**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

**Расчетно-графическая работа**

по «Информационная система “Регистрации авиабилетов”»

по дисциплине «Информационные системы»

Факультет: АВТФ Преподаватель:

Группа: АВТ-518 Бычков М.И.

Студент: Беймо Ю. Д.

Вариант: 2

Оглавление

[Задание 3](#_Toc431641833)

[Планирование 4](#_Toc431641841)

[Back-End планирование 5](#_Toc431641842)

[Front-End планирование 6](#_Toc431641843)

[Этап Декомпозиции 7](#_Toc431641845)

[Проектирование 8](#_Toc431641847)

[BEпроектирование 9](#_Toc431641848)

[Среда проектирования 11](#_Toc431641850)

[BE реализация 12](#_Toc431641851)

[Физическая реализация 15](#_Toc431641852)

[Анализ результатов и выводы 18](#_Toc431641853)

[Приложение 19](#_Toc431641854)

# Задание

Разработать информационную систему, которая автоматизирует обслуживание пассажиров в аэропорту агентом по регистрации.

## Система должна:

1) содержать расписание рейсов из Москвы;

2) закрепить заданный рейс на стойку регистрации;

3) получить списки пассажиров на заданный рейс;

4) зарегистрировать подошедшего пассажира на рейс. Для этого надо указать номер пассажира в списке, номер паспорта, вес сдаваемого багажа и ручной клади;

5) снять пассажира с рейса, то есть отменить регистрацию, не удаляя его из списка;

6) повторно его зарегистрировать;

7) завершить регистрацию, выдав списки пассажиров прошедших регистрацию, неявившихся пассажиров, сводную ведомость регистрации с указанием количества пассажиров, с указанием пола, суммарный вес багажа и ручной клади и вес сверхнормативного багажа

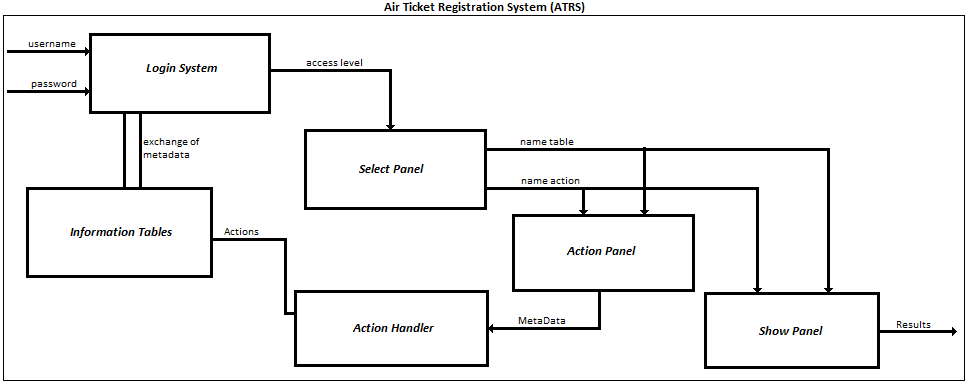
Планирование

Для реализации проекта потребовалось проанализировать задачу. В ходе анализа было выявлено, что разработку клиентского приложения необходимо разделить на две фундаментально различные части: Front-End и Back-End разработку.

Back-End – в данный раздел входит планирование, структуризация и разработка базы данных, составление необходимых запросов, которые удовлетворяют всем требованиям проекта.

Front-End – в данный раздел входит планирование, структуризация, оптимизация и разработка программного интерфейса пользователя.

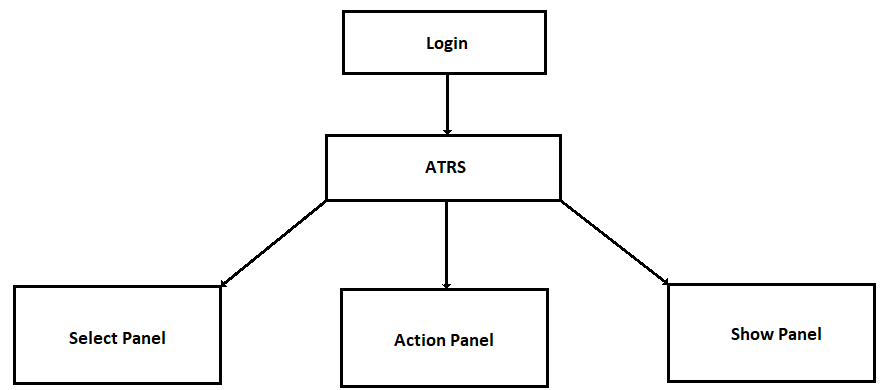
В процессе планирования также постоим диаграмму feo:



*Рис. 1 - Диаграмма feo для Air Ticket Registration System*

## Front-End планирование

Необходимо реализовать набор схем, удовлетворяющих требованиям проекта. Ниже представлена общая схема зависимостей для пользовательского интерфейса:



*Рис. 2 - общая схема зависимостей пользовательского интерфейса*

Back-End планирование

Разрабатываемая информационная система будет содержать:

* Информацию о городах
* Информацию об авиакомпаниях
* Информацию о типах самолетов
* Информацию о рейсах
* Информацию о пассажирах
* Информацию о регистрации

Ниже представлена структурная схема информационной системы со всеми связями:

Рейсы:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ИД | Откуда | Куда | Дата вылета | Дата прилёта | Компания | Самолёт | Платформа |

Пассажиры:

|  |  |  |  |
| --- | --- | --- | --- |
| ИД | № пасспорта | Вес багажа | Пол |

Города:

|  |  |
| --- | --- |
| ИД | Город |

Авиакомпании:

|  |  |
| --- | --- |
| ИД | Компания |

Типы самолетов:

|  |  |
| --- | --- |
| ИД | Тип |

Регистрация:

|  |  |  |  |
| --- | --- | --- | --- |
| ИД | ИД рейса | ИД пассажира | Явка |

Рис. 3 – Структура таблиц информационной системы.

