```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;

public abstract class Pessoa
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public string Cpf { get; set; }
    public string Email { get; set; }
}

public class Funcionario : Pessoa
{
    public DateTime DataAdmissao { get; set; }
    public string Funcao { get; set; }
    public double Salario { get; set; }
}

public class Aluno : Pessoa
{
    public int CursoId { get; set; }
    public Curso Curso { get; set; }
    public DateTime DataMatricula { get; set; }
    public string RA { get; set; }
}

public class Curso
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public int CoordenadorId { get; set; }
    public Coordenador Coordenador { get; set; }
    public int Duracao { get; set; }
    public List<Disciplina> Disciplinas { get; set; } = new List<Disciplina>();
}

public class Disciplina
{
    public int Id { get; set; }
    public string Nome { get; set; }
    public string Codigo { get; set; }
    public int Creditos { get; set; }
    public int ProfessorId { get; set; }
    public Professor Professor { get; set; }
    public int CursoId { get; set; }
    public Curso Curso { get; set; }
}
```

```
public class Coordenador : Funcionario
{
    public List<Curso> CursosCoordenados { get; set; } = new List<Curso>();
}

public class Professor : Funcionario
{
    public List<Disciplina> DisciplinasLecionadas { get; set; } = new List<Disciplina>();
}

public class GradeCurricular
{
    public List<Disciplina> Disciplinas { get; set; } = new List<Disciplina>();

    public void AdicionarDisciplina(Disciplina disciplina)
    {
        Disciplinas.Add(disciplina);
        Console.WriteLine($"Disciplina {disciplina.Nome} adicionada à grade.");
    }

    public void RemoverDisciplina(string codigo)
    {
        Disciplina disciplina = Disciplinas.Find(d => d.Codigo == codigo);
        if (disciplina != null)
        {
            Disciplinas.Remove(disciplina);
            Console.WriteLine($"Disciplina {disciplina.Nome} removida da grade.");
        }
        else
        {
            Console.WriteLine($"Disciplina com código {codigo} não encontrada na grade.");
        }
    }

    public void ExibirGrade()
    {
        Console.WriteLine("Grade Curricular:");
        foreach (var disciplina in Disciplinas)
        {
            Console.WriteLine($"Disciplina: {disciplina.Nome}, Código: {disciplina.Codigo},
Créditos: {disciplina.Creditos}, Professor: {disciplina.Professor.Nome}");
        }
    }
}

public class ApplicationContext : DbContext
{
```

```
    public DbSet<Pessoa> Pessoas { get; set; }
    public DbSet<Funcionario> Funcionarios { get; set; }
    public DbSet<Aluno> Alunos { get; set; }
    public DbSet<Curso> Cursos { get; set; }
    public DbSet<Disciplina> Disciplinas { get; set; }
    public DbSet<Coordenador> Coordenadores { get; set; }
    public DbSet<Professor> Professores { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlite("Data Source=academico.db");
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<Pessoa>()
            .HasDiscriminator<string>("PessoaType")
            .HasValue<Funcionario>("Funcionario")
            .HasValue<Aluno>("Aluno")
            .HasValue<Coordenador>("Coordenador")
            .HasValue<Professor>("Professor");

        modelBuilder.Entity<Curso>()
            .HasOne(c => c.Coordenador)
            .WithMany(c => c.CursosCoordenados)
            .HasForeignKey(c => c.CoordenadorId);

        modelBuilder.Entity<Disciplina>()
            .HasOne(d => d.Professor)
            .WithMany(p => p.DisciplinasLecionadas)
            .HasForeignKey(d => d.ProfessorId);

        modelBuilder.Entity<Disciplina>()
            .HasOne(d => d.Curso)
            .WithMany(c => c.Disciplinas)
            .HasForeignKey(d => d.CursoId);
    }
}

class Program
{
    static void Main(string[] args)
    {
        using (var context = new ApplicationContext())
        {
            context.Database.EnsureDeleted();
