Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных Технологий и Анализа Данных

Кафедра вычислительной техники

**Название работы** – “Клиент-серверное приложение”

Отчет по лабораторной работе “Лабораторная работа №2”

по дисциплине Управление данными

Вариант 4

Выполнил

Студент, номер группы ИСМб-19-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Д.Солопов

Дата: \_\_\_\_\_\_\_\_\_\_

Принял

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А.Харахинов

Дата: \_\_\_\_\_\_\_\_\_\_

Иркутск 2021 г.

Содержание

[Индивидуальное задание 3](#_Toc68532154)

[Серверная часть приложения 4](#_Toc68532155)

[Клиентская часть приложения 9](#_Toc68532156)

Индивидуальное задание

**Задание**: Разработать клиент-серверное приложение, которое реализует решение задачи индивидуального варианта.

**Вариант 4. Библиотека**

Имеются книги (регистрационный номер, количество страниц, год издания, раздел - учебник, художественная общественно-политическая и т.д.) и читатели (ФИО, домашний адрес, паспортные данные).

Необходимо регистрировать дату, когда какой - либо читатель берет или возвращает книгу.

Выходные документы:

1. список читателей, которые держат на руках книги более месяца, с перечислением названия книги и даты выдачи, упорядоченный по датам выдачи, с указанием количества книг, которые должен сдать каждый читатель;

2. для заданного читателя выдать список прочитанных им книг, сортируя по датам получения.

Серверная часть приложения

Для реализации серверной части приложения был использован MS SQL Server.

Код создания таблиц (SQL):

CREATE TABLE [dbo].[Book] (

[Register\_Number] BIGINT NOT NULL,

[Count\_Pages] SMALLINT NULL,

[Year\_Publishing] SMALLINT NULL,

[Section] NVARCHAR (150) NULL,

PRIMARY KEY CLUSTERED ([Register\_Number] ASC)

);

CREATE TABLE [dbo].[Reader] (

[Password\_Data] BIGINT NOT NULL,

[Home\_Address] NVARCHAR (80) NULL,

[Full\_Name] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Password\_Data] ASC)

);

CREATE TABLE [dbo].[RecordRegistration] (

[ID] INT NOT NULL,

[Book\_Register\_Number] BIGINT NOT NULL,

[Reader\_Password\_Data] BIGINT NOT NULL,

[Date\_Issue] DATE NULL,

[Date\_Return] DATE NULL,

PRIMARY KEY CLUSTERED ([ID] ASC),

FOREIGN KEY ([Book\_Register\_Number]) REFERENCES [dbo].[Book] ([Register\_Number]),

FOREIGN KEY ([Reader\_Password\_Data]) REFERENCES [dbo].[Reader] ([Password\_Data])

);

Для наиболее эффективного взаимодействия с данными в базе данных на сервере были реализованы хранимые процедуры, осуществляющие поведение Insert, Update, Delete и Select. Будет представлены некоторые хранимые процедуры для базы данных.

Код хранимой процедуры для записи данных в таблицу регистрации (WriteRegisterData):

CREATE PROCEDURE [dbo].[WriteRegisterData](

@id bigint,

@reg bigint,

@pwd bigint,

@dateIssue date,

@dateReturn date

)AS

BEGIN

IF(EXISTS(SELECT \* FROM [dbo].[RecordRegistration] WHERE ID=@id))

BEGIN

RETURN (-4);

END;

IF(NOT EXISTS(SELECT \* FROM [dbo].[Book] WHERE Register\_Number=@reg))

BEGIN

RETURN (-2);

END;

IF(NOT EXISTS(SELECT \* FROM [dbo].[Reader] WHERE Password\_Data=@pwd))

BEGIN

RETURN (-1);

END;

IF((@dateIssue > @dateReturn) OR EXISTS(SELECT \* FROM [dbo].[Book]

WHERE ([dbo].[Book].Register\_Number=@reg)

AND ((YEAR(@dateIssue) < [dbo].[Book].Year\_Publishing)

OR (YEAR(@dateReturn) > YEAR(GETDATE())))))

BEGIN

RETURN (-3);

END;

INSERT INTO [dbo].[RecordRegistration] (ID, Book\_Register\_Number, Reader\_Password\_Data, Date\_Issue, Date\_Return)

VALUES (@id, @reg, @pwd, @dateIssue, @dateReturn);

RETURN 0;

END;

В хранимой процедуре WriteRegisterData реализована логика добавления записи в таблицу RecordRegistration (таблица регистрации) с проверкой данных на корректность, а также проверкой данных на временную корректность (дата выдачи не может быть раньше даты публикации книги).

Хранимая процедура для обновления данных в таблице Book (UpdateBookData):

CREATE PROCEDURE [dbo].[UpdateBookData](

@reg bigint,

@newPages smallint,

@newYear smallint,

@newSection nvarchar(150)

)

AS

BEGIN

IF(@newYear > YEAR(GETDATE()))

BEGIN

RETURN (-1);

END;

IF(EXISTS(SELECT [dbo].[RecordRegistration].Date\_Issue, [dbo].[RecordRegistration].Book\_Register\_Number FROM [dbo].[RecordRegistration]

WHERE ([dbo].[RecordRegistration].Book\_Register\_Number=@reg) AND YEAR([dbo].[RecordRegistration].Date\_Issue) < @newYear))

BEGIN

RETURN (-2);

END;

UPDATE Book

SET Count\_Pages=@newPages, Year\_Publishing=@newYear, Section=@newSection

WHERE Register\_Number=@reg;

RETURN 0;

END;

Хранимая процедура UpdateBookData вызывается при обновлении данных в таблице. Каждое изменение происходит с помощью процедуры UpdateBookData. Хранимая процедура реализует логику проверки данных на временную корректность (например, нельзя установить дату публикации раньше той даты, которая фактически была зарегистрированна в таблице RecordRecistration).

Хранимая процедура DeleteReaderData для удаления данных из таблицы Reader:

CREATE PROCEDURE [dbo].[DeleteReaderData](

@pwd bigint

)

AS

BEGIN

IF(EXISTS(SELECT \* FROM [dbo].[RecordRegistration] WHERE Reader\_Password\_Data=@pwd))

BEGIN

RETURN (-1);

END;

DELETE [dbo].[Reader]

WHERE Password\_Data=@pwd;

END;

В данной хранимой процедуре реализуется логика удаления записи из таблицы Reader с проверкой на факт регистрации записи с данными паспортными данными в таблице регистрации RecordRegistration.

Хранимая процедура ReadRegisterData для чтения данных из таблицы RecordRegistration:

CREATE PROCEDURE [dbo].[ReadRegisterData]

AS

SELECT \* FROM [dbo].[RegisterView];

RETURN 0

Представление для таблицы RecordRegistration:

CREATE VIEW [dbo].[RegisterView]

AS SELECT \* FROM [dbo].[RecordRegistration];

Хранимая процедура ReadRegisterData считывает данные из представления RegisterView, которая в свою очередь возвращает данные таблицы RecordRegistration

Хранимая процедура TaskOneT для решения задачи 1 из индивидуального варианта:

CREATE PROCEDURE [dbo].[TaskOneT]

@monthValue int

AS

BEGIN

SELECT [dbo].[Reader].Full\_Name, [dbo].[Reader].Password\_Data, [dbo].[RecordRegistration].Book\_Register\_Number, [dbo].[RecordRegistration].Date\_Issue, [dbo].[RecordRegistration].Date\_Return

FROM [dbo].[RecordRegistration]

JOIN [dbo].[Reader]

ON [dbo].[RecordRegistration].Reader\_Password\_Data=[dbo].[Reader].Password\_Data

JOIN [dbo].[Book]

ON [dbo].[Book].Register\_Number=[dbo].[RecordRegistration].Book\_Register\_Number

WHERE ((MONTH([dbo].[RecordRegistration].Date\_Return) - MONTH([dbo].[RecordRegistration].Date\_Issue)) > @monthValue)

OR ((MONTH([dbo].[RecordRegistration].Date\_Return) - MONTH([dbo].[RecordRegistration].Date\_Issue)) + 12\*(YEAR([dbo].[RecordRegistration].Date\_Return) - YEAR([dbo].[RecordRegistration].Date\_Issue)) > @monthValue);

END;

Данная хранимая процедура возвращает таблицу, содержащая данные соответствующие требованиям задачи, с использованием JOIN … ON и фильтрацией, необходимая для решения задачи.

