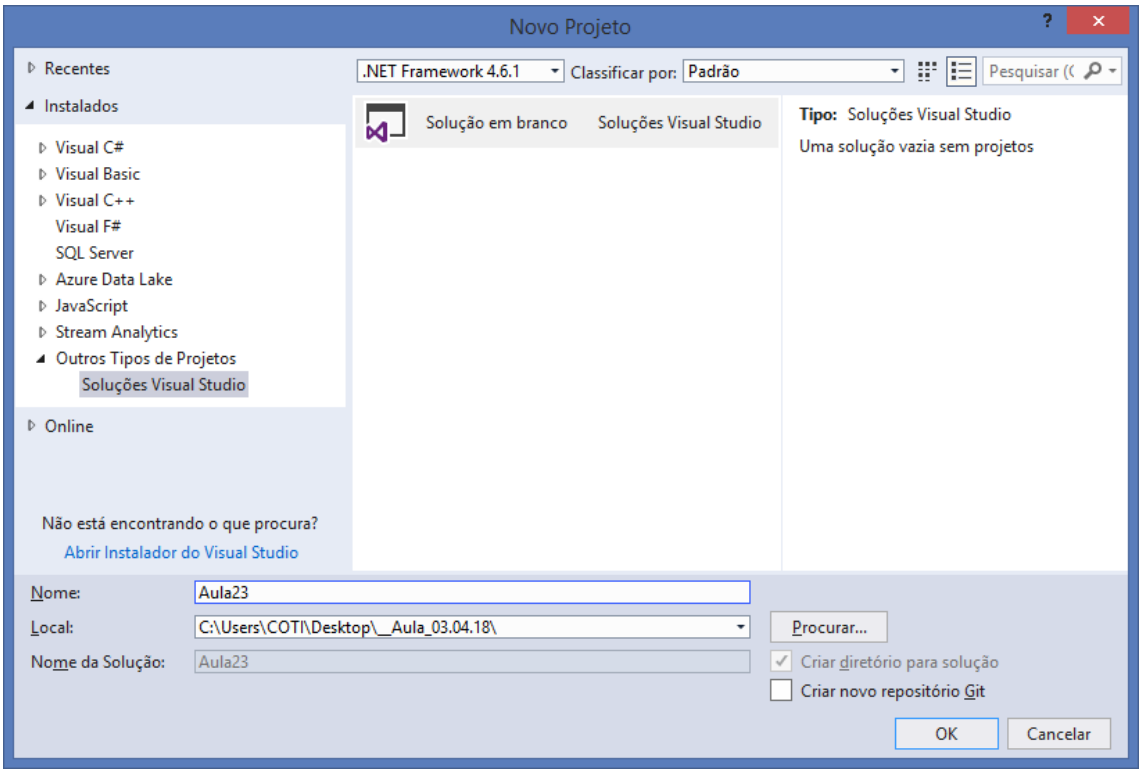
C#.NET WebDeveloper

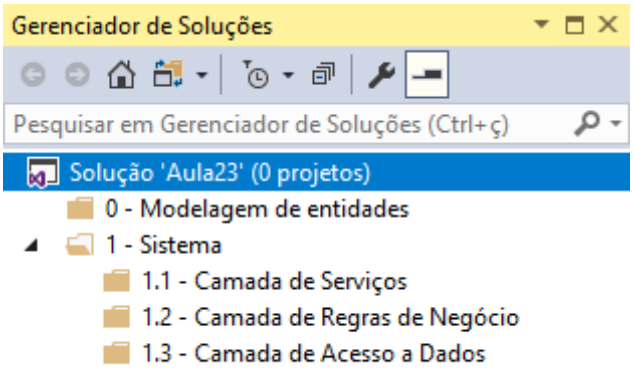
Terça-feira, 03 de Abril de 2018

Desenvolvimento web com Asp.Net WebApi. Injeção de Dependência com Simple Injector e Acesso a banco de dados com EntityFramework.

Aula 23

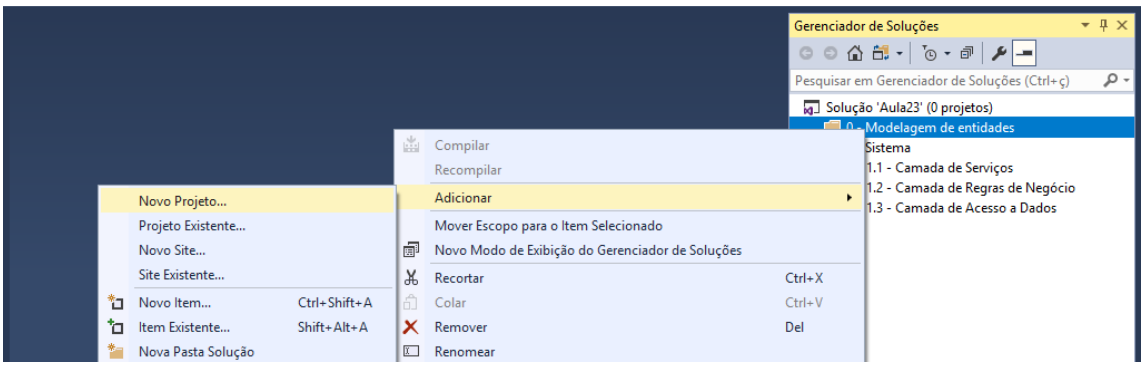
Criando um nova solution em branco:

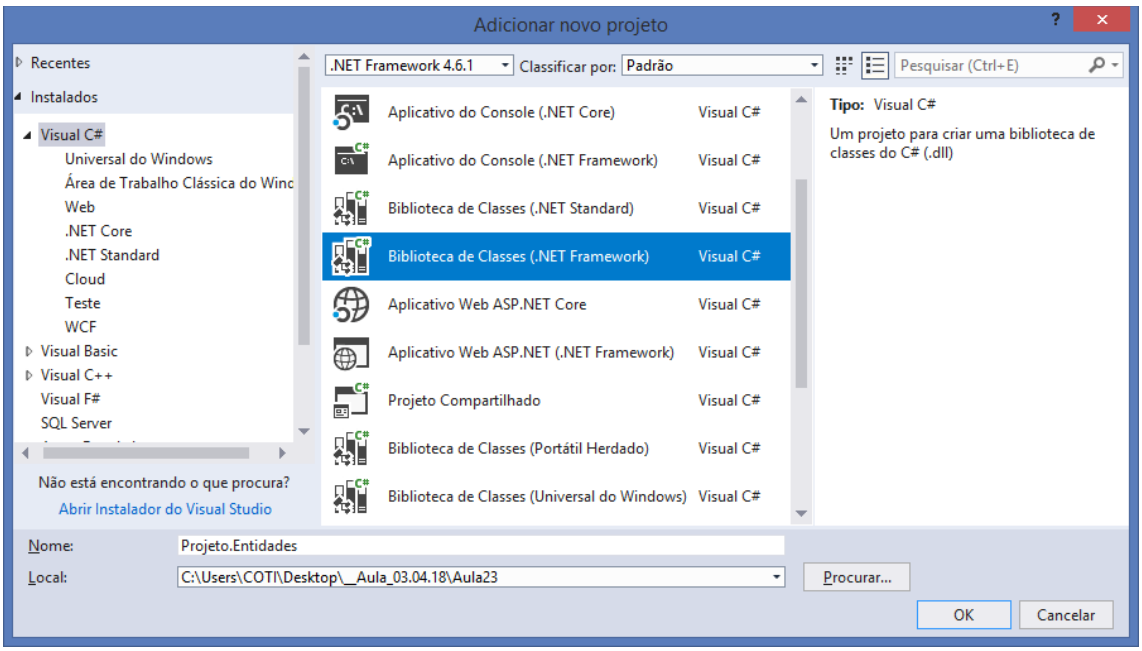




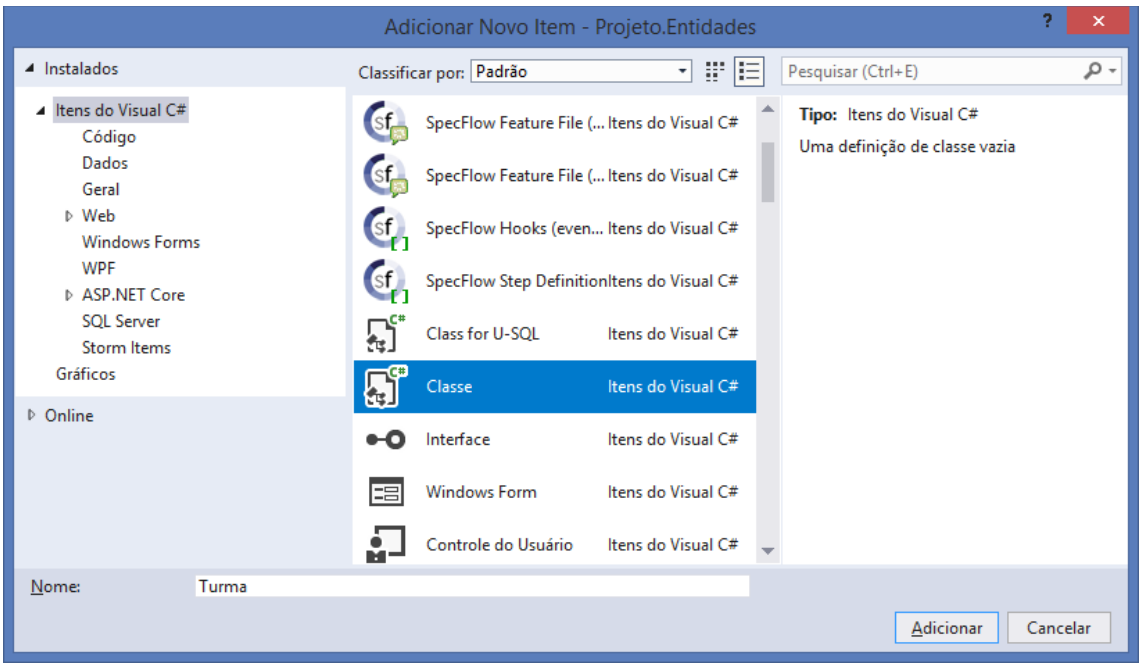
---

0 - Modelagem de entidades






```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.Entidades
{
    public class Aluno
    {
        public int IdAluno { get; set; }
        public string Nome { get; set; }
        public string Email { get; set; }
        public string Matricula { get; set; }

        public Aluno()
        {
            //construtor default..
        }

        //sobrecarga (overloading) de construtores..
        public Aluno(int idAluno, string nome, string email, string matricula)
        {
            IdAluno = idAluno;
            Nome = nome;
            Email = email;
            Matricula = matricula;
        }

        //sobrescrita de método (override)
        public override string ToString()
        {
            return $"Id: {IdAluno}, Nome: {Nome}, Email: {Email},
                    Matricula: {Matricula}";
        }
    }
}
```




