Федеральное государственное образовательное бюджетное учреждение  
высшего образования

«Финансовый университет при Правительстве Российской Федерации»

КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

**ОТЧЕТ**

**по результатам проектирования**

**и разработки программного приложения**

**по профессиональному модулю ПМ.03 Участие в интеграции программных модулей**

**специальности 09.02.03 Программирование в компьютерных системах**

Выполнил:

студент группы 4ПКС-220

Мамедов Руслан Эльшанович

Проверили:

преподаватели

\_\_\_\_\_\_\_\_ / Т.Г. Аксёнова /

\_\_\_\_\_\_\_\_/О.А. Тимакин/

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2023

**Содержание**

[Проектирование программного приложения 3](#_Toc154589727)

[Руководство оператора 4](#_Toc154589728)

[Назначение программы 4](#_Toc154589729)

[Условия выполнения программы 4](#_Toc154589730)

[Выполнение программы 4](#_Toc154589731)

[Сообщения оператору 6](#_Toc154589732)

[Листинг программных модулей 8](#_Toc154589733)

# Проектирование программного приложения

На рис. 1 представлена диаграмма вариантов использования.

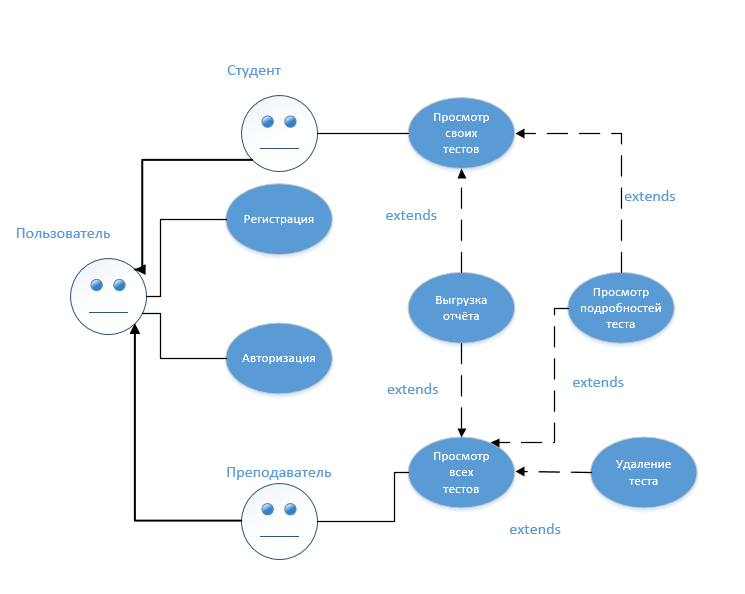


Рисунок 1. Диаграмма вариантов использования

На рис. 2 представлена ER-диаграмма.