# Проектирование информационной системы

## Графическое представление сущность-связь:

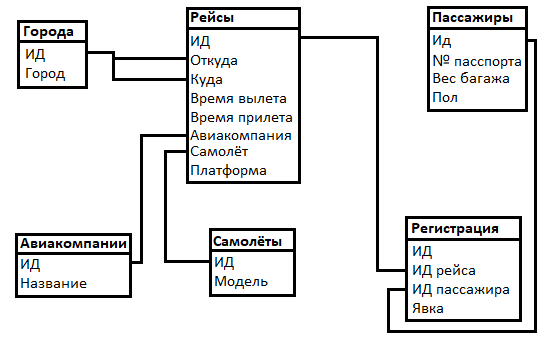


Рис. 4 – Графическое представление информационной системы.

## BEпроектирование

Ниже представлены все таблицы базы данных с заполненными связями:

Таблица 1. Сущность «Города»

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ИД (Первичный ключ) | integer |
| Город | character |

Таблица 2. Сущность «Рейсы»

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ИД (Первичный ключ) | integer |
| Откуда (Внешний ключ) | integer |
| Куда (Внешний ключ) | integer |
| Время вылета | datatime |
| Время прилёта | datatime |
| Авиакомпания (Внешний ключ) | integer |
| Самолёт (Внешний ключ) | integer |
| Платформа | integer |

Таблица 4. Сущность «Пассажиры»

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ИД (Первичный ключ) | integer |
| № пасспорта | character |
| Вес багажа | integer |
| Пол | character |

Таблица 5. Сущность «Авиакомпании»

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ИД (Первичный ключ) | integer |
| Компания | character |

Таблица 6. Сущность «Типы самолетов»

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ИД (Первичный ключ) | integer |
| Тип | character |

Таблица 7. Сущность «Регистрация»

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| ИД (Первичный ключ) | integer |
| ИД рейса (Внешний ключ) | integer |
| ИД пассажира (Внешний ключ) | integer |
| Явка | integer |

## Среда проектирования

Для разработки информационной системы были выбраны следующие инструментальные средства:

В качестве СУБД выбрана PostgreSQL 9.4. Данная система позволяет хранить все данные информационной системы в базе данных на сервере. Таким образом, все клиенты системы будут обращаться к одной базе данных посредством ИС. Преимуществами данной СУБД являются:

* быстродействие;
* высокий уровень безопасности;
* надежность;
* переносимость.

В качестве среды программирования выбран Microsoft Visual Studio 2013, в качестве языка программирования выбран C# - [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)[язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

C# является языком программирования, который разработан для создания множества приложений, работающих в среде .NET Framework. Язык C# прост, безопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую С-подобным языкам.

Visual C# — это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework 4.5 предоставляет доступ ко многим службам операционной системы и к другим полезным, хорошо спроектированным классам, что существенно ускоряет цикл разработки.

# 

# Физическая реализация

Было создано Windows-приложение, состоящее из пяти форм. Для работы с СУБД PostgreSQL потребовался драйвер Npgsql и библиотеки Mono.Security.dll, Npgsql.dll.

Для того чтобы, создать новое подключение к базе данных, нужно прописать строку подключения перед обращением к базе данных. Строка подключения в общем виде выглядит следующим образом:

NpgsqlConnectionconn = newNpgsqlConnection("server= 127.0.0.1; database= database; userid = user; password = password;");

Создали новый адаптер для заполнения набора данных с запросом:

NpgsqlDataAdapter adap = new Npgsql.NpgsqlDataAdapter("select \* from tablename", conn);

DataAdapter – составная часть провайдера данных. То есть подсоединенная компонента объектной модели ADO .NET. Используется для заполнения объекта DataSet и модификации источника данных. Выполняет функции посредника при взаимодействии БД и объекта DataSet.

Обеспечивает связь между источником данных и объектом DataSet. С одной стороны, база данных, с другой – DataSet. Извлечение данных и заполнение объекта DataSet – назначение DataAdapter'а.

С использованием функциональных возможностей DataAdapter'а, а конкретно с помощью метода Fill, который изменяет данные в DataSet, при выполнении метода Fill объект DataAdapter заполняет DataTable или DataSet данными, полученными из БД. После обработки данных, загруженных в память, с помощью метода Update можно записать модифицированные записи в БД.

Фактически, DataAdapter управляет обменом данных и обновлением содержимого источника данных.

DataAdapter представляет набор команд для подключения к базе данных и модификации данных.

Связали элемент dataGridView с набором данных с помощью команды: dataGridView1.DataSource = dataSet1.Tables[0];

Свойство DataSource возвращает или задает источник данных, данные которого отображает [DataGridView](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.datagridview(v=vs.110).aspx).

Аналогичным образом можно связать с базой данных такие объекты, как comboBox и listBox.

Так, например, для входа в систему после ввода логина и пароля

После того как пользователь завершает работу с данной ИС, все изменения *автоматически* сохраняются в БД, что способствует, продолжить работу с ИС с использованием всех последних изменений.

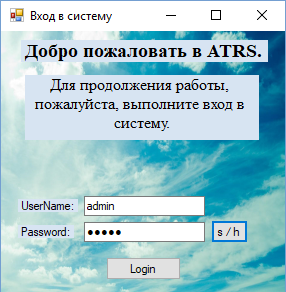


Рис.5 – Вход в информационную систему.

Кнопка s/h служит для отображения/скрытия символов пароля.

