МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Базы данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«Разработка *Windows* приложения базы данных в заданной предметной области»

Вариант №5

Выполнил: студент гр. ИТИ-22

Карпенко Д.Е.

Принял: преподаватель-стажёр

Карась О.В.

Гомель 2023

**Цель работы:** получить навыки использования объекты *ADO*.*NET* для создания приложений баз данных с удобным для пользователя графическим интерфейсом. Изучить технологию связывания элементов управления с источниками данных.

**Задание:**

Создать приложение баз данных с визуализацией посредством *Windows*-форм, соответствующее предметной области вашего варианта, которое будет выполнять следующие функции:

1. отображение данных, занесенных в таблицы в табличных элементах управления *Windows-* форм (формы);
2. ввод и редактирование записей таблиц;
3. удаление записей таблиц;
4. фильтрацию по одному или нескольким полям.

Форма (формы), отображающие данные таблиц, стоящих на стороне отношения «многие» схемы базы данных, должны выводить вместо числовых кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

Доступ к данным должен осуществляться средствами *ADO*.*NET*. Для визуализации данных следует использовать связывания элементов управления (типа *TextBox*, *ListBox*, *ComboBox*, *DataGridView*) с источниками объектами типа *DataSet*, *DataTable*, *DataView*.

Количество подлежащих разработке *Windows*-форм определяется исходя из требования, что они должны позволять в удобном для пользователя представлении: отображать, фильтровать, добавлять, удалять и редактирования информацию не менее чем из трех таблиц базы данных. Не менее, чем одна из таблиц должна находиться на стороне отношения «многие» схемы базы данных.

*Windows*-формы (Windows форма) должны содержать текстовые поля, поля со списком, табличные элементы управления, управляющие (командные) кнопки и надписи, возможно – меню, вкладки и др.

**Ход работы**

В ходе лабораторной были использованы средства *ADO*.*NET.* Код приложения представлен в приложении А.

На рисунке 1 представлена *ER*-диаграмма структуры БД.

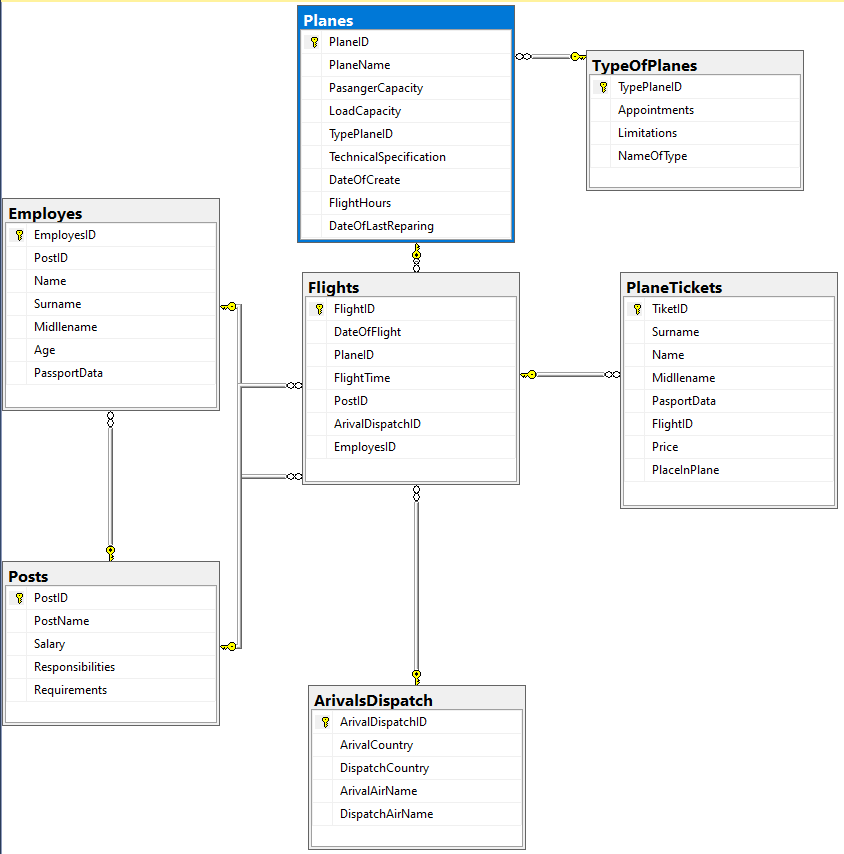
****

Рисунок 1 – *ER*-диаграмми

На рисунке 2 представлена структура таблицы «*ArivalsDispatch*»

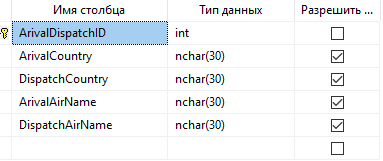


Рисунок 2 – Таблица «*ArivalsDispatch*»

На рисунке 3 представлена структура таблицы «*EmployesID*»

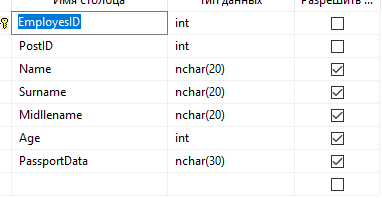


Рисунок 3 – Таблица «*EmployesID*»

На рисунке 4 представлена структура таблицы «*Flights*»

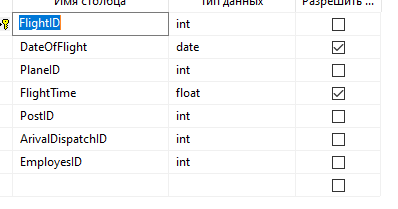
****

Рисунок 4 – Таблица «*Flights*»

На рисунке 5 представлена структура таблицы «*Planes*»

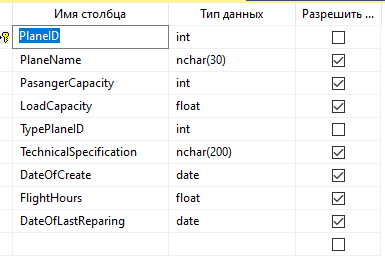


Рисунок 5 – Таблица «*Planes*»

На рисунке 6 представлена структура таблицы «*PlaneTikets*»

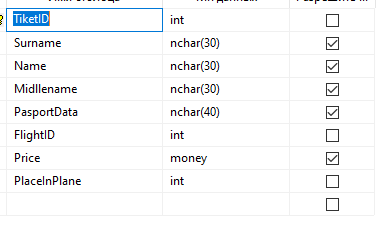


Рисунок 6 – Таблица «*PlaneTikets*»

На рисунке 7 представлена структура таблицы «*Posts*»

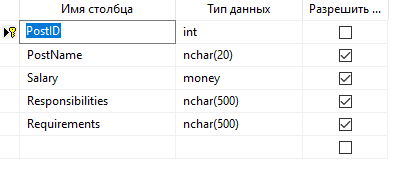


Рисунок 7 – Таблица «*Posts*»

На рисунке 8 представлена структура таблицы «*TypeOfPlanes*»

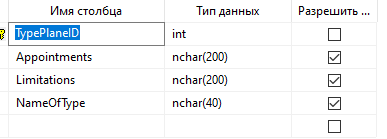


Рисунок 8 – Таблица «*TypeOfPlanes*»

На рисунке 8 представлена структура БД «*Airlines*»

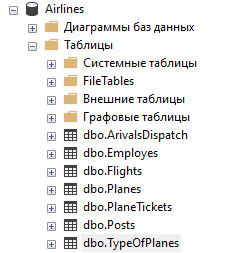


Рисунок 9 – Структура БД «*Airlines*»

На рисунке 10 представлены исходные данные заполненной таблицы «*ArivalDispatch*». Данная таблица является вспомогательной и в задании к ЛР её нету, в следствии этого она заполнена на 7 строк.

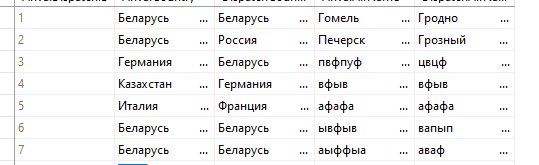


Рисунок 10 – Данные таблицы « *ArivalDispatch* »

На рисунке 11 представлены исходные данные заполненной таблицы «*Employes*».

