudent()

{

SurnameTB.Text = card.SingleOrDefault().stud\_Family;

NameTB.Text = card.SingleOrDefault().stud\_Name;

if (OtcheTB.Text == "Отсутствует")

{

OtcheCheck.IsChecked = false;

}

else

{

OtcheTB.Text = card.SingleOrDefault().stud\_Otche;

}

GenderCmb.SelectedItem = card.SingleOrDefault().gender;

DateBirthTB.SelectedDate = card.SingleOrDefault().birth\_date;

BirthPlaceTB.Text = card.SingleOrDefault().birth\_place;

MobPhoneBox.Text = card.SingleOrDefault().stud\_tel;

EmailBox.Text = card.SingleOrDefault().stud\_email;

TutorCmb.SelectedItem = card.SingleOrDefault().stud\_tutor;

GroupCmb.SelectedItem = card.SingleOrDefault().id\_group;

Photo\_byte = card.SingleOrDefault().stud\_photo;

HobbyTB.Text = act.SingleOrDefault().hobby;

HobbyAreaTB.Text = act.SingleOrDefault().hobby\_area;

InterestTB.Text = act.SingleOrDefault().freetime\_interest;

AddEduTB.Text = act.SingleOrDefault().addit\_edu;

SocialLifeTB.Text = act.SingleOrDefault().socialLife\_part;

EduRelCmb.Text = edu.SingleOrDefault().edu\_rel;

EduLevelCmb.Text = edu.SingleOrDefault().edu\_level;

InterestTB.Text = edu.SingleOrDefault().edu\_interest;

DebtTB.Text = edu.SingleOrDefault().edu\_debt;

StudBuildTB.Text = studData.SingleOrDefault().stud\_building;

RegAddres.Text = studData.SingleOrDefault().reg\_address;

FactAdress.Text = studData.SingleOrDefault().fact\_address;

HomePhoneBox.Text = studData.SingleOrDefault().home\_tel;

PrevEduTB.Text = studData.SingleOrDefault().prev\_edu;

PrevEndYearTBB.Text = studData.SingleOrDefault().prev\_edu\_endyear;

SeriaPassTB.Text = studData.SingleOrDefault().passport\_seria;

NumberPassTB.Text = studData.SingleOrDefault().passport\_num;

DatePass.SelectedDate = studData.SingleOrDefault().passport\_date;

WhoPassTB.Text = studData.SingleOrDefault().passport\_who;

SNILSTB.Text = studData.SingleOrDefault().SNILS;

if (studData.SingleOrDefault().police\_record\_check == "Состоит")

{

PoliceRecCheckBox.IsChecked = false;

RecReasonTB.Text = studData.SingleOrDefault().police\_record\_reason;

RecDate.SelectedDate = studData.SingleOrDefault().police\_record\_date;

PolieRecPlaceTB.Text = studData.SingleOrDefault().police\_record\_who;

}

else

{

}

if (studData.SingleOrDefault().internal\_record\_check == "Состоит")

{

InternalCheckBox.IsChecked = true;

InternalReasTB.Text = studData.SingleOrDefault().internal\_record\_reason;

InternalDate.SelectedDate = studData.SingleOrDefault().internal\_record\_date;

InternalTalkDate.SelectedDate = studData.SingleOrDefault().discuss\_date;

RemoveRecReason.Text = studData.SingleOrDefault().police\_record\_endreason;

RemoveRecDate.SelectedDate = studData.SingleOrDefault().police\_record\_end;

}

// StudGroupRel groupRel = new StudGroupRel();

StudPosCmb.Text = rel.SingleOrDefault().group\_position;

OpinionRelCmb.Text = rel.SingleOrDefault().rel\_opinion;

SocialTaskCmb.Text = rel.SingleOrDefault().socialTask\_opinion;

GroupInflCmb.Text = rel.SingleOrDefault().group\_influence;

FriendTB.Text = rel.SingleOrDefault().college\_friends;

// StudPersona pers = new StudPersona();

PhysCmb.Text = pers.SingleOrDefault().stud\_phys;

SubcultureTB.Text = pers.SingleOrDefault().stud\_subculture;

SpecHelpTB.Text = pers.SingleOrDefault().stud\_socialhelp;

CollRelCmb.Text = pers.SingleOrDefault().stud\_college\_rel;

EnvirRelTB.Text = pers.SingleOrDefault().stud\_envir;

ConflictTB.Text = pers.SingleOrDefault().stud\_conflict;

StudRoleCmb.Text = pers.SingleOrDefault().stud\_role;

GoodTB.Text = pers.SingleOrDefault().stud\_good\_behav;

BadTB.Text = pers.SingleOrDefault().stud\_bad\_behav;

CrimeTB.Text = pers.SingleOrDefault().stud\_crime;

ReasonCrimeTB.Text = pers.SingleOrDefault().stud\_crime\_reason;

BadHabitTB.Text = pers.SingleOrDefault().stud\_bad\_habit;

// Family family = new Family();

FamilyStatusCmb.Text = fam.SingleOrDefault().family\_status;

if (fam.SingleOrDefault().sis\_bro\_check == "Да")

{

SisBroCheckBox.IsChecked = true;

SisBroListTB.Text = fam.SingleOrDefault().sis\_bro\_comment;

}

else

{

SisBroCheckBox.IsChecked = false;

}

string stat = FamilyStatusCmb.Text;

if (stat == "Полная")

{

string m = fam.SingleOrDefault().mom\_FIO;

string[] Momwords = m.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

MomSurnameTB.Text = Momwords[0];

MomNameTB.Text = Momwords[1];

MomOtcheTB.Text = Momwords[2];

MomPhoneTB.Text = fam.SingleOrDefault().mom\_tel;

MomDate.SelectedDate = fam.SingleOrDefault().mom\_date;

MomWorkTB.Text = fam.SingleOrDefault().mom\_workplace;

MomWorkPosTB.Text = fam.SingleOrDefault().mom\_workpos;

string d = fam.SingleOrDefault().dad\_FIO;

string[] Dadwords = d.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

DadSurnameTB.Text = Dadwords[0];

DadNameTB.Text = Dadwords[1];

DadOtcheTB.Text = Dadwords[2];

DadPhoneTB.Text = fam.SingleOrDefault().dad\_tel;

