**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 2**

по дисциплине «Разработка приложений баз данных для информационных систем»

на тему: «Использование ENTITY FRAMEWORK и LINQ для работы с базами данных.»

Вариант 29

Выполнил: студент гр. ИТП-31

Зайцев А.В.

Принял: доцент

Асенчик О. Д.

Гомель 2020

**Цель работы:**

Ознакомиться с возможностями ENTITY FRAMEWORK и получить навыки написания LINQ запросов к объектам, связанным с таблицами базы данных СУБД MS SQL сервер.

**Задание:**

3.1. Создать с использованием .NET Core Entity Framework Core консольное приложение, содержащее набор классов, моделирующих предметную область соответствующей своему варианту и ранее созданную и заполненной тестовыми данными задания базой MS SQL Server. Для этого необходимо создать:

* Классы моделирующие не менее чем три таблицы базы данных согласно вашему варианту.
* Класс контекста данных.

3.2. Выполнить, используя объекты Entity Framework Core и LINQ:

1. Выборку всех данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один» – 1 шт.
2. Выборку данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один», отфильтрованные по определенному условию, налагающему ограничения на одно или несколько полей – 1 шт.
3. Выборку данных, сгруппированных по любому из полей данных с выводом какого-либо итогового результата (min, max, avg, сount или др.) по выбранному полю из таблицы, стоящей в схеме базы данных нас стороне отношения «многие» – 1 шт.
4. Выборку данных из двух полей двух таблиц, связанных между собой отношением «один-ко-многим» – 1 шт.
5. Выборку данных из двух таблиц, связанных между собой отношением «один-ко-многим» и отфильтрованным по некоторому условию, налагающему ограничения на значения одного или нескольких полей – 1 шт.
6. Вставку данных в таблицы, стоящей на стороне отношения «Один» – 1 шт.
7. Вставку данных в таблицы, стоящей на стороне отношения «Многие» – 1 шт.:
8. Удаление данных из таблицы, стоящей на стороне отношения «Один» – 1 шт.
9. Удаление данных из таблицы, стоящей на стороне отношения «Многие» – 1 шт.
10. Обновление удовлетворяющих определенному условию записей в любой из таблиц базы данных – 1 шт.

**Вариант задания:**

Предметная область задания изображена на рисунке 1.

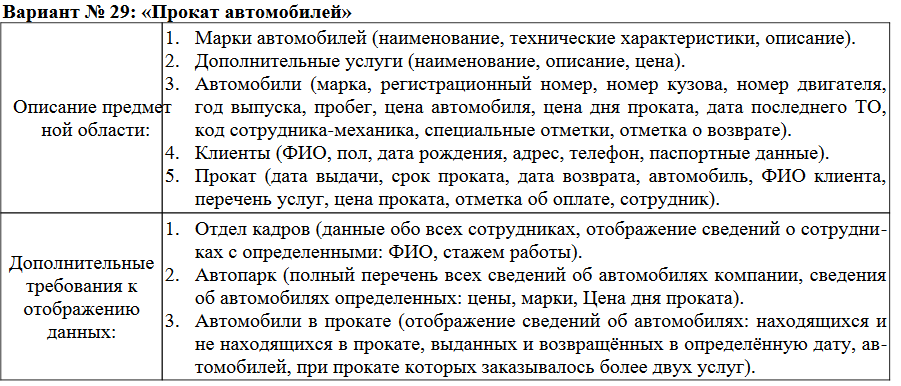
****

Рисунок 1 – Предметная область

**Ход работы:**

Задание 1

Изначально необходимо создать классы, моделирующие предметную область. Эти классы хранятся в папке *Models* (рисунок 2).

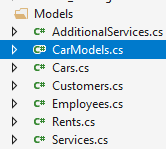


Рисунок 2 – содержимое папки *Models*

Структура созданного консольного приложения изображена на рисунке 3.

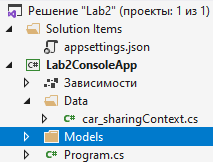


Рисунок 3 – Структура созданного приложения

Задание 2

Были созданы 10 *LINQ* запросов к базе данных. Каждому запросу соответствует отдельный метод. Результат выполнения запросов на рисунках 4 – 13.