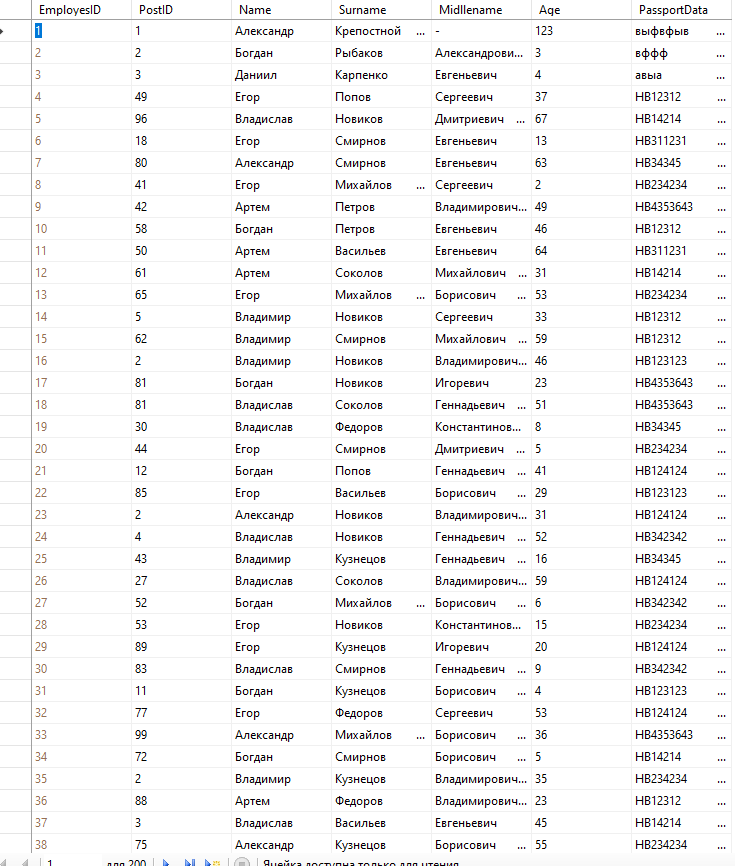


Рисунок 11 – Данные таблицы «*Employes*»

На рисунке 12 представлены исходные данные заполненных таблиц «*Flights*» и «*Planes*».

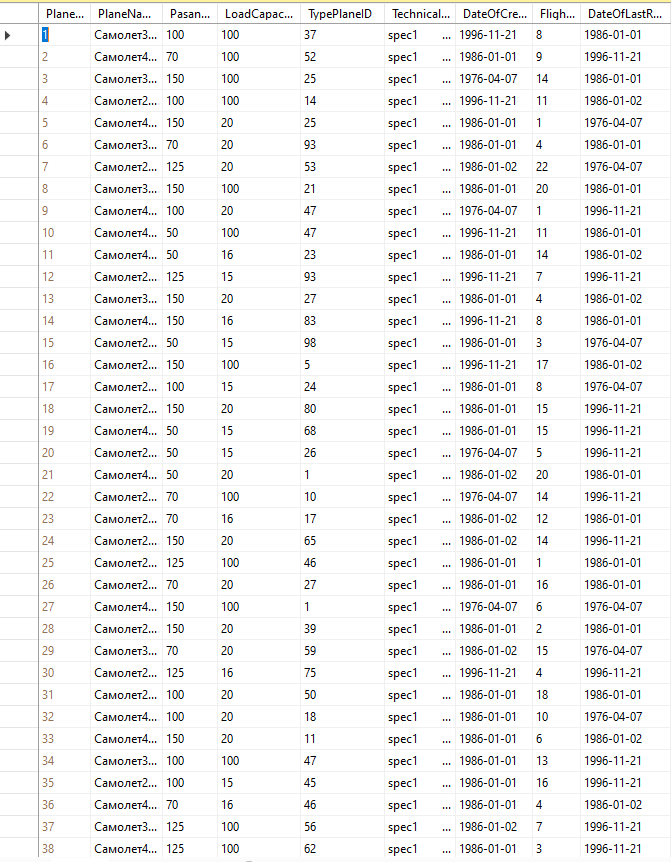
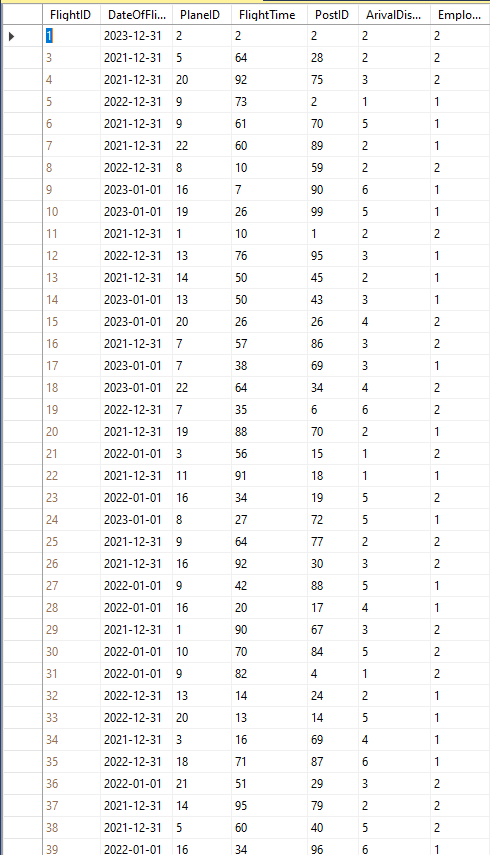


Рисунок 12 – Данные таблиц «*Flights*» и «*Planes*»

На рисунке 13 представлены исходные данные заполненной таблицы «*PlaneTkets*» и «*Posts*».

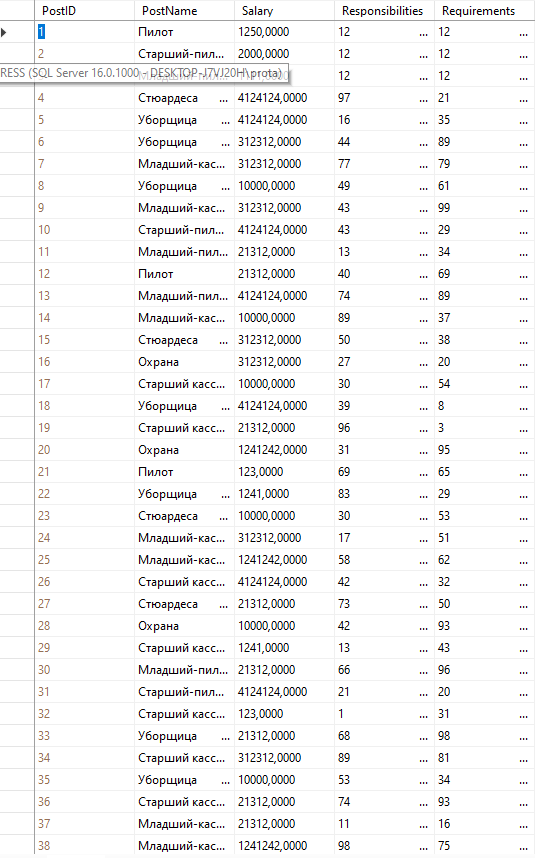


Рисунок 13 – Данные таблиц «*PlaneTkets*» и «*Posts*»

На рисунке 14 представлены исходные данные заполненной таблицы «*TypesOfPlanes*».

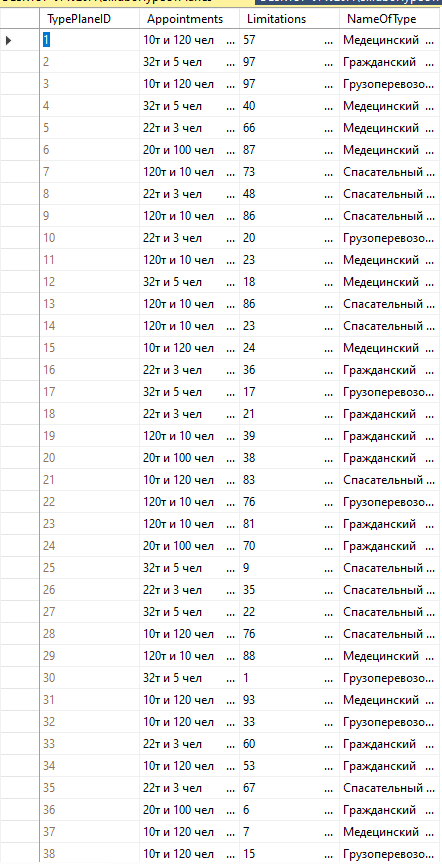


Рисунок 14 – Данные таблиц «*TypesOfPlanes*»

На рисунке 15 представлено главное меню программы c начальным выводом таблицы «*PlaneTikcets*».