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.Entidades
{
    public class Turma
    {
        public int IdTurma { get; set; }
        public string Curso { get; set; }
        public DateTime DataInicio { get; set; }
        public DateTime DataTermino { get; set; }

        public Turma()
        {
            //construtor default..
        }

        public Turma(int idTurma, string curso, DateTime dataInicio,
                     DateTime dataTermino)
        {
            IdTurma = idTurma;
            Curso = curso;
            DataInicio = dataInicio;
            DataTermino = dataTermino;
        }

        public override string ToString()
        {
            return $"Id: {IdTurma}, Curso: {Curso},
                    Inicio: {DataInicio}, Termino: {DataTermino}";
        }
    }
}
```

Relacionamento de multiplicidade Muitos para Muitos

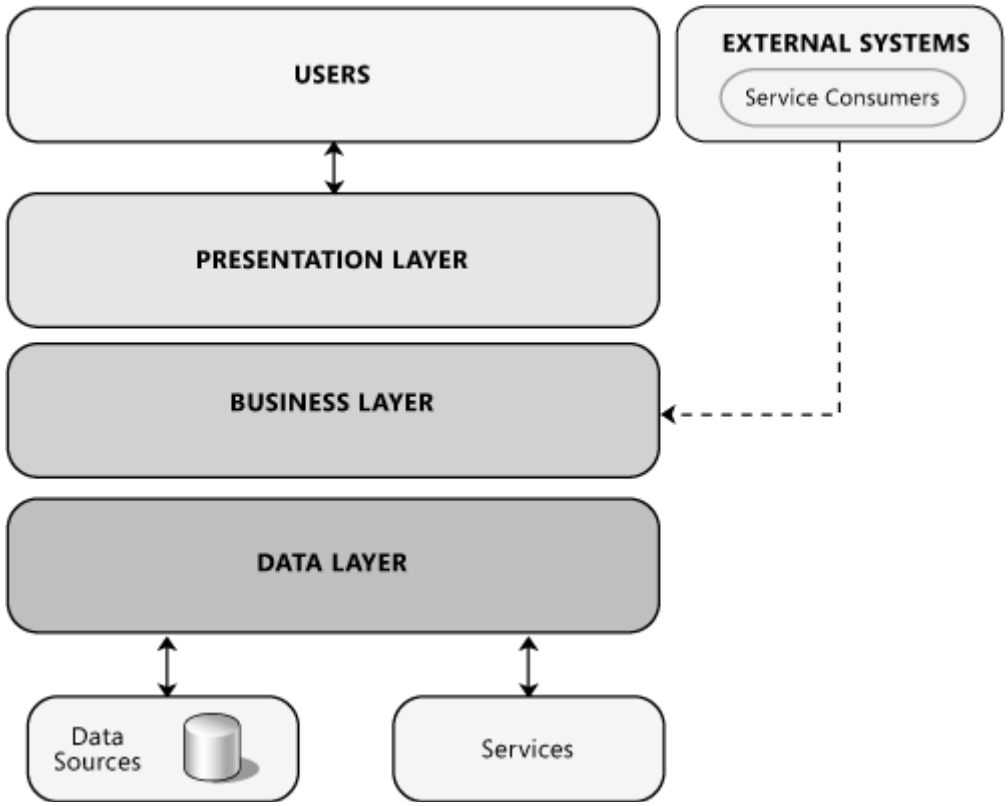
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.Entidades
{
    public class Turma
    {
        public virtual int IdTurma { get; set; }
        public virtual string Curso { get; set; }
        public virtual DateTime DataInicio { get; set; }
        public virtual DateTime DataTermino { get; set; }

        //Relacionamento TER-MUITOS
        public virtual List<Aluno> Alunos { get; set; }

        public Turma()
        {
            //construtor default..
        }

        public Turma(int idTurma, string curso, DateTime dataInicio,
                     DateTime dataTermino)
        {
            IdTurma = idTurma;
            Curso = curso;
            DataInicio = dataInicio;
            DataTermino = dataTermino;
        }

        public override string ToString()
        {
            return $"Id: {IdTurma}, Curso: {Curso},
                     Inicio: {DataInicio}, Termino: {DataTermino}";
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.Entidades
{
    public class Aluno
    {
        public virtual int IdAluno { get; set; }
        public virtual string Nome { get; set; }
        public virtual string Email { get; set; }
        public virtual string Matricula { get; set; }

        //Relacionamento TER-MUITOS..
        public virtual List<Turma> Turmas { get; set; }
```

```
        public Aluno()
        {
            //construtor default..
        }

        //sobrecarga (overloading) de construtores..
        public Aluno(int idAluno, string nome, string email, string matricula)
        {
            IdAluno = idAluno;
            Nome = nome;
            Email = email;
            Matricula = matricula;
        }

        //sobrescrita de método (override)
        public override string ToString()
        {
            return $"Id: {IdAluno}, Nome: {Nome}, Email: {Email},
                    Matricula: {Matricula}";
        }
    }
}
```