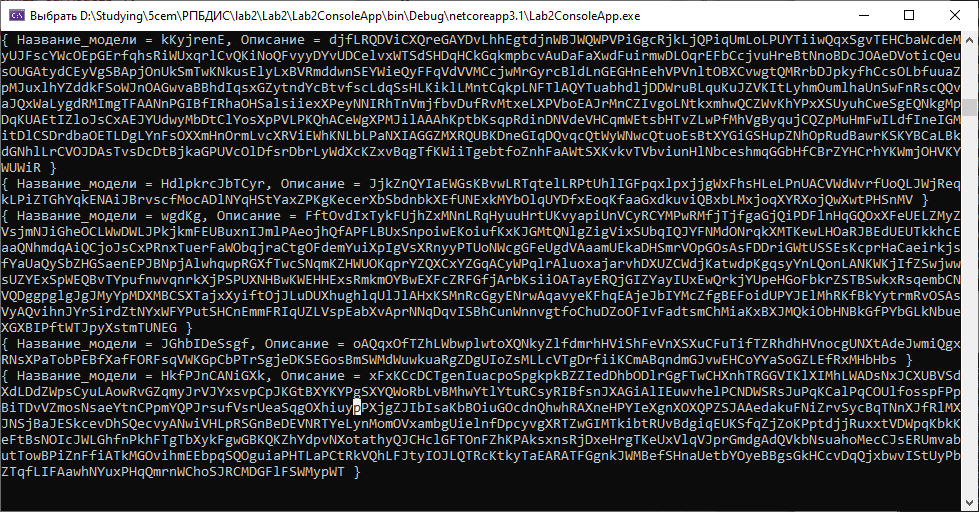


Рисунок 4 – Результат выполнения первого запроса

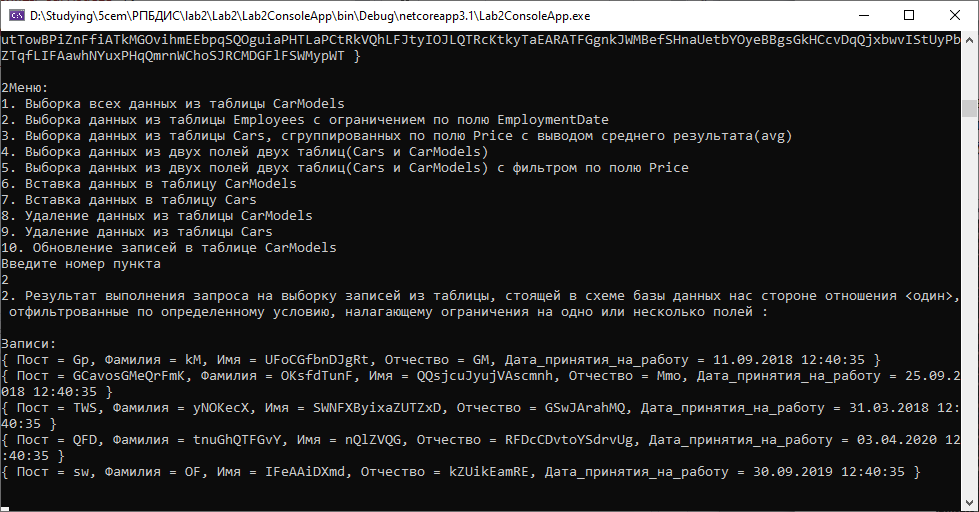


Рисунок 5 – Результат выполнения 2-го запроса

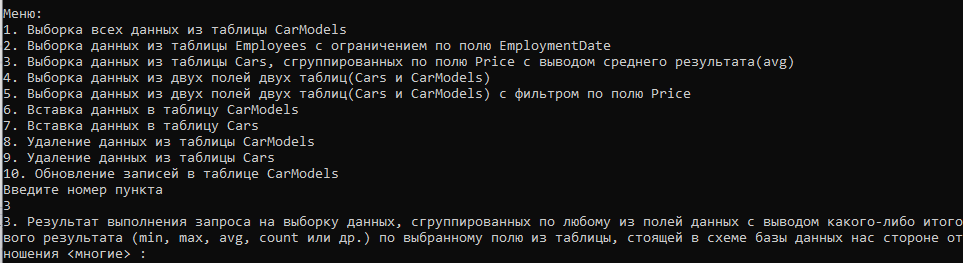


Рисунок 6 – Результат выполнения 3-го запроса

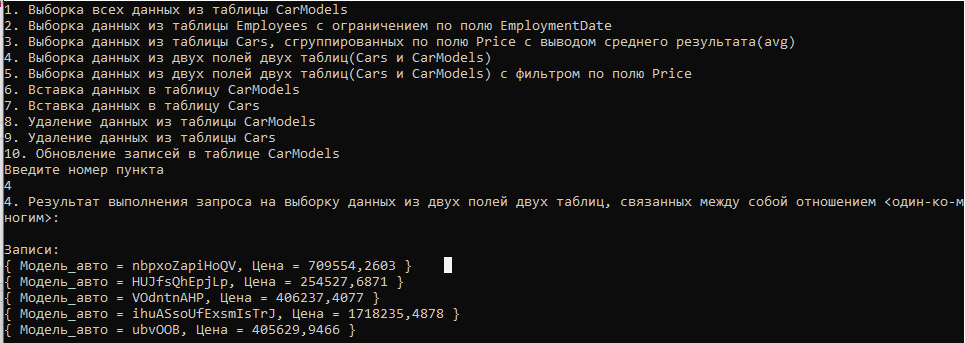


Рисунок 7 – Результат выполнения 4-го запроса