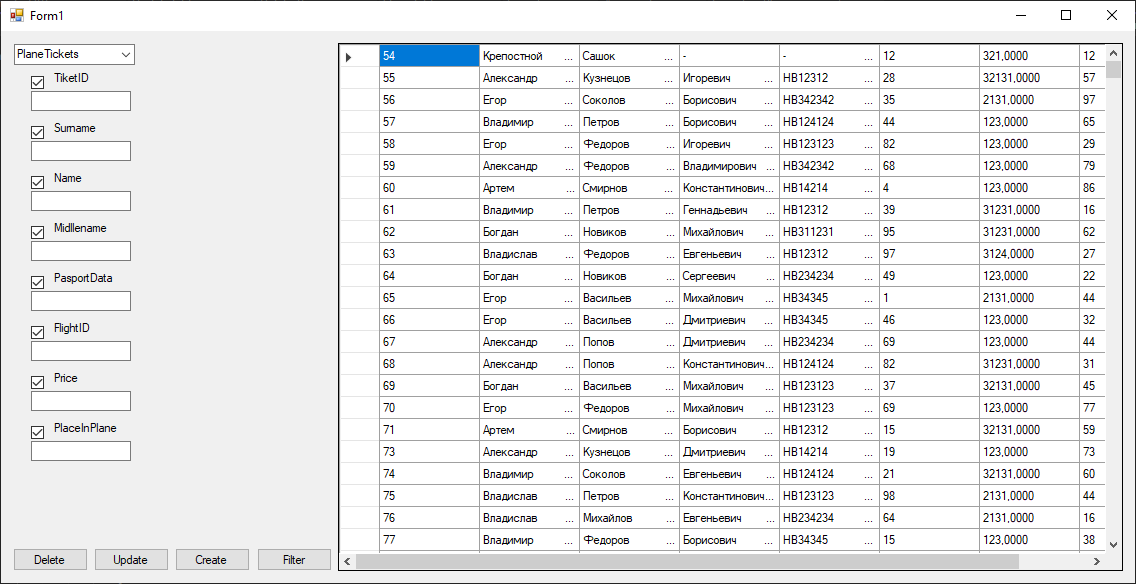


Рисунок 15 – Главное меню

На рисунке 16 представлен выбор таблицы, которую необходимо вывести и провести действия на удаление, добавление, обновление или фильтрацию.

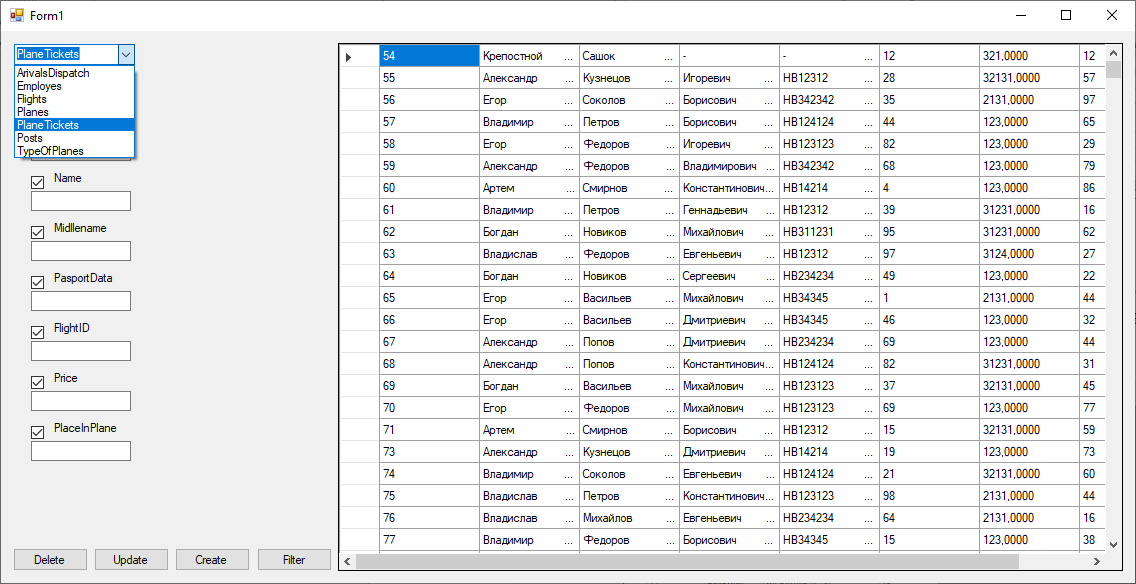


Рисунок 16 – Выбор таблицы

На рисунке 17 представлена работа фильтрации таблицы «*Doctors*» по полю «*Town*» из таблицы «*PlaneTikcets*»

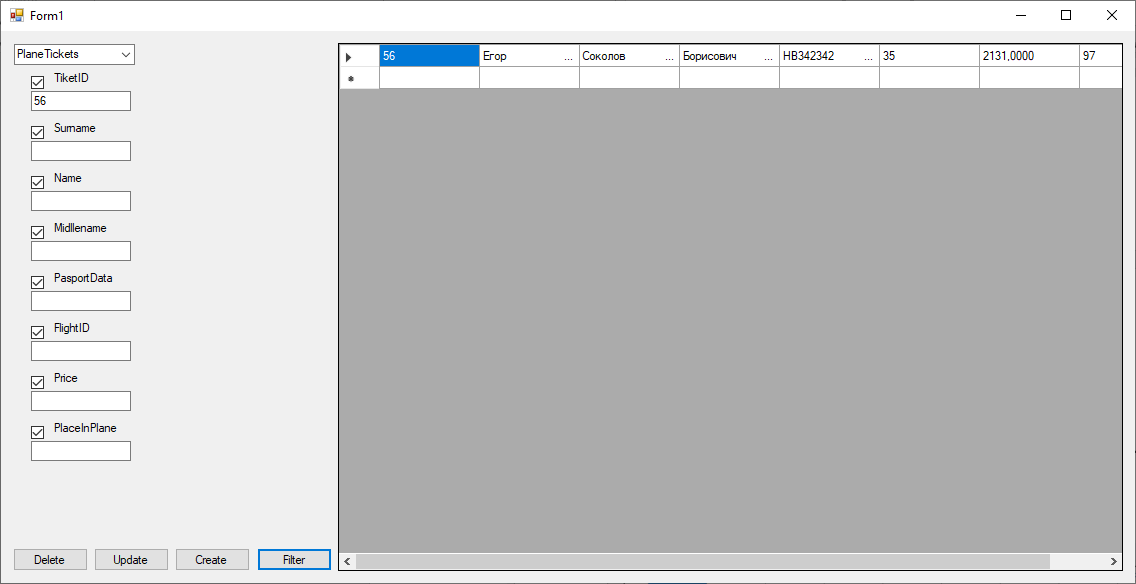


Рисунок 17 – Фильтрация по одному полю

На рисунке 19 представлена работа фильтрации таблицы «*Doctors*» по полю «*Town*» и «*House*» из таблицы « *PlaneTikcets* »

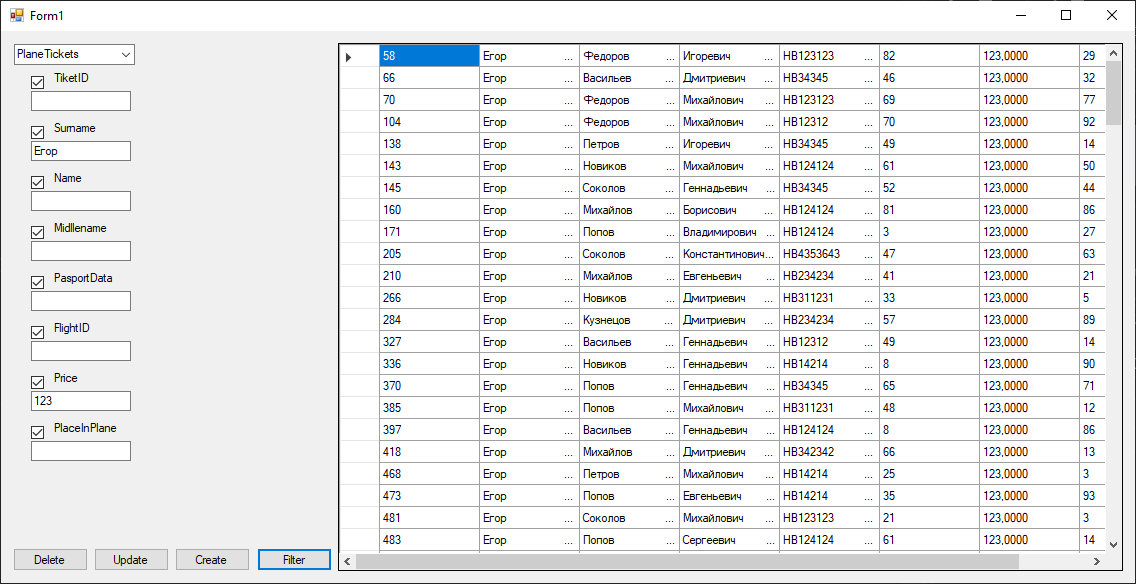


Рисунок 18 – Фильтрация по нескольким полям

На рисунке 20 представлено удаление выделенных строк в таблице по нажатию кнопки «*Delete*»

