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 №7**

з Організація баз даних

на тему: «Entity Framework. Взаимодействие с данными. Подходы.»

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 Entity Framework 6. Взаимодействие с разными подходами.(EF 6 Code-First, EF 6 Data Base First, EF 6 Model First)

**Ход работы:**

1. Пройти тест
2. Использовать в своём проекте класс логирования **(использовать текстовые файлы)**
3. Показать созданную базу данных
4. Подключить проект C# проверить соединение с БД вывести отладочную информацию в лог файл.
5. Создать **используя EF**и реализовать действия:
   1. Введение в Entity Framework
   2. Первое приложение с Entity Framework. Подход Code First
   3. Code First к существующей базе данных
   4. Соглашения по наименованию в Code Firs
   5. Автоматизация Code First
   6. Автоматизация Code First и EF Power Tools
   7. Database First
   8. Model First
6. Оформить отчёт и сохранить на сервере;
7. **Знать ответы на контрольные вопросы.**

**Содержание работы:**

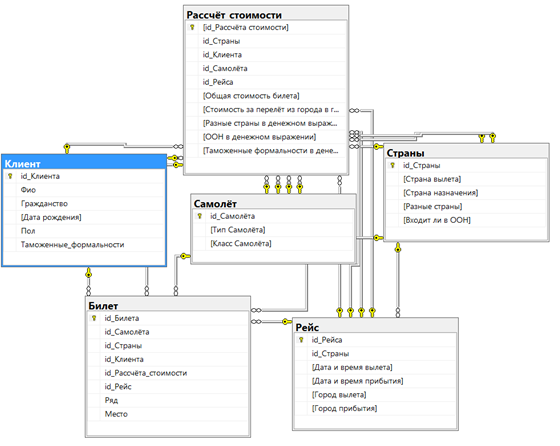
1. Для своего варианта на основе построенной логической модели своего варианта создать БД.
2. Подключить класс для вывода логирования.
3. Выполнить соединение с правильными и не правильными данными
4. Создать форму для входа и после успешной авторизации запустить основную, если авторизация не успешная, то главная форма не должна создаваться.
5. Для каждого пункта показать программный код и пример работы
6. Для своего варианта показать подходы **EF**
   * **1, 4, 7, 10, 13, 16, 19, 22 - Code First**
   * **2, 5, 8, 11, 14, 17, 20, 23 - Database First**
   * **3, 6, 9, 12, 15, 18, 21, 24 - Model First**
7. Показать пример работы с правильными и не правильными данными;

**Выполнение работы**

1. Для своего варианта на основе построенной логической модели своего

варианта создать БД;

**Логическая модель БД**



**Подключить класс для вывода логирования**

class ConnectedLayer

{

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

Logger logger = LogManager.GetCurrentClassLogger();

SqlConnectionStringBuilder builder;

public ConnectedLayer()

{

builder = new SqlConnectionStringBuilder(connectionString);

//ds = new DataSet();

var oSqlCon = new SqlConnection(connectionString);

using (SqlConnection connection = new SqlConnection(connectionString))

{

logger.Trace(

"Connection to" + Environment.NewLine +

"Data Source: " + oSqlCon.DataSource + Environment.NewLine +

"Database: " + oSqlCon.Database + Environment.NewLine +

"State: " + oSqlCon.State +

"User: " + builder.UserID +

"Catalog: " + builder.InitialCatalog +

"Server: " + builder.DataSource

);

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 logger = LogManager.GetCurrentClassLogger();

var oSqlCon = new SqlConnection(connectionString);

builder = new SqlConnectionStringBuilder(connectionString);

oSqlCon.Open();

SqlDataAdapter oSqlDtAdptr = new SqlDataAdapter(command, oSqlCon);

DataSet ds = new DataSet();

oSqlDtAdptr.Fill(ds);

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

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

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

oSqlCon.Close();

return ds;