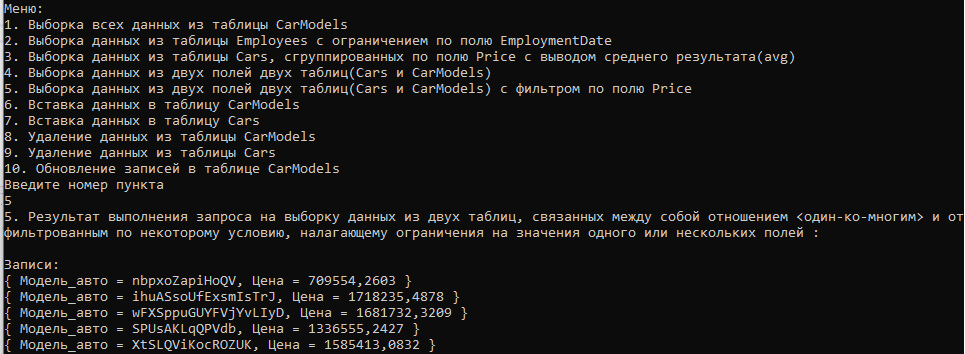


Рисунок 8 – Результат выполнения 5-го запроса

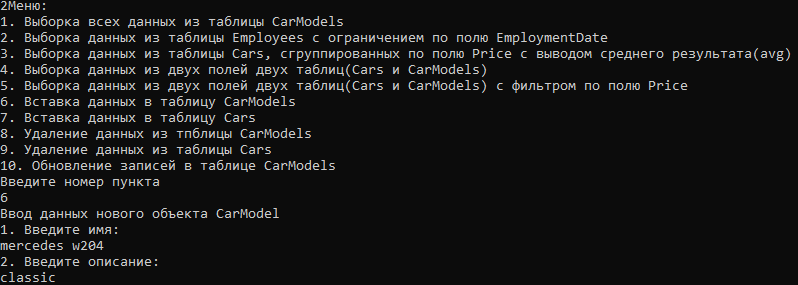


Рисунок 9 – Результат выполнения 6-го запроса

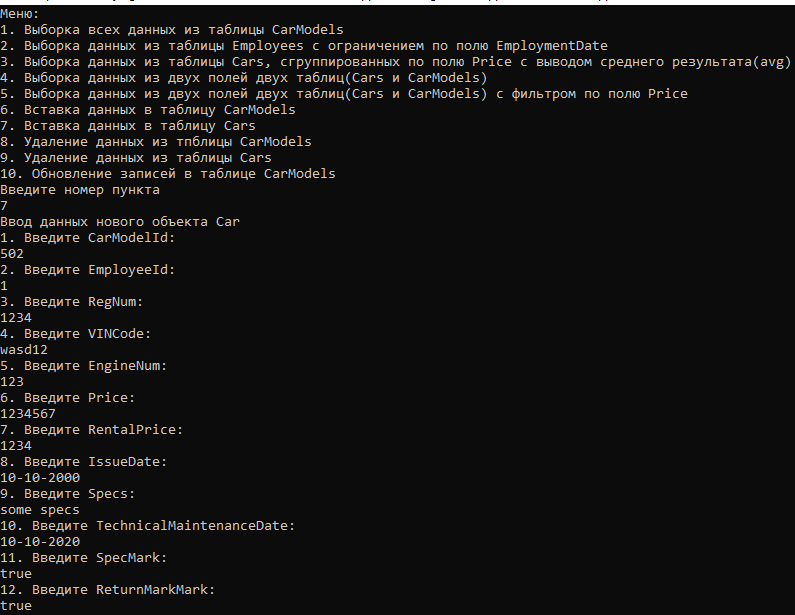


Рисунок 10 – Результат выполнения 7-го запроса

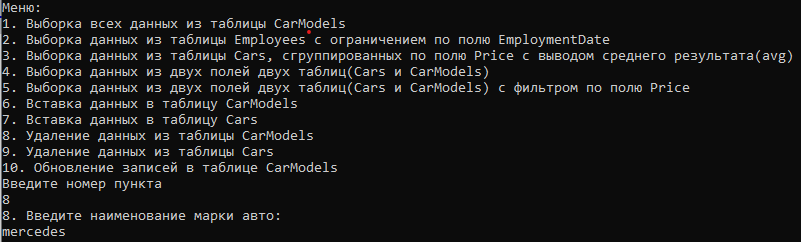


Рисунок 11 – Результат выполнения 8-го запроса