```

```
        context.Database.EnsureCreated();

        var coordenadorEngenharia = new Coordenador { Nome = "Carlos", Cpf =
"12345678900", Email = "carlos@exemplo.com", DataAdmissao = DateTime.Now, Funcao =
"Coordenador", Salario = 8000 };
        var coordenadorPsicologia = new Coordenador { Nome = "Maria", Cpf =
"22345678900", Email = "maria@exemplo.com", DataAdmissao = DateTime.Now, Funcao =
"Coordenador", Salario = 8500 };
        var coordenadorDireito = new Coordenador { Nome = "João", Cpf = "32345678900",
Email = "joao@exemplo.com", DataAdmissao = DateTime.Now, Funcao = "Coordenador",
Salario = 9000 };
        var coordenadorEnfermagem = new Coordenador { Nome = "Ana", Cpf =
"42345678900", Email = "ana@exemplo.com", DataAdmissao = DateTime.Now, Funcao =
"Coordenador", Salario = 8200 };

        var professorEngenharia = new Professor { Nome = "Ana", Cpf = "98765432100",
Email = "ana@exemplo.com", DataAdmissao = DateTime.Now, Funcao = "Professor",
Salario = 6000 };
        var professorPsicologia = new Professor { Nome = "Pedro", Cpf = "98765432101",
Email = "pedro@exemplo.com", DataAdmissao = DateTime.Now, Funcao = "Professor",
Salario = 6100 };
        var professorDireito = new Professor { Nome = "Lucas", Cpf = "98765432102", Email
= "lucas@exemplo.com", DataAdmissao = DateTime.Now, Funcao = "Professor", Salario =
6200 };
        var professorEnfermagem = new Professor { Nome = "Fernanda", Cpf =
"98765432103", Email = "fernanda@exemplo.com", DataAdmissao = DateTime.Now,
Funcao = "Professor", Salario = 6300 };

        var cursoEngenharia = new Curso { Nome = "Engenharia de Software", Coordenador
= coordenadorEngenharia, Duracao = 8 };
        var cursoPsicologia = new Curso { Nome = "Psicologia", Coordenador =
coordenadorPsicologia, Duracao = 10 };
        var cursoDireito = new Curso { Nome = "Direito", Coordenador = coordenadorDireito,
Duracao = 10 };
        var cursoEnfermagem = new Curso { Nome = "Enfermagem", Coordenador =
coordenadorEnfermagem, Duracao = 8 };

        var disciplinaEngenharia1 = new Disciplina { Nome = "Programação Orientada a
Objetos", Codigo = "POO101", Creditos = 4, Professor = professorEngenharia, Curso =
cursoEngenharia };
        var disciplinaEngenharia2 = new Disciplina { Nome = "Estruturas de Dados", Codigo
= "ED102", Creditos = 4, Professor = professorEngenharia, Curso = cursoEngenharia };

        var disciplinaPsicologia1 = new Disciplina { Nome = "Psicologia Geral", Codigo =
"PSI101", Creditos = 3, Professor = professorPsicologia, Curso = cursoPsicologia };
        var disciplinaPsicologia2 = new Disciplina { Nome = "Psicologia do
Desenvolvimento", Codigo = "PSI102", Creditos = 3, Professor = professorPsicologia, Curso
= cursoPsicologia };
```

```
        var disciplinaDireito1 = new Disciplina { Nome = "Direito Constitucional", Codigo =
"DIR101", Creditos = 5, Professor = professorDireito, Curso = cursoDireito };
        var disciplinaDireito2 = new Disciplina { Nome = "Direito Civil", Codigo = "DIR102",
Creditos = 5, Professor = professorDireito, Curso = cursoDireito };

        var disciplinaEnfermagem1 = new Disciplina { Nome = "Anatomia Humana", Codigo
= "ENF101", Creditos = 4, Professor = professorEnfermagem, Curso = cursoEnfermagem };
        var disciplinaEnfermagem2 = new Disciplina { Nome = "Enfermagem Clínica",
Codigo = "ENF102", Creditos = 4, Professor = professorEnfermagem, Curso =
cursoEnfermagem };

        context.Coordenadores.AddRange(coordenadorEngenharia, coordenadorPsicologia,
coordenadorDireito, coordenadorEnfermagem);
        context.Professores.AddRange(professorEngenharia, professorPsicologia,
professorDireito, professorEnfermagem);
        context.Cursos.AddRange(cursoEngenharia, cursoPsicologia, cursoDireito,
cursoEnfermagem);
        context.Disciplinas.AddRange(disciplinaEngenharia1, disciplinaEngenharia2,
disciplinaPsicologia1, disciplinaPsicologia2, disciplinaDireito1, disciplinaDireito2,
