МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

“Харківський авіаційний інститут”

Факультет СУЛА

Кафедра математичного моделювання та штучного інтелекту

**Лабораторна робота №2**

з Організація баз даних

на тему: «Архітектури технології ADO.NET. Сonnected layer»

Виконав: студент 3 курсу групи №335a

напряму підготовки(спеціальності):

122 Комп’ютерні науки

(шифр і назва напряму підготовки(спеціальності))

Гринюк М.О.

(прізвище й ініціали студента)

Прийняв: ст. викл., к.т.н. Коробчинський К.П.

(посада, науковий ступінь, прізвище й ініціали)

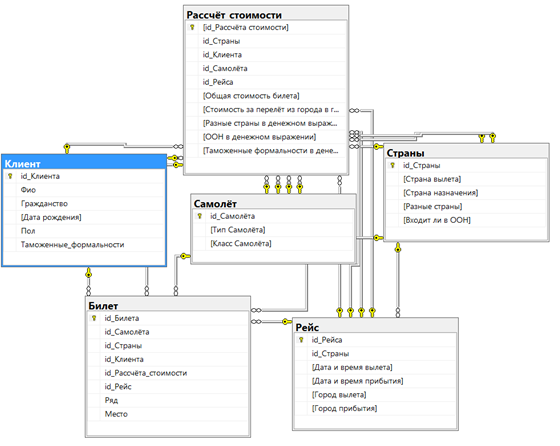
Харків – 2020

**Цель работы:** Изучить принципы работы вставки, удаления, чтения данных из базы данных с помощью класса connected layer.

**Ход работы:**

1. Пройти тест;
2. Добавить логическую модель предметной обрасти;
3. Создать добавить к своему проекту класс логирования;
4. Создать проект C#, создать **connected layer**и реализовать действия:
   1. Вставки;
   2. Чтения;
   3. Обновления;
   4. Удаления;
5. **Зайдите на сайт MSDN.**Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на занятии, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. *Сохраните ссылки и дайте им короткое описание;*
6. Для каждого пункта показать программный код и пример работы (скрин).
7. Оформить отчёт и сохранить на севере STM;
8. Уточнить сколько пар выполняется работа и когда необходимо её сдать.
9. Знать ответы на контрольные вопросы.

**Логическая модель БД**



**Выполнение работы**

1. **Создать проект C#, создать Connected Layer и реализовать действия:**
   1. Вставки;
   2. Чтения;
   3. Обновления;
   4. Удаления;

Команды представлены объектом интерфейса System.Data.IDbCommand. Провайдер для MS SQL предоставляет его реализацию в виде класса SqlCommand. Этот класс инкапсулирует sql-выражение, которое должно быть выполнено.

Чтобы выполнить команду, необходимо применить один из методов SqlCommand:

* ExecuteNonQuery: просто выполняет sql-выражение и возвращает количество измененных записей. Подходит для sql-выражений INSERT, UPDATE, DELETE.
* ExecuteReader: выполняет sql-выражение и возвращает строки из таблицы. Подходит для sql-выражения SELECT.
* ExecuteScalar: выполняет sql-выражение и возвращает одно скалярное значение, например, число. Подходит для sql-выражения SELECT в паре с одной из встроенных функций SQL, как например, Min, Max, Sum, Count.

Если мы хотим считывать данные, которые хранятся в таблице, то нам потребуется другой метод - ExecuteReader(). Этот метод возвращает объект **SqlDataReader**, который используется для чтения данных. Так, получим все данные из таблицы Users и выведем их в ListBox.

Для выполнения команды нам потребуется sql-выражение и объект подключения:

* 1. **Класс ConnectedLayer**

public class ConnectedLayer

{

public string connectionString = System.Configuration.ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;

private SqlCommand oSqlCom;

private SqlDataAdapter oSqlDtAdptr;

private DataSet ds;

Logger logger;

SqlConnectionStringBuilder sb;

//Constructor 1 : Accepts Sql server instance name,

// and the database name

public ConnectedLayer()

{

DataSet ds = new DataSet();

var oSqlCon = new SqlConnection(connectionString);

Logger logger = LogManager.GetCurrentClassLogger();

Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);

config.ConnectionStrings.ConnectionStrings.Add(

new ConnectionStringSettings("ConnectionStr", "SomeConnectionString")