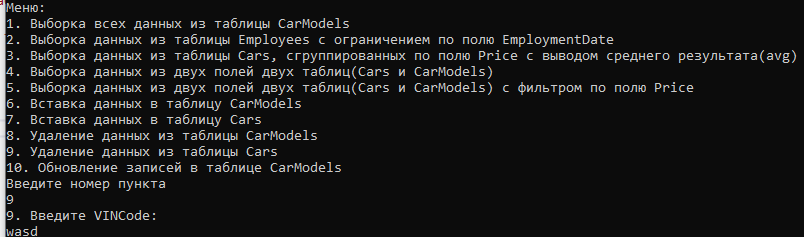


Рисунок 12 – Результат выполнения 9-го запроса

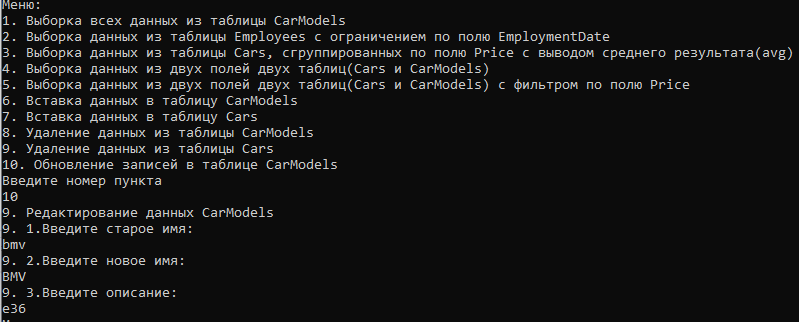


Рисунок 13 – Результат выполнения 10-го запроса

**Вывод:**

Были созданы классы, описывающие предметную область приложения. Они были переименованы и распределены по папкам. Далее было написано 10 *LINQ* запросов для всех *CRUD* операций к базе данных.