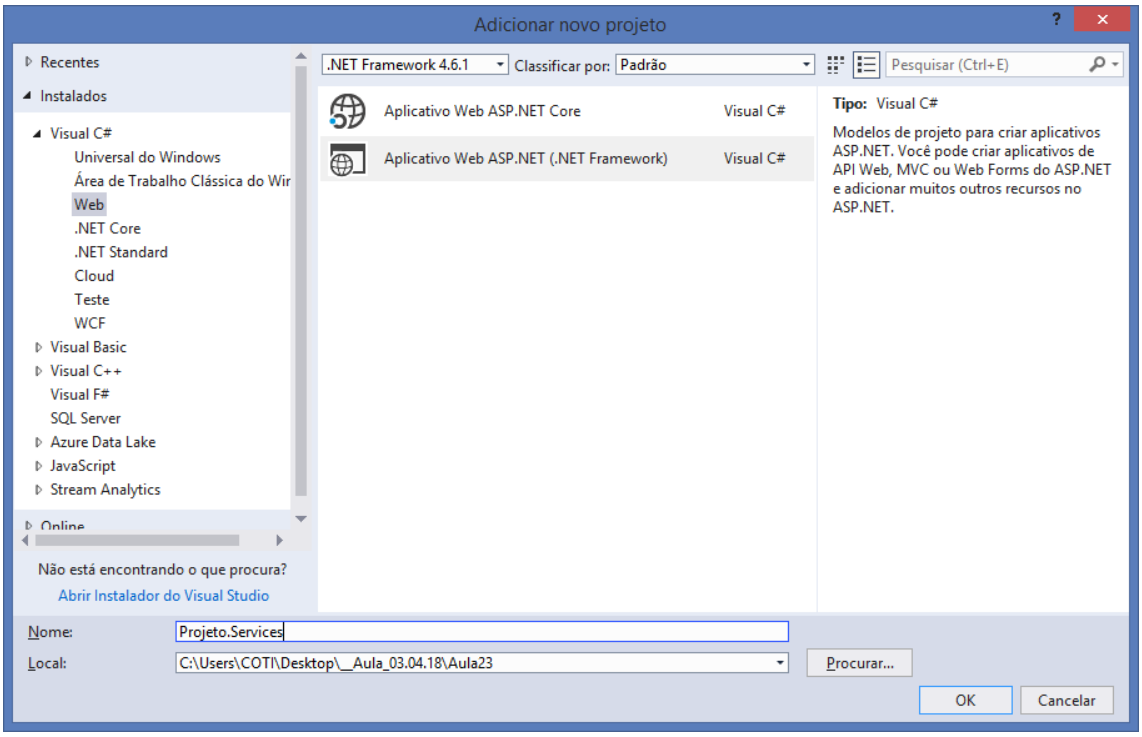
Arquitetura do projeto

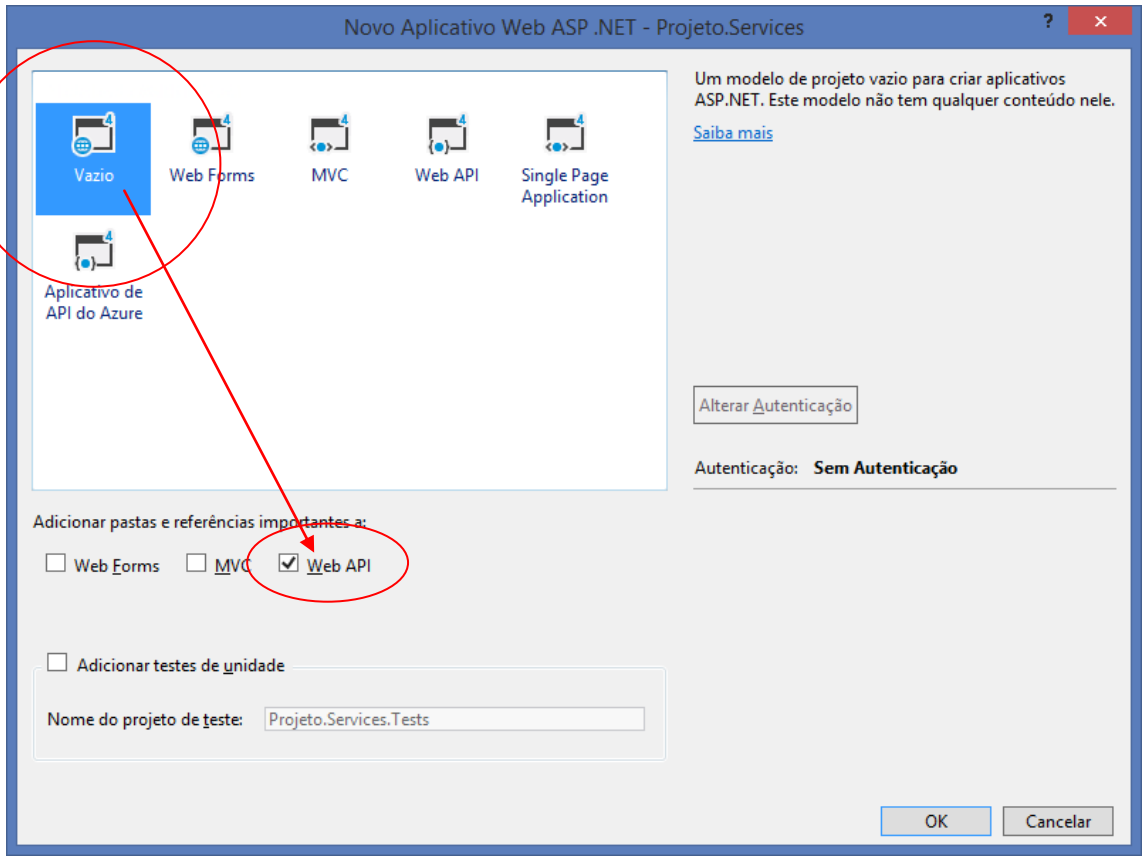
Desenvolvimento baseado em camadas



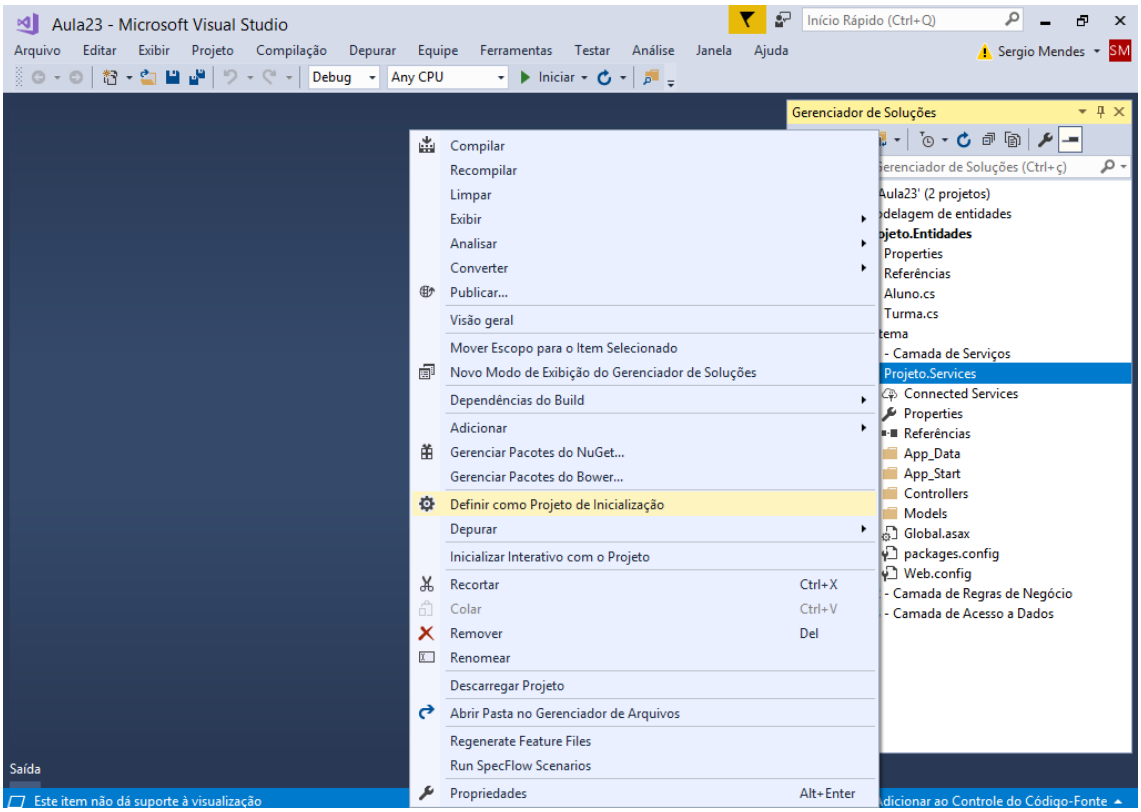
1.1 - Camada de Serviços

Projeto Asp.Net WebApplication (.NET Framework)





Definindo o projeto de incialização da Solution:

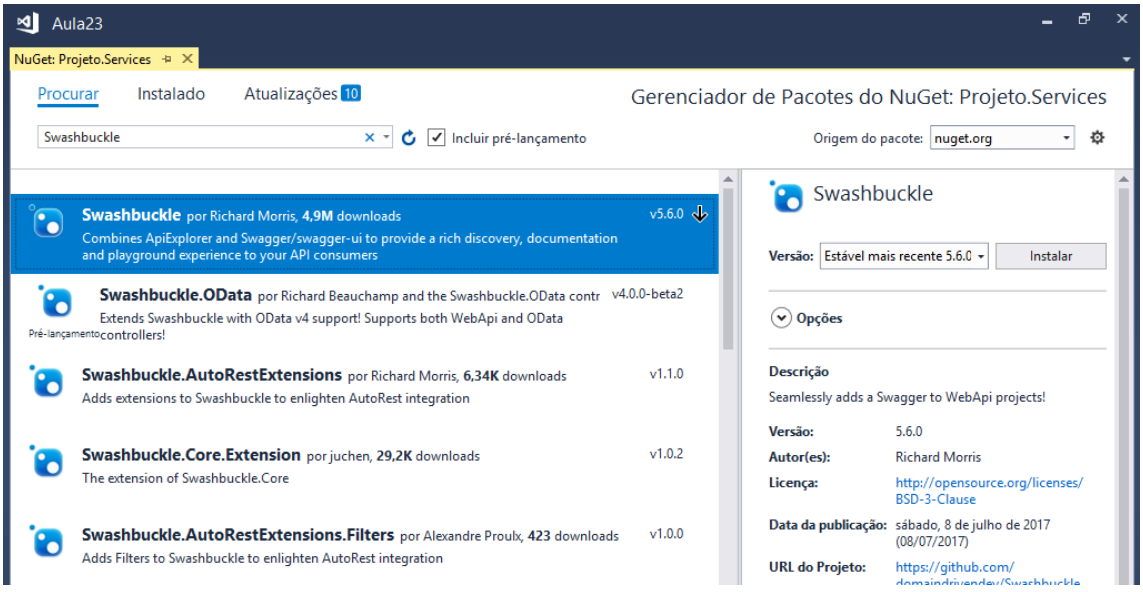


Swagger

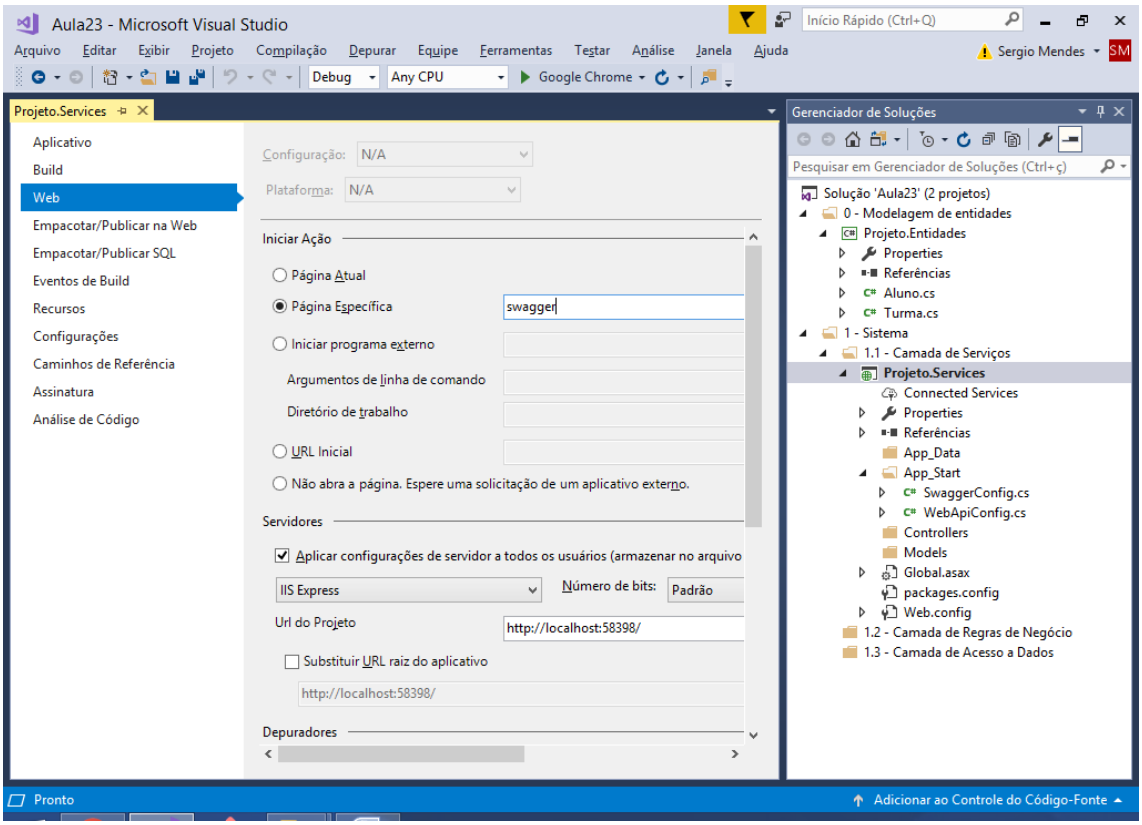
Framework para geração de documentação
de serviços em projetos do tipo WebApi.