После входа открывается окно работы с системой:

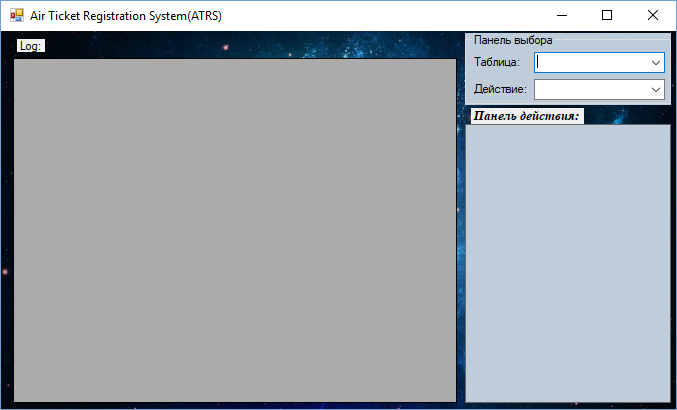


Рис.6 – Главное окно информационной системы.

Интерфейс состоит из 3 важных элементов: панель выбора, панель действия и панель отображения. Для работы с системой необходимо выбрать таблицу и действие в панели выбора. Набор для выбора зависит от уровня доступа.

## Работа с датой

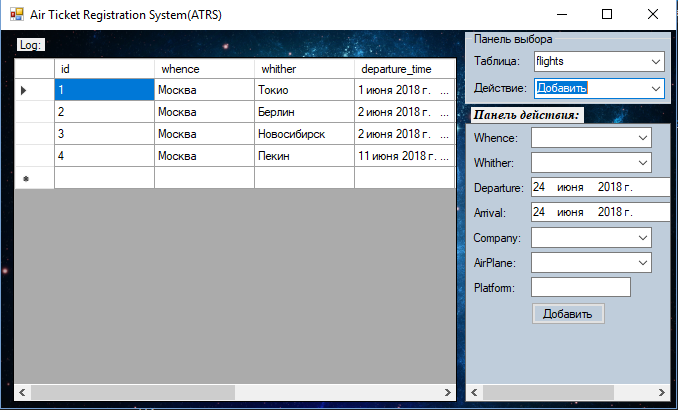


Рис.7 – Окно добавления рейса.

Для работы с датой используется элемент DateTimePicker.

DateTimePicker представляет раскрывающийся по нажатию календарь, в котором можно выбрать дату. собой элемент, который с помощью перемещения ползунка позволяет вводить числовые значения.

Наиболее важные свойства DateTimePicker:

- Format: определяет формат отображения даты в элементе управления.

- CustomFormat: задает формат отображения даты, если для свойства Format установлено значение Custom

- MinDate: минимальная дата, которую можно выбрать

- MaxDate: наибольшая дата, которую можно выбрать

- Value: определеяте текущее выбранное значение в DateTimePicker

- Text: представляет тот текст, который отображается в элементе

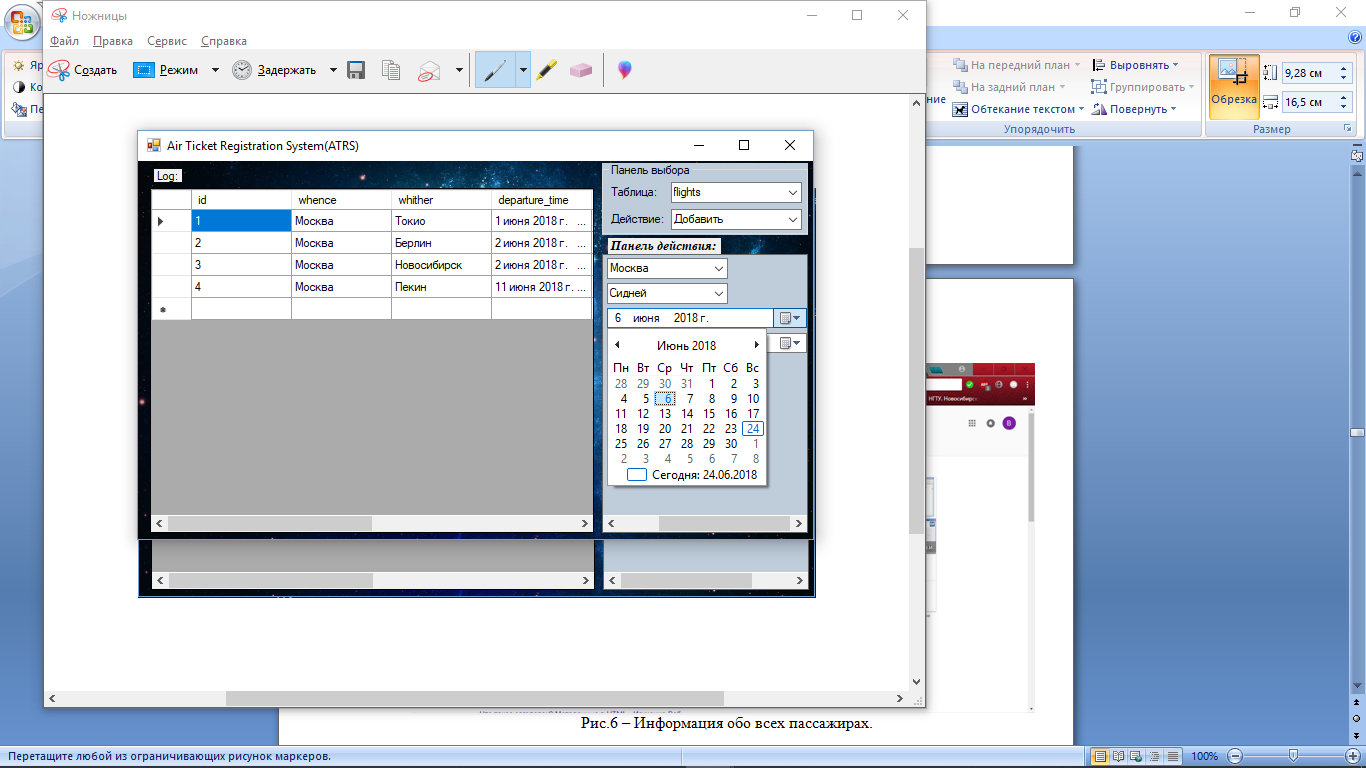


Рис.8 – Работа с датой

## Регистрация

Для начала добавим несколько пассажиров:

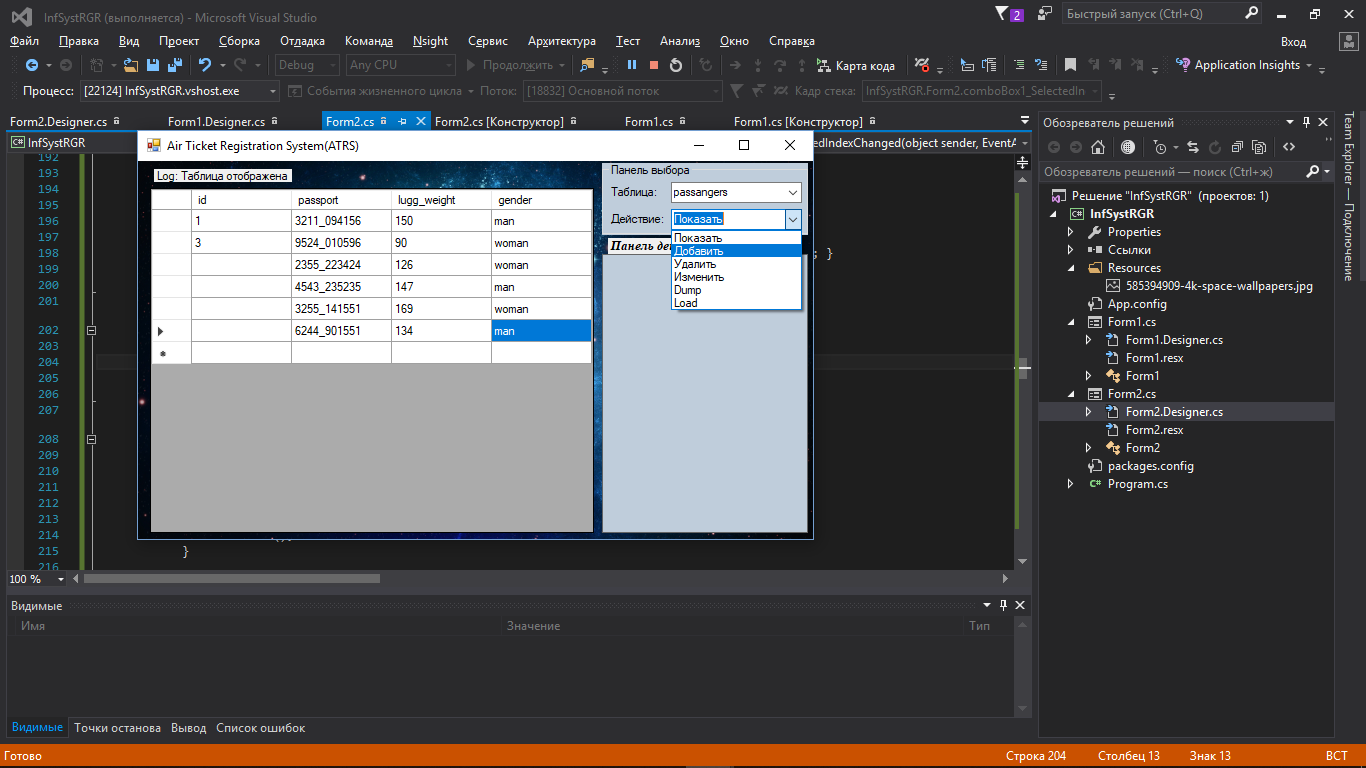


Рис. 9 - Регистрация пассажира

Добавление происходит вписыванием значения прямо в таблицу, а затем выбора действия "Добавить".

Если необходимо удалить запись, то после выбора действия нужно вписать ИД удаляемых записей через запятую.