**ПРИЛОЖЕНИЕ А**

Листинги классов

*Program.cs*

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using Microsoft.Extensions.Configuration;

using Microsoft.EntityFrameworkCore;

using System.IO;

namespace Lab2ConsoleApp

{

class Program

{

static void Main(string[] args)

{

using(car\_sharingContext db = new car\_sharingContext())

{

int menu = 1;

while (menu != 0)

{

Console.WriteLine("Меню:");

Console.WriteLine("1. Выборка всех данных из таблицы CarModels");

Console.WriteLine("2. Выборка данных из таблицы Employees с ограничением по полю EmploymentDate");

Console.WriteLine("3. Выборка данных из таблицы Cars, сгруппированных по полю Price с выводом среднего результата(avg)");

Console.WriteLine("4. Выборка данных из двух полей двух таблиц(Cars и CarModels)");

Console.WriteLine("5. Выборка данных из двух полей двух таблиц(Cars и CarModels) с фильтром по полю Price");

Console.WriteLine("6. Вставка данных в таблицу CarModels");

Console.WriteLine("7. Вставка данных в таблицу Cars");

Console.WriteLine("8. Удаление данных из таблицы CarModels");

Console.WriteLine("9. Удаление данных из таблицы Cars");

Console.WriteLine("10. Обновление записей в таблице CarModels");

Console.WriteLine("Введите номер пункта");

menu = Int32.Parse(Console.ReadLine());

switch (menu)

{

case 1:

Select(db, 1);

break;

case 2:

Select(db, 2);

break;

case 3:

Select(db, 3);

break;

case 4:

Select(db, 4);

break;

case 5:

Select(db, 5);

break;

case 6:

Insert(db, 6);

break;

case 7:

Insert(db, 7);

break;

case 8:

Delete(db, 8);

break;

case 9:

Delete(db, 9);

break;

case 10:

Update(db);

break;

case 0:

Console.WriteLine("Осуществление выхода из программы");

break;

default: Console.WriteLine("Осуществление выхода из программы"); break;

}

}

}

}

static void Print(string sqltext, IEnumerable items)

{

Console.WriteLine(sqltext);

Console.WriteLine("Записи: ");

foreach (var item in items)

{

Console.WriteLine(item.ToString());

}

Console.WriteLine();

Console.ReadKey();

}

static void Select(car\_sharingContext db, int num)

{

string comment = "";

//для наглядности выводим не более 5 записей

switch (num)

{

case 1:

var query1 = from c in db.CarModels

select new

{

Название\_модели = c.Name,

Описание = c.Description

};

comment = "1. Результат выполнения запроса на выборку всех записей из таблицы, стоящей в схеме базы данных нас стороне отношения «один»: \r\n";

Print(comment, query1.Take(200).ToList());

break;

case 2:

var query2 = from e in db.Employees

where (e.EmploymentDate > Convert.ToDateTime("01-01-2018"))

select new

{

Пост = e.Post,

Фамилия = e.Surname,

Имя = e.Name,

Отчество = e.Patronymic,

Дата\_принятия\_на\_работу = e.EmploymentDate

};

comment = "2. Результат выполнения запроса на выборку записей из таблицы, стоящей в схеме базы данных нас стороне отношения «один», отфильтрованные по определенному условию, налагающему ограничения на одно или несколько полей : \r\n";

Print(comment, query2.Take(5).ToList());

break;

case 3:

var query3 = from c in db.Cars

select new

{

Цена = c.Price

};

comment = "3. Результат выполнения запроса на выборку данных, сгруппированных по любому из полей данных с выводом какого-либо итогового результата (min, max, avg, сount или др.) по выбранному полю из таблицы, стоящей в схеме базы данных нас стороне отношения «многие» : \r\n";

Console.WriteLine(comment);

Console.WriteLine("Средняя цена автомобилей: " + query3.Take(20000).ToList().Average(n => n.Цена));

break;

case 4:

var query4 = from f in db.CarModels

join t in db.Cars

on f.CarModelId equals t.CarModelId

select new

{

Модель\_авто = f.Name,

Цена = t.Price

};

comment = "4. Результат выполнения запроса на выборку данных из двух полей двух таблиц, связанных между собой отношением «один-ко-многим»: \r\n";

Print(comment, query4.Take(5).ToList());

break;

case 5:

var query5 = from f in db.CarModels

join t in db.Cars

on f.CarModelId equals t.CarModelId

where t.Price > 600000

select new

{

Модель\_авто = f.Name,

Цена = t.Price

};

comment = "5. Результат выполнения запроса на выборку данных из двух таблиц, связанных между собой отношением «один-ко-многим» и отфильтрованным по некоторому условию, налагающему ограничения на значения одного или нескольких полей : \r\n";