}}

**Code First к существующей базе данных**

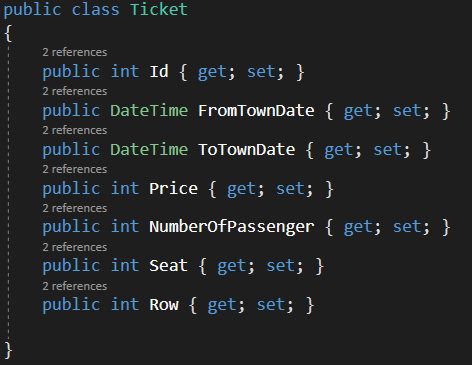
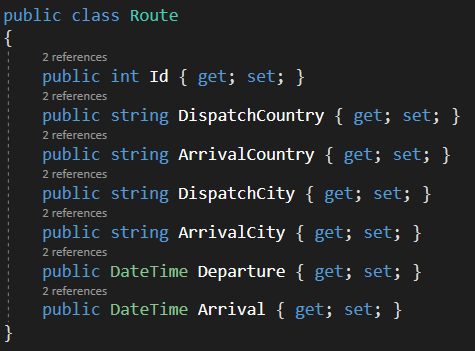
**App.config:**

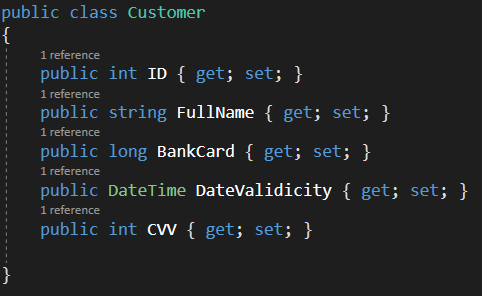
<connectionStrings>

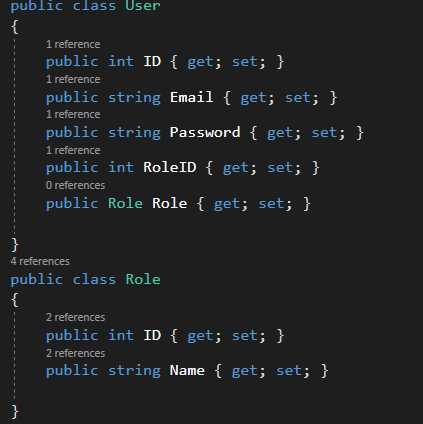
<add name="DBConnection" connectionString="data source=(localdb)\MSSQLLocalDB;Initial Catalog=userstore;Integrated Security=True;" providerName="System.Data.SqlClient" />

</connectionStrings>

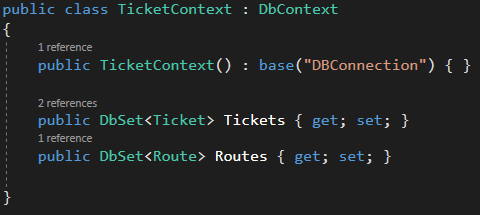
**Models:**





**Контекст данных:**



**Код программы:**

public partial class MainWindow : Window

{

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

Logger logger = LogManager.GetCurrentClassLogger();

public MainWindow()

{

logger.Trace("Windows is loaded");

using (TicketContext db = new TicketContext())

{

InitializeComponent();

Route route1 = new Route { Id = 1, DispatchCountry = "Ukraine", ArrivalCountry = "France", DispatchCity = "Kiev", ArrivalCity = "Paris", Arrival = DateTime.Now, Departure = DateTime.Now };

Route route2 = new Route { Id = 2, DispatchCountry = "Ukraine", ArrivalCountry = "Germany", DispatchCity = "Kiev", ArrivalCity = "Berlin", Arrival = DateTime.Now, Departure = DateTime.Now };

db.Routes.AddRange(new List<Route> { route1, route2 });

db.SaveChanges();

Ticket ticket1 = new Ticket {Id =1, FromTownDate = DateTime.Now, ToTownDate = DateTime.Now, Price = 500, NumberOfPassenger = 3, Row = 1, Seat = 1 };

Ticket ticket2 = new Ticket {Id =2, FromTownDate = DateTime.Now, ToTownDate = DateTime.Now, Price = 400, NumberOfPassenger = 3, Row = 1, Seat = 1 };

db.Tickets.AddRange(new List<Ticket> { ticket1, ticket2 });

db.SaveChanges();

Customer customer1 = new Customer { ID = 1, FullName = "Степанов Александр Адреевич", BankCard = 34561234098548393, DateValidicity = DateTime.Now, CVV = 322 };

logger.Trace(db.SaveChanges().ToString());

dataGrid.ItemsSource = db.Tickets.ToList();

}

}

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

{

logger.Trace($" Command: Select \* From {comboBox.SelectedIndex} ");

var oSqlCon = new SqlConnection(connectionString);