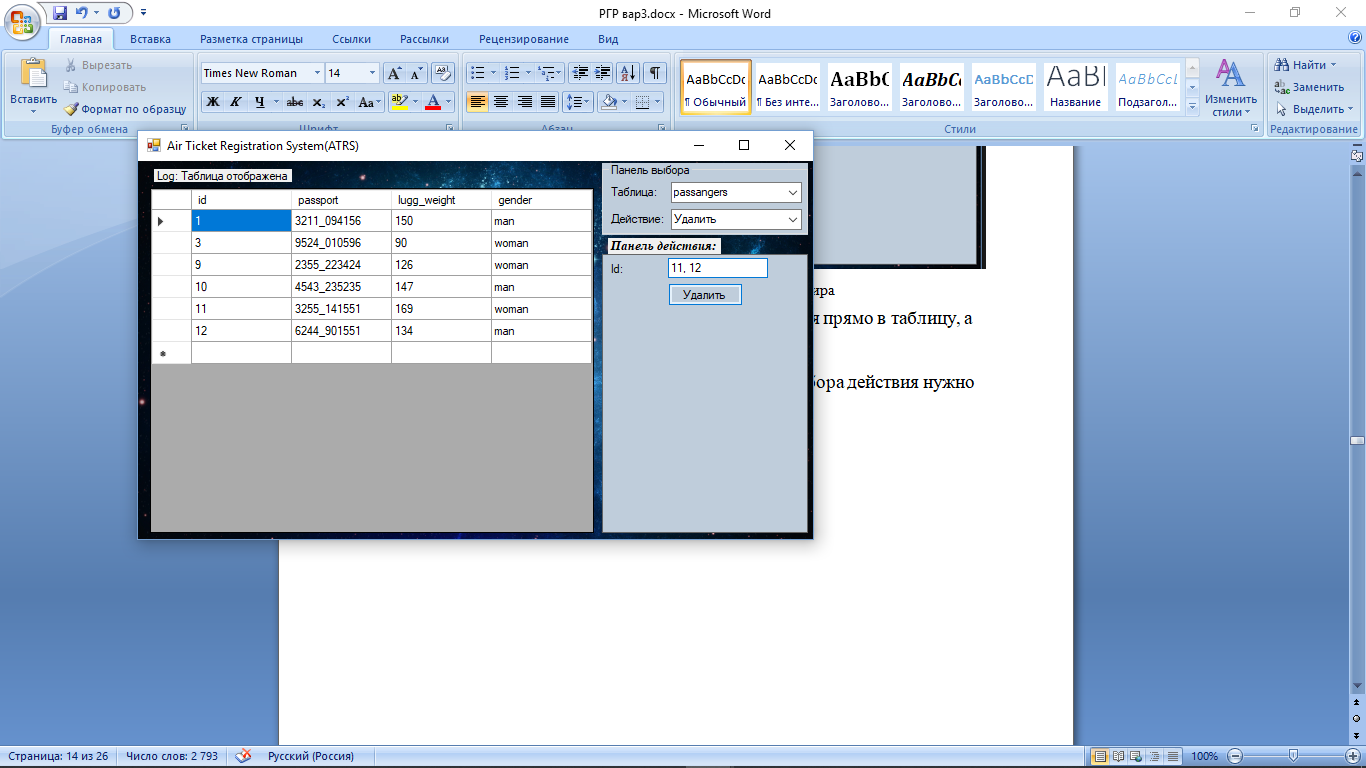


Рис. 10 - Удаление записей

Для регистрации пассажира на рейс необходимо указать ИД рейса и ИД пассажира, а также явку(1 - явился, 0 - не явился):

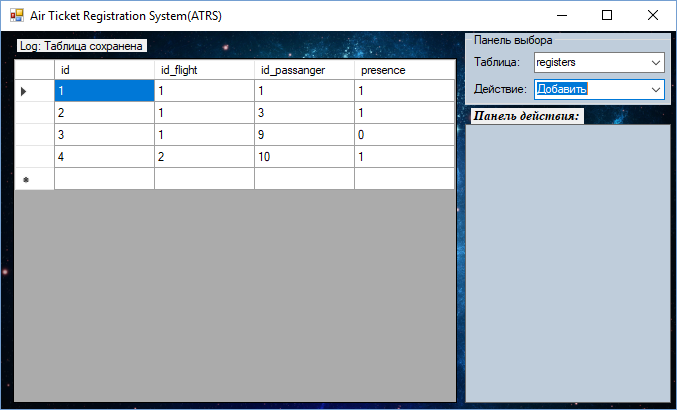


Рис. 11 - Регистрация пассажира на рейс

Dump и Загрузка БД:

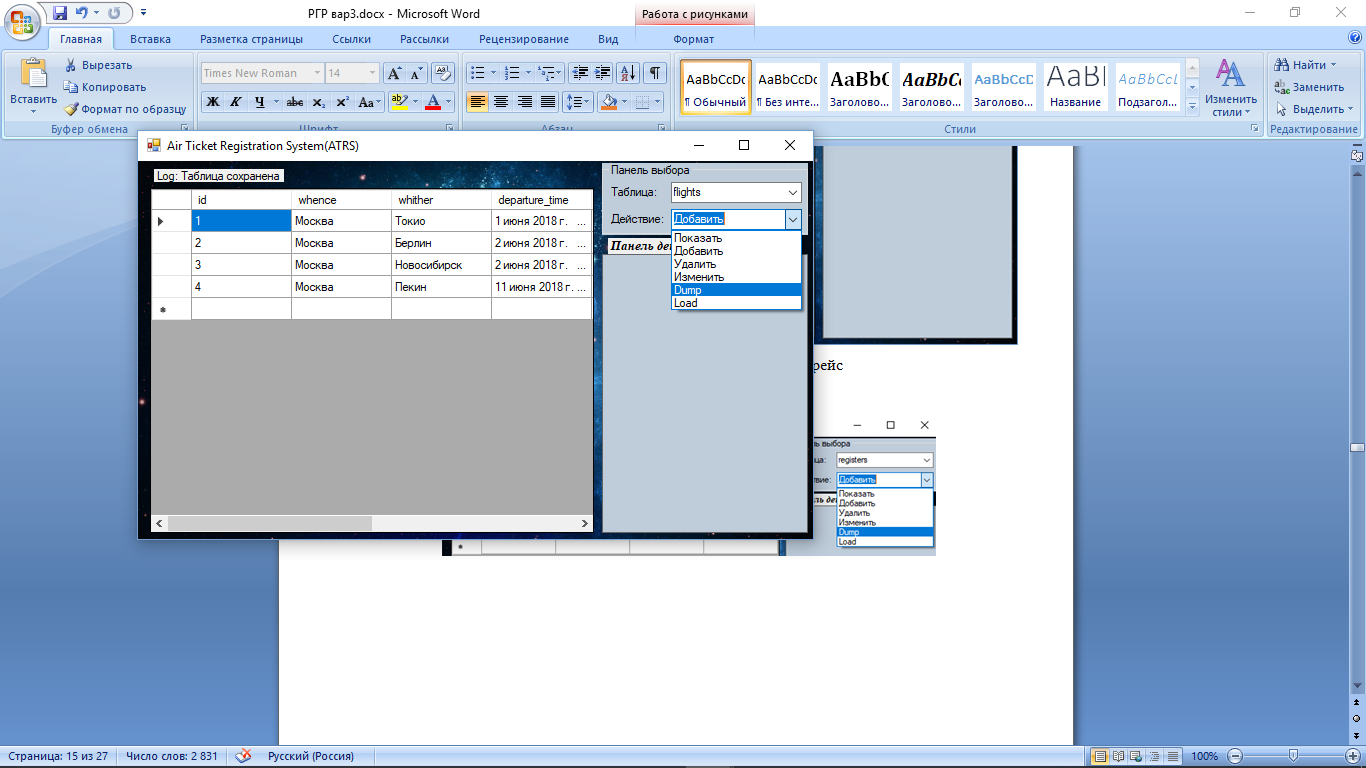


Рис. 12 - Dump BD

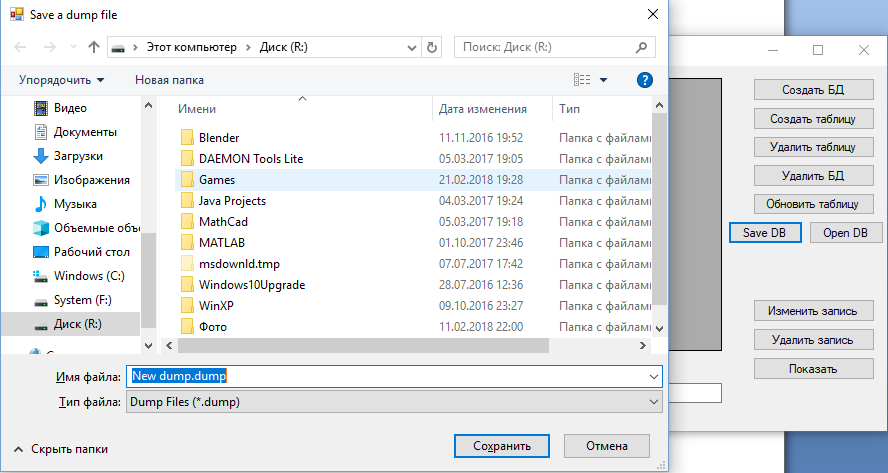


Рис. 13 - Dump BD продолжение**Анализ** результатов **и выводы:**

В проделанной расчетно-графической работе был разработан проект информационной системы, который автоматизирует процесс регистрации авиабилетов. Для выполнения реализации проекта этой работы были использованы знания и навыки, полученные на лабораторных работах. Предусмотрено сохранение резервной копии БД и ее развертывание при работе с информационной системой.

# Приложение

**Autorization.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace InfSystRGR

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

int acs\_lvl = 1;

if ((textBox1.Text == "") && (textBox2.Text == ""))

{

Form2 newForm = new Form2(acs\_lvl);

newForm.Show();

} else if ((textBox1.Text == "manager") && (textBox2.Text == "123"))

{

acs\_lvl = 2;

Form2 newForm = new Form2(acs\_lvl);

newForm.Show();

} else if ((textBox1.Text == "admin") && (textBox2.Text == "12213"))

{

acs\_lvl = 3;

Form2 newForm = new Form2(acs\_lvl);

newForm.Show();

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (textBox2.UseSystemPasswordChar == false) textBox2.UseSystemPasswordChar = true;

else textBox2.UseSystemPasswordChar = false;

}

}

}

**Flights.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

using System.IO;

using System.Diagnostics;

namespace InfSystRGR

{

public partial class Form2 : Form

{

string nameTable, nameAct;

string connectionString = "Server=localhost;Port=5432;User Id=postgres;Password=12213;Database=rgr\_atrs";

DataSet dataSet;

public NpgsqlDataAdapter dataAdapter;

public NpgsqlConnection npgsqlConnection;

public NpgsqlCommand npgComm;

int acs\_lvl = 0;

Button myButt = new Button();

ComboBox myWhence = new ComboBox();

ComboBox myWhither = new ComboBox();

DateTimePicker myDeparture = new DateTimePicker();

DateTimePicker myArrival = new DateTimePicker();

ComboBox myCompany = new ComboBox();

ComboBox myPlane = new ComboBox();

TextBox myText1 = new TextBox();

TextBox myText2 = new TextBox();

TextBox myText3 = new TextBox();

Label labWhence = new Label();

Label labWhither = new Label();

Label labDeparture = new Label();

Label labArrival = new Label();

Label labCompany = new Label();

Label labPlane = new Label();

Label labText1 = new Label();

Label labText2 = new Label();

Label labText3 = new Label();

public Form2()

{

InitializeComponent();

npgsqlConnection = new NpgsqlConnection();

npgComm = new NpgsqlCommand();

npgComm.Connection = npgsqlConnection;

npgsqlConnection.ConnectionString = connectionString;

LoadNameTables();

LoadNameAct();

loadControls();

}

public Form2(int acs\_lvl)

{

InitializeComponent();

npgsqlConnection = new NpgsqlConnection();

npgComm = new NpgsqlCommand();

npgComm.Connection = npgsqlConnection;

npgsqlConnection.ConnectionString = connectionString;

this.acs\_lvl = acs\_lvl;

LoadNameTables();

LoadNameAct();

loadControls();

}

public void loadControls()