Print(comment, query5.Take(5).ToList());

break;

}

}

static void Insert(car\_sharingContext db, int num)

{

switch (num)

{

case 6:

Console.WriteLine("Ввод данных нового объекта CarModel");

Console.WriteLine("1. Введите имя:");

string name = Console.ReadLine();

Console.WriteLine("2. Введите описание:");

string description = Console.ReadLine();

CarModels carModel = new CarModels { Name = name, Description = description };

db.CarModels.Add(carModel);

db.SaveChanges();

Console.ReadKey();

break;

case 7:

Console.WriteLine("Ввод данных нового объекта Car");

Console.WriteLine("1. Введите CarModelId:");

int carModelId = Int32.Parse(Console.ReadLine());

Console.WriteLine("2. Введите EmployeeId:");

int employeeId = Int32.Parse(Console.ReadLine());

Console.WriteLine("3. Введите RegNum:");

int regNum = Int32.Parse(Console.ReadLine());

Console.WriteLine("4. Введите VINCode:");

string vinCode = Console.ReadLine();

Console.WriteLine("5. Введите EngineNum:");

int engineNum = Int32.Parse(Console.ReadLine());

Console.WriteLine("6. Введите Price:");

decimal price = decimal.Parse(Console.ReadLine());

Console.WriteLine("7. Введите RentalPrice:");

decimal rentalPrice = decimal.Parse(Console.ReadLine());

Console.WriteLine("8. Введите IssueDate:");

DateTime issueDate = DateTime.Parse(Console.ReadLine());

Console.WriteLine("9. Введите Specs:");

string specs = Console.ReadLine();

Console.WriteLine("10. Введите TechnicalMaintenanceDate:");

DateTime technicalMaintenanceDategNum = DateTime.Parse(Console.ReadLine());

Console.WriteLine("11. Введите SpecMark:");

bool specMark = bool.Parse(Console.ReadLine());

Console.WriteLine("12. Введите ReturnMarkMark:");

bool returnMark = bool.Parse(Console.ReadLine());

Cars car = new Cars

{

CarModelId = carModelId,

EmployeeId = employeeId,

RegNum = regNum,

Vincode = vinCode,

EngineNum = engineNum,

Price = price,

RentalPrice = rentalPrice,

IssueDate = issueDate,

Specs = specs,

TechnicalMaintenanceDate = technicalMaintenanceDategNum,

SpecMark = specMark,

ReturnMark = returnMark

};

db.Cars.Add(car);

db.SaveChanges();

Console.ReadKey();

break;

}

}

static void Delete(car\_sharingContext db, int num)

{

switch (num)

{

case 8:

Console.WriteLine("8. Введите наименование марки авто: ");

string name = Console.ReadLine();

var carModels = db.CarModels.Where(c => c.Name == name);

var cars = db.Cars.Include("CarModel").Where(o => (o.CarModel.Name == name));

db.CarModels.RemoveRange(carModels);

db.SaveChanges();

db.Cars.RemoveRange(cars);

db.SaveChanges();

Console.ReadKey();

break;

case 9:

Console.WriteLine("9. Введите VINCode: ");

string vinCode = Console.ReadLine();

var car = db.Cars.Where(c => c.Vincode == vinCode);

db.Cars.RemoveRange(car);

db.SaveChanges();

Console.ReadKey();

break;

}

}

static void Update(car\_sharingContext db)

{

Console.WriteLine("9. Редактирование данных CarModels");

Console.WriteLine("9. 1.Введите старое имя: ");

string oldName = Console.ReadLine();

Console.WriteLine("9. 2.Введите новое имя: ");

string newName = Console.ReadLine();

Console.WriteLine("9. 3.Введите описание: ");

string description = Console.ReadLine();

var carModel = db.CarModels.Where(c => c.Name == oldName).FirstOrDefault();

if(carModel != null)

{

carModel.Name = newName;

carModel.Description = description;

}

db.SaveChanges();

Console.ReadKey();