);

config.Save();

ConnectionStringsSection section = config.GetSection("connectionStrings") as ConnectionStringsSection;

if (section.SectionInformation.IsProtected)

{

section.SectionInformation.UnprotectSection();

}

else

{

section.SectionInformation.ProtectSection("DataProtectionConfigurationProvider");

}

config.Save();

}

public DataSet Read(string command)

{

logger = LogManager.GetCurrentClassLogger();

var oSqlCon = new SqlConnection(connectionString);

sb = new SqlConnectionStringBuilder(connectionString);

oSqlCon.Open();

oSqlDtAdptr = new SqlDataAdapter(command, oSqlCon);

ds = new DataSet();

oSqlDtAdptr.Fill(ds);

logger.Trace("UserId: "+sb.UserID);

logger.Trace("UserCommand: " + command);

logger.Trace("User's DataSet: " + ds);

oSqlCon.Close();

return ds;

}

public int InsertUpdateDelete(string SqlCommandAsString)

{

logger = LogManager.GetCurrentClassLogger();

sb = new SqlConnectionStringBuilder(connectionString);

var oSqlCon = new SqlConnection(connectionString);

try

{

// check whether the connection is not open

if (oSqlCon.State != ConnectionState.Open)

{

oSqlCon.Open();

}

using (oSqlCom = new SqlCommand())

{

// set the connection for the commnad

oSqlCom.Connection = oSqlCon;

// assign the insert query as a text to the sql command

oSqlCom.CommandText = SqlCommandAsString;

// this will return no of rows affected, by executing the query

return oSqlCom.ExecuteNonQuery();

}

}

catch (Exception Ex)

{

logger.Trace("Connection: " + sb.ConnectionString);

logger.Trace("InitialCatalog: " + sb.InitialCatalog);

logger.Trace("UserId: " + sb.UserID);

logger.Trace("UserCommand: " + oSqlCom.CommandText);

logger.Trace("User's DataSet: " + ds);

throw Ex;

}

finally

{

if (oSqlCon.State == ConnectionState.Open)

{

oSqlCon.Close();

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public DataTable FillDataSet(string SqlSelectCommandAsString, DataTable table)

{

logger = LogManager.GetCurrentClassLogger();

var oSqlCon = new SqlConnection(connectionString);

try

{

// check whether the connection is not open

if (oSqlCon.State != ConnectionState.Open)

{

oSqlCon.Open();

}

using (oSqlCom = new SqlCommand())

{

// set the connection for the commnad

oSqlCom.Connection = oSqlCon;

// assign the insert query as a text to the sql command

oSqlCom.CommandText = SqlSelectCommandAsString;

using (oSqlDtAdptr = new SqlDataAdapter())

{

oSqlDtAdptr.SelectCommand = oSqlCom;

oSqlDtAdptr.Fill(table);

return table;

}

}

}

catch (Exception Ex)

{

logger.Trace("Connection: " + sb.ConnectionString);

logger.Trace("InitialCatalog: " + sb.InitialCatalog);

logger.Trace("UserId: " + sb.UserID);

logger.Trace("UserCommand: " + oSqlCom.CommandText);

logger.Trace("User's DataSet: " + ds);

throw Ex;

}

finally

{

if (oSqlCon.State == ConnectionState.Open)

{

oSqlCon.Close();

}

}

}

public void PrintDataset(DataSet dt)

{

logger = LogManager.GetCurrentClassLogger();

var oSqlCon = new SqlConnection(connectionString);

try

{

DataTableReader dtReader = dt.CreateDataReader();

while (dtReader.Read())

{

for (int i = 0; i < dtReader.FieldCount; i++)

Console.Write("{0}\t", dtReader.GetValue(i).ToString().Trim());

Console.WriteLine();

}

dtReader.Close();

}

catch (Exception Ex)

{

logger.Trace("Connection: " + sb.ConnectionString);

logger.Trace("InitialCatalog: " + sb.InitialCatalog);

logger.Trace("UserId: " + sb.UserID);

logger.Trace("UserCommand: " + oSqlCom.CommandText);

logger.Trace("User's DataSet: " + ds);

throw Ex;

}

finally

{

if (oSqlCon.State == ConnectionState.Open)

{

oSqlCon.Close();