{

labWhence.Text = "Whence:";

labWhence.AutoSize = true;

labWhither.Text = "Whither:";

labWhither.AutoSize = true;

labDeparture.Text = "Departure:";

labDeparture.AutoSize = true;

labArrival.Text = "Arrival:";

labArrival.AutoSize = true;

labCompany.Text = "Company:";

labCompany.AutoSize = true;

labPlane.Text = "AirPlane:";

labPlane.AutoSize = true;

myButt.AutoSize = true;

labText1.AutoSize = true;

labText2.AutoSize = true;

labText3.AutoSize = true;

myButt.Click += new System.EventHandler(myButt\_Click);

}

public void LoadNameCity()

{

npgsqlConnection.Open();

npgComm.CommandText = "select name from dir\_city";

NpgsqlDataReader dr = npgComm.ExecuteReader();

myPlane.Items.Clear();

while (dr.Read())

{

string nameComp = dr["name"].ToString();

myWhence.Items.Add(nameComp);

myWhither.Items.Add(nameComp);

}

npgsqlConnection.Close();

}

public void LoadNameComp()

{

npgsqlConnection.Open();

npgComm.CommandText = "select name from dir\_company";

NpgsqlDataReader dr = npgComm.ExecuteReader();

myCompany.Items.Clear();

while (dr.Read())

{

string nameComp = dr["name"].ToString();

myCompany.Items.Add(nameComp);

}

npgsqlConnection.Close();

}

public void LoadNamePlane()

{

npgsqlConnection.Open();

npgComm.CommandText = "select name from dir\_plane";

NpgsqlDataReader dr = npgComm.ExecuteReader();

myPlane.Items.Clear();

while (dr.Read())

{

string nameComp = dr["name"].ToString();

myPlane.Items.Add(nameComp);

}

npgsqlConnection.Close();

}

public void LoadNameTables()

{

npgsqlConnection.Open();

npgComm.CommandText = "select table\_Name from information\_Schema.Tables where table\_schema='public'";

NpgsqlDataReader dr = npgComm.ExecuteReader();

comboBox1.Items.Clear();

while (dr.Read())

{

nameTable = dr["TABLE\_NAME"].ToString();

comboBox1.Items.Add(nameTable);

}

npgsqlConnection.Close();

}

public void LoadNameAct()

{

comboBox2.Items.Add("Показать");

if (acs\_lvl > 1)

{

comboBox2.Items.Add("Добавить");

comboBox2.Items.Add("Удалить");

comboBox2.Items.Add("Изменить");

if (acs\_lvl > 2)

{

comboBox2.Items.Add("Dump");

comboBox2.Items.Add("Load");

}

}

}

public void showTable()

{

if (nameTable == "flights")

{

try

{

string commSellect = "select \* from flights";

//string commSellect = "select flights.id as " + '"' + "id" + '"' + ", whence, whither, departure\_time,";

//commSellect += "arrival\_time, dir\_company.name as " + '"' + "company" + '"' + ", dir\_plane.name";

//commSellect += " as " + '"' + "airplane" + '"' + ", platform from flights join dir\_company on ";

//commSellect += "flights.id\_company = dir\_company.id join dir\_plane on flights.id\_airplane = dir\_plane.id";

npgsqlConnection.Open();

dataAdapter = new NpgsqlDataAdapter(commSellect, npgsqlConnection);

NpgsqlCommandBuilder commandBuilder = new NpgsqlCommandBuilder(dataAdapter);

dataSet = new DataSet();

dataAdapter.Fill(dataSet, nameTable);

dataGridView1.DataSource = dataSet.Tables[0];

npgsqlConnection.Close();

}

catch (Exception ex) { label4.Text = "Log: Show fails"; MessageBox.Show(ex.ToString()); }

}

else

{

try

{

npgsqlConnection.Open();

dataAdapter = new NpgsqlDataAdapter("select \* from " + nameTable, npgsqlConnection);

NpgsqlCommandBuilder commandBuilder = new NpgsqlCommandBuilder(dataAdapter);

dataSet = new DataSet();

dataAdapter.Fill(dataSet, nameTable);

dataGridView1.DataSource = dataSet.Tables[0];

npgsqlConnection.Close();

label4.Text = "Log: Таблица отображена";

}

catch (Exception ex) { label4.Text = "Log: Show fails"; MessageBox.Show(ex.ToString()); }

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

nameTable = comboBox1.SelectedItem.ToString();

showTable();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

panel1.Controls.Clear();

nameAct = comboBox2.SelectedItem.ToString();

if (nameAct == "Показать")

{

showTable();

}

if (nameAct == "Добавить")

{

myButt.Text = "Добавить";

if (nameTable == "flights")

{

LoadNameCity();

LoadNameComp();

LoadNamePlane();

labText1.Text = "Platform:";

labWhence.Location = new Point(5, 7);

myWhence.Location = new Point(65, 3);

labWhither.Location = new Point(5, 32);

myWhither.Location = new Point(65, 28);

labDeparture.Location = new Point(5, 57);

myDeparture.Location = new Point(65, 53);

labArrival.Location = new Point(5, 82);

myArrival.Location = new Point(65, 78);

labCompany.Location = new Point(5, 107);

myCompany.Location = new Point(65, 103);

labPlane.Location = new Point(5, 132);

myPlane.Location = new Point(65, 128);

labText1.Location = new Point(5, 157);

myText1.Location = new Point(65, 153);

myButt.Location = new Point(65, 178);

panel1.Controls.Add(labWhence);

panel1.Controls.Add(myWhence);

panel1.Controls.Add(labWhither);

panel1.Controls.Add(myWhither);

panel1.Controls.Add(labDeparture);

panel1.Controls.Add(myDeparture);

panel1.Controls.Add(labArrival);

panel1.Controls.Add(myArrival);

panel1.Controls.Add(labCompany);

panel1.Controls.Add(myCompany);

panel1.Controls.Add(labPlane);

panel1.Controls.Add(myPlane);

panel1.Controls.Add(labText1);

panel1.Controls.Add(myText1);

panel1.Controls.Add(myButt);