- Instalando:

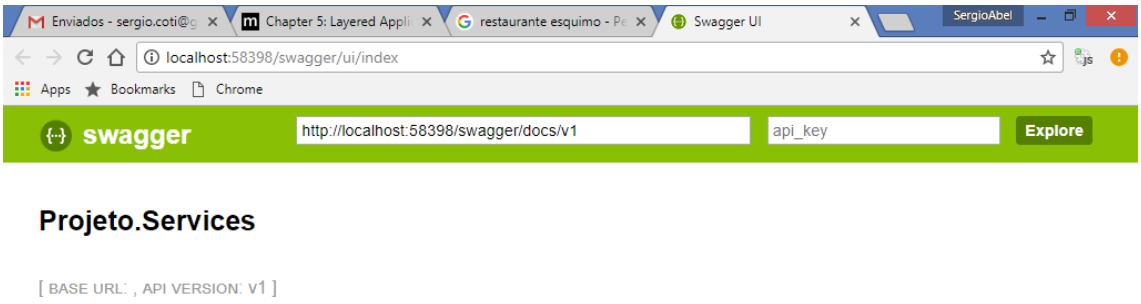
Gerenciar pacotes do Nuget



Configurando a página inicial do projeto:

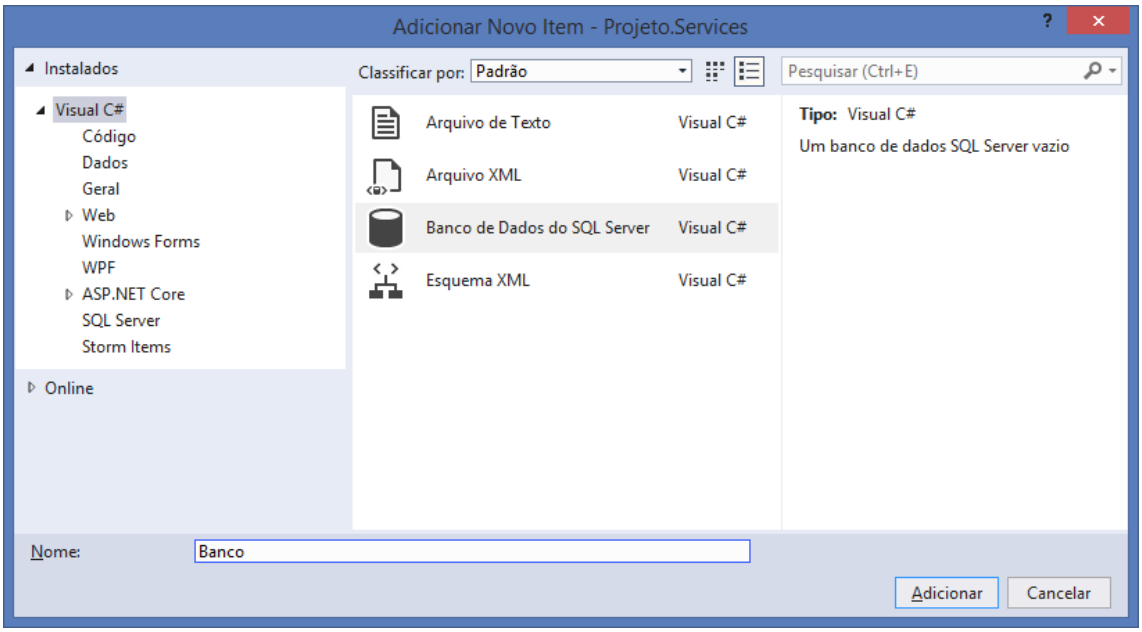


http://localhost:58398/swagger/ui/index



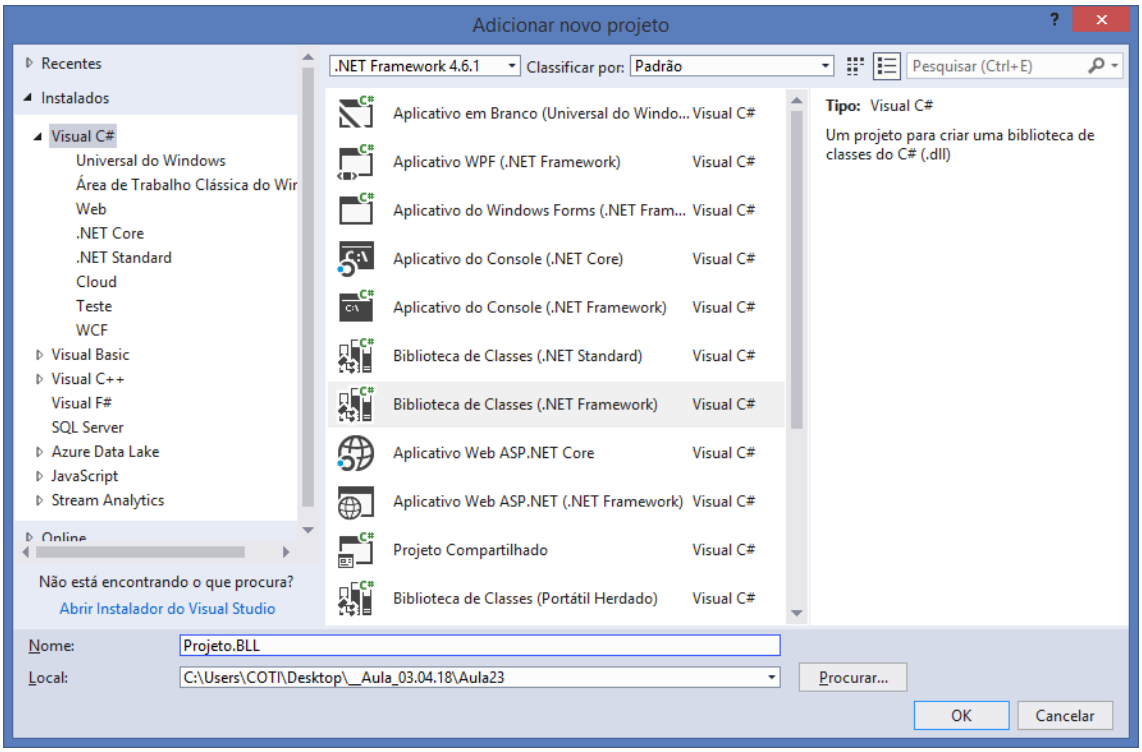
Criando a base de dados:

MDF - Master Database File



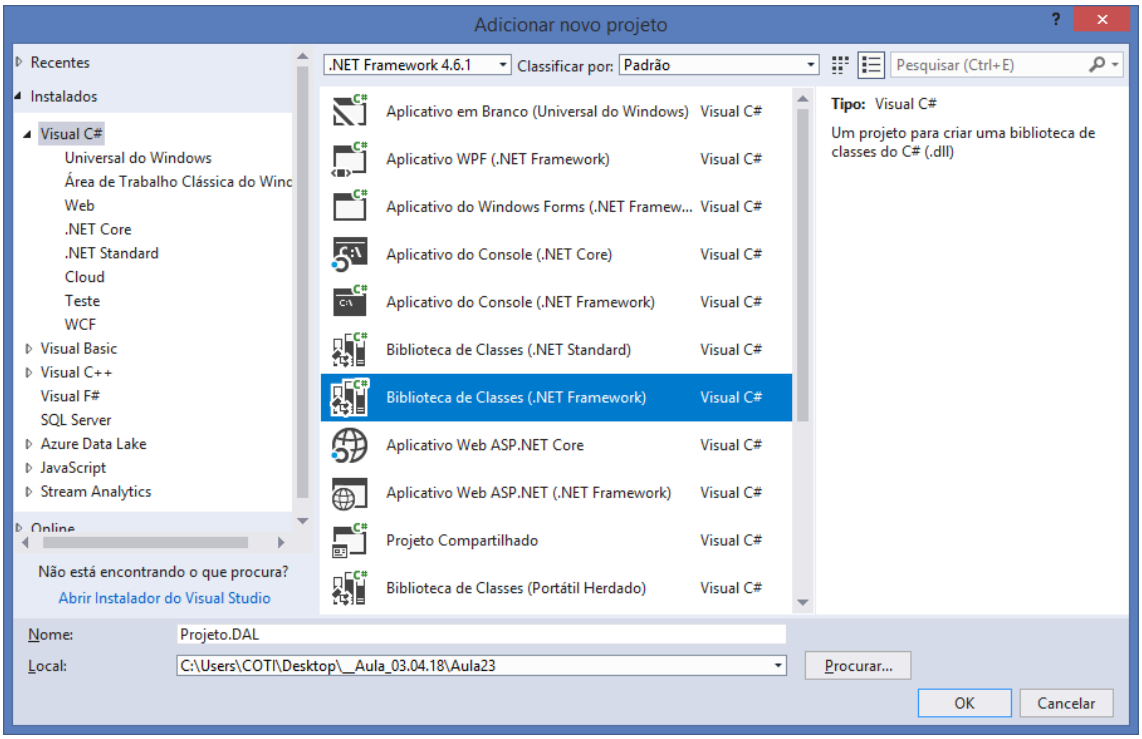
1.2 - Camada de Regras de Negócio

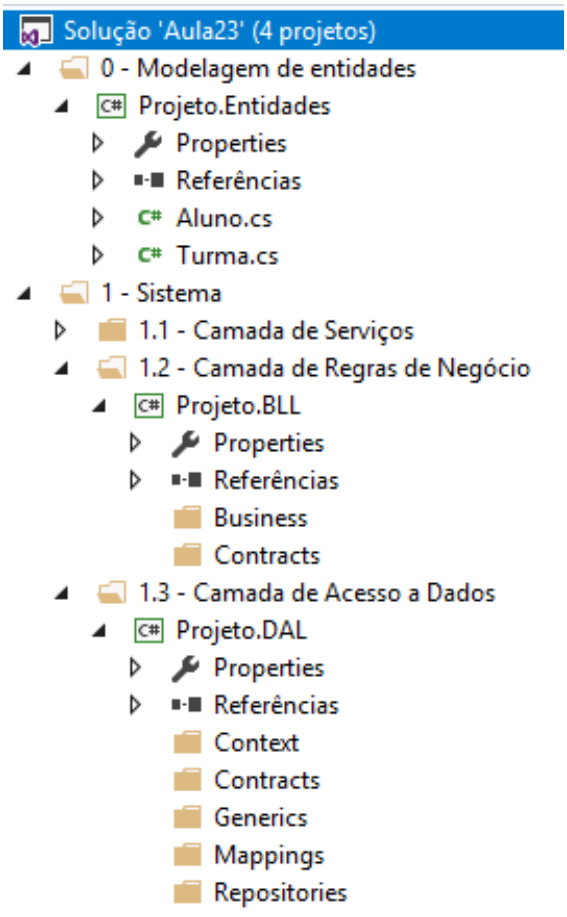
BLL - Business Logic Layer



1.3 - Camada de Acesso a dados

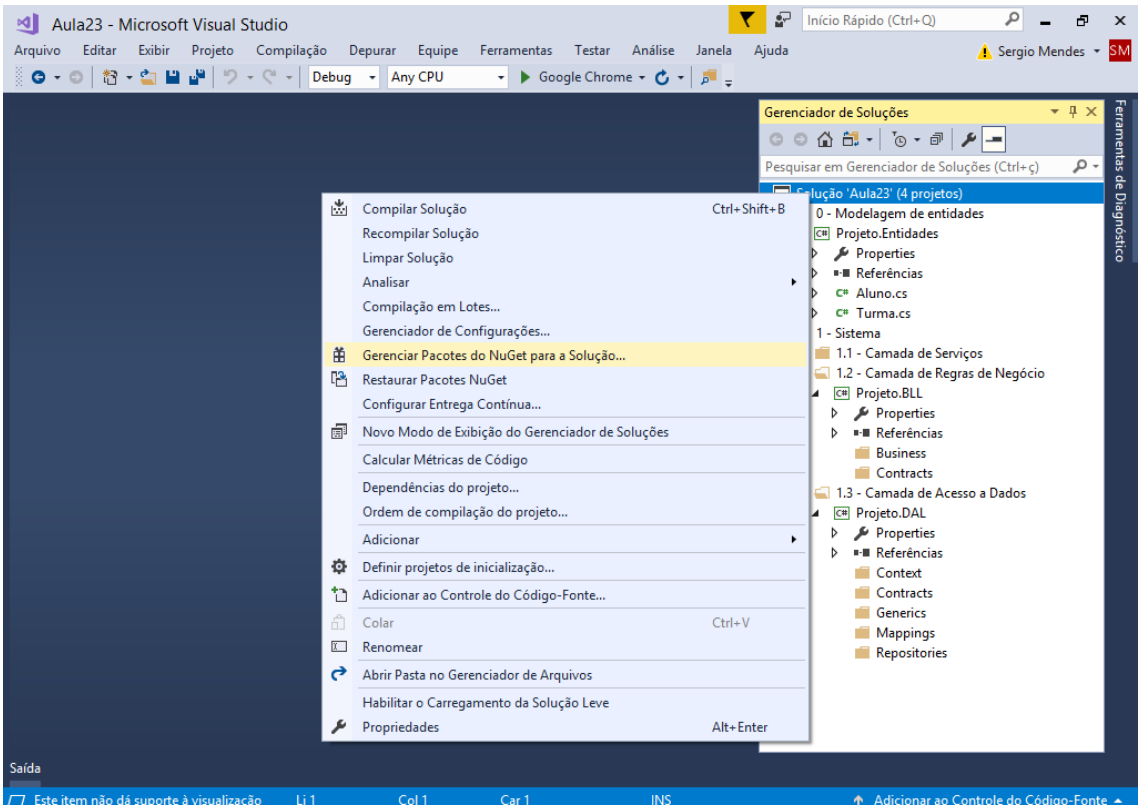
DAL - Data Access Layer

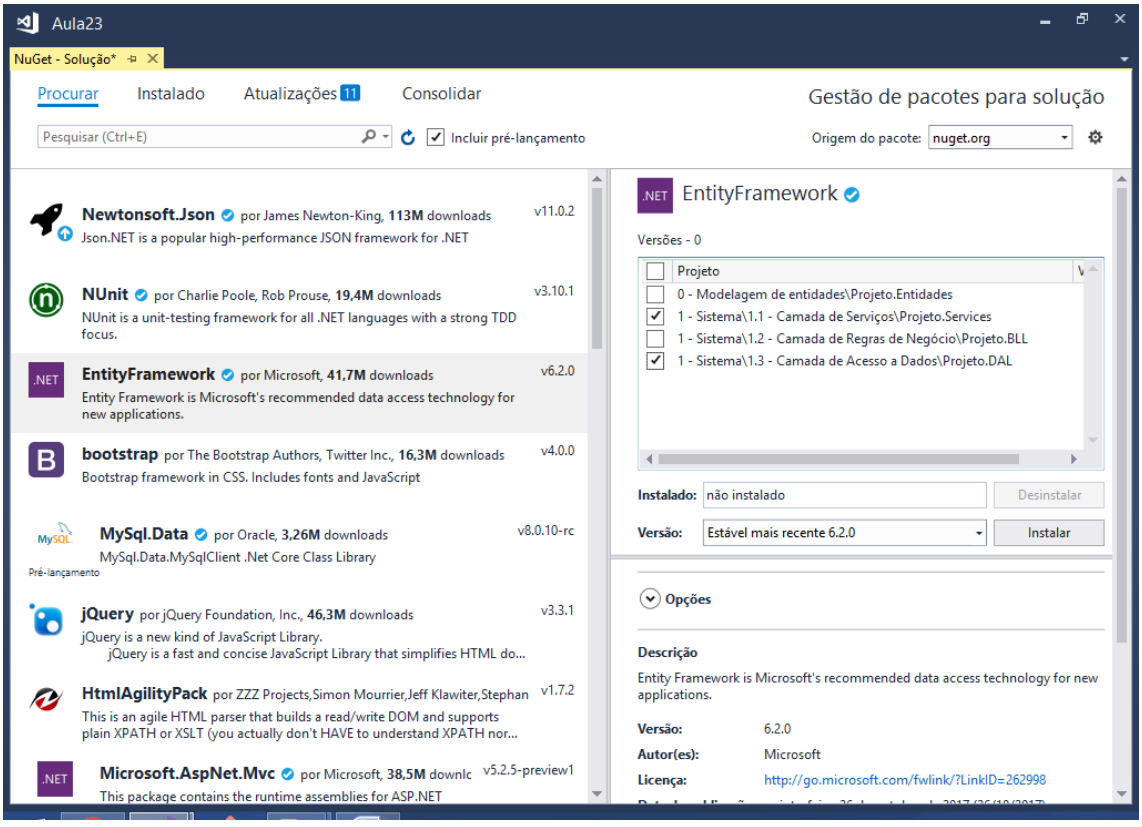




Instalando o EntityFramework

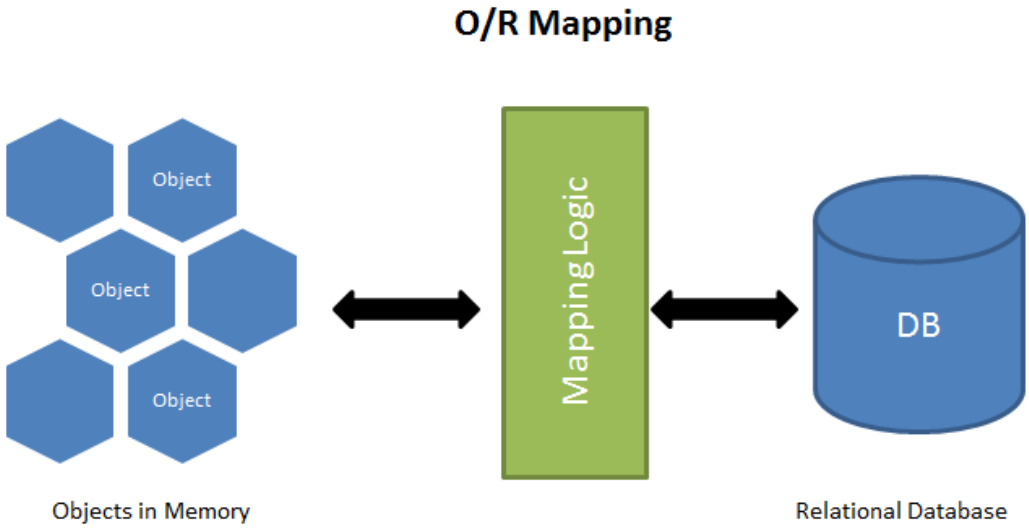
Gerenciador de pacotes do Nuget



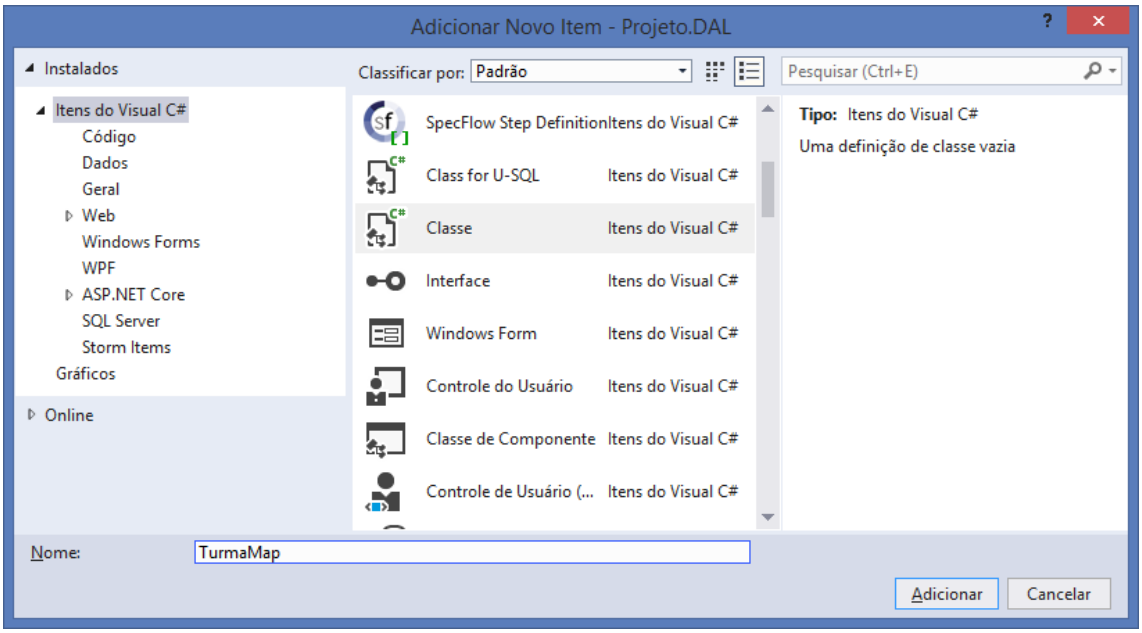


ORM - Mapeamento Objeto Relacional

Mapear as classes de entidade para o banco de dados

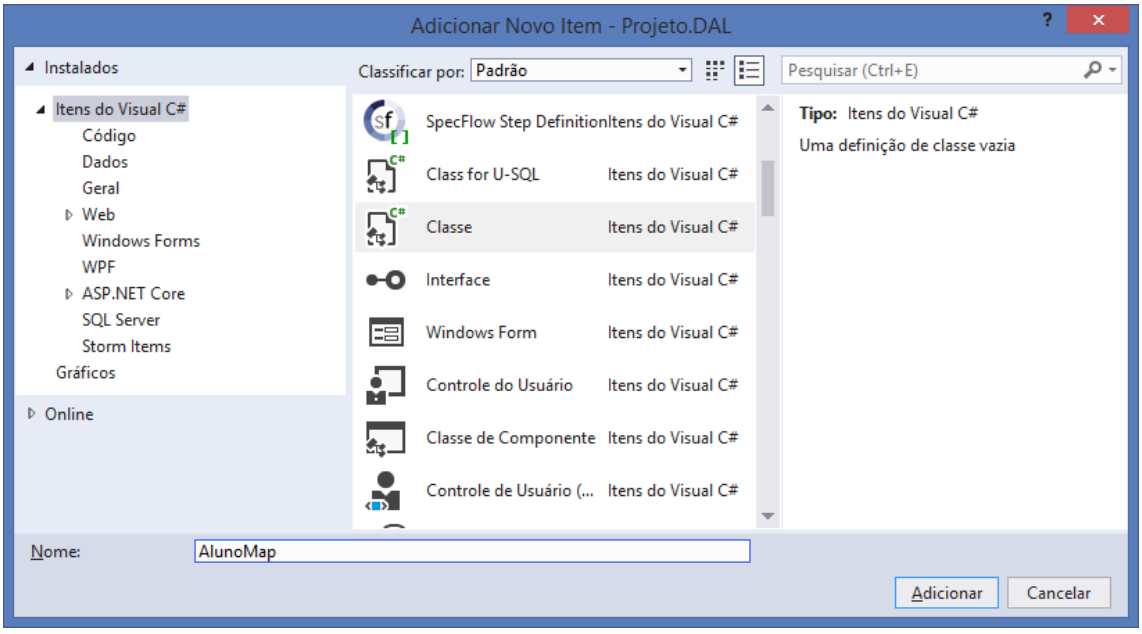


Mapeamento da entidade Turma:




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entidades; //importando..
using System.Data.Entity.ModelConfiguration; //mapeamento..

namespace Projeto.DAL.Mappings
{
```

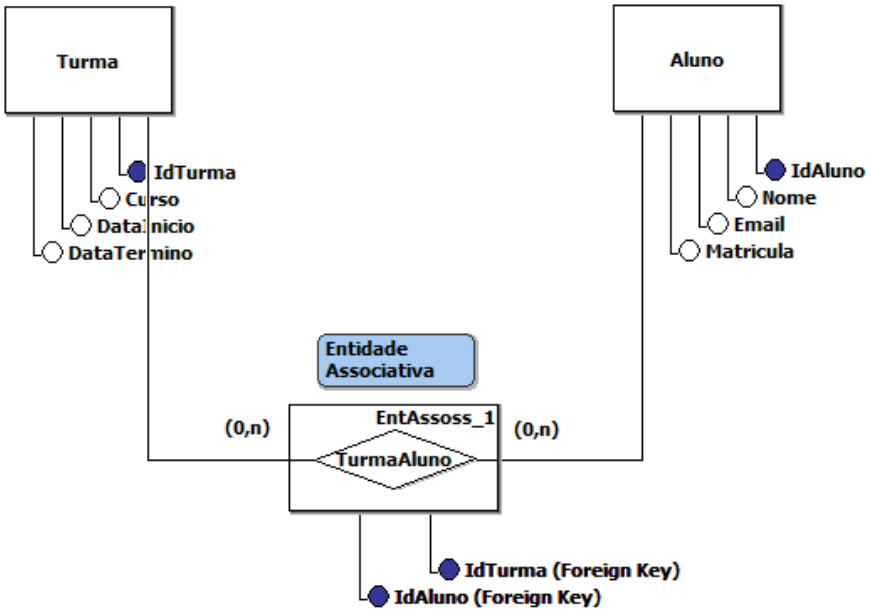
```
    //classe de mapeamento para a entidade 'Turma'
    public class TurmaMap : EntityTypeConfiguration<Turma>
    {
        //construtor..
        public TurmaMap()
        {
            //nome da tabela..
            ToTable("Turma");

            //chave primária..
            HasKey(t => t.IdTurma);

            //demais campos da tabela..
            Property(t => t.IdTurma)
                .HasColumnName("IdTurma");

            Property(t => t.Curso)
                .HasColumnName("Curso")
                .HasMaxLength(50)
                .IsRequired();

            Property(t => t.DataInicio)
                .HasColumnName("DataInicio")
                .IsRequired();

            Property(t => t.DataTermino)
                .HasColumnName("DataTermino")
                .IsRequired();
        }
    }
}
```




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entidades; //classes de entidade..
```

```
using System.Data.Entity.ModelConfiguration; //mapeamentos..

namespace Projeto.DAL.Mappings
{
    public class AlunoMap : EntityTypeConfiguration<Aluno>
    {
        public AlunoMap()
        {
            //nome da tabela..
            ToTable("Aluno");

            //chave primária..
            HasKey(a => a.IdAluno);

            //demais campos..
            Property(a => a.IdAluno)
                .HasColumnName("IdAluno");

            Property(a => a.Nome)
                .HasColumnName("Nome")
                .HasMaxLength(50)
                .IsRequired();

            Property(a => a.Email)
                .HasColumnName("Email")
                .HasMaxLength(50)
                .IsRequired();

            Property(a => a.Matricula)
                .HasColumnName("Matricula")
                .HasMaxLength(20)
                .IsRequired();
        }
    }
}
```

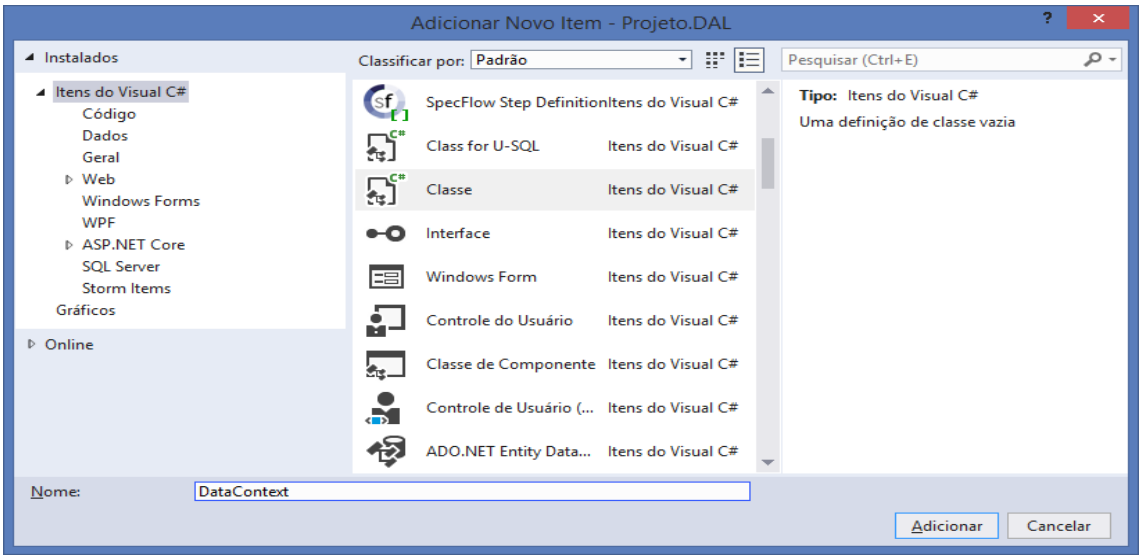
Mapeamento de relacionamento muitos para muitos:



Mapeando o relacionamento muitos para muitos
Fazendo o mapeamento da entidade associativa

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entidades; //importando..
using System.Data.Entity.ModelConfiguration; //mapeamento..

namespace Projeto.DAL.Mappings
{
    //classe de mapeamento para a entidade 'Turma'
    public class TurmaMap : EntityTypeConfiguration<Turma>
    {
        //construtor..
        public TurmaMap()
        {
            //nome da tabela..
            ToTable("Turma");

            //chave primária..
            HasKey(t => t.IdTurma);

            //demais campos da tabela..
            Property(t => t.IdTurma)
                .HasColumnName("IdTurma");

            Property(t => t.Curso)
                .HasColumnName("Curso")
                .HasMaxLength(50)
                .IsRequired();

            Property(t => t.DataInicio)
                .HasColumnName("DataInicio")
                .IsRequired();

            Property(t => t.DataTermino)
                .HasColumnName("DataTermino")
                .IsRequired();

            //mapeamento do relacionamento NpN
            //e da tabela associativa..
            HasMany(t => t.Alunos) //Turma TEM MUITOS Alunos
                .WithMany(a => a.Turmas) //Aluno TEM MUITAS Turmas
                .Map( //Mapeando a tabela associativa
                    m => {
                        m.ToTable("TurmaAluno"); //nome da tabela associativa
                        m.MapLeftKey("IdTurma"); //FK com a entidade Turma
                        m.MapRightKey("IdAluno"); //FK com a entidade Aluno
                    }
                );
        }

    }

}
```

Classe de conexão do EntityFramework com o banco de dados:




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity; //entity framework..
using System.Configuration; //connectionstring..
using Projeto.Entidades; //classes de entidade
using Projeto.DAL.Mappings; //classes de mapeamento..

namespace Projeto.DAL.Context
{
    //Regra 1) Herdar a Classe DbContext
    public class DataContext : DbContext
    {
        //Regra 2) Construtor que envie para DbContext a connectionstring..
        public DataContext()
            : base(ConfigurationManager.ConnectionStrings
                    ["aula"].ConnectionString)
        {
            //envia para o construtor da classe DbContext (base)
            //o endereço da connectionstring para que a conexão seja aberta..
        }

        //Regra 3) Sobrescrever o método OnModelCreating..
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            //adicionar cada classe de mapeamento..
            modelBuilder.Configurations.Add(new TurmaMap());
            modelBuilder.Configurations.Add(new AlunoMap());
        }

        //Regra 4) Declarar uma propriedade DbSet para cada entidade..
        public DbSet<Turma> Turma { get; set; }
        public DbSet<Aluno> Aluno { get; set; }
    }
}
```