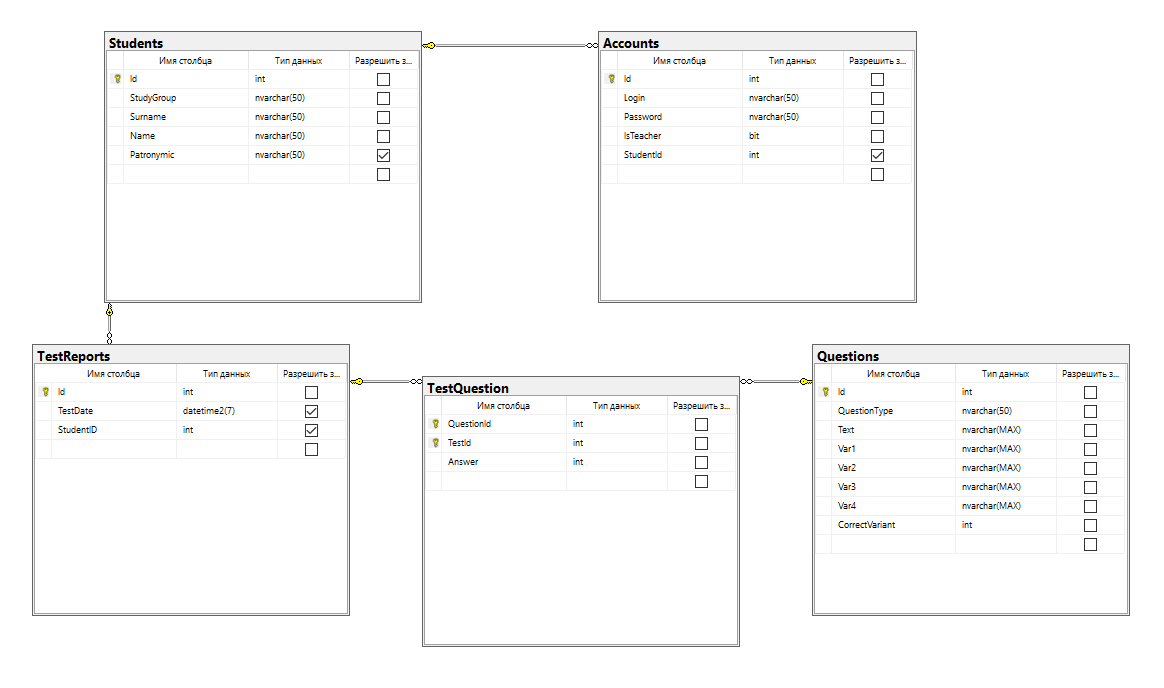


Рисунок 2. ER-диаграмма

# Руководство оператора

## Назначение программы

Программа предназначена для просмотра результатов тестирования студентов. В приложении предусмотрен подробный просмотр каждого тестирования. Присутствует разделение на роли «Преподаватель» и «Студент», преподаватель может просматривать все записи о тестированиях, а также удалять их, студент может просматривать только свои результаты.

Вход в приложение осуществляется через логин и пароль. При регистрации предлагается выбор роли аккаунта, для аккаунта студента необходимо заполнить личную информацию.

## Условия выполнения программы

Для выполнения программы необходимо иметь программное средство .NET Framework и аппаратные средства, соответствующие минимальным системным требованиям операционной системы Windows 10.

## Выполнение программы

При запуске программы оператору необходимо авторизоваться, введя логин и пароль (рис. 3).

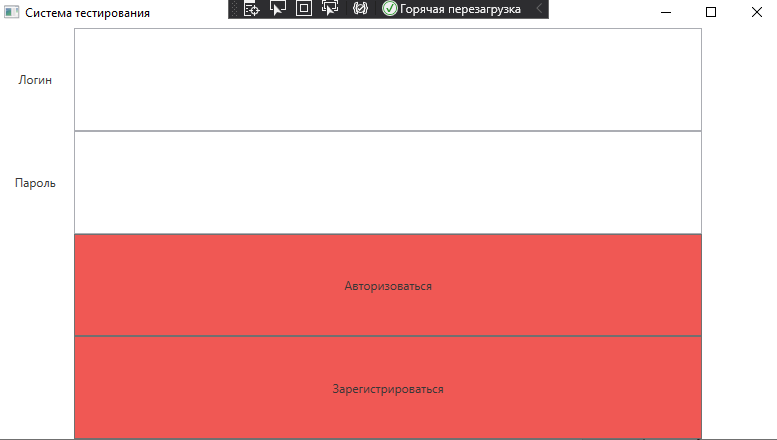


Рисунок 3. Главное меню программы

В случае отсутствия аккаунта необходимо зарегистрироваться (рис. 4).



Рисунок 4. Окно регистрации

Преподаватель видит все результаты тестов (рис. 5).

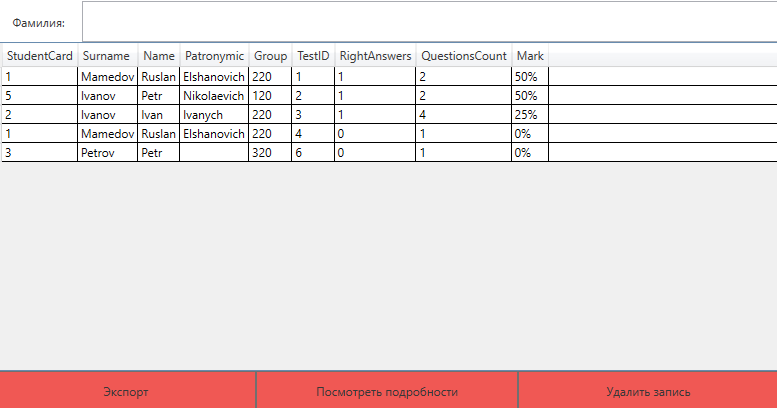


Рисунок 5. Окно преподавателя

Преподаватель может сортировать таблицу по столбцу, для этого необходимо по нему кликнуть (рис. 6).