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();

string command = $"Select \* From {comboBox.SelectedIndex}";

SqlDataAdapter oSqlDtAdptr = new SqlDataAdapter(command, oSqlCon);

DataSet ds = new DataSet();

oSqlDtAdptr.Fill(ds);

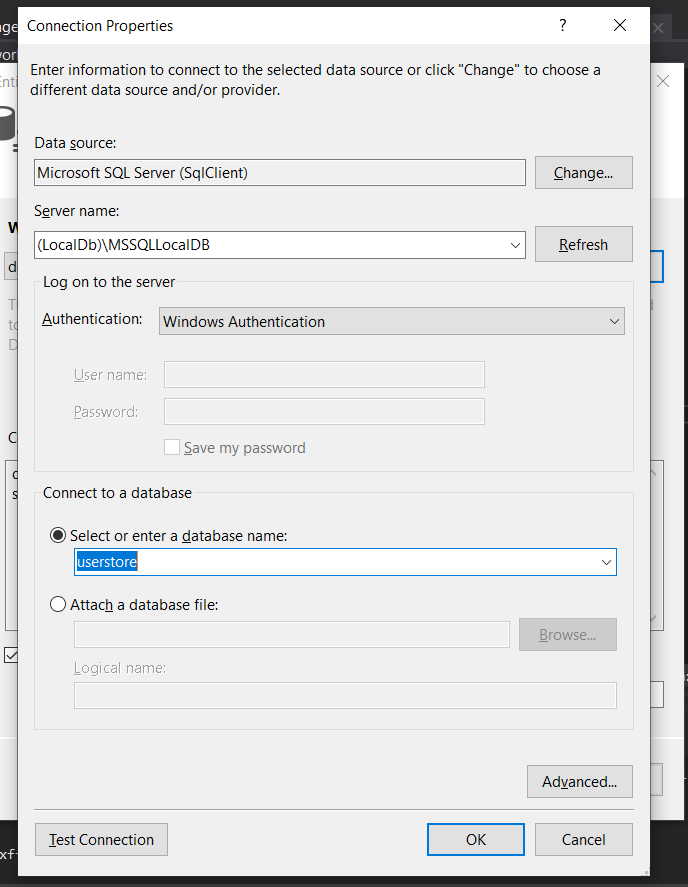
dataGrid.ItemsSource = ds.Tables;

}

}

}

**Автоматизация Code First**



**Model:**

namespace EntityFrameworkBD7.App\_Data

{

using System;

using System.Data.Entity;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

public partial class TicketContext2 : DbContext

{

public TicketContext2()

: base("name=TicketContext2")

{

}

public virtual DbSet<C\_\_MigrationHistory> C\_\_MigrationHistory { get; set; }

public virtual DbSet<Routes> Routes { get; set; }

public virtual DbSet<Tickets> Tickets { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

}

}

}

**Tables:**

namespace EntityFrameworkBD7.App\_Data

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

public partial class Routes

{

public int Id { get; set; }

public string DispatchCountry { get; set; }

public string ArrivalCountry { get; set; }

public string DispatchCity { get; set; }

public string ArrivalCity { get; set; }

public DateTime Departure { get; set; }

public DateTime Arrival { get; set; }

}

}

namespace EntityFrameworkBD7.App\_Data

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

public partial class Tickets

{

public int Id { get; set; }

public DateTime FromTownDate { get; set; }

public DateTime ToTownDate { get; set; }

public int Price { get; set; }

public int NumberOfPassenger { get; set; }

public int Seat { get; set; }

public int Row { get; set; }

}

}

**MigrationHistory:**

namespace EntityFrameworkBD7.App\_Data

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

[Table("\_\_MigrationHistory")]

public partial class C\_\_MigrationHistory

{

[Key]

[Column(Order = 0)]

[StringLength(150)]

public string MigrationId { get; set; }

[Key]

[Column(Order = 1)]

[StringLength(300)]

public string ContextKey { get; set; }

[Required]

public byte[] Model { get; set; }

[Required]