}

}

}

**Form1**  
 public partial class Form1 : Form

{

ConnectedLayer layer;

public Form1()

{

InitializeComponent();

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

dataGridView1.AllowUserToAddRows = false;

dataGridView1.ReadOnly = true;

}

private void Insert(object sender, EventArgs e)

{

layer = new ConnectedLayer();

string comm = $"Insert into dbo.Самолёт Values ('{textBox1.Text}','{textBox2.Text}')";

layer.InsertUpdateDelete(comm);

}

private void Refreshing(object sender, EventArgs e)

{

// Обновление формы

Form1 form1 = new Form1();

this.Hide();

form1.ShowDialog();

}

private void Delete(object sender, EventArgs e)

{

layer = new ConnectedLayer();

string comm = $"Delete From dbo.Самолёт where [Тип самолёта] = '{dataGridView1.CurrentCell.Value.ToString()}';";

layer.InsertUpdateDelete(comm);

}

private void Update(object sender, EventArgs e)

{

layer = new ConnectedLayer();

string comm = $"Update dbo.Самолёт SET [Тип самолёта] = '{textBox1.Text}' WHERE [Тип самолёта] = '{dataGridView1.CurrentCell.Value.ToString()}';";

layer.InsertUpdateDelete(comm);

}

private void Reading(object sender, EventArgs e)

{

// Чтение таблицы

layer = new ConnectedLayer();

string commandsel = "Select [Тип самолёта],[Класс самолёта] from dbo.Самолёт";

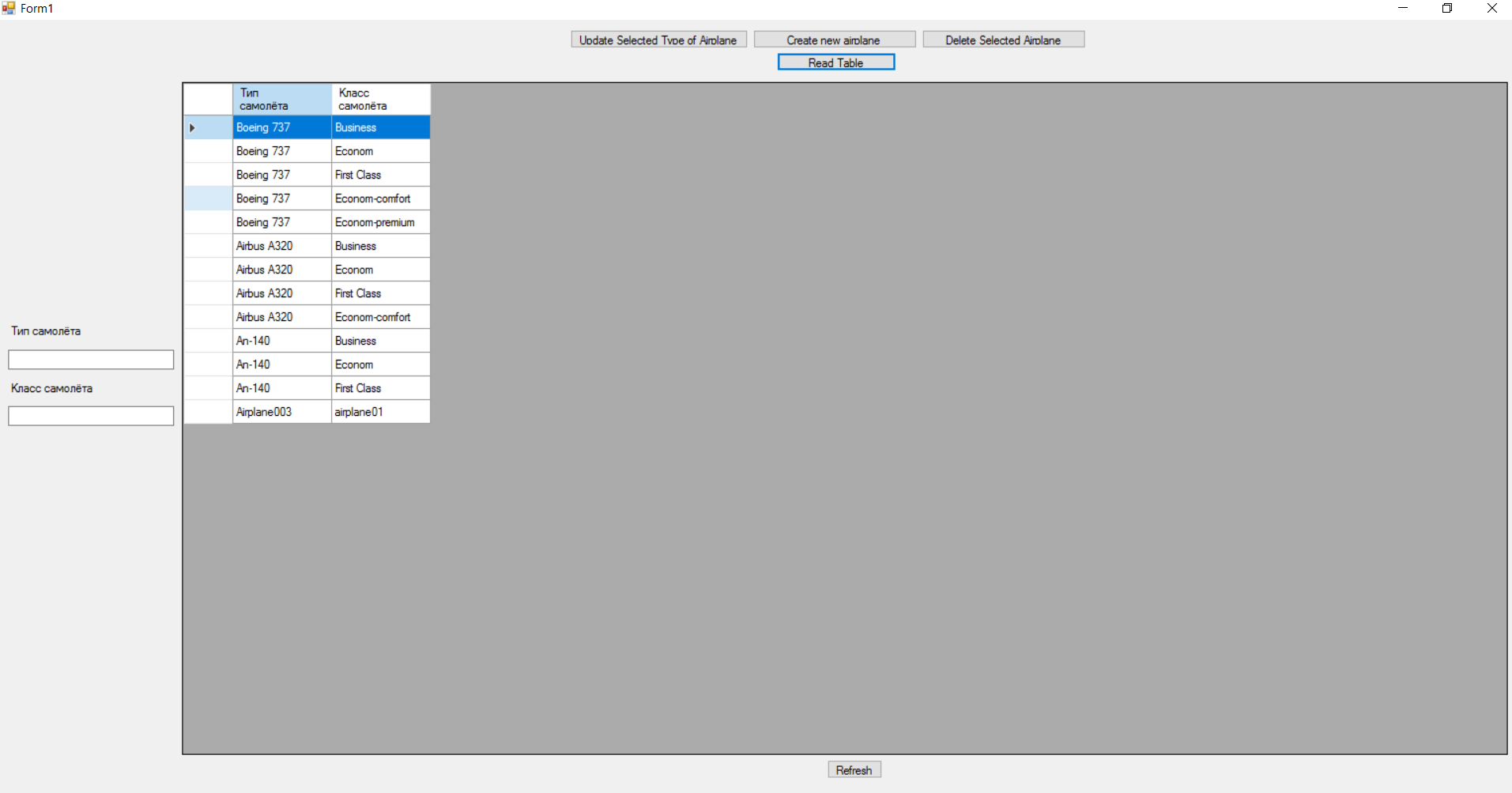
dataGridView1.DataSource = layer.Read(commandsel).Tables[0];