Хранимая процедура TaskTwot для решения задачи 2 из индивидуального варианта:

CREATE PROCEDURE [dbo].[TaskTwot]

@pwd bigint

AS

SELECT [dbo].[RecordRegistration].Book\_Register\_Number, [dbo].[RecordRegistration].Date\_Issue, [dbo].[Book].Count\_Pages, [dbo].[Book].Section

FROM [dbo].[RecordRegistration], [dbo].[Book] WHERE

([dbo].[RecordRegistration].Reader\_Password\_Data=@pwd) AND ([dbo].[RecordRegistration].Book\_Register\_Number=[dbo].[Book].Register\_Number);

RETURN 0

Хранимая процедура DefineBookCounter для подсчёта количества книг, зарегистрированных под определённым читателем:

CREATE PROCEDURE [dbo].[DefineBookCounter](

@pwd bigint

)

AS

BEGIN

RETURN (SELECT COUNT(\*) FROM [dbo].[RecordRegistration] WHERE [dbo].[RecordRegistration].Reader\_Password\_Data=@pwd);

END;

Клиентская часть приложения

Для реализации клиентской части приложения был использован язык C# со средой разработки Visual Studio Community, с возможностью взаимодействия с базой данных.

Интерфейс приложения предоставляет возможность взаимодействия с тремя таблицами: таблицей читателей (Reader), таблицей книг (Book) и таблицей регистрации (RecordRegistration).

Для переключения между таблицами используется верхнее меню (с помощью элемента управления MenuStrip).