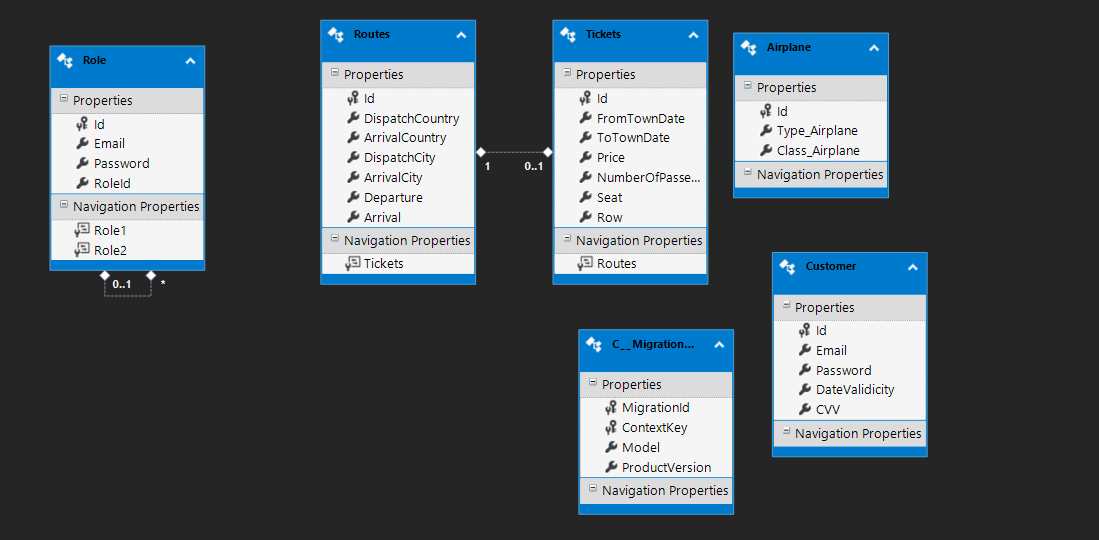
[StringLength(32)]

public string ProductVersion { get; set; }

}

}

**Database First**



public partial class AirMoximusEntities : DbContext

{

public AirMoximusEntities()

: base("name=AirMoximusEntities")

{

}

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

throw new UnintentionalCodeFirstException();

}

public virtual DbSet<C\_\_MigrationHistory> C\_\_MigrationHistory { get; set; }

public virtual DbSet<Airplane> Airplane { get; set; }

public virtual DbSet<Customer> Customer { get; set; }

public virtual DbSet<ExceptionDetails> ExceptionDetails { get; set; }

public virtual DbSet<Role> Role { get; set; }

public virtual DbSet<Route> Route { get; set; }

public virtual DbSet<Routes> Routes { get; set; }

public virtual DbSet<Table> Table { get; set; }

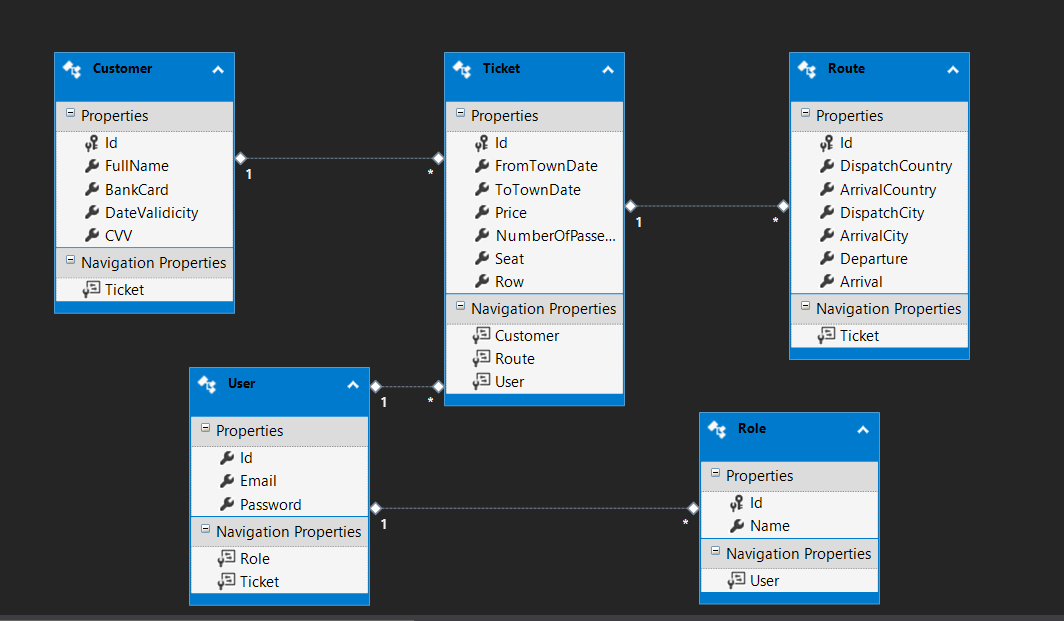
public virtual DbSet<Ticket> Ticket { get; set; }