DadDate.SelectedDate = fam.SingleOrDefault().dad\_date;

DadWorkTB.Text = fam.SingleOrDefault().dad\_workplace;

DadWorkPosTB.Text = fam.SingleOrDefault().dad\_workpos;

}

else if (stat == "Только мать")

{

string m = fam.SingleOrDefault().mom\_FIO;

string[] Momwords = m.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

MomSurnameTB.Text = Momwords[0];

MomNameTB.Text = Momwords[1];

MomOtcheTB.Text = Momwords[2];

MomPhoneTB.Text = fam.SingleOrDefault().mom\_tel;

MomDate.SelectedDate = fam.SingleOrDefault().mom\_date;

MomWorkTB.Text = fam.SingleOrDefault().mom\_workplace;

MomWorkPosTB.Text = fam.SingleOrDefault().mom\_workpos;

}

else if (stat == "Только отец")

{

string d = fam.SingleOrDefault().dad\_FIO;

string[] Dadwords = d.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

DadSurnameTB.Text = Dadwords[0];

DadNameTB.Text = Dadwords[1];

DadOtcheTB.Text = Dadwords[2];

DadPhoneTB.Text = fam.SingleOrDefault().dad\_tel;

DadDate.SelectedDate = fam.SingleOrDefault().dad\_date;

DadWorkTB.Text = fam.SingleOrDefault().dad\_workplace;

DadWorkPosTB.Text = fam.SingleOrDefault().dad\_workpos;

}

else if (stat == "Опекун")

{

string o = fam.SingleOrDefault().caretaker\_FIO;

string[] Carewords = o.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

CaretSurname.Text = Carewords[0];

CaretName.Text = Carewords[1];

CaretOtche.Text = Carewords[2];

CaretPhoneTB.Text = fam.SingleOrDefault().caretaker\_tel;

CareDate.SelectedDate = fam.SingleOrDefault().caretaker\_date;

CaretWorkTB.Text = fam.SingleOrDefault().caretaker\_workplace;

CaretWorkPosTB.Text = fam.SingleOrDefault().caretaker\_workpos;

}

else

{

}

// FamilySufficiency famSuf = new FamilySufficiency();

HouseTB.Text = suff.SingleOrDefault().family\_house;

HousingTB.Text = suff.SingleOrDefault().family\_house\_condition;

FamilyInfTB.Text = suff.SingleOrDefault().family\_influence;

FamilyRelTB.Text = suff.SingleOrDefault().family\_rel;

FamlyHelpTB.Text = suff.SingleOrDefault().family\_help;

FamilyConnectTB.Text = suff.SingleOrDefault().family\_connection;

CollegeInformTB.Text = suff.SingleOrDefault().college\_inform;

}

private void SaveStudent()

{

//TextValidation();

var stud = db.Students.Where(q => q.ID\_stud == StudEditCheck.StudIDCheck).FirstOrDefault();

stud.stud\_family = SurnameTB.Text;

stud.stud\_name = NameTB.Text;

if (OtcheCheck.IsChecked == true)

{

stud.stud\_otche = "Отсутствует";

}

else

{

stud.stud\_otche = OtcheTB.Text;

}

stud.gender = GenderCmb.Text;

stud.date\_birth = (DateTime)DateBirthTB.SelectedDate;

stud.place\_birth = BirthPlaceTB.Text;

stud.stud\_tel = MobPhoneBox.Text;

stud.stud\_email = EmailBox.Text;

stud.stud\_image = Photo\_byte;

stud.ID\_tutor = (TutorCmb.SelectedItem as TutorStud).tutorID;

stud.ID\_group = (GroupCmb.SelectedItem as GroupStudCl).groupID;

stud.ID\_studdata = StudEditCheck.StudIDCheck;

stud.ID\_family = StudEditCheck.StudIDCheck;

stud.ID\_stud\_activ = StudEditCheck.StudIDCheck;

stud.ID\_edu\_process = StudEditCheck.StudIDCheck;

stud.ID\_group\_rel = StudEditCheck.StudIDCheck;

stud.ID\_stud\_persona = StudEditCheck.StudIDCheck;

var activ = db.StudActivity.Where(q => q.ID\_stud\_activ == StudEditCheck.StudIDCheck).FirstOrDefault();

activ.Hobby = HobbyTB.Text;

activ.Hobby\_area = HobbyAreaTB.Text;

activ.Freetime\_interest = InterestTB.Text;

activ.Addit\_edu = AddEduTB.Text;

activ.SocialLife\_part = SocialLifeTB.Text;

activ.Freetime\_interest = FreeTimeTB.Text;

var edu = db.EduProcess.Where(q => q.ID\_edu\_process == StudEditCheck.StudIDCheck).FirstOrDefault();

edu.Edu\_rel = EduRelCmb.Text;

edu.Edu\_level = EduLevelCmb.Text;

edu.Edu\_interest = InterestTB.Text;

edu.Edu\_debt = DebtTB.Text;

var data = db.StudentData.Where(q => q.ID\_studdata == StudEditCheck.StudIDCheck).FirstOrDefault();

data.stud\_building = StudBuildTB.Text;

data.reg\_address = RegAddres.Text;

data.fact\_address = FactAdress.Text;

data.home\_tel = HomePhoneBox.Text;

data.prev\_edu = PrevEduTB.Text;

data.prev\_edu\_endyear = PrevEndYearTBB.Text;

data.passport\_seria = SeriaPassTB.Text;

data.passport\_number = NumberPassTB.Text;

data.passport\_date = (DateTime)DatePass.SelectedDate;

data.passport\_who = WhoPassTB.Text;

data.SNILS = SNILSTB.Text;

if (PoliceRecCheckBox.IsChecked == false)

{

data.police\_record\_check = "Состоит";

data.police\_record\_reason = RecReasonTB.Text;

data.police\_record\_date = RecDate.SelectedDate;

data.police\_record\_who = PolieRecPlaceTB.Text;

}

else

{

data.police\_record\_check = "Не состоит";

}

if (InternalCheckBox.IsChecked == true)

{

data.internal\_record\_check = "Состоит";

data.internal\_record\_reason = InternalReasTB.Text;

data.internal\_record\_date = InternalDate.SelectedDate;

data.discuss\_date = InternalTalkDate.SelectedDate;

data.police\_record\_endreason = RemoveRecReason.Text;

data.police\_record\_end = RemoveRecDate.SelectedDate;