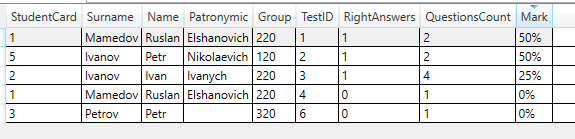


Рисунок 6. Сортировка таблицы по столбцу

Присутствует возможность фильтрации по фамилии студента, для этого необходимо ввести её часть в строку над таблицей (рис. 7).

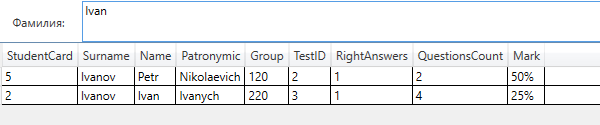


Рисунок 7. Фильтрация данных

Нажав на кнопку «Экспорт» оператор получает документ, со всеми данными из таблицы (рис. 8).

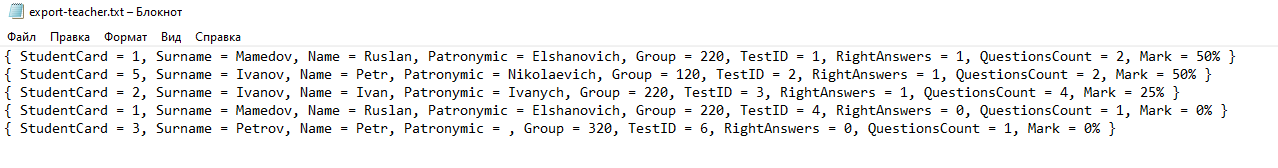


Рисунок 8. Документ с экспортированными данными

При нажатии на кнопку «Посмотреть подробности» откроется список вопросов в тесте, который покажет все варианты ответа для каждого вопроса, правильные и данные пользователем ответы (рис. 9).

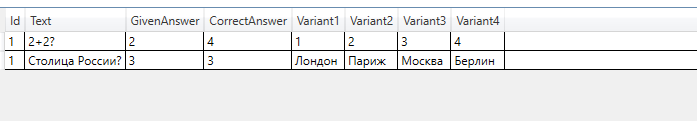


Рисунок 9. Подробная информация о тесте

Кнопка «Удалить запись» сотрёт информацию о тесте в базе данных.

Возможности оператора с ролью «Студент» отличаются в следующих пунктах:

* студент видит только свои тесты;
* отсутствует возможность удаления записей;
* фильтрация происходит по id теста.

## Сообщения оператору

При авторизации оператору может быть выдано сообщение «Неправильный логин или пароль», это означает что в базе данных нет аккаунта с таким логином и паролем.

При регистрации аккаунту оператору могут выданы сообщение «Поле должно быть заполнено», где слово «Поле» будет заменено на то, что необходимо заполнить (например, логин или пароль).

Также при регистрации может быть выдано сообщение «Логин занят», которое означает, что аккаунт с таким логином уже существует в базе данных.

При завершении экспорта данных будет отправлено сообщение Экспорт завершён.

# Листинг программных модулей

MainWindow.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;

namespace \_220\_Mamedov

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

new Registration().Show();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

var ctx = ExamEntities.GetContext();

if(!ctx.Accounts.Any(x => x.Login == LoginInput.Text && x.Password == PasswordInput.Password))

{

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

return;

}

var account = ctx.Accounts.Where(x => x.Login == LoginInput.Text && x.Password == PasswordInput.Password).First();

Application.Current.Properties["account\_id"] = account.Id;

if (account.IsTeacher)

{

new TeacherWindow().Show();

}

else

{

new StudentWindow().Show();

}

this.Close();

}

}

}

Registration.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 \_220\_Mamedov

{

/// <summary>

/// Логика взаимодействия для Registration.xaml

/// </summary>

public partial class Registration : Window

{

public Registration()

{

InitializeComponent();

}

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

{

var ctx = ExamEntities.GetContext();

if (string.IsNullOrWhiteSpace(LoginInput.Text))

{

MessageBox.Show("Логин должен быть заполнен");

return;

}

if (string.IsNullOrWhiteSpace(PasswordInput.Password))

{

MessageBox.Show("Пароль должен быть заполнен");

return;

}

if (ctx.Accounts.Any(x => x.Login == LoginInput.Text))

{

MessageBox.Show("Логин занят");

return;

}

if (!AccChecker.IsChecked.Value)