disciplinaEnfermagem1, disciplinaEnfermagem2);

        context.SaveChanges();
    }

    int opcao;
    do
    {
        Console.WriteLine("\nSistema de Gestão Acadêmica");
        Console.WriteLine("1. Selecionar Curso");
        Console.WriteLine("2. Adicionar Disciplina à Grade");
        Console.WriteLine("3. Remover Disciplina da Grade");
        Console.WriteLine("4. Exibir Grade Curricular");
        Console.WriteLine("0. Sair");
        Console.Write("Escolha uma opção: ");
        opcao = int.Parse(Console.ReadLine());

        switch (opcao)
        {
            case 1:
                SelecionarCurso();
                break;
            case 2:
                AdicionarDisciplina();
                break;
            case 3:
                RemoverDisciplina();
                break;
```

```
                case 4:
                    ExibirGrade();
                    break;
                case 0:
                    Console.WriteLine("Saindo...");
                    break;
                default:
                    Console.WriteLine("Opção inválida. Tente novamente.");
                    break;
            }
        } while (opcao != 0);
    }

    static Curso cursoSelecionado;
    static GradeCurricular grade = new GradeCurricular();

    static void SelecionarCurso()
    {
        using (var context = new ApplicationContext())
        {
            var cursos = context.Cursos.Include(c => c.Coordenador).Include(c =>
c.Disciplinas).ThenInclude(d => d.Professor).ToList();

            Console.WriteLine("\nCursos Disponíveis:");
            for (int i = 0; i < cursos.Count; i++)
            {
                Console.WriteLine($"{i + 1}. {cursos[i].Nome}");
            }
            Console.Write("Escolha um curso: ");
            int escolha = int.Parse(Console.ReadLine());
            cursoSelecionado = cursos[escolha - 1];
            Console.WriteLine($"\nCurso Selecionado: {cursoSelecionado.Nome}");
            Console.WriteLine($"Coordenador: {cursoSelecionado.Coordenador.Nome}");
            Console.WriteLine($"Duração: {cursoSelecionado.Duracao} semestres");
            Console.WriteLine("Disciplinas:");
            foreach (var disciplina in cursoSelecionado.Disciplinas)
            {
                Console.WriteLine($"- {disciplina.Nome} (Código: {disciplina.Codigo}, Créditos:
{disciplina.Creditos}, Professor: {disciplina.Professor.Nome})");
            }
        }
    }

    static void AdicionarDisciplina()
    {
        if (cursoSelecionado == null)
        {
            Console.WriteLine("Nenhum curso selecionado.");
```

```
            return;
        }

        using (var context = new ApplicationContext())
        {
            var curso = context.Cursos.Include(c => c.Disciplinas).ThenInclude(d =>
d.Professor).FirstOrDefault(c => c.Id == cursoSelecionado.Id);

            if (curso != null)
            {
                Console.WriteLine("\nDisciplinas Disponíveis:");
                foreach (var disciplina in curso.Disciplinas)
                {
                    Console.WriteLine($"- {disciplina.Nome} (Código: {disciplina.Codigo})");
                }
                Console.Write("Digite o código da disciplina que deseja adicionar: ");
                string codigo = Console.ReadLine();
                Disciplina disciplinaSelecionada = curso.Disciplinas.Find(d => d.Codigo ==
codigo);
                if (disciplinaSelecionada != null)
                {
                    grade.AdicionarDisciplina(disciplinaSelecionada);
                }
                else
                {
                    Console.WriteLine("Disciplina não encontrada.");
                }
            }
        }
    }

    static void RemoverDisciplina()
    {
        Console.Write("Digite o código da disciplina que deseja remover: ");
        string codigo = Console.ReadLine();
        grade.RemoverDisciplina(codigo);
    }

    static void ExibirGrade()
    {
        grade.ExibirGrade();
    }
}
```