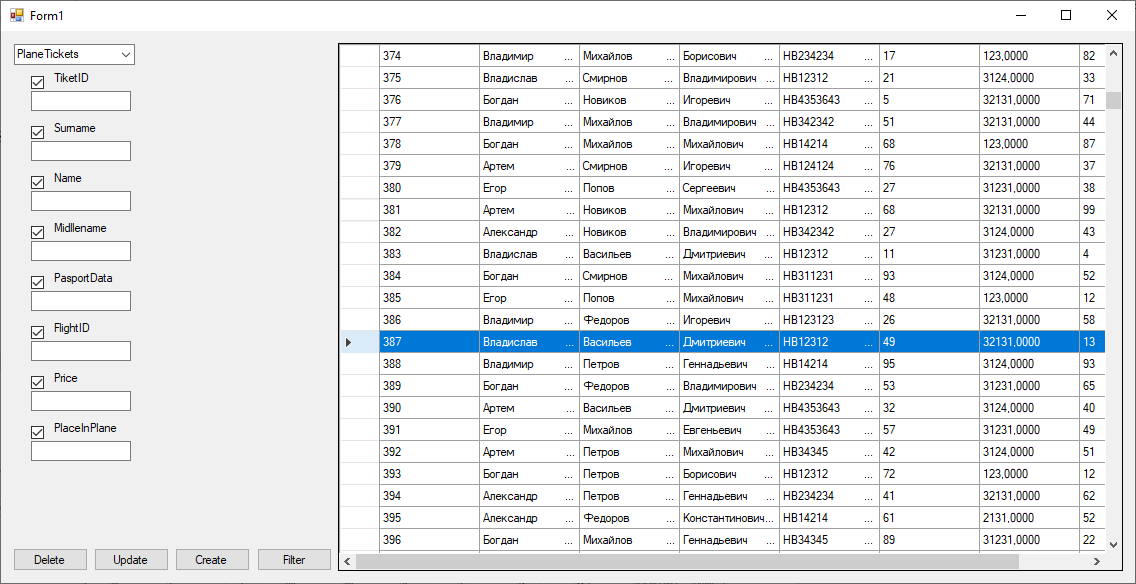


Рисунок 20 – Подготовка к удалению

На рисунке 21 представлена таблицы «*PlaneTikcets*» после завершения удаления выбранных записей.

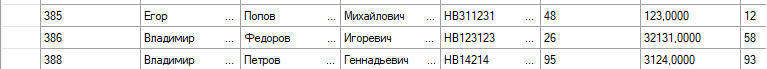


Рисунок 21 – Таблица после удаления записей

На рисунке 22 представлено открытие окна добавления и обновления строки в таблице «*PlaneTikcets*» по нажатию кнопки «*Confirm*» на главном окне.

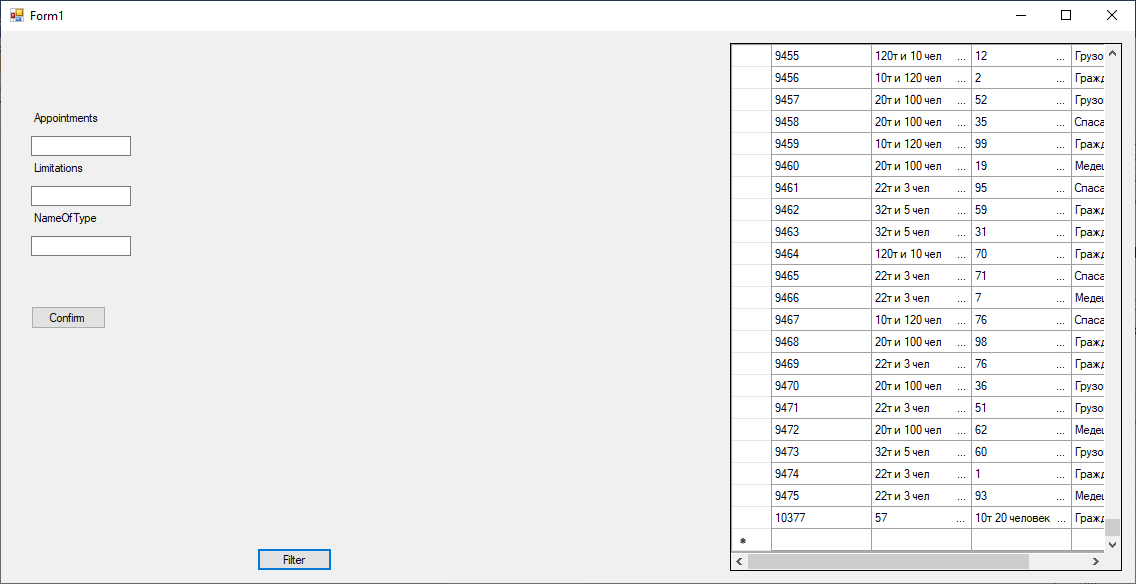


Рисунок 22 – Окно взаимодействия с таблицей

На рисунке 23 представлено заполненное окно данными для добавления записи.

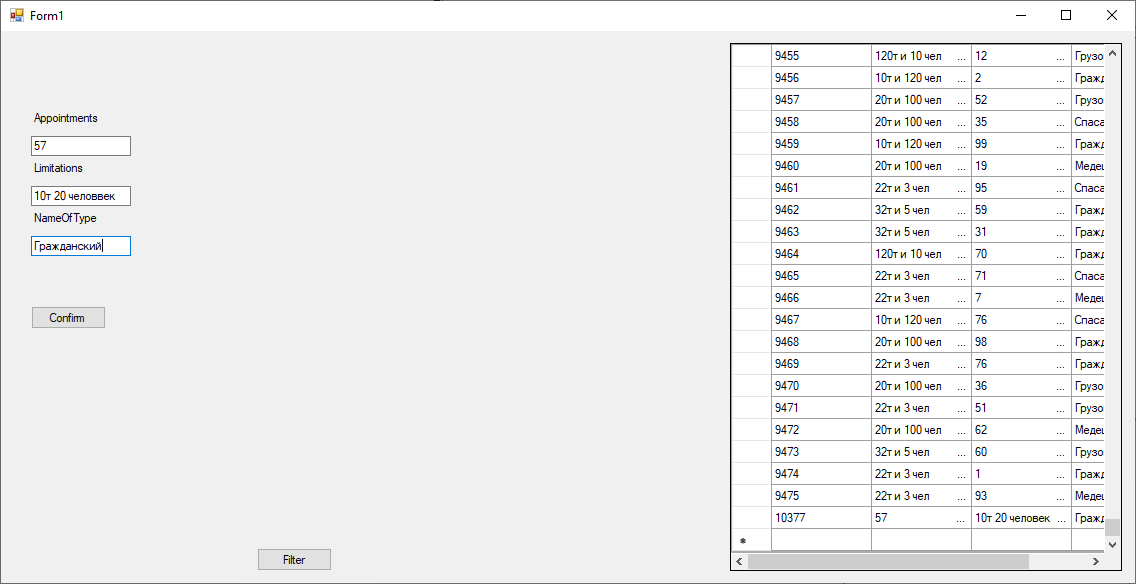


Рисунок 23 – Подготовка к добавлению записи

На рисунке 24 представлена добавленная запись после обновления таблицы.



Рисунок 24 – Добавленная запись

На рисунке 25 представлен процесс подготовки к обновлению выбранной записи на те, что ввел пользователь.

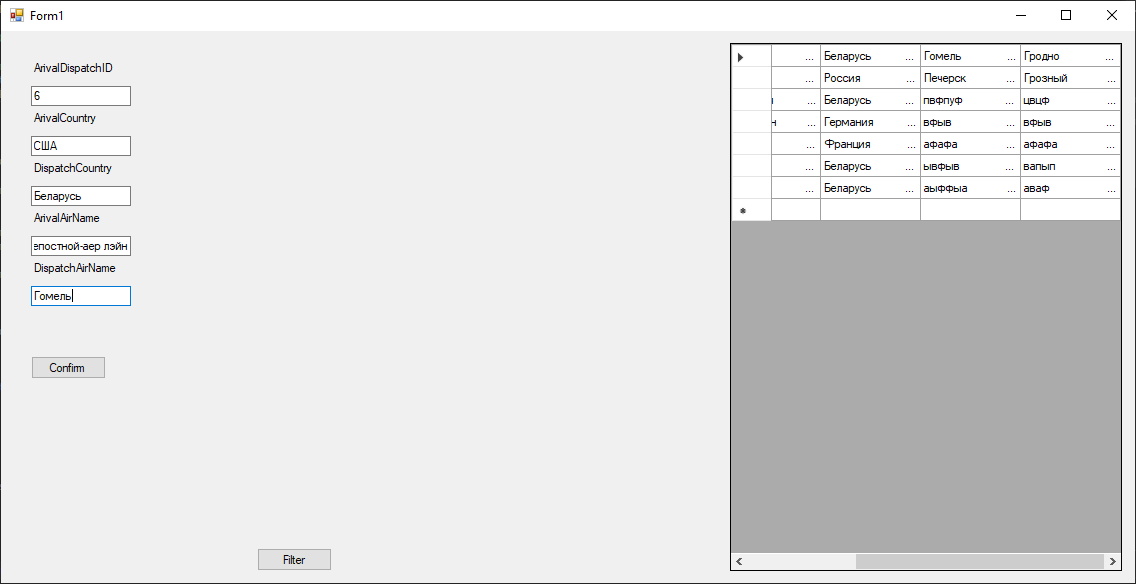


Рисунок 25 – Подготовка к обновлению записи

На рисунке 26 представлена обновленная запись и информация об обновлении.