} else

{

try { dataAdapter.Update(dataSet, nameTable); label4.Text = "Log: Таблица сохранена"; }

catch (Exception ex) { label4.Text = "Log: update fails"; MessageBox.Show(ex.ToString()); }

}

}

if (nameAct == "Изменить")

{

myButt.Text = "Изменить";

if (nameTable == "flights")

{

LoadNameCity();

LoadNameComp();

LoadNamePlane();

labText1.Text = "Platform:";

labText2.Text = "Id:";

labText2.Location = new Point(5, 7);

myText2.Location = new Point(65, 3);

labWhence.Location = new Point(5, 32);

myWhence.Location = new Point(65, 28);

labWhither.Location = new Point(5, 57);

myWhither.Location = new Point(65, 53);

labDeparture.Location = new Point(5, 82);

myDeparture.Location = new Point(65, 78);

labArrival.Location = new Point(5, 107);

myArrival.Location = new Point(65, 103);

labCompany.Location = new Point(5, 132);

myCompany.Location = new Point(65, 128);

labPlane.Location = new Point(5, 157);

myPlane.Location = new Point(65, 153);

labText1.Location = new Point(5, 182);

myText1.Location = new Point(65, 178);

myButt.Location = new Point(65, 203);

panel1.Controls.Add(labText2);

panel1.Controls.Add(myText2);

panel1.Controls.Add(labWhence);

panel1.Controls.Add(myWhence);

panel1.Controls.Add(labWhither);

panel1.Controls.Add(myWhither);

panel1.Controls.Add(labDeparture);

panel1.Controls.Add(myDeparture);

panel1.Controls.Add(labArrival);

panel1.Controls.Add(myArrival);

panel1.Controls.Add(labCompany);

panel1.Controls.Add(myCompany);

panel1.Controls.Add(labPlane);

panel1.Controls.Add(myPlane);

panel1.Controls.Add(labText1);

panel1.Controls.Add(myText1);

panel1.Controls.Add(myButt);

}

else

{

labText1.Text = "Id:";

labText2.Text = "Поле:";

labText3.Text = "Значение:";

labText1.Location = new Point(5, 7);

myText1.Location = new Point(65, 3);

labText2.Location = new Point(5, 32);

myText2.Location = new Point(65, 28);

labText3.Location = new Point(5, 57);

myText3.Location = new Point(65, 53);

myButt.Location = new Point(65, 78);

panel1.Controls.Add(labText1);

panel1.Controls.Add(myText1);

panel1.Controls.Add(labText2);

panel1.Controls.Add(myText2);

panel1.Controls.Add(labText3);

panel1.Controls.Add(myText3);

panel1.Controls.Add(myButt);

}

}

if (nameAct == "Удалить")

{

myButt.Text = "Удалить";

labText1.Text = "Id:";

labText1.Location = new Point(5, 7);

myText1.Location = new Point(65, 3);

myButt.Location = new Point(65, 28);

panel1.Controls.Add(labText1);

panel1.Controls.Add(myText1);

panel1.Controls.Add(myButt);

}

}

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

{

if ((nameAct == "Добавить") && (nameTable == "flights"))

{

npgsqlConnection.Open();

string strSQL = "INSERT INTO flights(whence, whither, departure\_time, arrival\_time, company, airplane, platform) values("+"'"+myWhence.Text+"'"+", "+"'"+myWhither.Text+"'"+", " + "'" + myDeparture.Text + "'" + ", " + "'" + myArrival.Text + "'" + ", " + "'" + myCompany.Text + "'" + ", " + "'" + myPlane.Text + "'" + ", " + Convert.ToInt32(myText1.Text) + ")";

npgComm.CommandText = strSQL;

try

{

npgComm.ExecuteNonQuery();

label4.Text = "Log: Добавление записи успешно выполнено.";

}

catch (Exception ex)

{

label4.Text = "Log: Добавление записи провалилось";

MessageBox.Show(ex.ToString());

}

npgsqlConnection.Close();

showTable();

}

if (nameAct == "Удалить")

{

npgsqlConnection.Open();

string strSQL = "DELETE FROM "+nameTable+" where id IN ("+myText1.Text+")";

npgComm.CommandText = strSQL;

try

{

npgComm.ExecuteNonQuery();

label4.Text = "Log: Удаление записи успешно выполнено.";

}

catch (Exception ex)

{

label4.Text = "Log: Удаление записи провалилось";

MessageBox.Show(ex.ToString());

}

npgsqlConnection.Close();

showTable();

}

}

private void button9\_Click(object sender, EventArgs e)

{

saveFileDialog1.Filter = "Dump Files|\*.dump|All Files|\*.\*";

saveFileDialog1.Title = "Save a dump file";

saveFileDialog1.FileName = "New dump";

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

{

Dump(saveFileDialog1.FileName.Replace(@"\", @"\\").Replace(" ", "\" \""), "labsyst");

label1.Text = "Log: DB Saved";

}

}

private void button10\_Click(object sender, EventArgs e)

{

openFileDialog1.Filter = "Dump Files|\*.dump|All Files|\*.\*";

openFileDialog1.Title = "Select a dump file";

openFileDialog1.FileName = "";

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

{

Restore(openFileDialog1.FileName.Replace(@"\", @"\\").Replace(" ", "\" \""), "labsyst");

label1.Text = "Log: DB Opened";

}

}

}

}