{

if (string.IsNullOrWhiteSpace(GroupInput.Text))

{

MessageBox.Show("Группа должна быть заполнена");

return;

}

if (string.IsNullOrWhiteSpace(SurnameInput.Text))

{

MessageBox.Show("Фамилия должна быть заполнена");

return;

}

if (string.IsNullOrWhiteSpace(NameInput.Text))

{

MessageBox.Show("Имя должно быть заполнено");

return;

}

}

var account = new Accounts();

account.Login = LoginInput.Text;

account.Password = PasswordInput.Password;

account.IsTeacher = AccChecker.IsChecked.Value;

ctx.Accounts.Add(account);

ctx.SaveChanges();

if (!AccChecker.IsChecked.Value)

{

var student = new Students();

student.Name = NameInput.Text;

student.Surname = SurnameInput.Text;

student.Patronymic = PatronymicInput.Text;

student.StudyGroup = GroupInput.Text;

ctx.Students.Add(student);

ctx.SaveChanges();

ctx.Accounts.ToList().Last().StudentId = ctx.Students.ToList().Last().Id;

ctx.SaveChanges();

}

MessageBox.Show("Регистрация успешна!");

this.Close();

}

private void AccChecker\_Clicked(object sender, RoutedEventArgs e)

{

if (AccChecker.IsChecked.Value)

{

PatronymicLabel.Visibility = Visibility.Hidden;

PatronymicInput.Visibility = Visibility.Hidden;

SurnameInput.Visibility = Visibility.Hidden;

SurnameLabel.Visibility = Visibility.Hidden;

NameLabel.Visibility = Visibility.Hidden;

NameInput.Visibility = Visibility.Hidden;

GroupInput.Visibility = Visibility.Hidden;

GroupLabel.Visibility = Visibility.Hidden;

}

else

{

PatronymicLabel.Visibility = Visibility.Visible;

PatronymicInput.Visibility = Visibility.Visible;

SurnameInput.Visibility = Visibility.Visible;

SurnameLabel.Visibility = Visibility.Visible;

NameLabel.Visibility = Visibility.Visible;

NameInput.Visibility = Visibility.Visible;

GroupInput.Visibility = Visibility.Visible;

GroupLabel.Visibility = Visibility.Visible;

}

}

}

}

StudentWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

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 \_220\_Mamedov

{

/// <summary>

/// Логика взаимодействия для StudentWindow.xaml

/// </summary>

public partial class StudentWindow : Window

{

private int \_id;

public StudentWindow()

{

InitializeComponent();

var context = ExamEntities.GetContext();

var account\_id = ((int)Application.Current.Properties["account\_id"]);

\_id = context.Accounts.Where(x => x.Id == account\_id).First().StudentId.Value;

var table = from rep in context.TestReports

join qr in context.TestQuestion on rep.Id equals qr.TestId

where \_id == rep.StudentID

group rep by new { rep.Id, rep.StudentID } into grep

select new

{

TestID = grep.Key.Id,

TestDate = context.TestReports.Where(x => x.Id == grep.Key.Id).FirstOrDefault().TestDate,

RightAnswers = context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count(),

QuestionsCount = grep.Count(),

Mark = (context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count() / (float)grep.Count()) \* 100 + "%"

};

Table.ItemsSource = table.ToList();

}

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

{

using (var file = new StreamWriter($"export-student.txt", false))

{

StringBuilder sb = new StringBuilder();

foreach (var item in Table.Items)

{

sb.AppendLine(item.ToString());

}

file.Write(sb.ToString());

}

MessageBox.Show("Экспорт завершён");

}

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

{

var context = ExamEntities.GetContext();

var table = from rep in context.TestReports

join qr in context.TestQuestion on rep.Id equals qr.TestId

where \_id == rep.StudentID

group rep by new { rep.Id, rep.StudentID } into grep

select new

{

TestID = grep.Key.Id,

TestDate = context.TestReports.Where(x => x.Id == grep.Key.Id).FirstOrDefault().TestDate,

RightAnswers = context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count(),

QuestionsCount = grep.Count(),

Mark = (context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count() / (float)grep.Count()) \* 100 + "%"

};

if (!string.IsNullOrWhiteSpace(SearchBox.Text))

{

table = table.Where(x => x.TestID.ToString().StartsWith(SearchBox.Text));

}

Table.ItemsSource = table.ToList();

}

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

{

if (Table.SelectedValue != null)