}

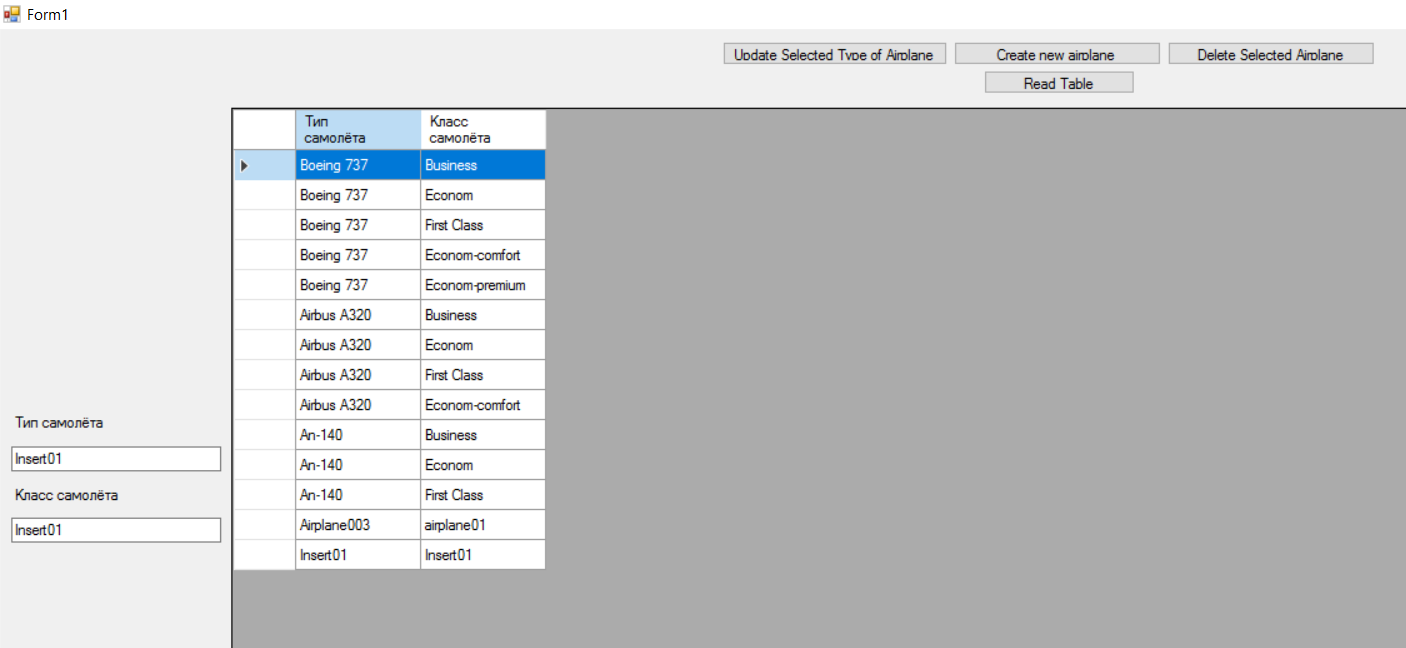
}

**Скриншоты работы:**

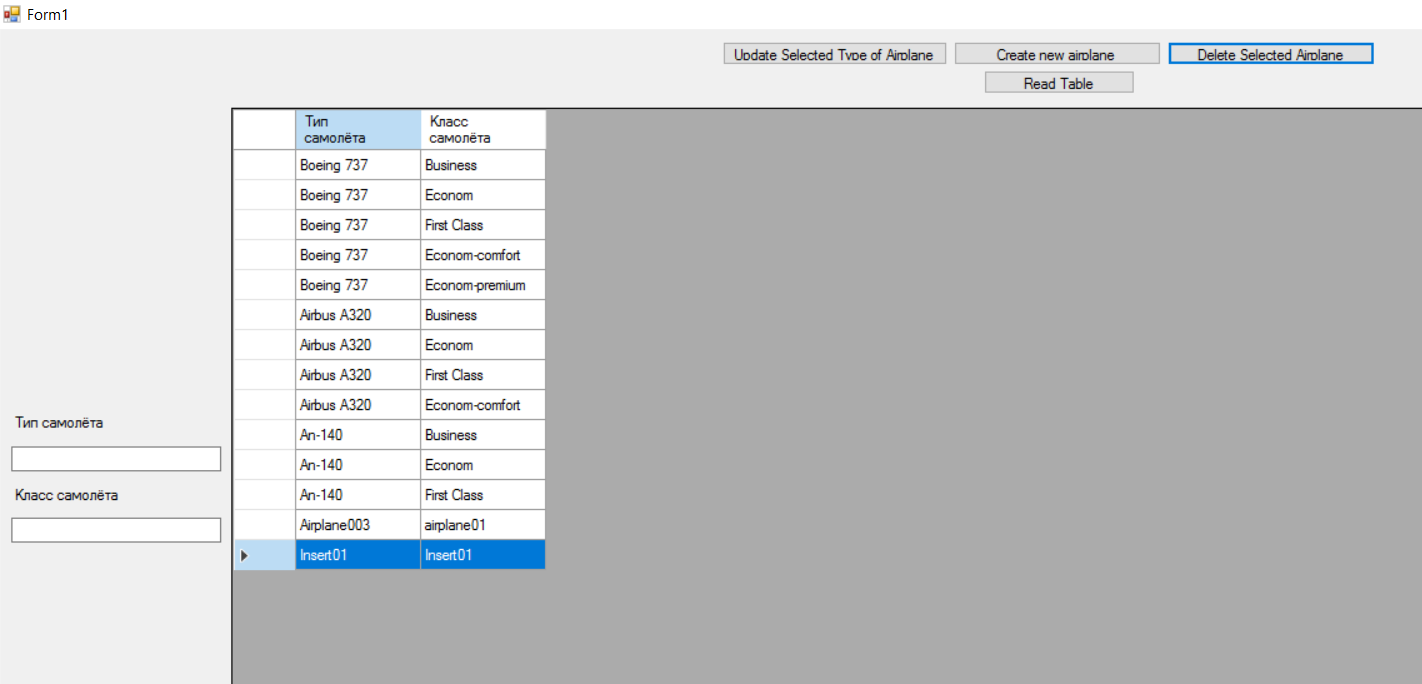