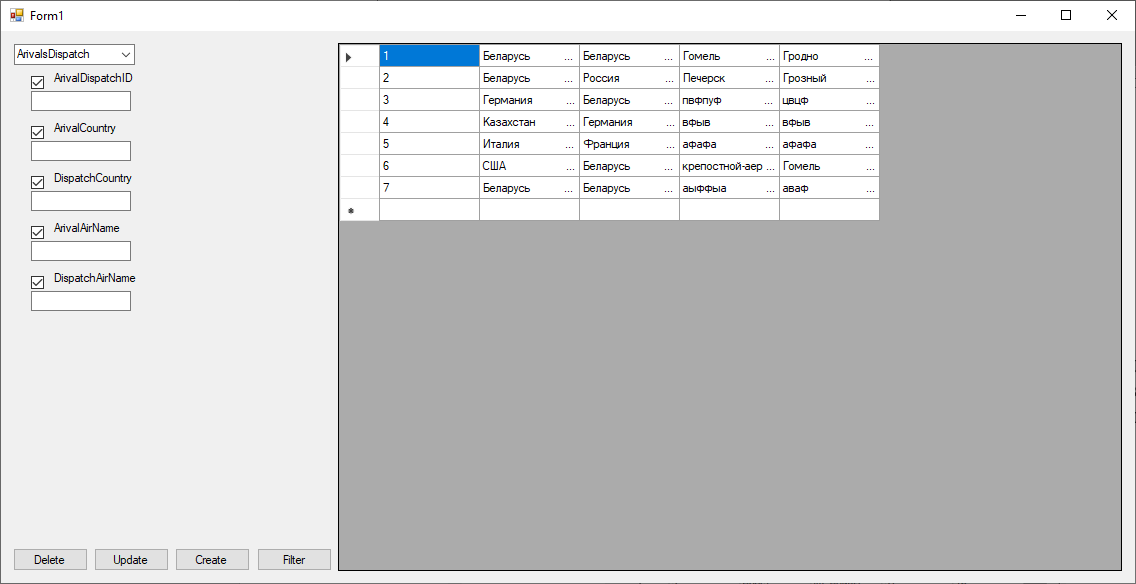


Рисунок 26 – Обновленная запись

**Вывод**: в результате выполнения лабораторной работы были получены знания и навыки в использовании объектов *ADO*.*NET* для создания приложений баз данных с удобным для пользователя графическим интерфейсом. Также были изучены технология связывания элементов управления с источниками данных.

**ПРИЛОЖЕНИЕ А**

**Текст скриптов**

**Файл *EmployesInsert.sql*:**

CREATE PROCEDURE InsertInformationEmployes(@Name nchar(20),@Surname nchar(20), @Midellname nchar(20), @Age int, @PasportData nchar(30), @PostID int)

AS

INSERT INTO Employees([Name], Surname, Midllename, Age, PassportData, PostID)

VALUES (@Name, @Surname, @Midellname, @Age, @PasportData, @PostID)

**Файл *FlightsInsert.sql*:**

CREATE PROCEDURE FlightsInsert(@DateOfFlight date, @ArivalDispatchID int, @PlaneID int, @FlightTime float, @PostID int)

AS

INSERT INTO Flights (DateOfFlight, ArivalDispatchID,PlaneID, FlightTime, PostID)

VALUES (@DateOfFlight, @ArivalDispatchID, @PlaneID, @FlightTime, @PostID)

**Файл *PlanesInsert.sql*:**

CREATE PROCEDURE PlanesInsert(@PlaneName nchar(40), @PasangerCapacity int, @LoadCapacity float, @TypePlaneID int, @TechnicalSpecification nchar(200),

@DateOfCreate date, @FlightHours float, @DateOfLastReparing date)

AS

INSERT INTO Planes (PlaneName, PasangerCapacity, LoadCapacity, TypePlaneID, TechnicalSpecification, DateOfCreate, FlightHours, DateOfLastReparing)

VALUES (@PlaneName, @PasangerCapacity, @LoadCapacity, @TypePlaneID, @TechnicalSpecification, @DateOfCreate, @FlightHours, @DateOfLastReparing)

**Файл *InsertPlaneTickets.sql*:**

CREATE PROCEDURE InsertPlaneTickets(@Surname nchar(30), @Name nchar(30), @Medellname nchar(30), @PasportData nchar(40), @FlightID int, @Price money, @PlaceInPlane int)

AS

INSERT INTO PlaneTickets(Surname, [Name], Midllename, PasportData, FlightID, Price, PlaceInPlane)

VALUES (@Surname, @Name, @Medellname, @PasportData, @FlightID, @Price, @PlaceInPlane)

**Файл *InsertPost.sql*:**

CREATE PROCEDURE InsertPost(@NameOfPost nchar(20), @Salary money, @Responsibilities nchar(500), @Requirements nchar(500))

AS

INSERT INTO Posts(PostName, Salary, Responsibilities, Requirements)

VALUES (@NameOfPost, @Salary, @Responsibilities, @Requirements)

**Файл *TypeOfPlanesInsert.sql*:**

CREATE PROCEDURE TypeOfPlanesInsert(@NameOfType nchar(30), @Appointments nchar(500), @Limitations nchar(500))

AS

INSERT INTO TypeOfPlanes(NameOfType, Appointments, Limitations)

VALUES (@NameOfType, @Appointments, @Limitations)

**Файл *DeleteFlights.sql*:**

CREATE PROCEDURE DeleteFlights(@FlightID int)

AS

DELETE FROM Flights

WHERE FlightID = @FlightID

**Файл *UpdateFlights.sql*:**

CREATE PROCEDURE UpdateFlights(@FlightID int, @DateOfFlight date,@ArivalDispatchID int,@PlaneID int, @FlightTime float, @PostID int, @EmployesID int)

AS

UPDATE Flights

SET DateOfFlight = @DateOfFlight, PlaneID = @PlaneID, FlightTime = @FlightTime, PostID = @PostID, EmployesID = @EmployesID, ArivalDispatchID = @ArivalDispatchID

**Файл *Main.cs*:**

**using Logic;**

**using System;**

**using System.Collections.Generic;**

**using System.Data;**

**using System.Windows.Forms;**

**namespace Laba\_6\_DB**

**{**

**public partial class Form1 : Form**

**{**

**int oldIndex;**

**ShowTables showTables = new ShowTables(@"Data Source= DESKTOP-J7VJ20H\SQLEXPRESS;Initial Catalog=Airlines;Integrated Security=True");**