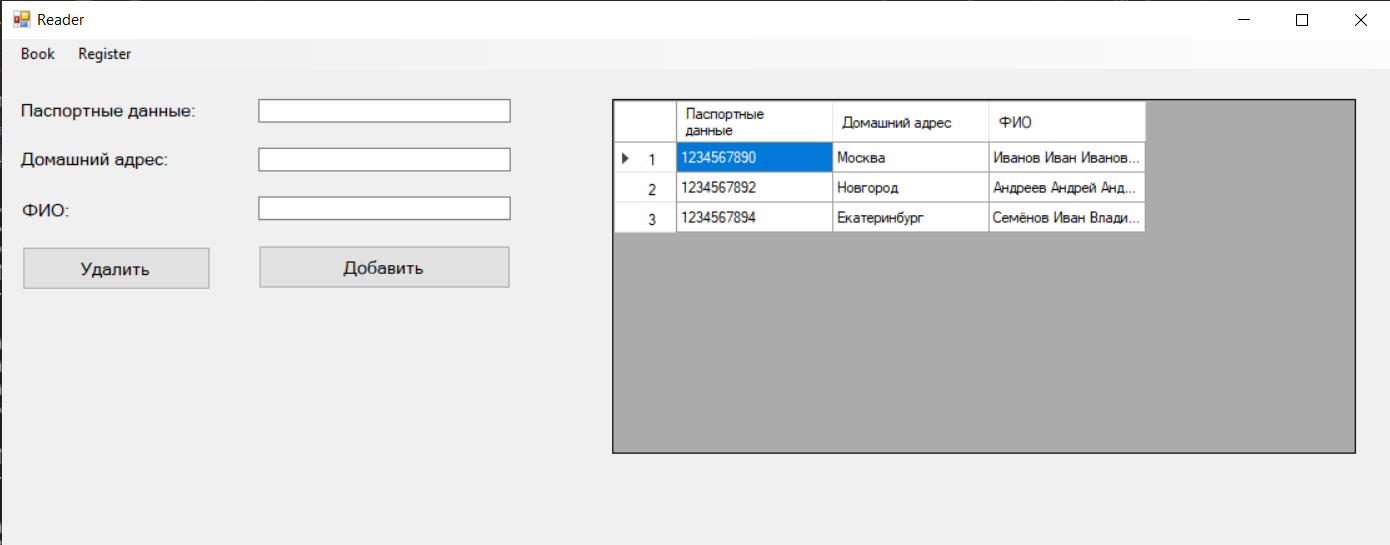


Рисунок 1 – Интерфейс для взаимодействия с таблицей Reader

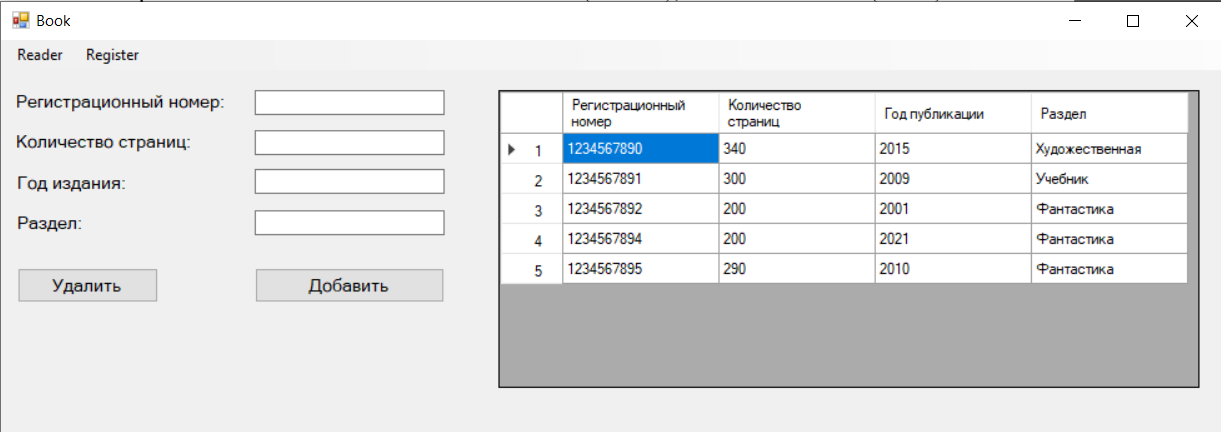


Рисунок 2 – Интерфейс для взаимодействия с таблицей Book

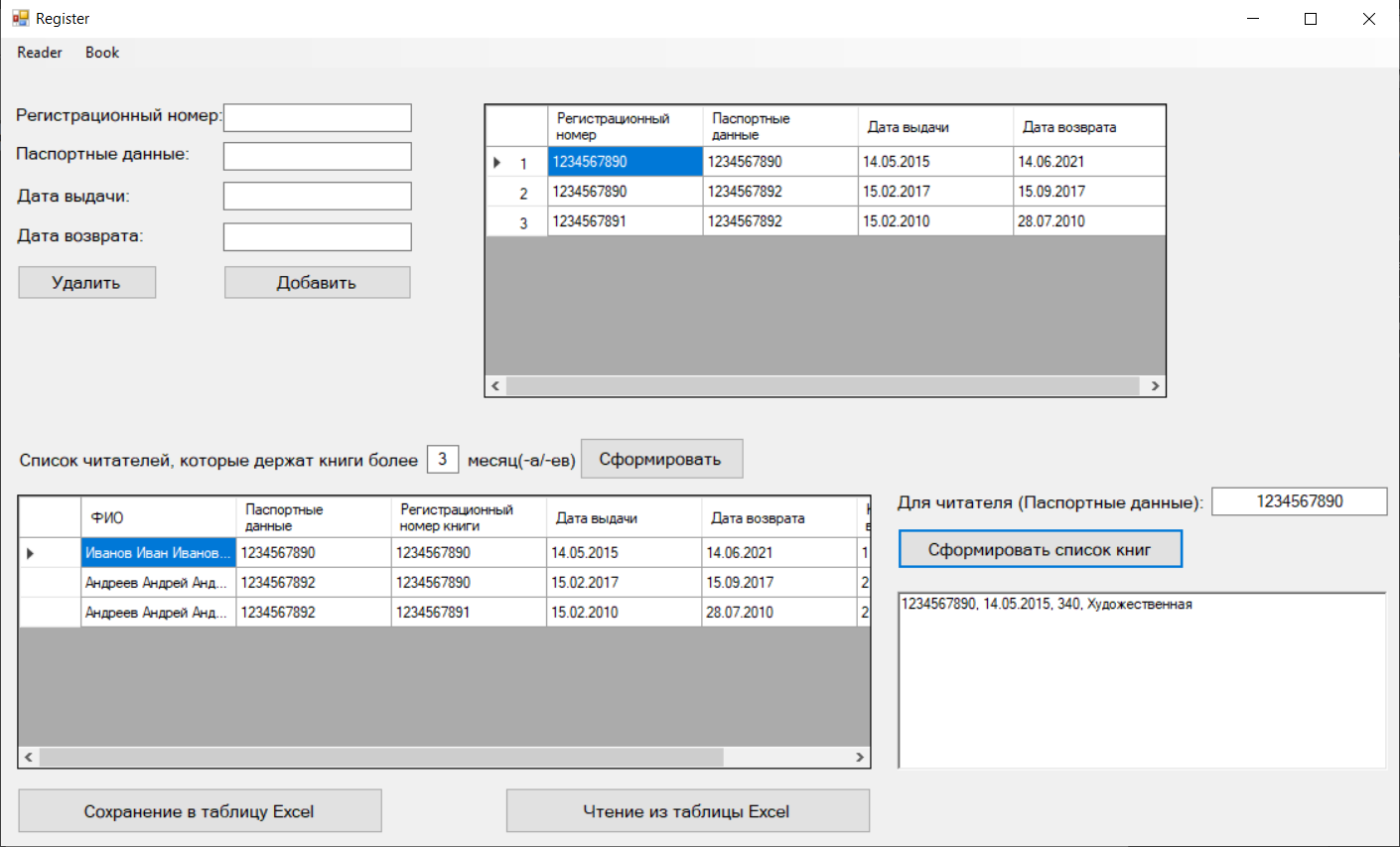


Рисунок 3 – Интерфейс взаимодействия с таблицей RecordRegistration

С помощью каждого окна можно манипулировать поведением Insert, Delete, Update и Select каждой таблицы. Для уменьшения громоздкости кода и увеличения скорости разработки были использованы адаптеры. Каждая форма реализует свою логику взаимодействия с той таблицей базы данных, к которой она привязана. Данные из базы данных привязываются посредством указания в DataSource каждого элемента управления DataGridView хранимой процедуры, осуществляющей возврат данных из соответствующей таблицы.

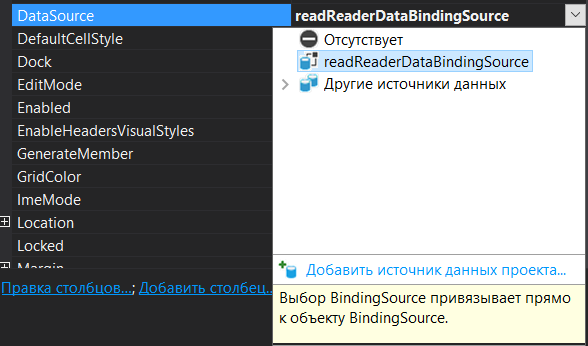


Рисунок 4 – Связывание элемента управления DataGridView формы Reader (элемент управления DataGridView)

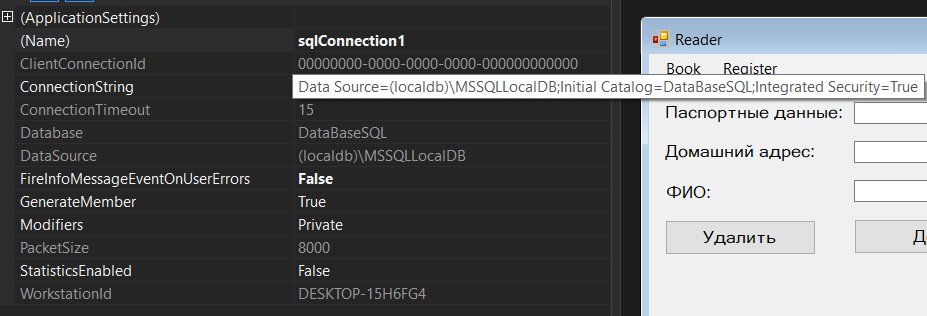


Рисунок 5 – Настроенное подключение к базе данных для формы Reader (элемент управления SqlConnection)

Код для записи данных в таблицу RecordRegistration:

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

{

if (!Reader.CheckTextBoxes(new List<TextBox> { \_txtPasswData1, \_txtPasswData2, \_txtDataIssue, \_txtDataReturn }))

{

MessageBox.Show("Ошибка: не все текстовые поля заполнены!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!Book.registerNumberValidate(\_txtPasswData1.Text))

{

MessageBox.Show("Ошибка: не корректный регистрационный номер!" +

" Номер должен состоять из числового набора длиной равной " + Book.MAX\_SIZE\_REGNUM.ToString(), "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!Reader.passwordDataValidate(\_txtPasswData2.Text))

{

MessageBox.Show("Ошибка: не корректные паспортные данные! Паспортные даные" +

" могут состоять только из набора цифр, длина которых строго равна " + Reader.MAX\_SIZE\_PSWD.ToString(), "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

\_txtPasswData1.Text = \_txtPasswData1.Text.Trim(new char[] { ' ' });

\_txtPasswData2.Text = \_txtPasswData2.Text.Trim(new char[] { ' ' });

\_txtDataIssue.Text = \_txtDataIssue.Text.Trim(new char[] { ' ' });

\_txtDataReturn.Text = \_txtDataReturn.Text.Trim(new char[] { ' ' });

if((!checkDateEntry(\_txtDataIssue.Text)) || (!checkDateEntry(\_txtDataReturn.Text))){

MessageBox.Show("Ошибка: неверный формат даты! Дата выдачи и возврата должна быть" +

" в формате ДД.ММ.ГГГГ", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

addDataToDataBase(\_txtPasswData1.Text, \_txtPasswData2.Text, \_txtDataIssue.Text, \_txtDataReturn.Text);