{

var id = (int)Table.SelectedValue.GetType().GetProperty("TestID").GetValue(Table.SelectedValue, null);

new TestInfo(id).Show();

}

}

}

}

TeacherWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

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 \_220\_Mamedov

{

/// <summary>

/// Логика взаимодействия для TeacherWindow.xaml

/// </summary>

public partial class TeacherWindow : Window

{

public TeacherWindow()

{

InitializeComponent();

var context = ExamEntities.GetContext();

var table = from rep in context.TestReports

join qr in context.TestQuestion on rep.Id equals qr.TestId

group rep by new { rep.Id, rep.StudentID } into grep

select new

{

StudentCard = grep.Key.StudentID,

Surname = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Surname,

Name = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Name,

Patronymic = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Patronymic,

Group = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().StudyGroup,

TestID = grep.Key.Id,

RightAnswers = context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count(),

QuestionsCount = grep.Count(),

Mark = (context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count()/(float)grep.Count()) \* 100 + "%"

};

Table.ItemsSource = table.ToList();

}

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

{

using (var file = new StreamWriter($"export-teacher.txt", false))

{

StringBuilder sb = new StringBuilder();

foreach (var item in Table.Items)

{

sb.AppendLine(item.ToString());

}

file.Write(sb.ToString());

}

MessageBox.Show("Экспорт завершён");

}

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

{

var context = ExamEntities.GetContext();

var table = from rep in context.TestReports

join qr in context.TestQuestion on rep.Id equals qr.TestId

group rep by new { rep.Id, rep.StudentID } into grep

select new

{

StudentCard = grep.Key.StudentID,

Surname = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Surname,

Name = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Name,

Patronymic = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Patronymic,

Group = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().StudyGroup,

TestID = grep.Key.Id,

RightAnswers = context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count(),

QuestionsCount = grep.Count(),

Mark = (context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count() / (float)grep.Count()) \* 100 + "%"

};

if (!string.IsNullOrWhiteSpace(SearchBox.Text))

{

table = table.Where(x => x.Surname.ToString().StartsWith(SearchBox.Text));

}

Table.ItemsSource = table.ToList();

}

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

{

if (Table.SelectedValue != null)

{

var id = (int)Table.SelectedValue.GetType().GetProperty("TestID").GetValue(Table.SelectedValue, null);

new TestInfo(id).Show();

}

}

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

{

var context = ExamEntities.GetContext();

if (Table.SelectedValue != null)

{

var id = (int)Table.SelectedValue.GetType().GetProperty("TestID").GetValue(Table.SelectedValue, null);

var entry = context.TestReports.Where(x => x.Id == id).First();

context.TestReports.Remove(entry);

context.SaveChanges();

}

var table = from rep in context.TestReports

join qr in context.TestQuestion on rep.Id equals qr.TestId

group rep by new { rep.Id, rep.StudentID } into grep

select new

{

StudentCard = grep.Key.StudentID,

Surname = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Surname,

Name = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Name,

Patronymic = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().Patronymic,

Group = context.Students.Where(x => x.Id == grep.Key.StudentID).FirstOrDefault().StudyGroup,

TestID = grep.Key.Id,

RightAnswers = context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count(),

QuestionsCount = grep.Count(),

Mark = (context.TestQuestion.Where(x => x.TestId == grep.Key.Id)

.Where(x => x.Answer == context.Questions.Where(q => q.Id == x.QuestionId).FirstOrDefault().CorrectVariant).Count() / (float)grep.Count()) \* 100 + "%"

};

Table.ItemsSource = table.ToList();

}

}

}

TestInfo.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 \_220\_Mamedov

{

/// <summary>

/// Логика взаимодействия для TestInfo.xaml

/// </summary>

public partial class TestInfo : Window

{

private int \_id;

public TestInfo(int id)

{

\_id = id;

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

var context = ExamEntities.GetContext();

var table = from rep in context.TestReports

where rep.Id == \_id

join qr in context.TestQuestion on rep.Id equals qr.TestId

join q in context.Questions on qr.QuestionId equals q.Id

select new

{

rep.Id,

q.Text,

GivenAnswer = qr.Answer,

CorrectAnswer = q.CorrectVariant,

Variant1 = q.Var1,

Variant2 = q.Var2,

Variant3 = q.Var3,

Variant4 = q.Var4

};

QuestionsGrid.ItemsSource = table.ToList();

}

}

}