**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 9

Выполнил: студент гр. ИТП-31

Коршиков Н.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.

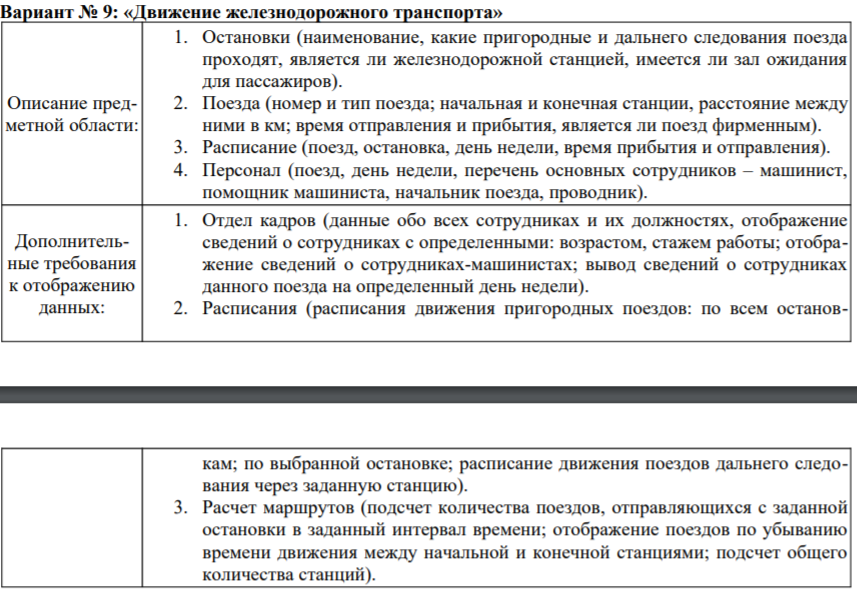


Рисунок 1 – Предметная область

**Ход работы:**

Задание 1

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

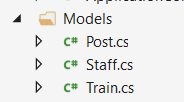


Рисунок 2 – содержимое папки *Models*

Структура созданного консольного приложения изображена на рисунке 3.