}

private void addDataToDataBase(String reg, String pwd, String dateIssue, String dateReturn)

{

if (!CheckOrientedData(reg, pwd, dateIssue, dateReturn))

{

MessageBox.Show("Ошибка: добавление записи невозможно! Запись содержит не корректные данные!",

"Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string[] dateString = dateIssue.Split(new char[] { '.' });

DateTime date1 = new DateTime(int.Parse(dateString[2]), int.Parse(dateString[1]),

int.Parse(dateString[0]));

dateString = dateReturn.Split(new char[] { '.' });

DateTime date2 = new DateTime(int.Parse(dateString[2]), int.Parse(dateString[1]),

int.Parse(dateString[0]));

this.sqlConnection1.Open();

this.sqlInsertCommand1.Parameters["@id"].Value = dataGridView1.Rows.Count;

this.sqlInsertCommand1.Parameters["@reg"].Value = long.Parse(reg);

this.sqlInsertCommand1.Parameters["@pwd"].Value = long.Parse(pwd);

this.sqlInsertCommand1.Parameters["@dateIssue"].Value = date1;

this.sqlInsertCommand1.Parameters["@dateReturn"].Value = date2;

this.sqlInsertCommand1.ExecuteScalar();

int result = Convert.ToInt32(this.sqlInsertCommand1.Parameters["@RETURN\_VALUE"].Value);

this.sqlConnection1.Close();

string[] names = { "регистрационные данные", "паспортные данные" };

if ((result < 0) && (result > (-3)))

{

MessageBox.Show("Ошибка: в записи содержаться не существующие "

+ names[(result == (-2)) ? 0 : 1] + "!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if (result == (-3))

{

MessageBox.Show("Ошибка: дата возврата не может быть раньше даты получения, дата возврата не может быть" +

" позднее текущей даты, установленной на сервере и дата получения не может быть раньше даты публикации книги!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if (result == (-4))

{

MessageBox.Show("Ошибка: записи с данным id не присутствует в базе данных, изменение невозможно!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

this.readRegisterDataTableAdapter.Fill(this.dataBaseSQLDataSet2.ReadRegisterData);

return;

}

При нажатии на кнопку “Добавить” на заранее заполненной форме Register, запускается логика добавления данных в таблицу RecordRegistration хранящейся на сервере.

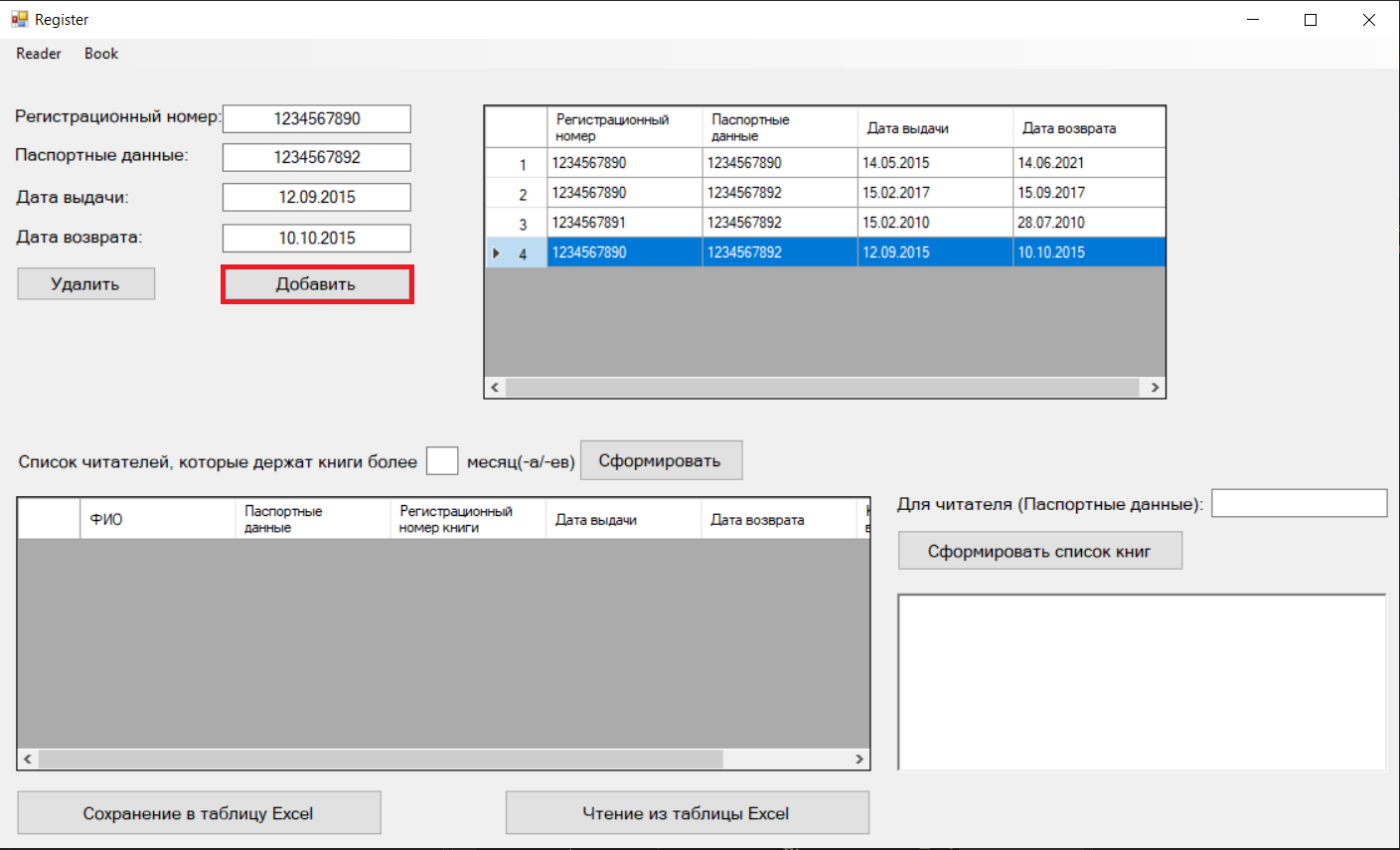


Рисунок 6 – Добавление данных в таблицу RecordRegistration

Код для изменения данных в таблице Book:

private void dataGridView1\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

if (dataGridView1.Rows.Count <= 0)

return;

int index = e.RowIndex;

if (index == dataGridView1.Rows.Count)

return;

if ((!Reader.checkRow(dataGridView1.Rows[index])) || (!CheckOrientedData(

dataGridView1.Rows[index].Cells[0].Value.ToString(),

dataGridView1.Rows[index].Cells[1].Value.ToString(),

dataGridView1.Rows[index].Cells[2].Value.ToString(),

dataGridView1.Rows[index].Cells[3].Value.ToString())))

{

MessageBox.Show("Ошибка: введены не корректные параметры! Изменение данных не возможно осуществить!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

this.readBookDataTableAdapter.Fill(this.dataBaseSQLDataSet1.ReadBookData);

return;

}

this.sqlConnection1.Open();

this.sqlUpdateCommand1.Parameters["@reg"].Value = long.Parse(dataGridView1.Rows[index].Cells[0].Value.ToString());

this.sqlUpdateCommand1.Parameters["@newPages"].Value = short.Parse(dataGridView1.Rows[index].Cells[1].Value.ToString());

this.sqlUpdateCommand1.Parameters["@newYear"].Value = short.Parse(dataGridView1.Rows[index].Cells[2].Value.ToString());

this.sqlUpdateCommand1.Parameters["@newSection"].Value = dataGridView1.Rows[index].Cells[3].Value.ToString();

this.sqlUpdateCommand1.ExecuteScalar();

int result = Convert.ToInt32(this.sqlUpdateCommand1.Parameters["@RETURN\_VALUE"].Value);

this.sqlConnection1.Close();

if (result == (-1))