1. **Reading**



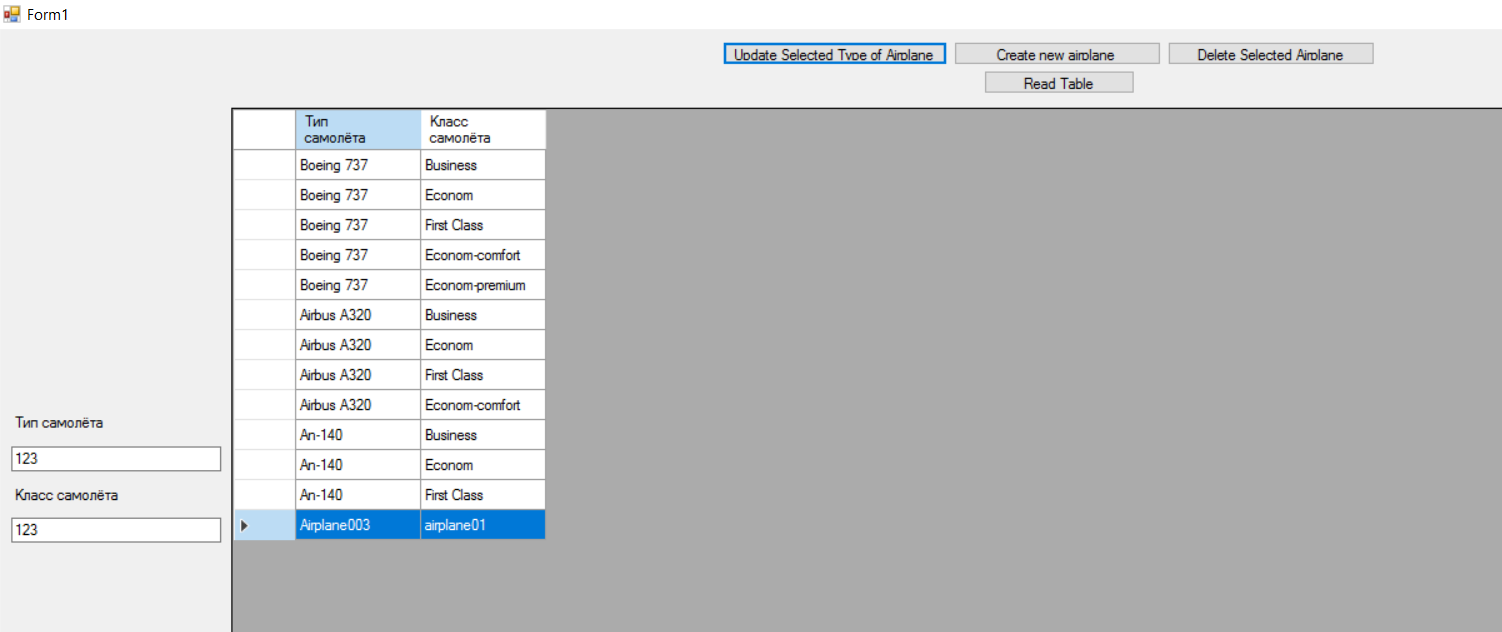
1. **Insert**

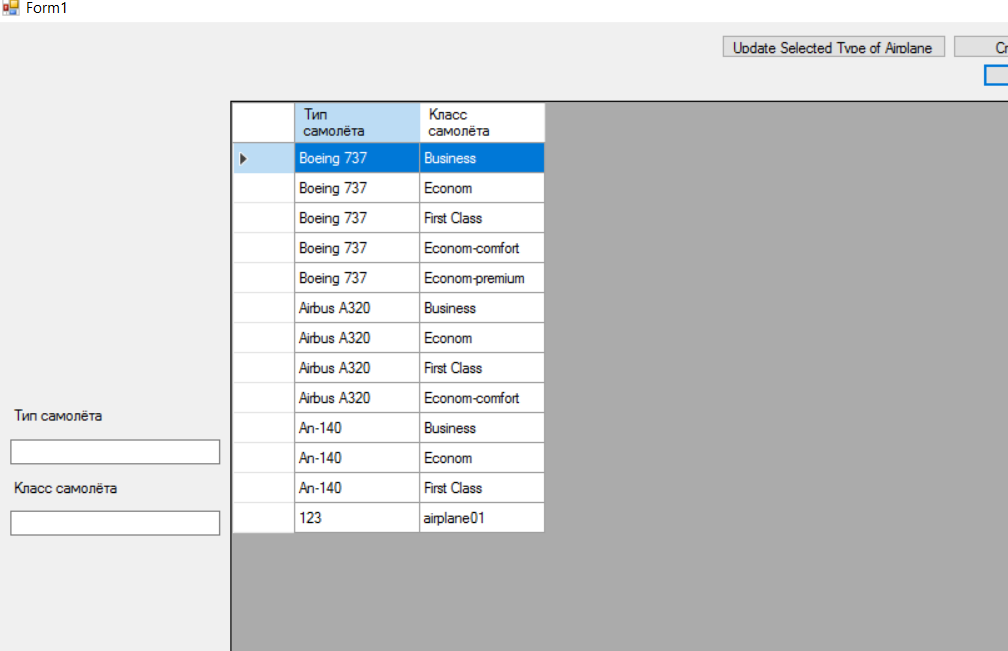