}

var groupRel = db.StudGroupRel.Where(q => q.ID\_group\_rel == StudEditCheck.StudIDCheck).FirstOrDefault();

groupRel.Group\_position = StudPosCmb.Text;

groupRel.Rel\_opinion = OpinionRelCmb.Text;

groupRel.SocialTask\_opinion = SocialTaskCmb.Text;

groupRel.Group\_influence = GroupInflCmb.Text;

groupRel.College\_friends = FriendTB.Text;

var pers = db.StudPersona.Where(q => q.ID\_stud\_persona == StudEditCheck.StudIDCheck).FirstOrDefault();

pers.Stud\_phys = PhysCmb.Text;

pers.Stud\_subculture = SubcultureTB.Text;

pers.Stud\_socialhelp = SpecHelpTB.Text;

pers.Stud\_college\_rel = CollRelCmb.Text;

pers.Stud\_envir = EnvirRelTB.Text;

pers.Stud\_conflict = ConflictTB.Text;

pers.Stud\_role = StudRoleCmb.Text;

pers.Stud\_good\_behav = GoodTB.Text;

pers.Stud\_bad\_behav = BadTB.Text;

pers.Stud\_crime = CrimeTB.Text;

pers.Stud\_crime\_reason = ReasonCrimeTB.Text;

pers.Stud\_bad\_habit = BadHabitTB.Text;

var family = db.Family.Where(q => q.ID\_family == StudEditCheck.StudIDCheck).FirstOrDefault();

family.Family\_status = FamilyStatusCmb.Text;

if (SisBroCheckBox.IsChecked == true)

{

family.Sis\_bro\_check = "Да";

family.Sis\_bro\_comment = SisBroListTB.Text;

}

else

{

family.Sis\_bro\_check = "Нет";

}

string stat = FamilyStatusCmb.Text;

if (stat == "Полная")

{

family.Parents\_check = "Полная";

family.Mom\_FIO = MomSurnameTB.Text + " " + MomNameTB.Text + " " + MomOtcheTB.Text;

family.Mom\_tel = MomPhoneTB.Text;

family.Mom\_date = MomDate.SelectedDate;

family.Mom\_workplace = MomWorkTB.Text;

family.Mom\_workpos = MomWorkPosTB.Text;

family.Dad\_FIO = DadSurnameTB.Text + " " + DadNameTB.Text + " " + DadOtcheTB.Text;

family.Dad\_tel = MomPhoneTB.Text;

family.Dad\_date = MomDate.SelectedDate;

family.Dad\_workplace = MomWorkTB.Text;

family.Dad\_workpos = MomWorkPosTB.Text;

family.Caretaker\_check = "Нет";

}

else if (stat == "Только мать")

{

family.Mom\_FIO = MomSurnameTB.Text + " " + MomNameTB.Text + " " + MomOtcheTB.Text;

family.Mom\_tel = MomPhoneTB.Text;

family.Mom\_date = MomDate.SelectedDate;

family.Mom\_workplace = MomWorkTB.Text;

family.Mom\_workpos = MomWorkPosTB.Text;

family.Caretaker\_check = "Нет";

}

else if (stat == "Только отец")

{

family.Dad\_FIO = DadSurnameTB.Text + " " + DadNameTB.Text + " " + DadOtcheTB.Text;

family.Dad\_tel = MomPhoneTB.Text;

family.Dad\_date = MomDate.SelectedDate;

family.Dad\_workplace = MomWorkTB.Text;

family.Dad\_workpos = MomWorkPosTB.Text;

family.Caretaker\_check = "Нет";

}

else if (stat == "Опекун")

{

family.Caretaker\_check = "Да";

family.Caretaker\_FIO = CaretSurname + " " + CaretName.Text + " " + CaretOtche.Text;

family.Caretaker\_tel = CaretPhoneTB.Text;

family.Caretaker\_date = CareDate.SelectedDate;

family.Caretaker\_workplace = CaretWorkTB.Text;

family.Caretaker\_workpos = CaretWorkPosTB.Text;

}

else

{

}

var famSuf = db.FamilySufficiency.Where(q => q.ID\_family\_suff == StudEditCheck.StudIDCheck).FirstOrDefault();

famSuf.family\_house = HouseTB.Text;

famSuf.family\_house\_condition = HousingTB.Text;

famSuf.family\_influence = FamilyInfTB.Text;

famSuf.family\_rel = FamilyRelTB.Text;

famSuf.family\_help = FamlyHelpTB.Text;

famSuf.family\_connection = FamilyConnectTB.Text;

famSuf.college\_inform = CollegeInformTB.Text;

MessageBoxResult ad = MessageBox.Show("Вы уверены в изменении студента?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

private void Window\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

try

{

this.DragMove();

}

catch

{

}

}

public void TextValidationTextBox(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space) e.Handled = true;

}

private void NumberValidationTextBox(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void NumberValidationNumberDate(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9.]+");

e.Handled = regex.IsMatch(e.Text);

}

private void NumberValidationNumberPassport(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

private void TextValidationTextBox(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^a-zA-ZА-яА-я]");

e.Handled = regex.IsMatch(e.Text);

}

private void NumberValidationNumberClass(object sender, TextCompositionEventArgs e)

{

TextBox box = (TextBox)sender;

Regex regex = new Regex("[^91]+");

e.Handled = regex.IsMatch(e.Text);

}

public void AddPhoto()

{

OpenFileDialog open = new OpenFileDialog();

open.Title = "Select a picture";

open.Filter = "All supported graphics|\*.jpg;\*.jpeg;\*.png|" +

"JPEG (\*.jpg;\*.jpeg)|\*.jpg;\*.jpeg|" +

"Portable Network Graphic (\*.png)|\*.png";

if (open.ShowDialog() == true)

{

Photo\_byte = File.ReadAllBytes(open.FileName); // получаем байты выбранного файла

PhotoStud.Source = new BitmapImage(new Uri(open.FileName));

}

}

private void PoliceRecCheckBox\_Checked(object sender, RoutedEventArgs e)

{

RecDate.IsEnabled = true;

RecReasonTB.IsEnabled = true;

PolieRecPlaceTB.IsEnabled = true;

}

private void PoliceRecCheckBox\_Unchecked(object sender, RoutedEventArgs e)