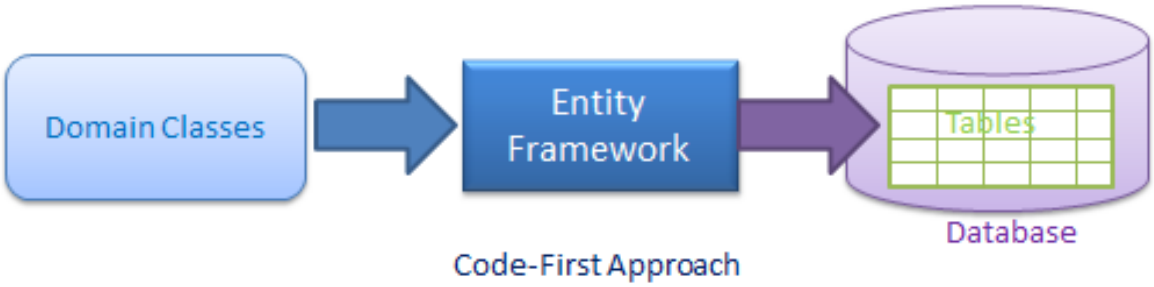
\Web.config.xml

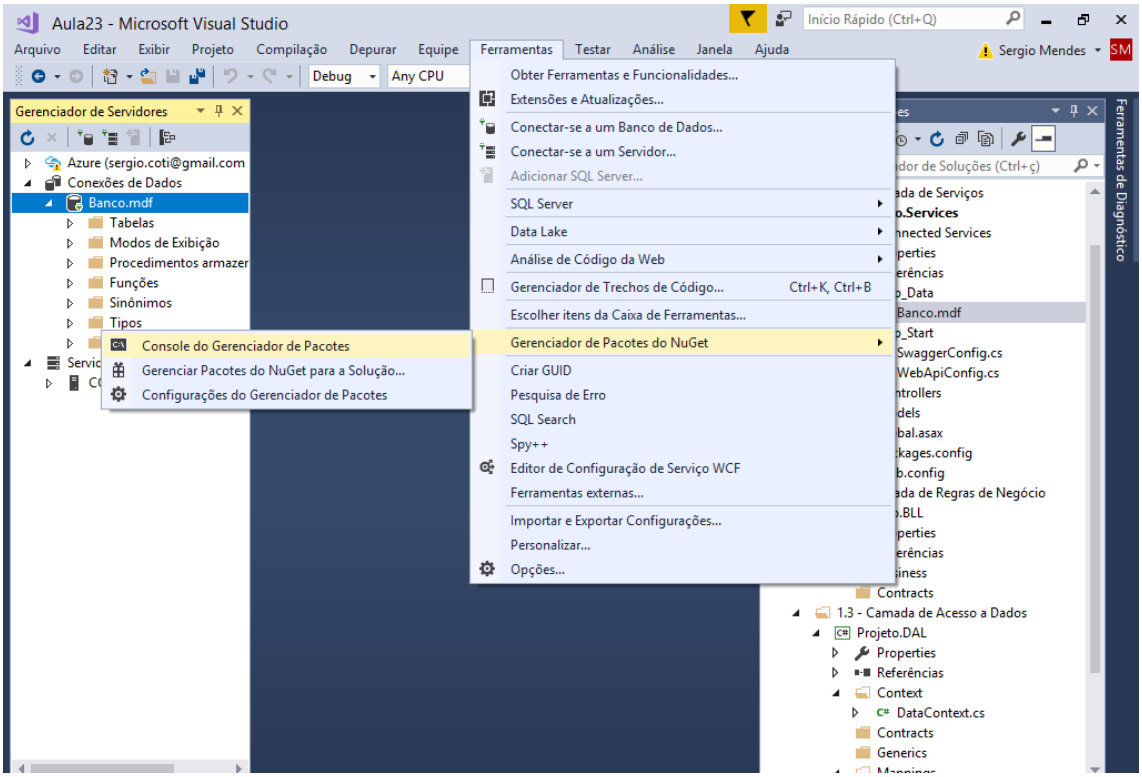
Mapeamento da connectionstring

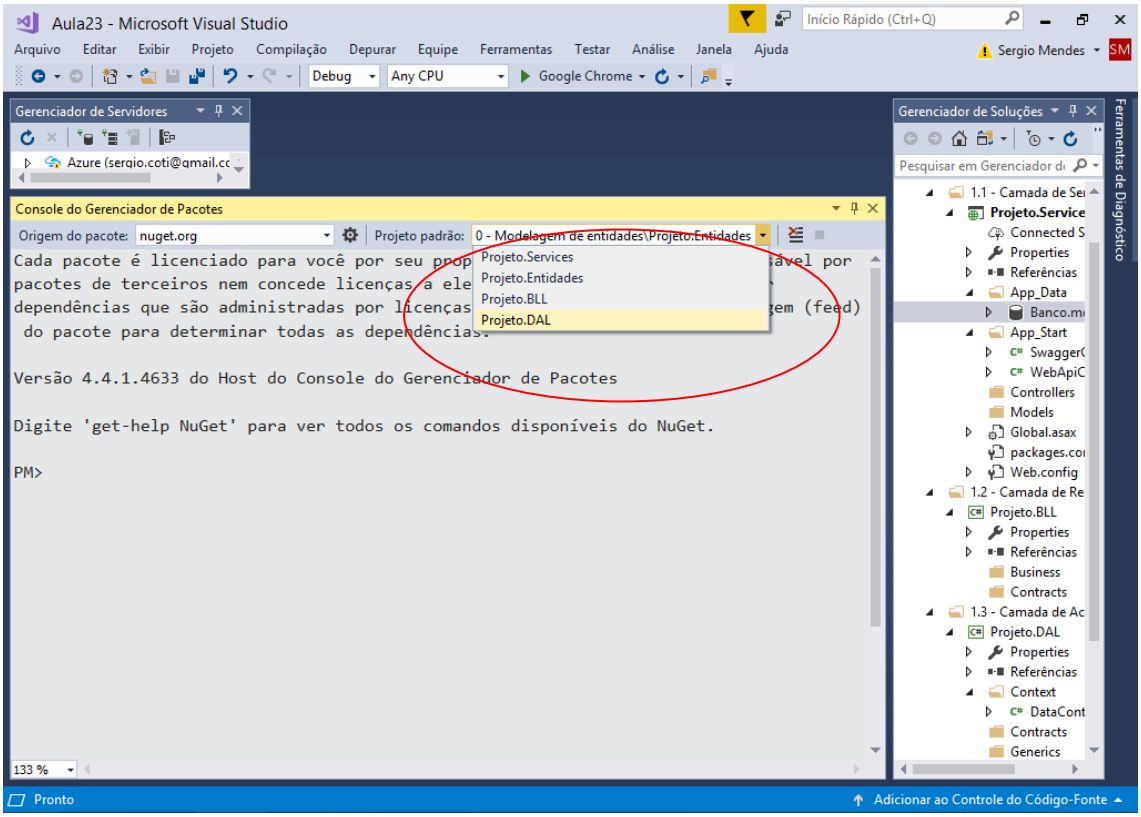
```
<!-- Mapeamento da connectionstring -->
<connectionStrings>
  <add
      name="aula"
      connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
    AttachDbFilename=C:\Users\COTI\Desktop\__Aula_03.04.18\
    Aula23\Projeto.Services\App_Data\Banco.mdf;Integrated Security=True"
    />
</connectionStrings>
```

Migrations (CodeFirst)

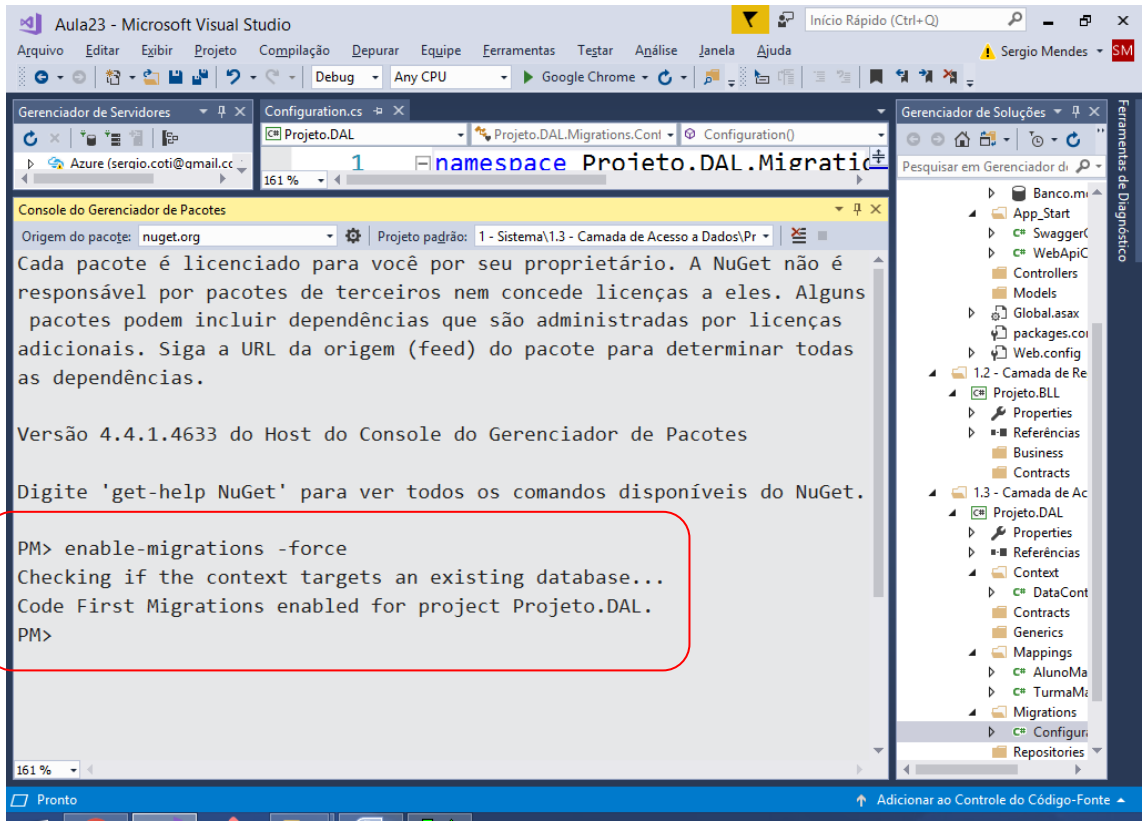
Gerando o conteudo do banco de dados (tabelas) baseado no mapeamento das classes de entidade feito pelo EntityFramework.