1. **Delete**

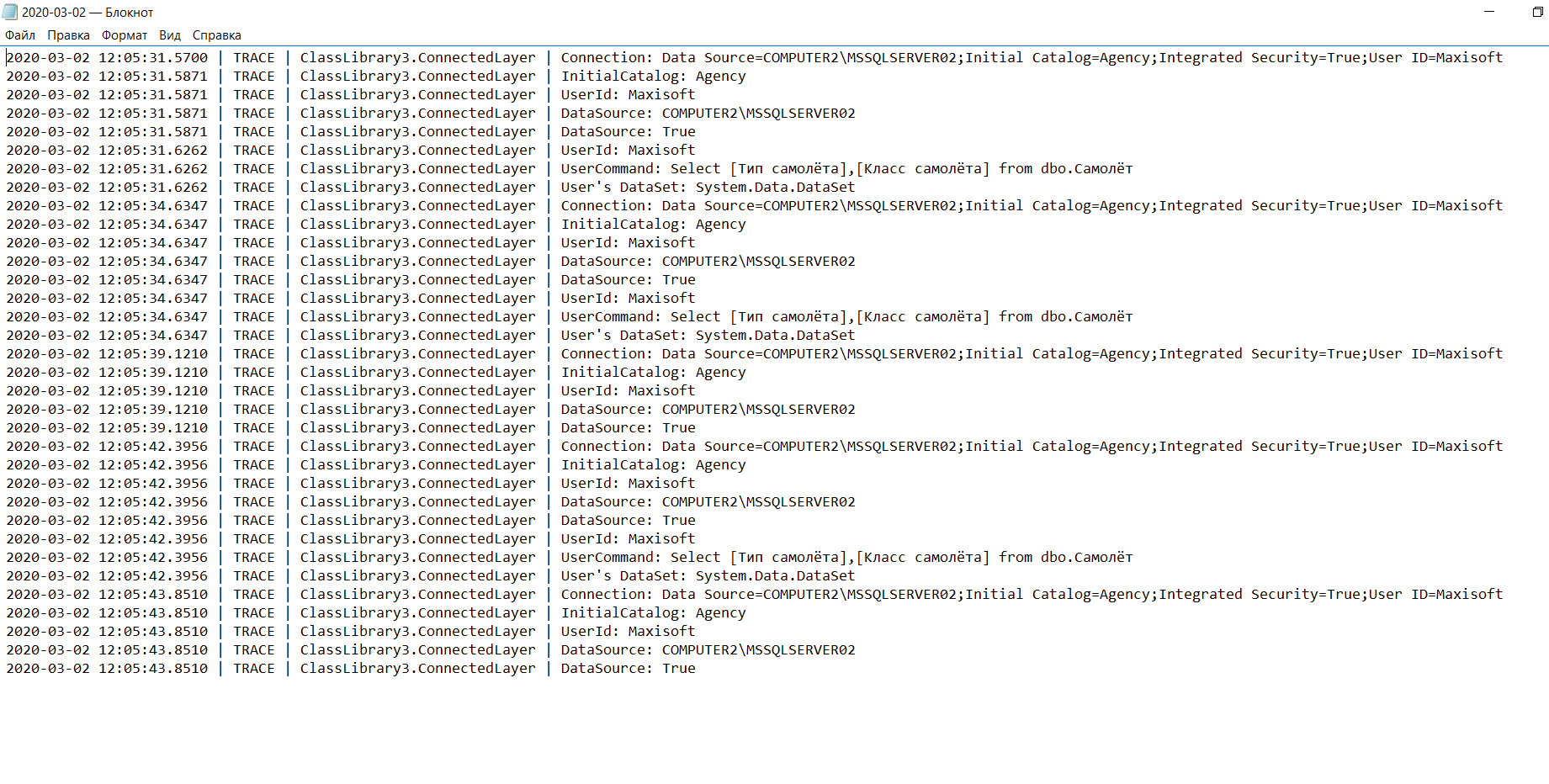


1. **Update**





1. **Лог Файл**



**Вывод:** Изучили принципы работы вставки, удаления, чтения данных из базы данных с помощью класса connected layer.