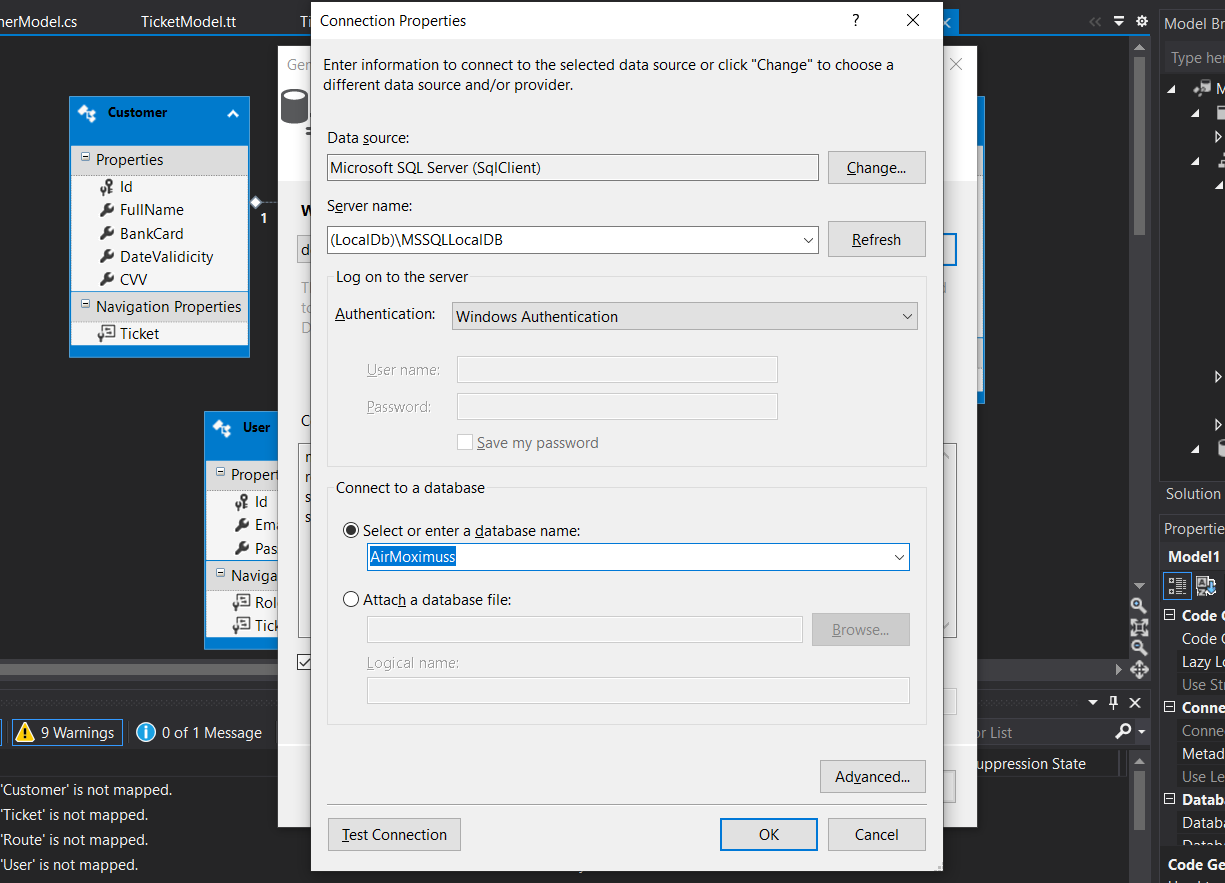
public virtual DbSet<Tickets> Tickets { get; set; }

}

## Model First

## Модель:





**Вывод:** Изучили принципы работы Entity Framework 6. Взаимодействие с разными подходами(EF 6 Code-First, EF 6 Data Base First, EF 6 Model First).