{

MessageBox.Show("Ошибка: дата публикации не должна быть позднее текущей даты, установленной на сервере! Изменение невозможно!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

this.readBookDataTableAdapter.Fill(this.dataBaseSQLDataSet1.ReadBookData);

return;

}

else if (result == (-2))

{

MessageBox.Show("Ошибка: дата публикации не должна быть позднее дат, которые уже зарегистрированы для данной книги! Для изменения даты публикации" +

" книги необходимо удалить записи, содержащие даты выдачи книги позднее года публикации, из таблицы регистрации!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

this.readBookDataTableAdapter.Fill(this.dataBaseSQLDataSet1.ReadBookData);

return;

}

}

Данный код запускается при изменении значения каждой отдельной ячейки таблицы, визуально представленной с помощью элемента управления DataGridView на форме Book (и механизм обновления схож для всех других форм, как и механизм Insert, Update, Delete и Select в целом).

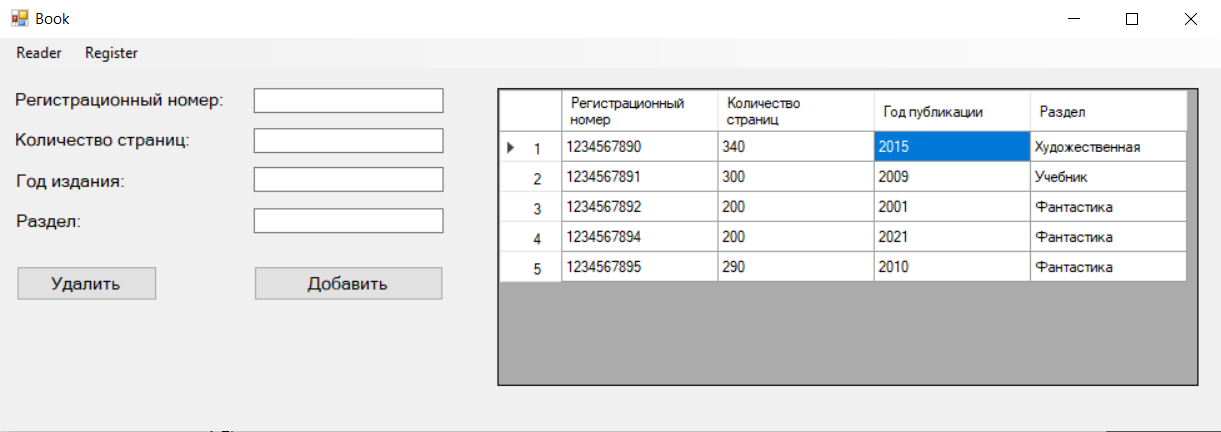


Рисунок 7 – До изменения данных в таблице Book

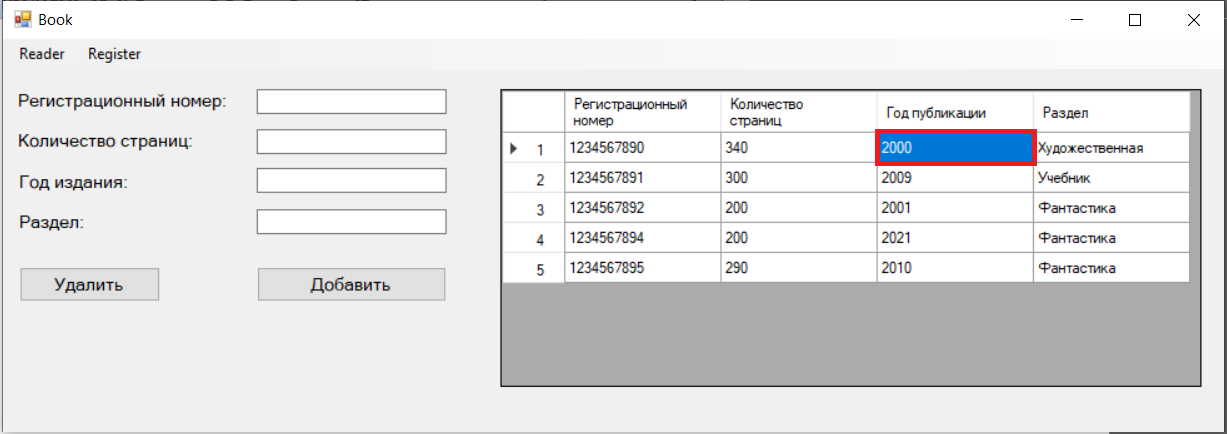


Рисунок 8 – Таблица Book после обновления

Для удаления каждой записи в таблице Reader (и в других таблицах), достаточно выбрать строку таблицы и нажать на кнопку “Удалить”.

Код для удаления записи в таблице Book:

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

{

if (dataGridView1.CurrentRow == null)

{

MessageBox.Show("Ошибка: не выбрана строка для удаления", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int index = dataGridView1.CurrentRow.Index;

String pwd = dataGridView1.Rows[index].Cells[0].Value.ToString();

this.sqlConnection1.Open();

this.sqlDeleteCommand1.Parameters["@pwd"].Value = long.Parse(pwd);

this.sqlDeleteCommand1.ExecuteScalar();

int result = Convert.ToInt32(this.sqlDeleteCommand1.Parameters["@RETURN\_VALUE"].Value);

this.sqlConnection1.Close();

if (result < 0)

{

MessageBox.Show("Ошибка: невозможно удалить данную запись!" +

" Данные паспортные данные зарегистрированы! Чтобы удалить данную" +

" запись необходимо удалить запись с данными паспортными данными!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

this.readReaderDataTableAdapter.Fill(this.dataBaseSQLDataSet.ReadReaderData);

}

Все данные вводимые пользователем в форму проходят через методы проверки, для примера будет представлен код проверки данных с формы Register:

private bool CheckOrientedData(String reg, String pwd, String dateIssue, String dateReturn)

{

if ((!Reader.passwordDataValidate(pwd))

|| (!Book.registerNumberValidate(reg))

|| (!checkDateEntry(dateIssue))

|| (!checkDateEntry(dateReturn)))

{

return false;

}

try

{

long.Parse(reg);

long.Parse(pwd);

}

catch (Exception) { return false; }

return true;

}

private bool checkDateEntry(String date)

{

if (date.Length != MAX\_SIZE\_DATA)

return false;

if(date.Count((char i) => (i == '.')) != 2)

{

return false;

}

foreach (var i in date)

if ((!Char.IsDigit(i)) && (i != '.'))

return false;

string[] dateSplit = date.Split(new char[] { '.' });

if ((dateSplit[2].Length != 4) || (dateSplit[1].Length != 2) || (dateSplit[0].Length != 2))

return false;

try

{

int value = int.Parse(dateSplit[0]);

if ((value <= 0) || (value > 31))

return false;

value = int.Parse(dateSplit[1]);

if ((value <= 0) || (value > 12))

return false;

value = int.Parse(dateSplit[2]);

if ((value <= 0) || (value > 9999))

return false;

}

catch (Exception)

{

return false;

}

return true;

}

При некорректности данных запись и обновления данных невозможно (см. рис. 9-11).