{

RecDate.IsEnabled = false;

RecReasonTB.IsEnabled = false;

PolieRecPlaceTB.IsEnabled = false;

}

private void MomExpand(object sender, RoutedEventArgs e)

{

DadState.IsExpanded = false;

CaretakerState.IsExpanded = false;

Sis\_broState.IsExpanded = false;

}

private void DadExpend(object sender, RoutedEventArgs e)

{

MomState.IsExpanded = false;

CaretakerState.IsExpanded = false;

Sis\_broState.IsExpanded = false;

}

private void CaretakerExp(object sender, RoutedEventArgs e)

{

DadState.IsExpanded = false;

MomState.IsExpanded = false;

Sis\_broState.IsExpanded = false;

}

private void SisBroExp(object sender, RoutedEventArgs e)

{

DadState.IsExpanded = false;

CaretakerState.IsExpanded = false;

MomState.IsExpanded = false;

}

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

{

SaveStudent();

}

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

{

var uriSource = new Uri(@"/Resources/stud\_icon1.png", UriKind.Relative);

PhotoStud.Source = new BitmapImage(uriSource);

}

private void OtcheCheck\_Checked(object sender, RoutedEventArgs e)

{

OtcheTB.IsEnabled = false;

}

private void OtcheCheck\_Unchecked(object sender, RoutedEventArgs e)

{

OtcheTB.IsEnabled = true;

}

private void PoliceRecCheckBox\_Checked\_1(object sender, RoutedEventArgs e)

{

PolieRecPlaceTB.IsEnabled = false;

RecDate.IsEnabled = false;

RecReasonTB.IsEnabled = false;

}

private void GroupCmb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

cmbAdd();

}

private void FamilyStatusCmb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (FamilyStatusCmb.SelectedIndex == 0)

{

MomState.Visibility = Visibility.Visible;

DadState.Visibility = Visibility.Visible;

CaretakerState.Visibility = Visibility.Hidden;

}

if (FamilyStatusCmb.SelectedIndex == 1)

{

MomState.Visibility = Visibility.Visible;

MomState.IsExpanded = true;

DadState.Visibility = Visibility.Hidden;

CaretakerState.Visibility = Visibility.Hidden;

}

if (FamilyStatusCmb.SelectedIndex == 2)

{

MomState.Visibility = Visibility.Hidden;

DadState.Visibility = Visibility.Visible;

MomState.IsExpanded = true;

CaretakerState.Visibility = Visibility.Hidden;

}

if (FamilyStatusCmb.SelectedIndex == 3)

{

MomState.Visibility = Visibility.Hidden;

DadState.Visibility = Visibility.Hidden;

CaretakerState.Visibility = Visibility.Visible;

CaretakerState.IsExpanded = true;

}

else

{

MomState.IsEnabled = false;

DadState.IsEnabled = false;

CaretakerState.IsEnabled = false;

}

}

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

{

}

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

{

try

{

RecStudentMakeRecord();

}

catch(Exception Ex)

{

MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void InternalCheckBox\_Checked(object sender, RoutedEventArgs e)

{

InternalDate.IsEnabled = true;

InternalReasTB.IsEnabled = true;

InternalTalkDate.IsEnabled = true;

RemoveRecDate.IsEnabled = true;

RemoveRecReason.IsEnabled = true;

}

private void InternalCheckBox\_Unchecked(object sender, RoutedEventArgs e)

{

InternalDate.IsEnabled = false;

InternalReasTB.IsEnabled = false;

InternalTalkDate.IsEnabled = false;

RemoveRecDate.IsEnabled = false;

RemoveRecReason.IsEnabled = false;

}

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

{

WindowState = WindowState.Minimized;

}

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

{

StudHub a = new StudHub();

a.Show();

Close();

}

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

{

AddPhoto();

}

public void RecStudentMakeRecord()

{

try

{

Microsoft.Office.Interop.Word.Document doc = null;

Microsoft.Office.Interop.Word.\_Application oWord = new Microsoft.Office.Interop.Word.Application();

Microsoft.Office.Interop.Word.Application app = new Microsoft.Office.Interop.Word.Application();

// Путь до шаблона документа

string source = System.IO.Path.Combine(Environment.CurrentDirectory, "Resources", "Forms", "Учётная карточка студента.docx");

// Открываем

doc = app.Documents.Open(source);

doc.Activate();

doc.Bookmarks["StudFIO"].Range.Text = NameTB.Text + " " + SurnameTB.Text + " " + OtcheTB.Text;

doc.Bookmarks["StudBirth"].Range.Text = DateBirthTB.Text;

doc.Bookmarks["GroupName"].Range.Text = GroupCmb.Text;

doc.Bookmarks["Speciality"].Range.Text = SpecialCmb.Text;

doc.Bookmarks["DecreeStart"].Range.Text = DecreeName.Text;

doc.Bookmarks["StudEndYear"].Range.Text = EndDate.Text;

doc.Bookmarks["FactAdress"].Range.Text = FactAdress.Text;

doc.Bookmarks["RegAdress"].Range.Text = RegAddres.Text;

doc.Bookmarks["MomFIO"].Range.Text = MomNameTB.Text + " " + MomSurnameTB.Text + " " + MomOtcheTB.Text;

doc.Bookmarks["MomDate"].Range.Text = MomDate.SelectedDate.ToString();

doc.Bookmarks["MomWork"].Range.Text = MomWorkTB.Text;

doc.Bookmarks["MomTel"].Range.Text = MomPhoneTB.Text;

doc.Bookmarks["MomWorkPos"].Range.Text = MomWorkPosTB.Text;

doc.Bookmarks["CareFIO"].Range.Text = CaretName.Text + " " + CaretSurname.Text + " " + CaretOtche.Text;

doc.Bookmarks["CareDate"].Range.Text = CareDate.SelectedDate.ToString();

doc.Bookmarks["CareWork"].Range.Text = CaretWorkTB.Text;

doc.Bookmarks["CareTel"].Range.Text = CaretPhoneTB.Text;

doc.Bookmarks["CareWorkPos"].Range.Text = CaretWorkPosTB.Text;

doc.Bookmarks["DadFIO"].Range.Text = DadNameTB.Text + " " + DadSurnameTB.Text + " " + DadOtcheTB.Text;

doc.Bookmarks["DadDate"].Range.Text = DadDate.SelectedDate.ToString();

doc.Bookmarks["DadWork"].Range.Text = DadWorkTB.Text;

doc.Bookmarks["DadTel"].Range.Text = DadPhoneTB.Text;

doc.Bookmarks["DadWorkPos"].Range.Text = DadWorkPosTB.Text;

doc.Bookmarks["SisBro"].Range.Text = SisBroListTB.Text;

doc.Bookmarks["Suffiency"].Range.Text = SufficiencyCmb.Text;

if (PoliceRecCheckBox.IsChecked == false)