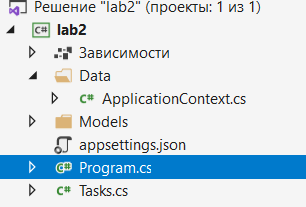


Рисунок 3 – Структура созданного приложения

Задание 2

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

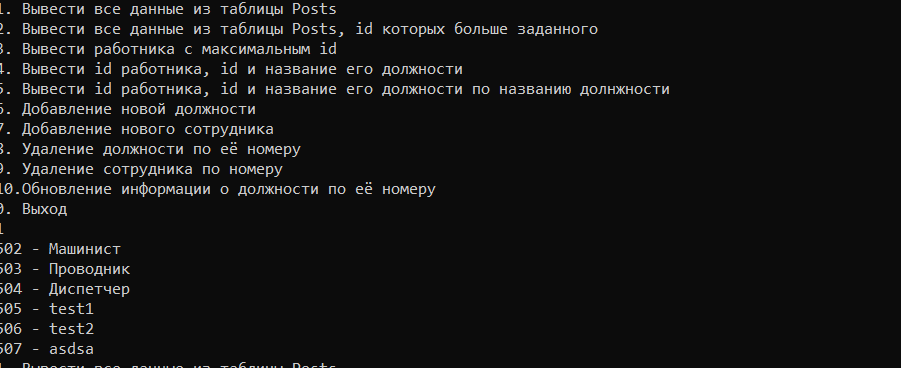


Рисунок 4 – Результат выполнения первого запроса

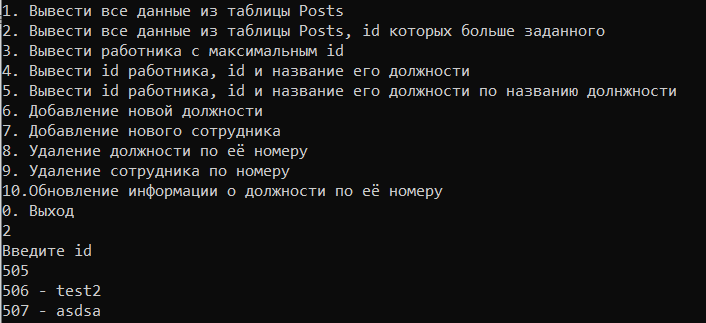


Рисунок 5 – Результат выполнения 2-го запроса

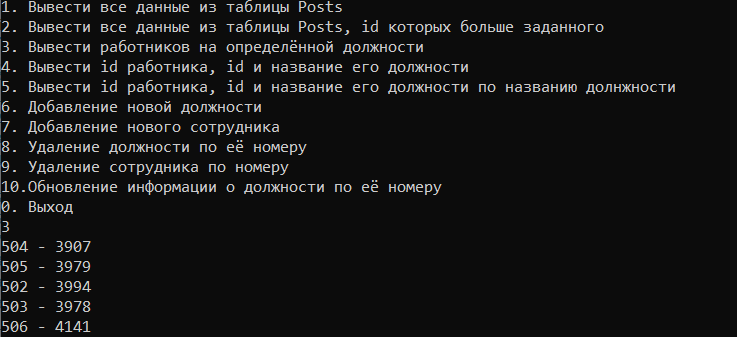


Рисунок 6 – Результат выполнения 3-го запроса

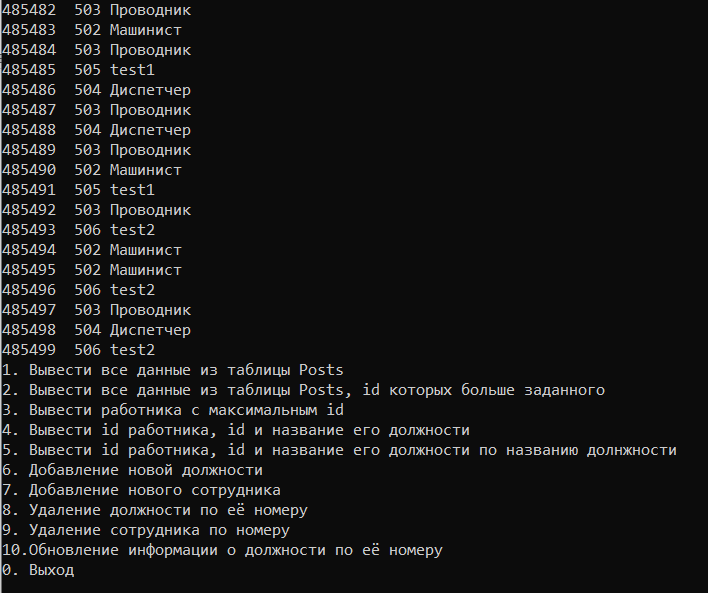


Рисунок 7 – Результат выполнения 4-го запроса

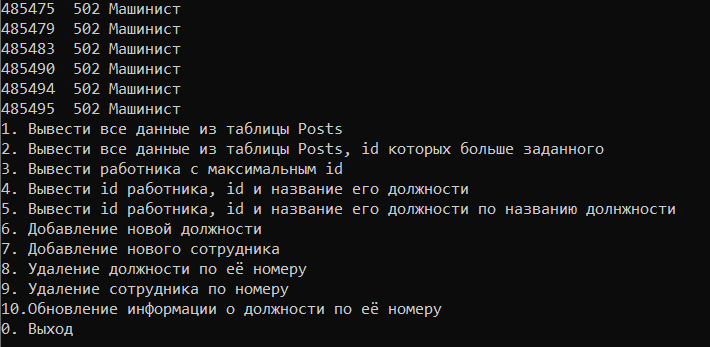


Рисунок 8 – Результат выполнения 5-го запроса

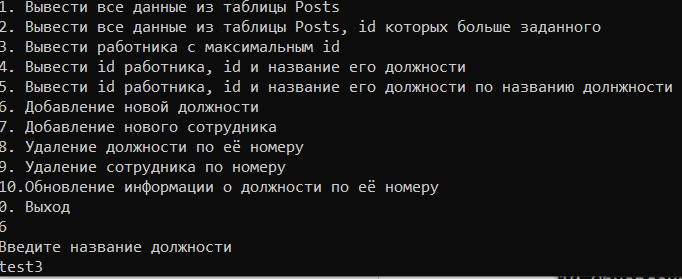


Рисунок 9 – Результат выполнения 6-го запроса

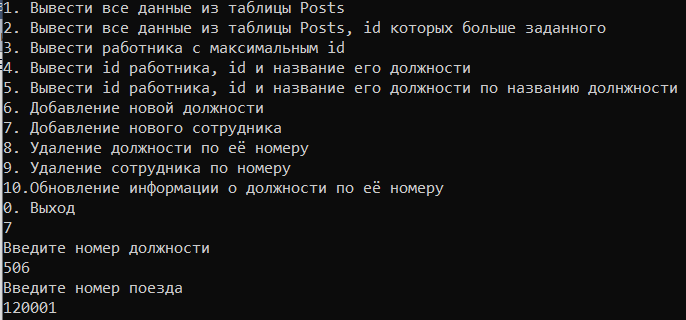


Рисунок 10 – Результат выполнения 7-го запроса

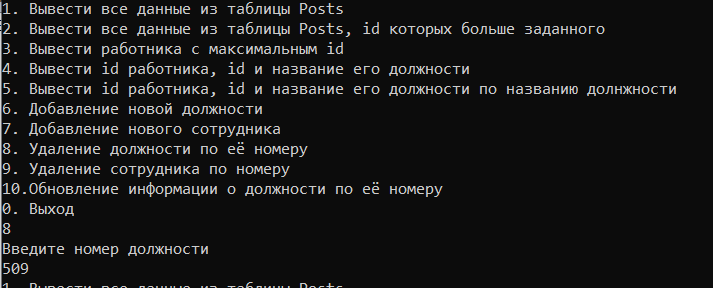


Рисунок 11 – Результат выполнения 8-го запроса

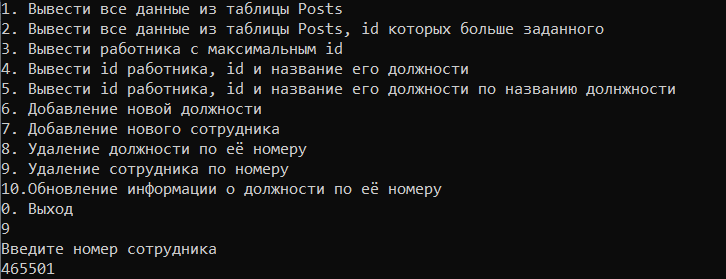


Рисунок 12 – Результат выполнения 9-го запроса

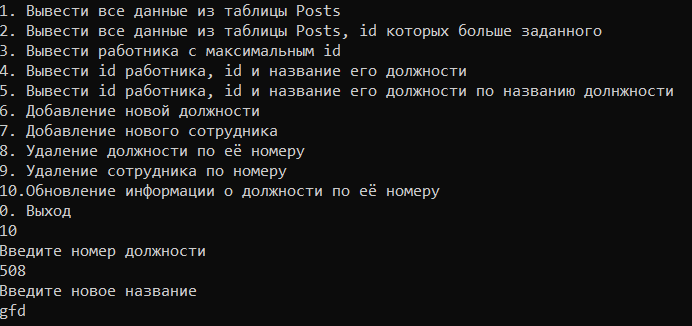


Рисунок 13 – Результат выполнения 10-го запроса

**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**

Листинг класса с запросами

*Tasks.cs*

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab2

{

class Tasks

{

public void Task1(DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

var posts = db.Posts.ToList();

foreach (Post p in posts)

{

Console.WriteLine($"{p.postId} - {p.nameOfPost}");

}

}

}

public void Task2(int i, DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

var posts = (from post in db.Posts

where post.postId > i

select post).ToList();

foreach (Post p in posts)

{

Console.WriteLine($"{p.postId} - {p.nameOfPost}");

}

}

}

public void Task3(DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

var posts = (from staff in db.Staffs

group staff by staff.postId into pos

select new {

НомерДолжности = pos.Key,

КоличествоСотрудников = pos.Count()

}).ToList();