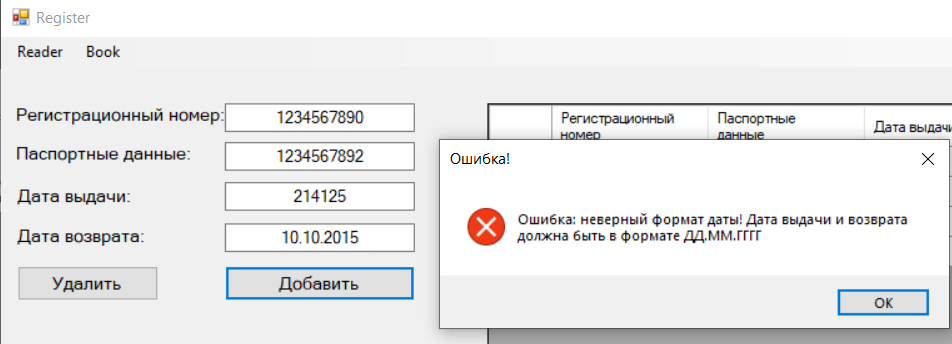


Рисунок 9 – Попытка добавить запись с некорректными данными

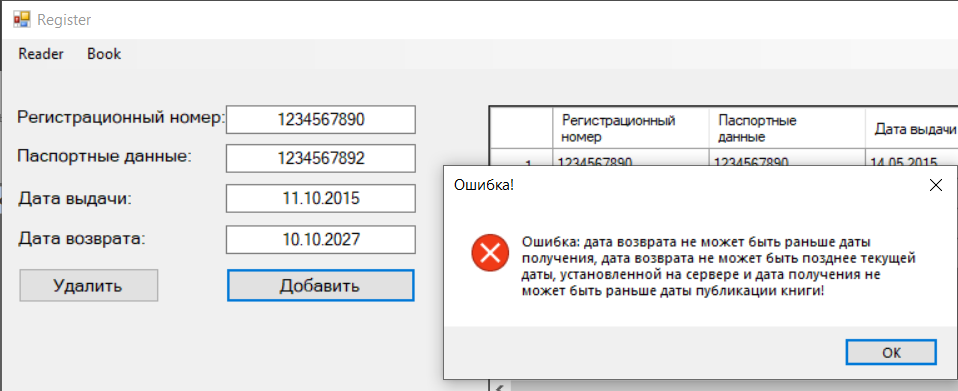


Рисунок 10 – Попытка добавления записи с некорректными данными (ошибка в указании временного интервала – возврат позднее фактической даты, установленной на сервере)

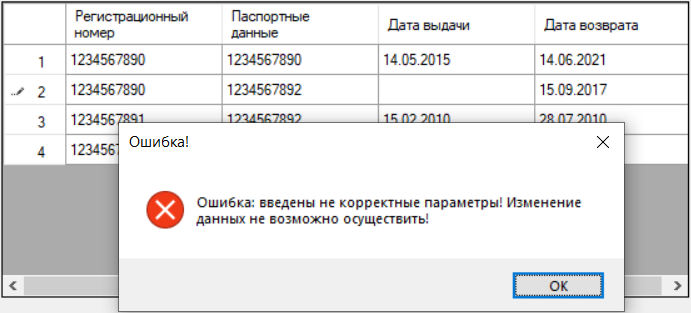


Рисунок 11 – Попытка обновления таблицы RecordRegistration некорректными данными

Аналогичные проверки реализованы для всех таблиц. Например, проверка на корректность для таблицы Book представлена на рисунке 12 (попытка обновления не корректными данными).

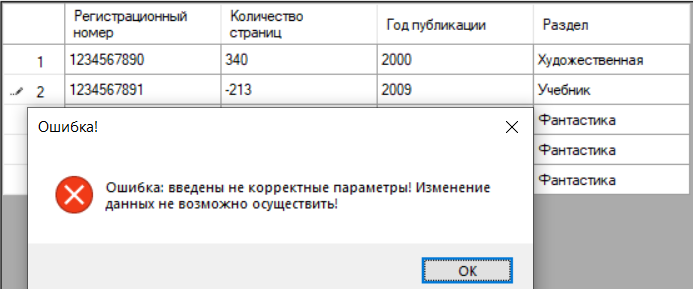


Рисунок 12 – Ошибка при обновлении таблицы Book не корректными данными

Для решения заданий индивидуального варианта необходимо перейти на форму Register и взаимодействовать с интерфейсом решения соответствующих задач.

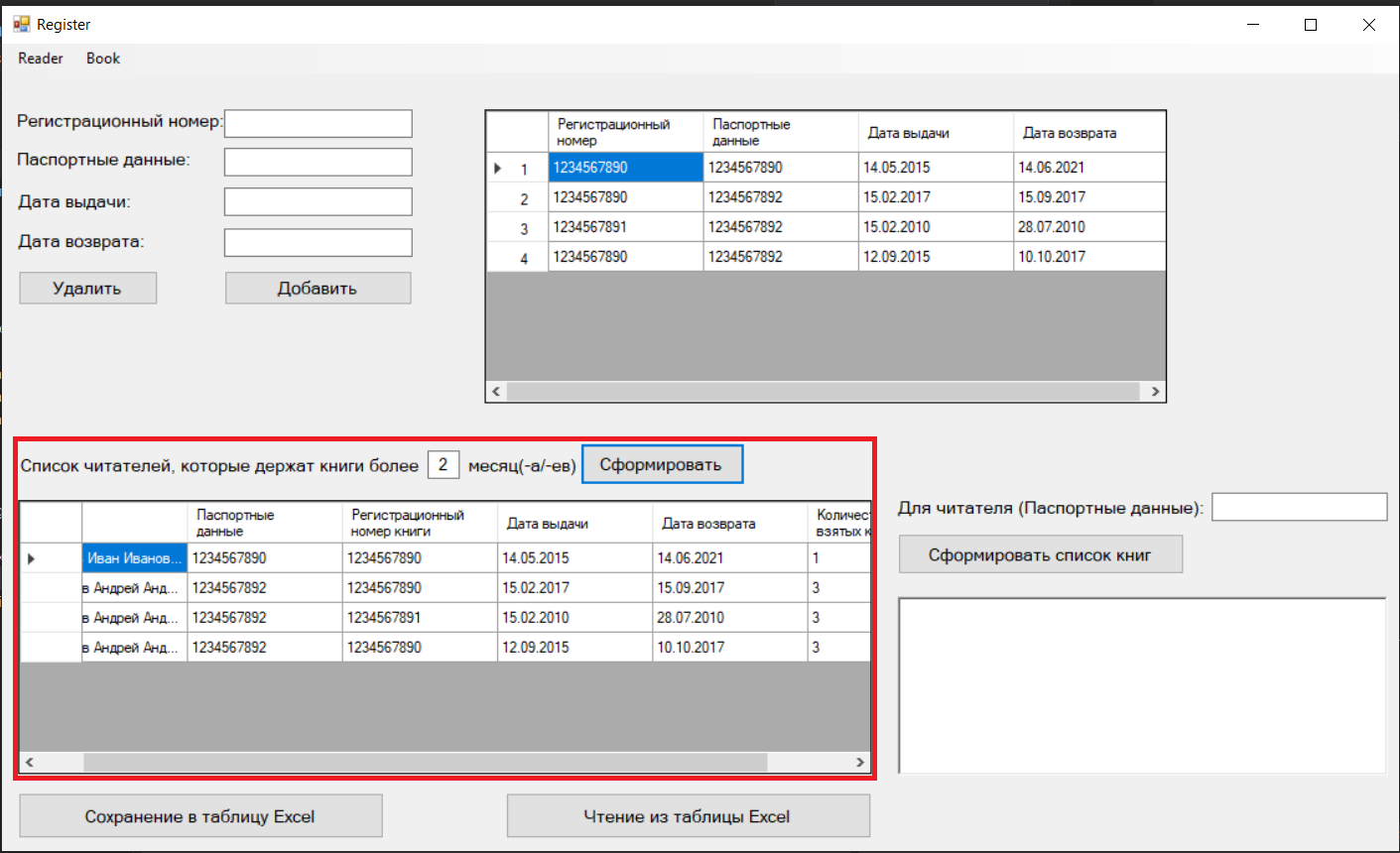


Рисунок 13 – Решение задания №1 задачи индивидуального варианта

Код решения задания №1:

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

{

if(textBox1.Text.Length == 0)

{

MessageBox.Show("Ошибка: не введено число месяц(-а/-ев)!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

dataGridView2.Rows.Clear();

using (SqlConnection sqlConnection = new SqlConnection("Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=DataBaseSQL;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"))

{

sqlConnection.Open();

using (SqlCommand command = new SqlCommand("dbo.TaskOneT", sqlConnection))

{

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@monthValue", int.Parse(textBox1.Text));

SqlDataReader sqlReader = command.ExecuteReader();

if (sqlReader.HasRows)

{

while (sqlReader.Read())

{

String fullName = sqlReader.GetValue(0).ToString();

String pwd = sqlReader.GetValue(1).ToString();

String reg = sqlReader.GetValue(2).ToString();

while (pwd.Length != 10)

pwd = ("0" + pwd);

while (reg.Length != 10)

reg = ("0" + reg);

dataGridView2.Rows.Add(fullName, pwd, reg,

sqlReader.GetValue(3).ToString().Split(new char[] { ' ' })[0],

sqlReader.GetValue(4).ToString().Split(new char[] { ' ' })[0],

"0");

}

}

sqlReader.Close();

}

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

using (SqlCommand command2 = new SqlCommand("dbo.DefineBookCounter", sqlConnection))

{

command2.CommandType = System.Data.CommandType.StoredProcedure;

SqlParameter valueReturn = new SqlParameter("valueReturn", SqlDbType.Int);

valueReturn.Direction = ParameterDirection.ReturnValue;

command2.Parameters.Add(valueReturn);

command2.Parameters.AddWithValue("@pwd", long.Parse(dataGridView2.Rows[i].Cells[1].Value.ToString()));

command2.ExecuteScalar();

dataGridView2.Rows[i].Cells[dataGridView2.Rows[i].Cells.Count - 1].Value = Convert.ToString(valueReturn.Value);

}

}

sqlConnection.Close();

}

}

После решения задания сформированную таблицу можно записать в файл таблицы Excel, нажав на кнопку “Сохранить в таблицу” (см. рис. 14).

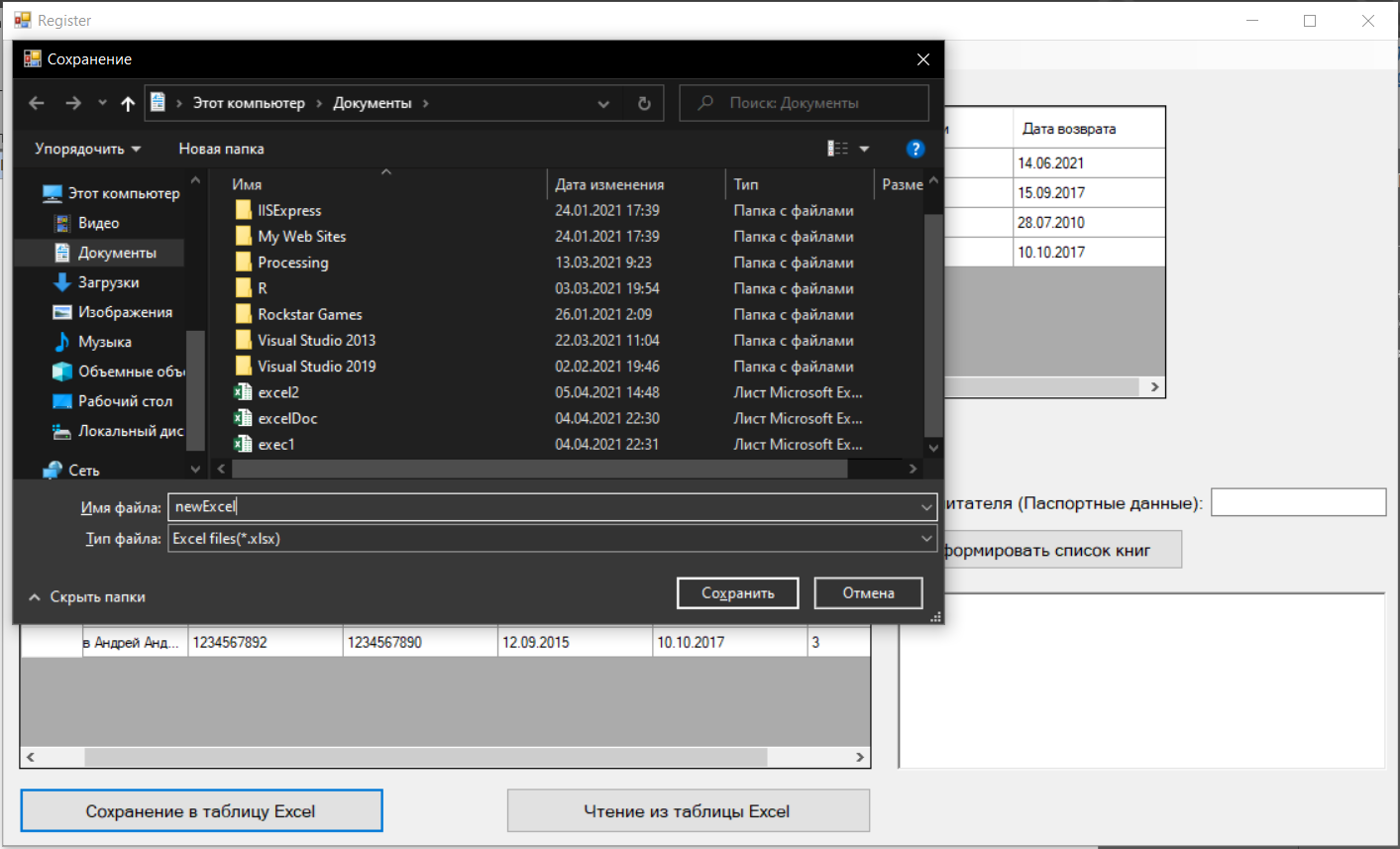


Рисунок 14 – Сохранение данных в Excel

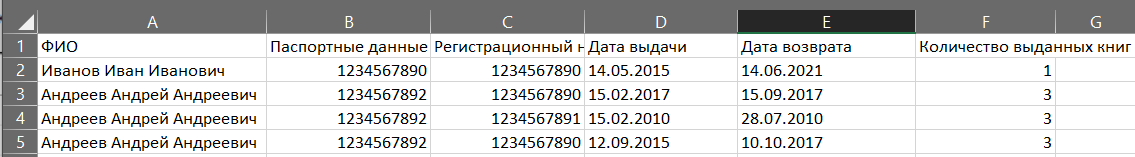


Рисунок 15 – Содержимое файла newExcel после записи данных из таблицы.

