ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Факультет Информационных технологий и программной инженерии
Кафедра Систем обработки данных

ОТЧЕТ

по практическому занятию №6

Тема: «Работа с Entity Framework»

Выполнил: студент 3 курса, гр. ИБ-31вп
_____ Ворожцов А. Е.

Принял:
_____ Медведев С. А.

Санкт-Петербург, 2025 г.

Содержание

Работа с Entity Framework

1. Сгенерированный код

Все сущности и контекст получены из скрипта `db/init/03-generator.sql`. Они расположены в `app/Entities.cs` и `app/DbContext.cs` (namespace `MyProjectNamespace`).

`app/Entities.cs`:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.EntityFrameworkCore;

namespace MyProjectNamespace
{
    [Table("conditions")]
    [PrimaryKey(nameof(Id))]
    public class Conditions
    {
        [Column("id")]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Column("name")]
        public string Name { get; set; } = null!;
    }

    [Table("cultures")]
    [PrimaryKey(nameof(Id))]
    public class Cultures
    {
        [Column("id")]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Column("name")]
        public string Name { get; set; } = null!;
    }

    [Table("experiments")]
    [PrimaryKey(nameof(Id))]
    public class Experiments
    {
        [Column("id")]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Column("year")]
        public int? Year { get; set; }
```

```
        [Column("field")]
        public int? Field { get; set; }

        [Column("culture_id")]
        public int? Culture_id { get; set; }

        [Column("predecessor_id")]
        public int? Predecessor_id { get; set; }

        [Column("condition_id")]
        public int? Condition_id { get; set; }

        [Column("yield")]
        public decimal? Yield { get; set; }
    }

    [Table("predecessors")]
    [PrimaryKey(nameof(Id))]
    public class Predecessors
    {
        [Column("id")]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [Column("name")]
        public string Name { get; set; } = null!;
    }
}
```

app/DbContext.cs:

```
using Microsoft.EntityFrameworkCore;

namespace MyProjectNamespace
{
    public class PublicDbContext : DbContext
    {
        public PublicDbContext(DbContextOptions<PublicDbContext> options) : base(options) { }
        public DbSet<Cultures> cultures { get; set; }
        public DbSet<Experiments> experiments { get; set; }
        public DbSet<Predecessors> predecessors { get; set; }
        public DbSet<Conditions> conditions { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            // Конфигурации моделей можно добавить здесь
        }
```

```
    }
}
```

2. Выбор таблицы

Для демонстрации CRUD-операций использована таблица `experiments` (см. `Experiments` в `app/Entities.cs`).

3. Вставка строки и вывод идентификатора

```
async Task<Experiments> AddExperimentAsync(PublicDbContext dbContext)
{
    var entity = new Experiments
    {
        Year = 2024,
        Field = 50,
        Culture_id = 3,
        Predecessor_id = 4,
        Condition_id = 2,
        Yield = 30
    };

    dbContext.experiments.Add(entity);
    await dbContext.SaveChangesAsync();
    Console.WriteLine($"Inserted ID: {entity.Id}");
    return entity;
}
```

Лог после запуска:

```
Inserted ID: 611
ID | Year | Field | Culture_id | Predecessor_id | Condition_id | Yield
-----------------------------------------------------------------------
611 | 2024 | 50 | 3 | 4 | 2 | 30
```

4. Вывод всего содержимого (рефлексия)

```
var experimentProperties = typeof(Experiments).GetProperties();

void PrintTableHeader()
{
    var header = string.Join(" | ", experimentProperties.Select(p => p.Name));
    Console.WriteLine(header);
    Console.WriteLine(new string('-', header.Length));
}
```

```
void PrintRow(Experiments exp)
{
    var values = experimentProperties.Select(p =>
    {
        var value = p.GetValue(exp);
        return value switch
        {
            null => "null",
            decimal d => d.ToString("0.##"),
            _ => value?.ToString() ?? string.Empty
        };
    });
    Console.WriteLine(string.Join(" | ", values));
}
```

Лог:

```
ID | Year | Field | Culture_id | Predecessor_id | Condition_id | Yield
-----------------------------------------------------------------------
607 | 2024 | 50 | 3 | 4 | 2 | 30
606 | 2024 | 50 | 3 | 4 | 2 | 30
... (обрезано)
```

5. Обновление записи

```
async Task<Experiments?> UpdateExperimentAsync(PublicDbContext dbContext, int id)
{
    var entity = await dbContext.experiments.FirstOrDefaultAsync(e => e.Id == id);
    if (entity is null)
    {
        return null;
    }

    entity.Yield = (entity.Yield ?? 0) + 5;
    entity.Field = (entity.Field ?? 0) + 1;
    entity.Year = (entity.Year ?? DateTime.UtcNow.Year) + 1;

    await dbContext.SaveChangesAsync();
    await dbContext.Entry(entity).ReloadAsync();
    return entity;
}
```

Лог после обновления:

```
ID | Year | Field | Culture_id | Predecessor_id | Condition_id | Yield
-----------------------------------------------------------------------
611 | 2025 | 51 | 3 | 4 | 2 | 35
```

6. Удаление записи

```
async Task<bool> DeleteExperimentAsync(PublicDbContext dbContext, int id)
{
    var entity = await dbContext.experiments.FirstOrDefaultAsync(e => e.Id == id);
    if (entity is null)
    {
        return false;
    }

    dbContext.experiments.Remove(entity);
    await dbContext.SaveChangesAsync();
    return true;
}
```

Лог после удаления:

```
Эксперимент с ID 611 удалён.
ID | Year | Field | Culture_id | Predecessor_id | Condition_id | Yield
-----------------------------------------------------------------------
607 | 2024 | 50 | 3 | 4 | 2 | 30
...
```