{

doc.Bookmarks["PoliceRecCheck"].Range.Text = "Есть";

doc.Bookmarks["PoliceRecWho"].Range.Text = PolieRecPlaceTB.Text + "; " + RecDate.Text + "; " + RecReasonTB.Text;

}

else

{

doc.Bookmarks["PoliceRecCheck"].Range.Text = "Отсутствует";

}

if (InternalCheckBox.IsChecked == true)

{

doc.Bookmarks["InternalRec"].Range.Text = InternalDate.SelectedDate.ToString() + "; " + InternalReasTB.Text; ;

doc.Bookmarks["DiscussDate"].Range.Text = InternalTalkDate.SelectedDate.ToString();

doc.Bookmarks["RecEnd"].Range.Text = RemoveRecDate.SelectedDate.ToString() + "; " + RemoveRecReason.Text;

}

else

{

doc.Bookmarks["InternalRec"].Range.Text = "Отсутствует";

}

doc.Bookmarks["FamConnect"].Range.Text = FamilyConnectTB.Text;

doc.Bookmarks["FamHouse"].Range.Text = HouseTB.Text;

doc.Bookmarks["FamHouseCond"].Range.Text = HousingTB.Text;

doc.Bookmarks["FamInfluence"].Range.Text = FamilyInfTB.Text;

doc.Bookmarks["FamInform"].Range.Text = FamilyInfTB.Text;

doc.Bookmarks["FamNeedHelp"].Range.Text = FamlyHelpTB.Text;

doc.Bookmarks["FamRel"].Range.Text = FamilyRelTB.Text;

doc.Bookmarks["FamResources"].Range.Text = SufficiencyCmb.Text;

doc.Bookmarks["FreeTime"].Range.Text = FreeTimeTB.Text;

doc.Bookmarks["Friends"].Range.Text = FriendTB.Text;

doc.Bookmarks["Good"].Range.Text = GoodTB.Text;

doc.Bookmarks["GroupPos"].Range.Text = StudPosCmb.Text;

doc.Bookmarks["GroupRel"].Range.Text = OpinionRelCmb.Text;

doc.Bookmarks["Hobby"].Range.Text = HobbyTB.Text;

doc.Bookmarks["HobbyArea"].Range.Text = HobbyAreaTB.Text;

doc.Bookmarks["Influence"].Range.Text = GroupInflCmb.Text;

doc.Bookmarks["Phys"].Range.Text = PhysCmb.Text;

doc.Bookmarks["SocHelp"].Range.Text = SpecHelpTB.Text;

doc.Bookmarks["SocLife"].Range.Text = SocialLifeTB.Text;

doc.Bookmarks["SocTaskRel"].Range.Text = SocialTaskCmb.Text;

doc.Bookmarks["StudBuilding"].Range.Text = StudBuildTB.Text;

doc.Bookmarks["StudID"].Range.Text = StudEditCheck.StudIDCheck.ToString();

doc.Bookmarks["StudRole"].Range.Text = StudRoleCmb.Text;

doc.Bookmarks["Subculture"].Range.Text = SubcultureTB.Text;

doc.Bookmarks["TutorFIO"].Range.Text = TutorCmb.Text;

doc.Bookmarks["FormDate"].Range.Text = DateTime.Now.ToString();

doc.Bookmarks["EduRel"].Range.Text = EduRelCmb.Text;

doc.Bookmarks["EduLevel"].Range.Text = EduLevelCmb.Text;

doc.Bookmarks["EnvirRel"].Range.Text = EnvirRelTB.Text;

doc.Bookmarks["EduInterest"].Range.Text = InterestTB.Text;

doc.Bookmarks["Debt"].Range.Text = DebtTB.Text;

doc.Bookmarks["Crime"].Range.Text = CrimeTB.Text;

doc.Bookmarks["CrimeReason"].Range.Text = ReasonCrimeTB.Text;

doc.Bookmarks["Conflict"].Range.Text = ConflictTB.Text;

doc.Bookmarks["CollRel"].Range.Text = CollRelCmb.Text;

doc.Bookmarks["BadHabits"].Range.Text = BadHabitTB.Text;

doc.Bookmarks["Bad"].Range.Text = BadTB.Text;

doc.Bookmarks["AdditEdu"].Range.Text = AddEduTB.Text;

doc.Bookmarks["DateNow"].Range.Text = DateTime.Now.ToString("D");

string CardSource = System.IO.Path.Combine(Environment.CurrentDirectory, "Records"); ;

doc.SaveAs($@"{CardSource}\{SurnameTB.Text} {NameTB.Text} {OtcheTB.Text}");

doc.Close();

doc = null;

app.Quit();

MessageBox.Show($@"Карточка сформирована и располагается в {CardSource}");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

}

StudHub.xaml.cs

using StudentGroups.Class.BaseClass;

using System;

using System.Collections.Generic;

using System.Data.Common;

using System.Data.SQLite;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using DataTable = System.Data.DataTable;

using Microsoft.Office.Interop.Excel;

using StudentGroups.Class;

namespace StudentGroups.Windows.MainPages.StudentsWind

{

/// <summary>

/// Логика взаимодействия для StudHub.xaml

/// </summary>

public partial class StudHub : System.Windows.Window

{

Data.StudCheckDBEntities db = new Data.StudCheckDBEntities();

List<StudList> stdlist = new List<StudList>();

public StudHub()

{

InitializeComponent();

Load();

}

private void Load()

{

StudDG.ItemsSource = null;

foreach (var x in db.Students.ToList())