Код для взаимодействия с документом Excel:

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

{

if(dataGridView2.Rows.Count == 0)

{

MessageBox.Show("Ошибка: нет данных для сохранения!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string filename = saveFileDialog1.FileName;

Excel.Application excelApp = new Excel.Application();

excelApp.Workbooks.Add();

Excel.Worksheet workSheet = excelApp.ActiveSheet;

workSheet.Cells[1, "A"] = "ФИО";

workSheet.Cells[1, "B"] = "Паспортные данные";

workSheet.Cells[1, "C"] = "Регистрационный номер книги";

workSheet.Cells[1, "D"] = "Дата выдачи";

workSheet.Cells[1, "E"] = "Дата возврата";

workSheet.Cells[1, "F"] = "Количество выданных книг";

for(int i = 0; i < dataGridView2.Rows.Count; i++)

{

workSheet.Cells[(i + 2), "A"] = dataGridView2.Rows[i].Cells[0].Value.ToString();

workSheet.Cells[(i + 2), "B"] = dataGridView2.Rows[i].Cells[1].Value.ToString();

workSheet.Cells[(i + 2), "C"] = dataGridView2.Rows[i].Cells[2].Value.ToString();

workSheet.Cells[(i + 2), "D"] = dataGridView2.Rows[i].Cells[3].Value.ToString();

workSheet.Cells[(i + 2), "E"] = dataGridView2.Rows[i].Cells[4].Value.ToString();

workSheet.Cells[(i + 2), "F"] = dataGridView2.Rows[i].Cells[5].Value.ToString();

}

try

{

workSheet.SaveAs(filename);

MessageBox.Show("Данные сохранены!", "Информация",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch(Exception)

{

MessageBox.Show("Ошибка: неудалось сохранить файл!", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

excelApp.Quit();

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

dataGridView2.Rows.Clear();

string filename = openFileDialog1.FileName;

Excel.Application excelApp = new Excel.Application();

Excel.Workbook excelBook = excelApp.Workbooks.Open(filename);

Excel.\_Worksheet excelSheet = excelBook.Sheets[1];

Excel.Range excelRange = excelSheet.UsedRange;

int rows = excelRange.Rows.Count;

int cols = excelRange.Columns.Count;

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

for (int i = 1; i <= rows; i++)

{

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

for (int j = 1; j <= cols; j++)

{

if (excelRange.Cells[i, j] != null && excelRange.Cells[i, j].Value2 != null)

maping[(i-1)].Add(excelRange.Cells[i, j].Value2.ToString());

}

}

excelBook.Close();

excelApp.Quit();

System.Runtime.InteropServices.Marshal.ReleaseComObject(excelApp);

for (int i = 1; i < maping.Count; i++)

{

dataGridView2.Rows.Add(

maping[i][0],

maping[i][1],

maping[i][2],

maping[i][3],

maping[i][4],

maping[i][5]);

}

dataGridView2.Refresh();

MessageBox.Show("Данные считаны!", "Информация",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

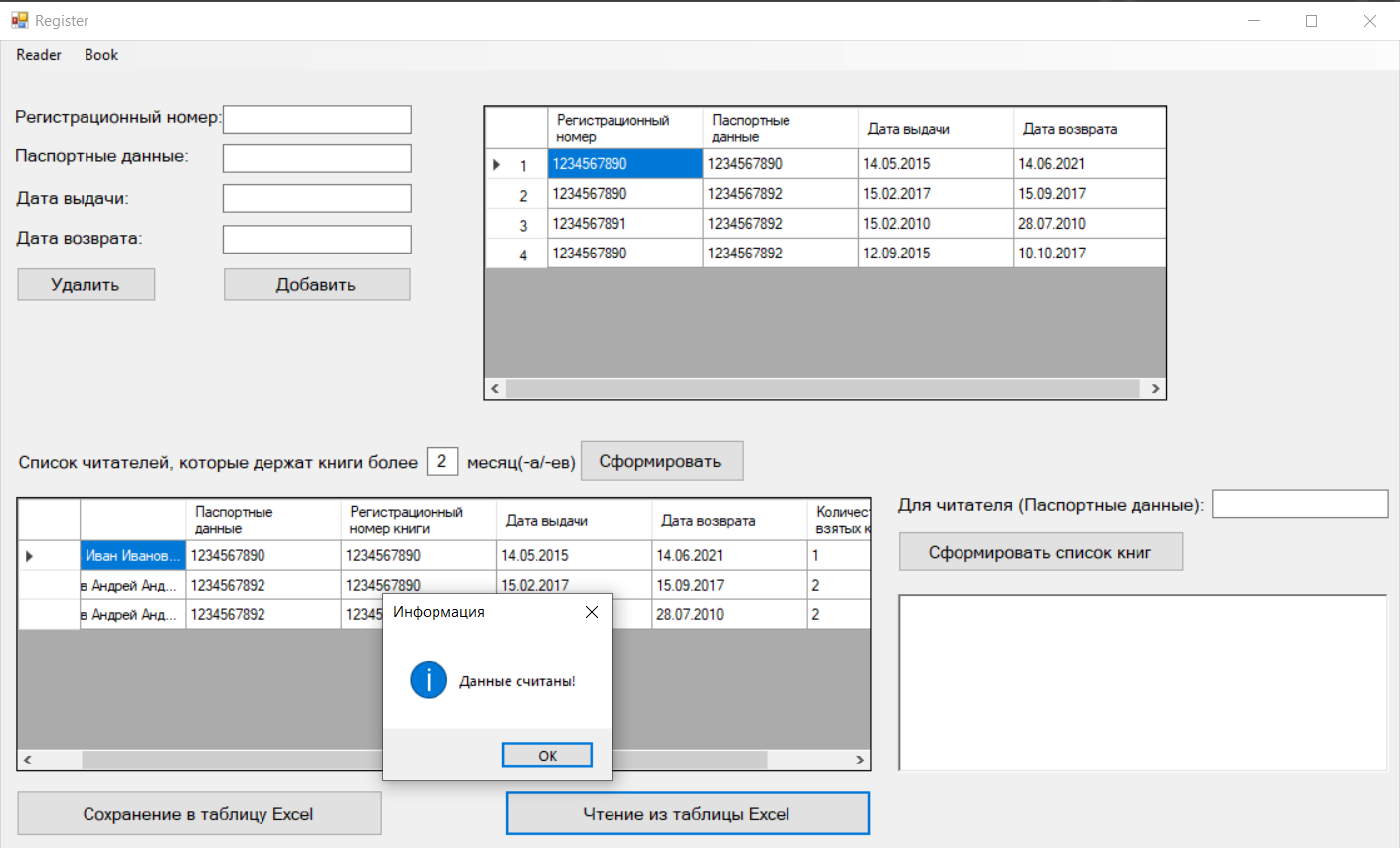


Рисунок 16 – Чтение таблицы Excel

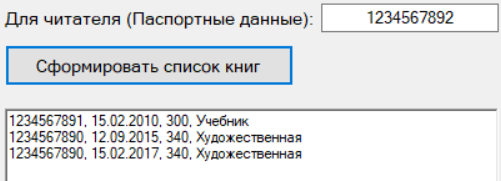


Рисунок 17 – Решение задачи 2 индивидуального варианта