**System.Timers.Timer timer = new System.Timers.Timer();**

**System.Timers.Timer chekerOfFilter = new System.Timers.Timer();**

**public Form1()**

**{**

**InitializeComponent();**

**InitializetionMainTable();**

**ChoseTable.SelectedIndex = 4;**

**timer = new System.Timers.Timer(500);**

**timer.Enabled = true;**

**timer.Elapsed += Timer\_Elapsed;**

**}**

**private void Timer\_Elapsed(object sender, System.Timers.ElapsedEventArgs e)**

**{**

**if (ChoseTable.InvokeRequired)**

**{**

**ChoseTable.Invoke(new Action(() =>**

**{**

**int index = ChoseTable.SelectedIndex;**

**// действия с полученным индексом**

**if (oldIndex != index)**

**{**

**oldIndex = index;**

**FillWFATable(index);**

**}**

**}));**

**}**

**else**

**{**

**int index = ChoseTable.SelectedIndex;**

**// действия с полученным индексом**

**if (oldIndex != index)**

**{**

**oldIndex = index;**

**FillWFATable(index);**

**}**

**}**

**}**

**private void DeleteFilter()**

**{**

**int j = 0;**

**int k = 0;**

**int g = 0;**

**for (int i = 0; i < this.Controls.Count; i++)**

**{**

**if (this.Controls[i].Name == "chekBox" + j)**

**{**

**this.Controls.RemoveAt(i);**

**j++;**

**i--;**

**}**

**}**

**for (int i = 0; i < this.Controls.Count; i++)**

**{**

**if (this.Controls[i].Name == "textBox" + k)**

**{**

**this.Controls.RemoveAt(i);**

**k++;**

**i--;**

**}**

**}**

**for (int i = 0; i < this.Controls.Count; i++)**

**{**

**if (this.Controls[i].Name == "label" + g)**

**{**

**this.Controls.RemoveAt(i);**

**g++;**

**i--;**

**}**

**}**

**}**

**private void CreateChekBoxs(List<string> columsNames)**

**{**

**DeleteFilter();**

**for (int i = 0; i < columsNames.Count; i++)**

**{**

**System.Windows.Forms.TextBox textBox = new System.Windows.Forms.TextBox();**

**textBox.Name = "textBox" + i;**

**textBox.Location = new System.Drawing.Point(30, 60 + 50 \* i);**

**this.Controls.Add(textBox);**

**}**

**for (int i = 0; i < columsNames.Count; i++)**

**{**

**Label label = new Label();**

**label.Name = "label" + i;**

**label.Text = columsNames[i];**

**label.Location = new System.Drawing.Point(50, 40 + 50 \* i);**

**this.Controls.Add(label);**

**}**

**for (int i = 0; i < columsNames.Count; i++)**

**{**

**CheckBox checkBox = new CheckBox();**

**checkBox.Name = "chekBox" + i;**

**checkBox.Checked = true;**

**checkBox.Location = new System.Drawing.Point(30, 40 + 50 \* i);**

**this.Controls.Add(checkBox);**

**}**

**}**

**List<string> columnsName;**

**private void ShowTable(DataTable dataTable)**

**{**

**AirlinesGridView.DataSource = dataTable;**

**AirlinesGridView.AutoGenerateColumns = true;**

**List<string> columnsName = new List<string>();**

**foreach (DataGridViewColumn column in AirlinesGridView.Columns)**

**{**

**columnsName.Add(column.Name);**

**}**

**this.columnsName = columnsName;**

**CreateChekBoxs(columnsName);**

**}**

**private void FillWFATable(int index)**

**{**

**DataTable dt = new DataTable();**

**switch (index)**

**{**

**case 0:**

**dt = showTables.ShowArivalsDispatchTable();**

**ShowTable(dt);**

**break;**

**case 1:**

**dt = showTables.ShowEmployesTable();**

**ShowTable(dt);**

**break;**

**case 2:**

**dt = showTables.ShowFlightsTable();**

**ShowTable(dt);**

**break;**

**case 3:**

**dt = showTables.ShowPlanesTable();**

**ShowTable(dt);**

**break;**

**case 4:**

**dt = showTables.ShowTkitTable();**

**ShowTable(dt);**

**break;**

**case 5:**

**dt = showTables.ShowPostsTable();**

**ShowTable(dt);**

**break;**

**case 6:**

**dt = showTables.ShowTypeOfPlanesTable();**

**ShowTable(dt);**

**break;**

**}**

**}**

**private void InitializetionMainTable()**

**{**

**DataTable dt = showTables.ShowTkitTable();**

**AirlinesGridView.DataSource = dt;**

**AirlinesGridView.AutoGenerateColumns = true;**

**}**

**private void DeleteButton\_Click(object sender, EventArgs e)**

**{**

**DataGridViewRow row = AirlinesGridView.CurrentRow;**

**DataGridViewCell cell = row.Cells[0];**

**int rowIndex = Convert.ToInt32(cell.Value);**

**DeleterRow deleter = new DeleterRow(@"Data Source= DESKTOP-J7VJ20H\SQLEXPRESS;Initial Catalog=Airlines;Integrated Security=True");**

**DeleteFromTable(deleter, rowIndex);**

**FillWFATable(oldIndex);**

**}**

**private void DeleteFromTable(DeleterRow deleterRow, int rowIndex)**

**{**

**switch (oldIndex)**

**{**

**case 0:**

**deleterRow.DeleteFormArivalsDispatch(rowIndex);**

**break;**

**case 1:**

**deleterRow.DeleteFormEmployes(rowIndex);**

**break;**

**case 2:**

**deleterRow.DeleteFormFlights(rowIndex);**

**break;**

**case 3:**

**deleterRow.DeleteFormPlanes(rowIndex);**

**break;**

**case 4:**

**deleterRow.DeleteFormPlaneTickets(rowIndex);**

**break;**

**case 5:**

**deleterRow.DeleteFormPosts(rowIndex);**

**break;**

**case 6:**

**deleterRow.DeleteFormTypeOfPlanes(rowIndex);**

**break;**

**}**

**}**

**private void TurnOfTools()**

**{**

**AirlinesGridView.Width = AirlinesGridView.Width / 2;**

**AirlinesGridView.Location = new System.Drawing.Point(AirlinesGridView.Location.X + AirlinesGridView.Width, AirlinesGridView.Location.Y);**

**UpdateButton.Visible = false;**

**DeleteButton.Visible = false;**

**CreateButton.Visible = false;**

**ChoseTable.Visible = false;**

**DeleteFilter();**

**}**

**private void TurnOnTools()**

**{**

**AirlinesGridView.Width = AirlinesGridView.Width \* 2;**

**AirlinesGridView.Location = new System.Drawing.Point(AirlinesGridView.Location.X - AirlinesGridView.Width / 2, AirlinesGridView.Location.Y);**

**UpdateButton.Visible = true;**

**DeleteButton.Visible = true;**

**CreateButton.Visible = true;**

**ChoseTable.Visible = true;**

**FillWFATable(oldIndex);**

**}**

**int count;**

**List<string> nameOfColumns = new List<string>();**

**List<string> data = new List<string>();**

**private void UpdateButton\_Click(object sender, EventArgs e)**

**{**

**TurnOfTools();**

**List<string> columnsName = new List<string>();**

**foreach (DataGridViewColumn column in AirlinesGridView.Columns)**

**{**

**columnsName.Add(column.Name);**

**}**

**count = columnsName.Count;**

**nameOfColumns = columnsName;**

**for (int i = 0; i < columnsName.Count; i++)**

**{**

**Label label = new Label();**

**label.Name = "label" + i;**

**label.Text = columnsName[i];**

**label.Visible = true;**

**label.Location = new System.Drawing.Point(30, 30 + 50 \* i);**

**this.Controls.Add(label);**

**}**

**for (int i = 0; i < columnsName.Count; i++)**

**{**

**System.Windows.Forms.TextBox textBox = new System.Windows.Forms.TextBox();**

**textBox.Name = "text" + i;**

**textBox.Visible = true;**

**textBox.Location = new System.Drawing.Point(30, 55 + 50 \* i);**

**this.Controls.Add(textBox);**

**}**

**System.Windows.Forms.Button button = new System.Windows.Forms.Button();**

**button.Location = new System.Drawing.Point(30, 75 + 50 \* columnsName.Count);**

**button.Name = "button1";**

**button.Text = "Confirm";**

**button.Click += Button\_Click;**

**this.Controls.Add(button);**

**}**

**private void Button\_Click(object sender, EventArgs e)**

**{**

**int k = 0, j = 0;**

**UpdateTable updateTable = new UpdateTable(@"Data Source= DESKTOP-J7VJ20H\SQLEXPRESS;Initial Catalog=Airlines;Integrated Security=True");**

**for (int i = 0; i < this.Controls.Count; i++)**

**{**

**if (this.Controls[i].Name == "text" + j)**

**{**

**data.Add(this.Controls[i].Text);**

**j++;**

**}**

**}**

**j = 0;**

**updateTable.UpdateTables(data, ChoseTable.Text, nameOfColumns);**

**for (int i = 0; i < this.Controls.Count; i++)**

**{**

**if (this.Controls[i].Name == "label" + k)**

**{**

**this.Controls.RemoveAt(i);**

**k++;**

**i = 0;**

**}**

**if (this.Controls[i].Name == "text" + j)**

**{**

**this.Controls.RemoveAt(i);**

**j++;**

**i = 0;**

**}**

**if (this.Controls[i].Name == "button1")**

**{**

**this.Controls.RemoveAt(i);**

**}**

**}**

**TurnOnTools();**

**}**

**private void CreateButton\_Click(object sender, EventArgs e)**

**{**

**TurnOfTools();**

**List<string> columnsName = new List<string>();**

**foreach (DataGridViewColumn column in AirlinesGridView.Columns)**

**{**

**columnsName.Add(column.Name);**

**}**

**count = columnsName.Count;**

**nameOfColumns = columnsName;**

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

**{**

**Label label = new Label();**

**label.Name = "label" + i;**

**label.Text = columnsName[i];**

**label.Visible = true;**

**label.Location = new System.Drawing.Point(30, 30 + 50 \* i);**

**this.Controls.Add(label);**

**}**

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

**{**

**System.Windows.Forms.TextBox textBox = new System.Windows.Forms.TextBox();**

**textBox.Name = "text" + i;**

**textBox.Visible = true;**

**textBox.Location = new System.Drawing.Point(30, 55 + 50 \* i);**

**this.Controls.Add(textBox);**

**}**

**System.Windows.Forms.Button button = new System.Windows.Forms.Button();**

**button.Location = new System.Drawing.Point(30, 75 + 50 \* columnsName.Count);**

**button.Name = "button1";**

**button.Text = "Confirm";**

**button.Click += Button\_Click1; ;**

**this.Controls.Add(button);**

**}**

**private void Button\_Click1(object sender, EventArgs e)**

**{**

**int k = 1, j = 1;**

**CreaterOfRows createrOfRows = new CreaterOfRows(@"Data Source= DESKTOP-J7VJ20H\SQLEXPRESS;Initial Catalog=Airlines;Integrated Security=True");**

**for (int i = 0; i < this.Controls.Count; i++)**

**{**

**if (this.Controls[i].Name == "text" + j)**

**{**

**data.Add(this.Controls[i].Text);**

**j++;**

**}**

**}**

**j = 1;**

**createrOfRows.CreateRowTable(data, ChoseTable.Text, nameOfColumns);**

**for (int i = 0; i < this.Controls.Count; i++)**

**{**

**if (this.Controls[i].Name == "label" + k)**

**{**

**this.Controls.RemoveAt(i);**

**k++;**

**i = 0;**

**}**

**if (this.Controls[i].Name == "text" + j)**

**{**

**this.Controls.RemoveAt(i);**

**j++;**

**i = 0;**

**}**

**if (this.Controls[i].Name == "button1")**

**{**

**this.Controls.RemoveAt(i);**

**}**

**}**

**TurnOnTools();**

**}**

**private void FilterButton\_Click(object sender, EventArgs e)**

**{**

**Filter filter = new Filter(@"Data Source= DESKTOP-J7VJ20H\SQLEXPRESS;Initial Catalog=Airlines;Integrated Security=True");**