{

StudList tmp = new StudList

{

Stud\_id = x.ID\_stud,

stud\_Family = x.stud\_family,

stud\_Name = x.stud\_name,

stud\_Otche = x.stud\_otche,

gender = x.gender,

birth\_date = (DateTime)x.date\_birth,

stud\_tel = x.stud\_tel,

stud\_group = db.Groups.Where(q => q.ID\_group == x.ID\_group).Single().group\_name,

stud\_email = x.stud\_email

};

stdlist.Add(tmp);

}

StudDG.ItemsSource = stdlist;

}

public void ExportToExcel()

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

excel.Visible = true;

excel.Interactive = false;

excel.ScreenUpdating = false;

excel.UserControl = false;

excel.DisplayAlerts = false;

for (int j = 2; j < StudDG.Columns.Count + 1; j++) //Столбцы

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[1, j + 2];

sheet1.Columns[j].NumberFormat = "@";

myRange.Value2 = StudDG.Columns[j - 1].Header;

myRange.Font.Name = "Times New Roman";

myRange.Cells.Font.Size = 16;

myRange.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.Style.WrapText = false;

myRange.Columns.ColumnWidth = 200;

}

Microsoft.Office.Interop.Excel.Range myRang2 = sheet1.get\_Range("A1", "C1");

myRang2.Value = "ФИО";

myRang2.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.Merge(Type.Missing);

myRang2.Font.Name = "Times New Roman";

myRang2.Font.Bold = true;

myRang2.Cells.Font.Size = 16;

sheet1.Columns.AutoFit();

sheet1.Rows.AutoFit();

excel.Interactive = true;

excel.ScreenUpdating = true;

excel.UserControl = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

ExportToExcel();

}

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

{

AddStudBook a = new AddStudBook();

a.Show();

Close();

}

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

{

WindowState = WindowState.Minimized;

}

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

{

MainWindow a = new MainWindow();

a.Show();

Close();

}

private void StudDG\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

StudList list = null;

list = StudDG.SelectedItem as StudList;

int b = list.Stud\_id;

StudEditCheck.StudIDCheck = b;

EditStudBook a = new EditStudBook();

a.Show();

Close();

}

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

{

MainWindow main = new MainWindow();

main.Show();

Close();

}

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

{

var stud = StudDG.SelectedItem as StudList;

var b = db.Students.Where(q => q.ID\_stud == stud.Stud\_id).FirstOrDefault();

MessageBoxResult ad = MessageBox.Show("Вы уверены в удалении студента?", "StudCheck", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

db.Students.Remove(b);

db.SaveChanges();

MessageBox.Show("Успешно", "StudCheck");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

}

}

GlavPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using StudentGroups.Class;

namespace StudentGroups.Windows.MainPages

{

/// <summary>

/// Логика взаимодействия для GlavPage.xaml

/// </summary>

public partial class GlavPage : Page

{

public GlavPage()

{

InitializeComponent();

}

}

}

GradePage.xaml.cs

using StudentGroups.Class;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace StudentGroups.Windows.MainPages

{

/// <summary>

/// Логика взаимодействия для JournalPage.xaml

/// </summary>

public partial class JournalPage : Page

{

Data.StudCheckDBEntities db = new Data.StudCheckDBEntities();

List<GradeClass> grClass = new List<GradeClass>();

public JournalPage()

{

InitializeComponent();

//Load();

}

private void Load()

{

var Tutorcheck = db.Tutors.Where(s => s.ID\_user == AuthRole.RoleChooser).FirstOrDefault();

GradeDG.ItemsSource = null;

foreach (var x in db.Groups.ToList())

{

GradeClass tmp = new GradeClass

{

Id\_group = x.ID\_group,

Group\_Name = x.group\_name,

Tutor\_FIO = db.Tutors.Where(q => q.ID\_group == x.ID\_group).FirstOrDefault().tutor\_family + " " + db.Tutors.Where(q => q.ID\_group == x.ID\_group).FirstOrDefault().tutor\_name + " " + db.Tutors.Where(q => q.ID\_group == x.ID\_group).FirstOrDefault().tutor\_otche,

Stud\_number = db.Students.Where(q => q.ID\_group == x.ID\_group).Count().ToString(),

Average\_grade = "4"

};

grClass.Add(tmp);

GradeDG.ItemsSource = grClass;

}

}

public void ExportToExcel()

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

excel.Visible = true;

excel.Interactive = false;

excel.ScreenUpdating = false;

excel.UserControl = false;

excel.DisplayAlerts = false;

for (int j = 2; j < GradeDG.Columns.Count + 1; j++) //Столбцы

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[1, j + 2];

sheet1.Columns[j].NumberFormat = "@";

myRange.Value2 = GradeDG.Columns[j - 1].Header;

myRange.Font.Name = "Times New Roman";

myRange.Cells.Font.Size = 16;

myRange.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.Style.WrapText = false;

myRange.Columns.ColumnWidth = 200;

}

Microsoft.Office.Interop.Excel.Range myRang2 = sheet1.get\_Range("A1", "C1");

myRang2.Value = "ФИО";

myRang2.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.Merge(Type.Missing);

myRang2.Font.Name = "Times New Roman";

myRang2.Font.Bold = true;

myRang2.Cells.Font.Size = 16;

sheet1.Columns.AutoFit();

sheet1.Rows.AutoFit();

excel.Interactive = true;

excel.ScreenUpdating = true;

excel.UserControl = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

ExportToExcel();

}

}

}

MainWindow.xaml.cs

using StudentGroups.Class;

using StudentGroups.Windows.MainPages;

using StudentGroups.Windows.MainPages.StudentsWind;

using System.Windows;

using System.Windows.Input;

namespace StudentGroups.Windows.MainPages

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

/\* if (Author.RoleChooser == 2)

{

resultWork.Visibility = Visibility.Collapsed;

}

else if (Author.RoleChooser == 3)

{

StudentBt.Visibility = Visibility.Collapsed;

}\*/

}

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

{

StudentsWind.StudHub a = new StudentsWind.StudHub();

a.Show();

Close();

}

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

{

Close();

}

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

{

Help a = new Help();

a.Show();

Close();

}

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

{

Spravka a = new Spravka();

a.Show();

Close();

}

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

{

}

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

{

AcadHub a = new AcadHub();

a.Show();

Close();

}

private void SlideToggle\_Checked(object sender, RoutedEventArgs e)

{

TutorPage tutor = new TutorPage();

StudPage stud = new StudPage();