}

}

}

*Car\_sharingContext.cs*

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.Extensions.Configuration;

using System.IO;

namespace Lab2ConsoleApp

{

public partial class car\_sharingContext : DbContext

{

public car\_sharingContext()

{

}

public car\_sharingContext(DbContextOptions<car\_sharingContext> options)

: base(options)

{

}

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

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

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

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

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

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

var builder = new ConfigurationBuilder();

// установка пути к текущему каталогу

builder.SetBasePath(Directory.GetCurrentDirectory());

// получаем конфигурацию из файла appsettings.json

builder.AddJsonFile("appsettings.json");

// создаем конфигурацию

var config = builder.Build();

// получаем строку подключения

string connectionString = config.GetConnectionString("SQLConnection");

var options = optionsBuilder

.UseSqlServer(connectionString)

.Options;

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<AdditionalServices>(entity =>

{

entity.HasOne(d => d.Rent)

.WithMany(p => p.AdditionalServices)

.HasForeignKey(d => d.RentId)

.HasConstraintName("FK\_\_Additiona\_\_RentI\_\_34C8D9D1");

entity.HasOne(d => d.Service)

.WithMany(p => p.AdditionalServices)

.HasForeignKey(d => d.ServiceId)

.HasConstraintName("FK\_\_Additiona\_\_Servi\_\_35BCFE0A");

});

modelBuilder.Entity<CarModels>(entity =>

{

entity.HasKey(e => e.CarModelId);

entity.Property(e => e.Description)

.HasMaxLength(1000)

.IsUnicode(false);

entity.Property(e => e.Name)

.HasMaxLength(20)

.IsUnicode(false);

});

modelBuilder.Entity<Cars>(entity =>

{

entity.HasKey(e => e.CarId);

entity.Property(e => e.IssueDate).HasColumnType("datetime");

entity.Property(e => e.Price).HasColumnType("money");

entity.Property(e => e.RentalPrice).HasColumnType("money");

entity.Property(e => e.Specs)

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.TechnicalMaintenanceDate).HasColumnType("datetime");

entity.Property(e => e.Vincode)

.HasColumnName("VINCode")

.HasMaxLength(20)

.IsUnicode(false);

entity.HasOne(d => d.CarModel)

.WithMany(p => p.Cars)

.HasForeignKey(d => d.CarModelId)

.HasConstraintName("FK\_\_Cars\_\_CarModelId\_\_286302EC");

entity.HasOne(d => d.Employee)

.WithMany(p => p.Cars)

.HasForeignKey(d => d.EmployeeId)

.HasConstraintName("FK\_\_Cars\_\_EmployeeId\_\_29572725");

});

modelBuilder.Entity<Customers>(entity =>

{

entity.HasKey(e => e.CustomerId);

entity.Property(e => e.Address)

.HasMaxLength(60)

.IsUnicode(false);

entity.Property(e => e.BirthDate).HasColumnType("datetime");

entity.Property(e => e.Name)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.PassportInfo)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.Patronymic)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.PhoneNum)

.HasMaxLength(13)

.IsUnicode(false);

entity.Property(e => e.Surname)

.HasMaxLength(20)

.IsUnicode(false);

});

modelBuilder.Entity<Employees>(entity =>

{

entity.HasKey(e => e.EmployeeId);

entity.Property(e => e.EmploymentDate).HasColumnType("datetime");

entity.Property(e => e.Name)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.Patronymic)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.Post)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.Surname)

.HasMaxLength(20)

.IsUnicode(false);

});

modelBuilder.Entity<Rents>(entity =>

{

entity.HasKey(e => e.RentId);

entity.Property(e => e.DeliveryDate).HasColumnType("datetime");

entity.Property(e => e.Price).HasColumnType("money");

entity.Property(e => e.ReturnDate).HasColumnType("datetime");

entity.HasOne(d => d.Car)

.WithMany(p => p.Rents)

.HasForeignKey(d => d.CarId)

.HasConstraintName("FK\_\_Rents\_\_CarId\_\_2E1BDC42");

entity.HasOne(d => d.Customer)

.WithMany(p => p.Rents)

.HasForeignKey(d => d.CustomerId)

.HasConstraintName("FK\_\_Rents\_\_CustomerI\_\_2F10007B");

entity.HasOne(d => d.Employee)