**List<string> columnsName = new List<string>();**

**foreach (string column in this.columnsName)**

**{**

**columnsName.Add(column);**

**}**

**int h = 0;**

**for (int k = 0; k < Controls.Count; k++)**

**{**

**if (Controls[k] is CheckBox checkBox)**

**{**

**if (!checkBox.Checked)**

**{**

**columnsName[h] = "";**

**h++;**

**}**

**}**

**}**

**string[] wordFilter = new string[columnsName.Count];**

**for (int i = 0; i < wordFilter.Length; i++)**

**{**

**wordFilter[i] = "";**

**}**

**int j = 0;**

**for (int i = 0; i < this.Controls.Count && wordFilter.Length != 0; i++)**

**{**

**if (this.Controls[i].Name == "textBox" + j)**

**{**

**wordFilter[j] = this.Controls[i].Text;**

**j++;**

**}**

**}**

**if (ChoseTable.InvokeRequired)**

**{**

**ChoseTable.Invoke(new Action(() =>**

**{**

**AirlinesGridView.Rows.Clear();**

**AirlinesGridView.DataSource = filter.FilterTable(columnsName, wordFilter, ChoseTable.Text); ;**

**AirlinesGridView.AutoGenerateColumns = true;**

**}));**

**}**

**else**

**{**

**AirlinesGridView.DataSource = filter.FilterTable(columnsName, wordFilter, ChoseTable.Text);**

**AirlinesGridView.AutoGenerateColumns = true;**

**}**

**}**

**}**

**}**

**Файл *CreaterOfRows.cs*:**

using System.Collections.Generic;

using System.Data.SqlClient;

namespace Logic

{

public class CreaterOfRows

{

string path;

public CreaterOfRows(string path) { this.path = path; }

private void AbsCreateRowInTable(string command)

{

using (SqlConnection connection = new SqlConnection(path))

{

SqlCommand query = new SqlCommand(command, connection);

connection.Open();

query.ExecuteNonQuery();

}

}

public void CreateRowTable(List<string> data, string nameTable, List<string> nameColumns)

{

string command = "INSERT INTO " + nameTable + " ( ";

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

{

if (nameColumns.Count - 1 != i)

command = command + " " + nameColumns[i] + ", ";

else

command = command + " " + nameColumns[i] + " ";

}

command = command + " ) VALUES ( ";

for (int i = 0; i < data.Count && data[i] != ""; i++)

{

if (int.TryParse(data[i], out int result) || (double.TryParse(data[i], out double res)))

{

if (data.Count - 1 == i)

{

command = command + " " + data[i] + " )";

}

else

{

command = command + " " + data[i] + ", ";

}

}

else

{

if (data.Count - 1 == i)

{

command = command + " \'" + data[i] + "\' )";

}

else

{

command = command + " \'" + data[i] + "\', ";

}

}

}

AbsCreateRowInTable(command);

}

}

}

**Файл *DeleterRow.cs*:**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Logic

{

public class DeleterRow

{

string path;

public DeleterRow(string path) { this.path = path; }

private void AbsDeleteRow(string command)

{

using (SqlConnection connection = new SqlConnection(path))

{

SqlCommand query = new SqlCommand(command, connection);

connection.Open();

query.ExecuteNonQuery();

}

}

public void DeleteFormArivalsDispatch(int id)

{

string command = "DELETE FROM ArivalsDispatch WHERE ArivalDispatchID = " + id;

AbsDeleteRow(command);

}

public void DeleteFormEmployes(int id)

{

string command = "DELETE FROM Employes WHERE EmployesID = " + id;

AbsDeleteRow(command);

}

public void DeleteFormFlights(int id)

{

string command = "DELETE FROM Flights WHERE FlightID = " + id;

AbsDeleteRow(command);

}

public void DeleteFormPlanes(int id)

{

string command = "DELETE FROM Planes WHERE PlaneID = " + id;

AbsDeleteRow(command);

}

public void DeleteFormPlaneTickets(int id)

{

string command = "DELETE FROM PlaneTickets WHERE TiketID = " + id;

AbsDeleteRow(command);

}

public void DeleteFormPosts(int id)

{

string command = "DELETE FROM Posts WHERE PostID = " + id;

AbsDeleteRow(command);

}

public void DeleteFormTypeOfPlanes(int id)

{

string command = "DELETE FROM TypeOfPlanes WHERE TypeOfPlaneID = " + id;

AbsDeleteRow(command);

}

}

}

**Файл *Filter.cs*:**

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

namespace Logic

{

public class Filter

{

string path;

public Filter(string path) { this.path = path; }

private DataTable AbsFilter(string command, string tableName)

{

DataTable dt = new DataTable(tableName);

using (SqlConnection connection = new SqlConnection(path))

{

SqlCommand query = new SqlCommand(command, connection);

SqlDataAdapter adapter = new SqlDataAdapter(query);

adapter.Fill(dt);

}

return dt;

}

public DataTable FilterTable(List<string> columnsName, string[] filtered, string tableName)

{

int count = 0;

string command = "SELECT ";

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

{

if (columnsName[i] != "")

{

if (columnsName.Count - 1 != i)

{

command = command + " " + columnsName[i] + ", ";

}

else

{

command = command + " " + columnsName[i] + " ";

}

count++;

}

}

command = command + " FROM " + tableName;

if (filtered.Length > 0)

{

int j = 0;

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

{

if (filtered[i] != "")

{

command = command + " WHERE ";

break;

}

}

double e;

int k;

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

{

if (filtered[i] != "")

{

if (int.TryParse(columnsName[i], out k) || double.TryParse(columnsName[i], out e))

{

if (j > 0 && j != filtered.Length - 1)

{

command = command + " AND ";

}

command = command + columnsName[i] + " = " + filtered[i];

j++;

}

else

{

if (j > 0 && j != filtered.Length - 1)

{

command = command + " AND ";

}

command = command + columnsName[i] + " = \'" + filtered[i] + "\'";

j++;

}

}

}

}

if(count != 0)

return AbsFilter(command, tableName);

else

return null;

}

}

}

**Файл *AddMedicalProcedures.cs*:**

**using System;**

**using System.Collections.Generic;**

**using System.ComponentModel;**

**using System.Data;**

**using System.Data.SqlClient;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**using System.Windows.Forms;**

**namespace lab6.AddRows**

**{**

**public partial class AddMedicalProcedures : Form**

**{**

**public string sqlConnection = @"Data Source=YOVOPC\SQLEXPRESS;Initial Catalog=PatientСare;Integrated Security=True";**

**public string sqlCommand;**

**public AddMedicalProcedures()**

**{**

**InitializeComponent();**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM Doctors", connection);**

**DataSet ds = new DataSet();**

**adapter.Fill(ds);**

**DataTable dt = ds.Tables[0];**

**StringBuilder sb = new StringBuilder();**

**foreach (DataColumn column in dt.Columns)**

**{**

**foreach (DataRow row in dt.Rows)**

**{**

**var cells = row.ItemArray;**

**foreach (object cell in cells)**

**sb.Append(cell + " ");**

**inputD.Items.Add(sb.ToString());**

**sb.Clear();**

**}**

**break;**

**}**

**}**

**inputD.Text = inputD.Items[0].ToString();**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM Patients", connection);**

**DataSet ds = new DataSet();**

**adapter.Fill(ds);**

**DataTable dt = ds.Tables[0];**

**StringBuilder sb = new StringBuilder();**

**foreach (DataColumn column in dt.Columns)**

**{**

**foreach (DataRow row in dt.Rows)**

**{**

**var cells = row.ItemArray;**

**foreach (object cell in cells)**

**sb.Append(cell + " ");**

**inputP.Items.Add(sb.ToString());**

**sb.Clear();**

**}**

**break;**

**}**

**}**

**inputP.Text = inputP.Items[0].ToString();**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM Diseases", connection);**

**DataSet ds = new DataSet();**

**adapter.Fill(ds);**

**DataTable dt = ds.Tables[0];**

**StringBuilder sb = new StringBuilder();**

**foreach (DataColumn column in dt.Columns)**

**{**

**foreach (DataRow row in dt.Rows)**

**{**

**var cells = row.ItemArray;**

**foreach (object cell in cells)**

**sb.Append(cell + " ");**

**inputDis.Items.Add(sb.ToString());**

**sb.Clear();**

**}**

**break;**

**}**

**}**

**inputDis.Text = inputDis.Items[0].ToString();**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter($"SELECT \* FROM Medications", connection);**