JournalPage journal = new JournalPage();

tutor.TutorSP.Visibility = Visibility.Visible;

stud.StudSP.Visibility = Visibility.Visible;

journal.GradeSP.Visibility = Visibility.Visible;

}

private void SlideToggle\_Unchecked(object sender, RoutedEventArgs e)

{

TutorPage tutor = new TutorPage();

StudPage stud = new StudPage();

JournalPage journal = new JournalPage();

tutor.TutorSP.Visibility = Visibility.Collapsed;

stud.StudSP.Visibility = Visibility.Collapsed;

journal.GradeSP.Visibility = Visibility.Collapsed;

}

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

{

Authorize a = new Authorize();

a.Show();

Close();

}

}

}

StudPage.xaml.cs

using StudentGroups.Data;

using System;

using System.Collections.Generic;

using System.Data;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Microsoft.Office.Interop.Excel;

using excel = Microsoft.Office.Interop.Excel;

using TextBox = System.Windows.Controls.TextBox;

using Window = System.Windows.Window;

using StudentGroups.Class;

using Page = System.Windows.Controls.Page;

using StudentGroups.Class.BaseClass;

namespace StudentGroups.Windows.MainPages

{

/// <summary>

/// Логика взаимодействия для StudPage.xaml

/// </summary>

public partial class StudPage : Page

{

Data.StudCheckDBEntities db = new Data.StudCheckDBEntities();

List<StudList> stdlist = new List<StudList>();

public StudPage()

{

InitializeComponent();

Load();

}

private void Load()

{

if (AuthRole.RoleChooser != 1)

{

var Tutorcheck = db.Tutors.Where(s => s.ID\_user == AuthRole.RoleChooser).FirstOrDefault();

StudDG.ItemsSource = null;

foreach (var x in db.Students.Where(q => q.ID\_group == Tutorcheck.ID\_group))

{

StudList tmp = new StudList

{

Stud\_id = x.ID\_stud,

stud\_Family = x.stud\_family,

stud\_Name = x.stud\_name,

stud\_Otche = x.stud\_otche,

stud\_group = db.Groups.Where(q => q.ID\_group == x.ID\_group).Single().group\_name,

};

stdlist.Add(tmp);

}

StudDG.ItemsSource = stdlist;

}

if (AuthRole.RoleChooser == 1)

{

StudDG.ItemsSource = null;

foreach (var x in db.Students.ToList())

{

StudList tmp = new StudList

{

Stud\_id = x.ID\_stud,

stud\_Family = x.stud\_family,

stud\_Name = x.stud\_name,

stud\_Otche = x.stud\_otche,

stud\_group = db.Groups.Where(q => q.ID\_group == x.ID\_group).Single().group\_name,

};

stdlist.Add(tmp);

}

StudDG.ItemsSource = stdlist;

}

else

{

}

}

private void SearchBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var SearchBox = sender as TextBox;

if (SearchVar.Text == "ID cneltynf")

{

if (SearchBox.Text != "")

{

var filList = stdlist.Where(x => x.Stud\_id.ToString().Contains(SearchBox.Text)).ToList();

StudDG.ItemsSource = null;

StudDG.ItemsSource = filList;

}

else

{

StudDG.ItemsSource = stdlist;

}

}

else if (SearchVar.Text == "Фамилия")

{

if (SearchBox.Text != "")

{

var filList = stdlist.Where(x => x.stud\_Family.ToString().Contains(SearchBox.Text)).ToList();

StudDG.ItemsSource = null;

StudDG.ItemsSource = filList;

}

else

{

StudDG.ItemsSource = stdlist;

}

}

else if (SearchVar.Text == "Группа")

{

if (SearchBox.Text != "")

{

var filList = stdlist.Where(x => x.stud\_group.ToString().Contains(SearchBox.Text)).ToList();

StudDG.ItemsSource = null;

StudDG.ItemsSource = filList;

}

else

{

StudDG.ItemsSource = stdlist;

}

}

}

public void ExportToExcel()

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

excel.Visible = true;

excel.Interactive = false;

excel.ScreenUpdating = false;

excel.UserControl = false;

excel.DisplayAlerts = false;

for (int j = 2; j < StudDG.Columns.Count + 1; j++) //Столбцы

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[1, j + 2];

sheet1.Columns[j].NumberFormat = "@";

myRange.Value2 = StudDG.Columns[j - 1].Header;

myRange.Font.Name = "Times New Roman";

myRange.Cells.Font.Size = 16;

myRange.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.Style.WrapText = false;

myRange.Columns.ColumnWidth = 200;

}

Microsoft.Office.Interop.Excel.Range myRang2 = sheet1.get\_Range("A1", "C1");

myRang2.Value = "ФИО";

myRang2.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.Merge(Type.Missing);

myRang2.Font.Name = "Times New Roman";

myRang2.Font.Bold = true;

myRang2.Cells.Font.Size = 16;

sheet1.Columns.AutoFit();

sheet1.Rows.AutoFit();

excel.Interactive = true;

excel.ScreenUpdating = true;

excel.UserControl = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

ExportToExcel();

}

}

}

TutorPage.xaml.cs

using StudentGroups.Class;

using StudentGroups.Class.BaseClass;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace StudentGroups.Windows.MainPages

{

/// <summary>

/// Логика взаимодействия для TutorPage.xaml

/// </summary>

public partial class TutorPage : Page

{

Data.StudCheckDBEntities db = new Data.StudCheckDBEntities();

List<TutorStud> ttrlist = new List<TutorStud>();

public TutorPage()

{

InitializeComponent();

Load();

}

public void Load()

{

TutorDG.ItemsSource = null;

foreach (var x in db.Tutors.ToList())

{

TutorStud tmp = new TutorStud()

{

tutorID = x.ID\_tutor,

tutorFIO = x.tutor\_name + " " + x.tutor\_family + " " + x.tutor\_otche,

tutorTel = x.tutor\_tel,

tutorEmail = x.tutor\_email,

tutorGroup = db.Groups.Where(q => q.ID\_group == x.ID\_group).Single().group\_name

};

ttrlist.Add(tmp);

TutorDG.ItemsSource = ttrlist;

}

}

private void SearchBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var SearchBox = sender as TextBox;

if (SearchVar.Text == "ID куратора")

{

if (SearchBox.Text != "")