.WithMany(p => p.Rents)

.HasForeignKey(d => d.EmployeeId)

.HasConstraintName("FK\_\_Rents\_\_EmployeeI\_\_300424B4");

});

modelBuilder.Entity<Services>(entity =>

{

entity.HasKey(e => e.ServiceId);

entity.Property(e => e.Description)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Name)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.Price).HasColumnType("money");

});

}

}

}

*AdditionalServices.cs*

using System;

using System.Collections.Generic;

namespace Lab2ConsoleApp

{

public partial class AdditionalServices

{

public int Id { get; set; }

public int? RentId { get; set; }

public int? ServiceId { get; set; }

public Rents Rent { get; set; }

public Services Service { get; set; }

}

}

*CarModels.cs*

using System;

using System.Collections.Generic;

namespace Lab2ConsoleApp

{

public partial class CarModels

{

public CarModels()

{

Cars = new HashSet<Cars>();

}

public int CarModelId { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public ICollection<Cars> Cars { get; set; }

}

}

*Cars.cs*

using System;

using System.Collections.Generic;

namespace Lab2ConsoleApp

{

public partial class Cars

{

public Cars()

{

Rents = new HashSet<Rents>();

}

public int CarId { get; set; }

public int? CarModelId { get; set; }

public int? RegNum { get; set; }

public string Vincode { get; set; }

public int? EngineNum { get; set; }

public decimal? Price { get; set; }

public decimal? RentalPrice { get; set; }

public DateTime? IssueDate { get; set; }

public string Specs { get; set; }

public DateTime? TechnicalMaintenanceDate { get; set; }

public bool? SpecMark { get; set; }

public bool? ReturnMark { get; set; }

public int? EmployeeId { get; set; }

public CarModels CarModel { get; set; }

public Employees Employee { get; set; }

public ICollection<Rents> Rents { get; set; }

}

}

*Customers.cs*

using System;

using System.Collections.Generic;

namespace Lab2ConsoleApp

{

public partial class Customers

{

public Customers()

{

Rents = new HashSet<Rents>();

}

public int CustomerId { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Patronymic { get; set; }

public string PhoneNum { get; set; }

public string Address { get; set; }

public DateTime? BirthDate { get; set; }

public string PassportInfo { get; set; }

public bool? Gender { get; set; }

public ICollection<Rents> Rents { get; set; }

}

}

*Employees.cs*

using System;

using System.Collections.Generic;

namespace Lab2ConsoleApp

{

public partial class Employees

{

public Employees()

{

Cars = new HashSet<Cars>();

Rents = new HashSet<Rents>();

}

public int EmployeeId { get; set; }

public string Post { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Patronymic { get; set; }

public DateTime? EmploymentDate { get; set; }

public ICollection<Cars> Cars { get; set; }

public ICollection<Rents> Rents { get; set; }

}

}

*Rents.cs*

using System;

using System.Collections.Generic;

namespace Lab2ConsoleApp

{

public partial class Rents

{

public Rents()

{

AdditionalServices = new HashSet<AdditionalServices>();

}

public int RentId { get; set; }

public DateTime? ReturnDate { get; set; }

public DateTime? DeliveryDate { get; set; }

public int? CarId { get; set; }

public int? CustomerId { get; set; }

public int? EmployeeId { get; set; }

public decimal? Price { get; set; }

public Cars Car { get; set; }

public Customers Customer { get; set; }

public Employees Employee { get; set; }

public ICollection<AdditionalServices> AdditionalServices { get; set; }

}

}

*Services.cs*

using System;

using System.Collections.Generic;

namespace Lab2ConsoleApp

{

public partial class Services

{

public Services()

{

AdditionalServices = new HashSet<AdditionalServices>();

}

public int ServiceId { get; set; }

public string Name { get; set; }

public decimal? Price { get; set; }

public string Description { get; set; }

public ICollection<AdditionalServices> AdditionalServices { get; set; }

}

}

*appsettings.json*

{

"ConnectionStrings": {

"SQLConnection": "Server=DESKTOP-J4QINE6\\SQLEXPRESS;Database=car\_sharing;Trusted\_Connection=True;"

}

}