foreach (var p in posts)

{

Console.WriteLine($"{p.НомерДолжности} - {p.КоличествоСотрудников}");

}

}

}

public void Task4(DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

var result = (from post in db.Posts

from staff in db.Staffs

where staff.postId == post.postId

select new { post, staff }).ToList();

foreach (var p in result)

{

Console.WriteLine($"{p.staff.staffId} {p.post.postId} {p.post.nameOfPost}");

}

}

}

public void Task5(string name, DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

var result = (from post in db.Posts

from staff in db.Staffs

where staff.postId == post.postId && post.nameOfPost == name

select new { post, staff }).ToList();

foreach (var p in result)

{

Console.WriteLine($"{p.staff.staffId} {p.post.postId} {p.post.nameOfPost}");

}

}

}

public void Task6(string name, DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

Post p = new Post(name);

db.Posts.Add(p);

db.SaveChanges();

}

}

public void Task7(int postId, int trainId, DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

Staff s = new Staff(trainId, postId);

db.Staffs.Add(s);

db.SaveChanges();

}

}

public void Task8(int postId, DbContextOptions<ApplicationContext> options)

{

using (ApplicationContext db = new ApplicationContext(options))

{

var posts = db.Posts.Where(p => p.postId == postId);

db.Posts.RemoveRange(posts);

db.SaveChanges();

}

}

public void Task9(int staffId, DbContextOptions<ApplicationContext> options)

{

using ApplicationContext db = new ApplicationContext(options);

var staffs = db.Staffs.Where(s => s.staffId == staffId);

db.Staffs.RemoveRange(staffs);

db.SaveChanges();

}

public void Task10(int postId, string nameOfPost, DbContextOptions<ApplicationContext> options)

{

using ApplicationContext db = new ApplicationContext(options);

Post posts = db.Posts.Where(s => s.postId == postId)

.FirstOrDefault();

posts.nameOfPost = nameOfPost;

db.SaveChanges();

}

}

}