{

var filList = ttrlist.Where(x => x.tutorID.ToString().Contains(SearchBox.Text)).ToList();

TutorDG.ItemsSource = null;

TutorDG.ItemsSource = filList;

}

else

{

TutorDG.ItemsSource = ttrlist;

}

}

else if (SearchVar.Text == "Фамилия")

{

if (SearchBox.Text != "")

{

var filList = ttrlist.Where(x => x.tutorFIO.ToString().Contains(SearchBox.Text)).ToList();

TutorDG.ItemsSource = null;

TutorDG.ItemsSource = filList;

}

else

{

TutorDG.ItemsSource = ttrlist;

}

}

else if (SearchVar.Text == "Группа")

{

if (SearchBox.Text != "")

{

var filList = ttrlist.Where(x => x.tutorGroup.ToString().Contains(SearchBox.Text)).ToList();

TutorDG.ItemsSource = null;

TutorDG.ItemsSource = filList;

}

else

{

TutorDG.ItemsSource = ttrlist;

}

}

else if (SearchVar.Text == "Телефон")

{

if (SearchBox.Text != "")

{

var filList = ttrlist.Where(x => x.tutorTel.ToString().Contains(SearchBox.Text)).ToList();

TutorDG.ItemsSource = null;

TutorDG.ItemsSource = filList;

}

else

{

TutorDG.ItemsSource = ttrlist;

}

}

else if (SearchVar.Text == "Email")

{

if (SearchBox.Text != "")

{

var filList = ttrlist.Where(x => x.tutorEmail.ToString().Contains(SearchBox.Text)).ToList();

TutorDG.ItemsSource = null;

TutorDG.ItemsSource = filList;

}

else

{

TutorDG.ItemsSource = ttrlist;

}

}

}

public void ExportToExcel()

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

excel.Visible = true;

excel.Interactive = false;

excel.ScreenUpdating = false;

excel.UserControl = false;

excel.DisplayAlerts = false;

for (int j = 2; j < TutorDG.Columns.Count + 1; j++) //Столбцы

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[1, j + 2];

sheet1.Columns[j].NumberFormat = "@";

myRange.Value2 = TutorDG.Columns[j - 1].Header;

myRange.Font.Name = "Times New Roman";

myRange.Cells.Font.Size = 16;

myRange.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRange.Style.WrapText = false;

myRange.Columns.ColumnWidth = 200;

}

Microsoft.Office.Interop.Excel.Range myRang2 = sheet1.get\_Range("A1", "C1");

myRang2.Value = "ФИО";

myRang2.VerticalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.HorizontalAlignment = Microsoft.Office.Interop.Excel.XlVAlign.xlVAlignCenter;

myRang2.Merge(Type.Missing);

myRang2.Font.Name = "Times New Roman";

myRang2.Font.Bold = true;

myRang2.Cells.Font.Size = 16;

sheet1.Columns.AutoFit();

sheet1.Rows.AutoFit();

excel.Interactive = true;

excel.ScreenUpdating = true;

excel.UserControl = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

ExportToExcel();

}

}

}

Authorize.xaml.cs

using StudentGroups.Class;

using System;

using System.Linq;

using System.Windows;

using System.Windows.Input;

namespace StudentGroups.Windows

{

/// <summary>

/// Логика взаимодействия для Authorize.xaml

/// </summary>

public partial class Authorize : Window

{

//StudCheckDataSet db = new StudCheckDataSet();

Data.StudCheckDBEntities db = new Data.StudCheckDBEntities();

public Authorize()

{

InitializeComponent();

}

private void Window\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed)

DragMove();

}

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

{

WindowState = WindowState.Minimized;

}

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

{

Close();

}

public void AuthorBt\_Click(object sender, RoutedEventArgs e)

{

try

{

var AuthPerson = db.Authorize.FirstOrDefault(x => x.user\_login == logBox.Text && x.user\_pass == passBox.Password);

if (AuthPerson == null)

{

MessageBox.Show("Такого пользователя нет!", "Ошибка при авторизации", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

switch (AuthPerson.ID\_role)

{

case 1:

MessageBox.Show("Здравствуйте, Администратор " + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

break;

default:

MessageBox.Show("Здравствуйте, Куратор " + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

break;

}

AuthRole.RoleChooser = AuthPerson.ID\_user;

MainPages.MainWindow a = new MainPages.MainWindow();

a.Show();

Close();

}

}

catch (Exception Ex)

{

MessageBox.Show("Ошибка!!! " + Ex.Message.ToString() + " Критическая работа приложения!!!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

Help.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using StudentGroups.Class;

namespace StudentGroups.Windows

{

/// <summary>

/// Логика взаимодействия для Help.xaml

/// </summary>

public partial class Help : Window

{

public Help()

{

InitializeComponent();

}

private void stud\_icon\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainPages.MainWindow a = new MainPages.MainWindow();

a.Show();

Close();

}

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

{

var subj = "STUDCHECK: " + themeCB.Text;

MessageBoxResult ad = MessageBox.Show("Вы подтверждаете отправку сообщения?", "STUDCHECK", MessageBoxButton.YesNo);

switch (ad)

{

case MessageBoxResult.Yes:

MailService ms = new MailService();

ms.Send("programmkst@yandex.ru", subj, LetterBox.Text);

MessageBox.Show("Успешно", " StudCheck ");

break;

case MessageBoxResult.No:

MessageBox.Show("Отмена", "StudCheck");

break;

}

}

}

}

Spravka.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace StudentGroups.Windows

{

/// <summary>

/// Логика взаимодействия для Spravka.xaml

/// </summary>

public partial class Spravka : Window

{

public Spravka()

{

InitializeComponent();

}

private void Image\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainPages.MainWindow a = new MainPages.MainWindow();

a.Show();

Close();

}

}

}

Start.xaml.cs

using StudentGroups.Class;

using System.Windows;

namespace StudentGroups.Windows

{

/// <summary>

/// Логика взаимодействия для Start.xaml

/// </summary>

public partial class Start : Window

{

public Start()

{

InitializeComponent();

AuthRole.RoleChooser = 0;

}

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

{

Windows.Authorize a = new Windows.Authorize();

a.Show();

Close();

}

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

{

WindowState = WindowState.Minimized;

}

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

{

Close();

}

}

}