**DataSet ds = new DataSet();**

**adapter.Fill(ds);**

**DataTable dt = ds.Tables[0];**

**StringBuilder sb = new StringBuilder();**

**foreach (DataColumn column in dt.Columns)**

**{**

**foreach (DataRow row in dt.Rows)**

**{**

**var cells = row.ItemArray;**

**foreach (object cell in cells)**

**sb.Append(cell + " ");**

**inputM.Items.Add(sb.ToString());**

**sb.Clear();**

**}**

**break;**

**}**

**}**

**inputM.Text = inputM.Items[0].ToString();**

**}**

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

**{**

**output.Visible = true;**

**output.Text = "Поля ввода очищены";**

**}**

**private void add\_Click(object sender, EventArgs e)**

**{**

**output.Visible = true;**

**try**

**{**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter(sqlCommand, connection);**

**DataSet ds = new DataSet();**

**adapter.Fill(ds);**

**DataTable dt = ds.Tables[0];**

**string[] doc = inputD.Text.Split(' ');**

**DataRow newRow = dt.NewRow();**

**newRow["DoctorId"] = Convert.ToInt32(doc[0]);**

**string[] pat = inputP.Text.Split(' ');**

**newRow["PatientId"] = Convert.ToInt32(pat[0]);**

**newRow["Date"] = $"{inputDate.Value.Year}-{inputDate.Value.Month}-{inputDate.Value.Day}";**

**string[] dis = inputDis.Text.Split(' ');**

**newRow["DiseasesId"] = Convert.ToInt32(doc[0]);**

**string[] med = inputM.Text.Split(' ');**

**newRow["MedicationsId"] = Convert.ToInt32(med[0]);**

**newRow["Dosage"] = Convert.ToInt32(inputDoz.Value);**

**dt.Rows.Add(newRow);**

**SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);**

**adapter.Update(ds);**

**ds.Clear();**

**adapter.Fill(ds);**

**output.Text = $"Добавлена запись: {doc[0]} {pat[0]} {inputDate.Value.Year}-{inputDate.Value.Month}-{inputDate.Value.Day} {dis[0]} {med[0]} {inputDoz.Value}";**

**}**

**}**

**catch**

**{**

**output.Text = "Ошибка конвертации данных";**

**}**

**}**

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

**{**

**output.Visible = true;**

**try**

**{**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**string[] id = comboBox1.Text.Split(' ');**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter();**

**SqlCommand command = new SqlCommand($"UPDATE MedicalProcedures SET DoctorId = @d, PatientId = @p, Date = @date, DiseasesId = @dis, MedicationsId = @med, Dosage = @dos WHERE MedicalId = @id", connection);**

**; string[] doc = inputD.Text.Split(' ');**

**SqlParameter param1 = new SqlParameter("@d", Convert.ToInt32(doc[0]));**

**command.Parameters.Add(param1);**

**string[] pat = inputP.Text.Split(' ');**

**SqlParameter param2 = new SqlParameter("@p", Convert.ToInt32(pat[0]));**

**command.Parameters.Add(param2);**

**SqlParameter param3 = new SqlParameter("@date", $"{inputDate.Value.Year}-{inputDate.Value.Month}-{inputDate.Value.Day}");**

**command.Parameters.Add(param3);**

**string[] dis = inputDis.Text.Split(' ');**

**SqlParameter param4 = new SqlParameter("@dis", Convert.ToInt32(doc[0]));**

**command.Parameters.Add(param4);**

**SqlParameter param5 = new SqlParameter("@id", Convert.ToInt32(id[0]));**

**command.Parameters.Add(param5);**

**string[] med = inputM.Text.Split(' ');**

**SqlParameter param6 = new SqlParameter("@med", Convert.ToInt32(med[0]));**

**command.Parameters.Add(param6);**

**SqlParameter param7 = new SqlParameter("@dos", Convert.ToInt32(inputDoz.Value));**

**command.Parameters.Add(param7);**

**adapter.UpdateCommand = command;**

**adapter.UpdateCommand.ExecuteNonQuery();**

**output.Text = $"Обновлена запись: {doc[0]} {pat[0]} {inputDate.Value.Year}-{inputDate.Value.Month}-{inputDate.Value.Day} {dis[0]} {med[0]} {inputDoz.Value}";**

**}**

**comboBox1.Items.Clear();**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter(sqlCommand, connection);**

**DataSet ds = new DataSet();**

**adapter.Fill(ds);**

**dataGridView1.DataSource = ds.Tables[0];**

**DataTable dt = ds.Tables[0];**

**StringBuilder sb = new StringBuilder();**

**foreach (DataColumn column in dt.Columns)**

**{**

**foreach (DataRow row in dt.Rows)**

**{**

**var cells = row.ItemArray;**

**foreach (object cell in cells)**

**sb.Append(cell + " ");**

**comboBox1.Items.Add(sb.ToString());**

**sb.Clear();**

**}**

**break;**

**}**

**}**

**comboBox1.Text = comboBox1.Items[0].ToString();**

**}**

**catch**

**{**

**output.Text = "Произошло недрозумение";**

**}**

**}**

**private void AddMedicalProcedures\_Load(object sender, EventArgs e)**

**{**

**using (SqlConnection connection = new SqlConnection(sqlConnection))**

**{**

**connection.Open();**

**SqlDataAdapter adapter = new SqlDataAdapter(sqlCommand, connection);**

**DataSet ds = new DataSet();**

**adapter.Fill(ds);**

**dataGridView1.DataSource = ds.Tables[0];**

**DataTable dt = ds.Tables[0];**

**StringBuilder sb = new StringBuilder();**

**foreach (DataColumn column in dt.Columns)**

**{**

**foreach (DataRow row in dt.Rows)**

**{**

**var cells = row.ItemArray;**

**foreach (object cell in cells)**

**sb.Append(cell + " ");**

**comboBox1.Items.Add(sb.ToString());**

**sb.Clear();**

**}**

**break;**

**}**

**}**

**comboBox1.Text = comboBox1.Items[0].ToString();**

**}**

**}**

**}**

**Файл *ShowTables.cs*:**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Logic

{

public class ShowTables

{

string path;

public ShowTables(string path) { this.path = path; }

private DataTable AbsTable(string tableName, string command)

{

DataTable dt = new DataTable(tableName);

using (SqlConnection connection = new SqlConnection(path))

{

SqlCommand query = new SqlCommand(command, connection);

// создаем объект SqlDataAdapter для заполнения объекта DataTable данными из базы данных

SqlDataAdapter adapter = new SqlDataAdapter(query);

adapter.Fill(dt);

}

return dt;

}

public DataTable ShowTkitTable()

{

string command = "SELECT \* FROM PlaneTickets";

string tableName = "PlaneTickets";

return AbsTable(tableName, command);

}

public DataTable ShowPostsTable()

{

string command = "SELECT \* FROM Posts";

string tableName = "Posts";

return AbsTable(tableName, command);

}

public DataTable ShowTypeOfPlanesTable()

{

string command = "SELECT \* FROM TypeOfPlanes";

string tableName = "TypeOfPlanes";

return AbsTable(tableName, command);

}

public DataTable ShowPlanesTable()

{

string command = "SELECT \* FROM Planes";

string tableName = "Planes";

return AbsTable(tableName, command);

}

public DataTable ShowFlightsTable()

{

string command = "SELECT \* FROM Flights";

string tableName = "Flights";

return AbsTable(tableName, command);

}

public DataTable ShowArivalsDispatchTable()

{

string command = "SELECT \* FROM ArivalsDispatch";

string tableName = "ArivalsDispatch";

return AbsTable(tableName, command);

}

public DataTable ShowEmployesTable()

{

string command = "SELECT \* FROM Employes";

string tableName = "Employes";

return AbsTable(tableName, command);

}

}

}

**Файл *UpdateTable.cs*:**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Logic

{

public class UpdateTable

{

string path;

public UpdateTable(string path) { this.path = path; }

private void AbsUpdateTable(string command)

{

using (SqlConnection connection = new SqlConnection(path))

{

SqlCommand query = new SqlCommand(command, connection);

connection.Open();

query.ExecuteNonQuery();

}

}

public void UpdateTables(List<string> data, string nameTable, List<string> nameColumns)

{

string command = "UPDATE " + nameTable + " SET ";

for (int i = 0; i < data.Count && data[i] != ""; i++)

{

if (int.TryParse(data[i], out int result) || (double.TryParse(data[i], out double res)))

{

if (data.Count - 1 == i)

{

command = command + nameColumns[i] + " = " + data[i] + " ";

command = command + " WHERE " + nameColumns[0] + " = " + data[0];

AbsUpdateTable(command);

}

else

{

if(i != 0)

command = command + nameColumns[i] + " = " + data[i] + ", ";

}

}

else

{

if (data.Count - 1 == i)

{

command = command + nameColumns[i] + " = \'" + data[i] + "\' ";

command = command + " WHERE " + nameColumns[0] + " = " + data[0];

AbsUpdateTable(command);

}

else

{

if(i != 0)

command = command + nameColumns[i] + " = \'" + data[i] + "\', ";

}

}

}

}

}

}