PM> enable-migrations -force



Classe gerada:

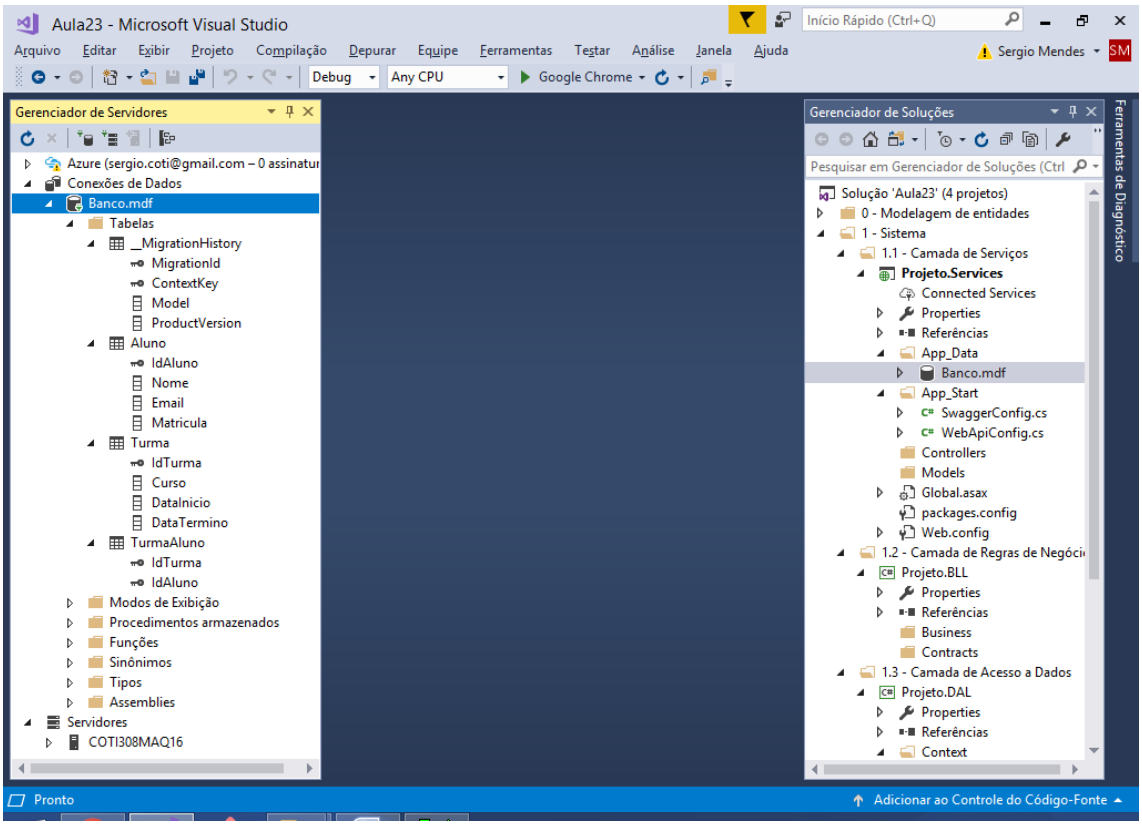
```
namespace Projeto.DAL.Migrations
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;

    internal sealed class Configuration : DbMigrationsConfiguration
              <Projeto.DAL.Context.DataContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
        }

        protected override void Seed(Projeto.DAL.Context.DataContext context)
        {
            //  This method will be called after migrating to the latest version.

            //  You can use the DbSet<T>.AddOrUpdate() helper extension method
            //  to avoid creating duplicate seed data.
        }
    }
}
```

Gerando as tabelas no banco de dados:

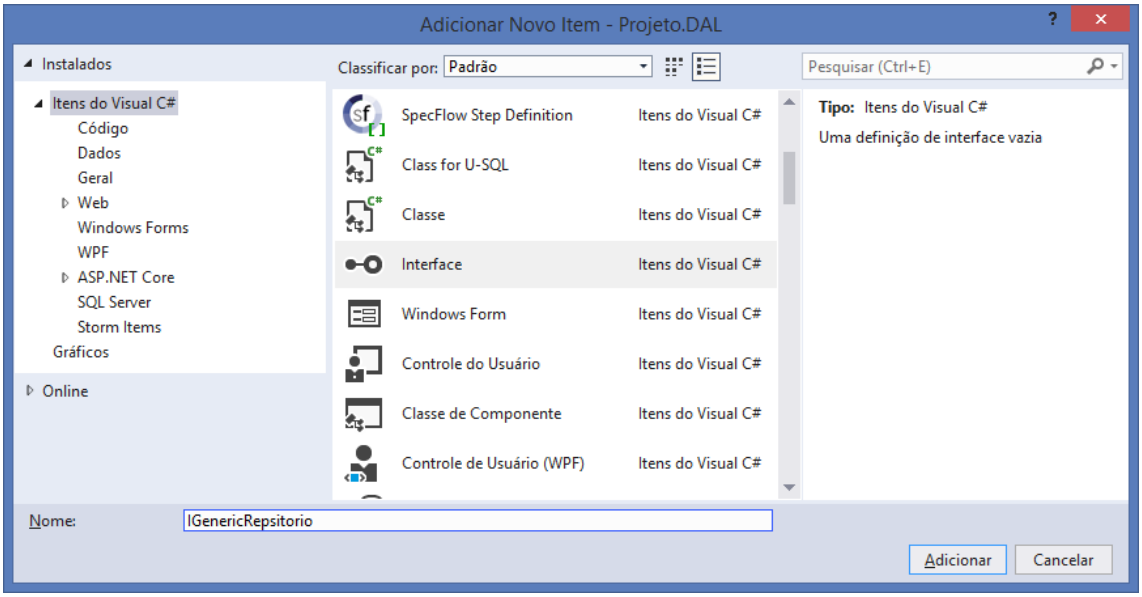
```
PM> update-database -verbose
```

```
CREATE TABLE [dbo].[Aluno] (
    [IdAluno] [int] NOT NULL IDENTITY,
    [Nome] [nvarchar](50) NOT NULL,
    [Email] [nvarchar](50) NOT NULL,
    [Matricula] [nvarchar](20) NOT NULL,
    CONSTRAINT [PK_dbo.Aluno] PRIMARY KEY ([IdAluno])
)

CREATE TABLE [dbo].[Turma] (
    [IdTurma] [int] NOT NULL IDENTITY,
    [Curso] [nvarchar](50) NOT NULL,
    [DataInicio] [datetime] NOT NULL,
    [DataTermino] [datetime] NOT NULL,
    CONSTRAINT [PK_dbo.Turma] PRIMARY KEY ([IdTurma])
)

CREATE TABLE [dbo].[TurmaAluno] (
    [IdTurma] [int] NOT NULL,
    [IdAluno] [int] NOT NULL,
    CONSTRAINT [PK_dbo.TurmaAluno] PRIMARY KEY ([IdTurma], [IdAluno])
)
```

No banco de dados:

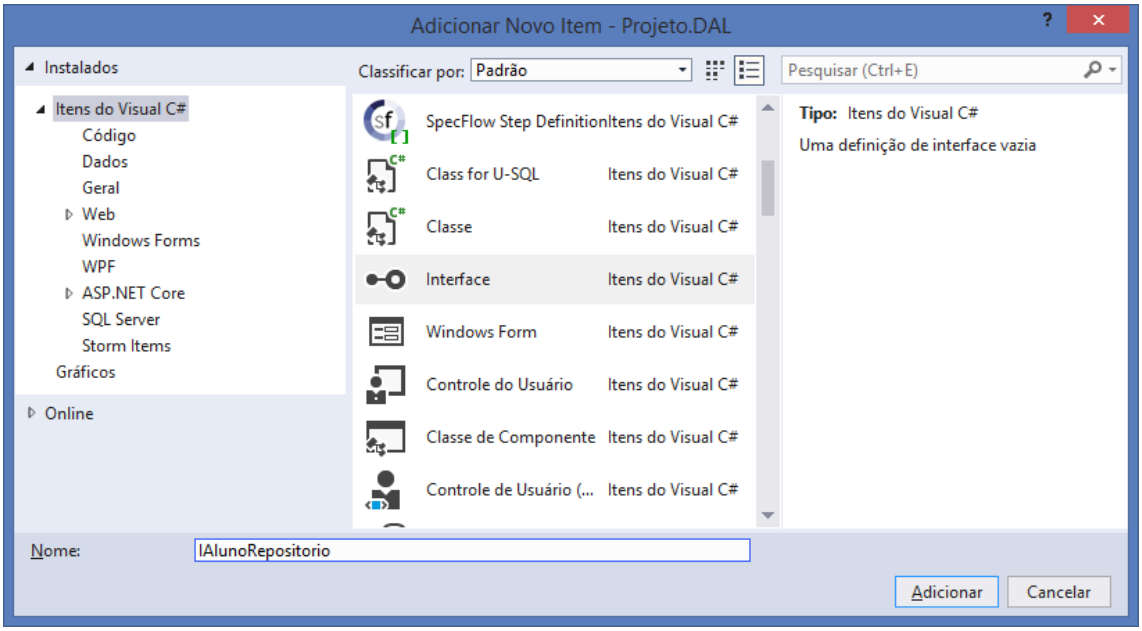


<u>Criando interfaces para cada classe que será programada no repositorio:</u>

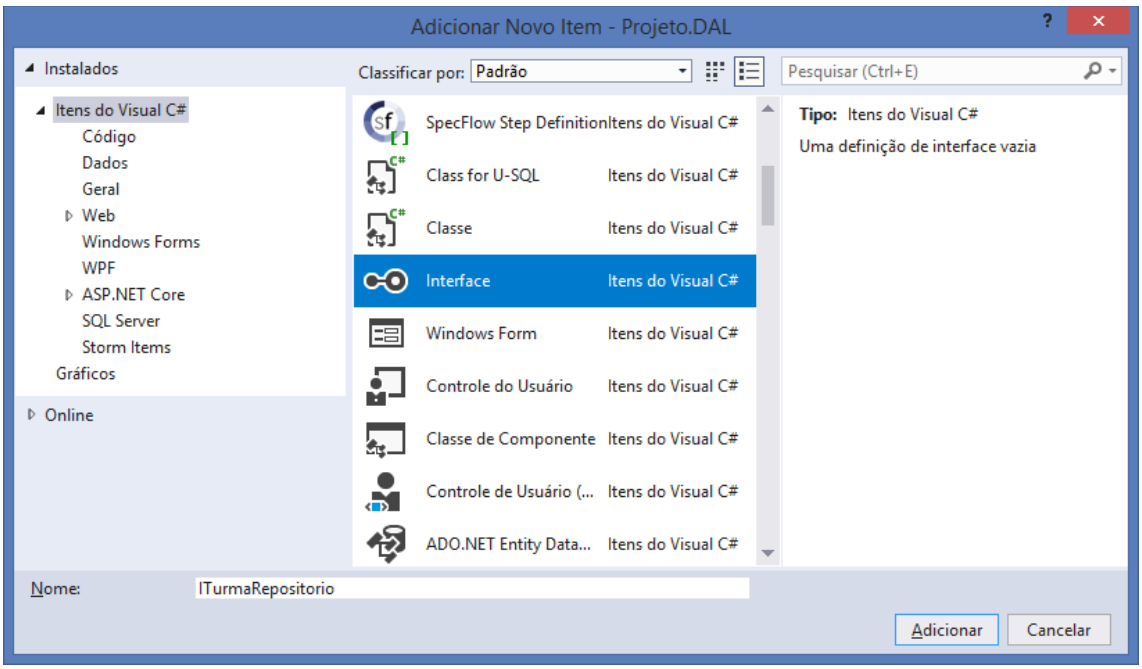



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Projeto.DAL.Contracts
{
    //<T> Tipo Genérico
    public interface IGenericRepositorio<T>
        where T : class
    {
        void Insert(T obj);

        void Update(T obj);

        void Delete(T obj);

        List<T> FindAll();

        T FindById(int id);
    }
}
```

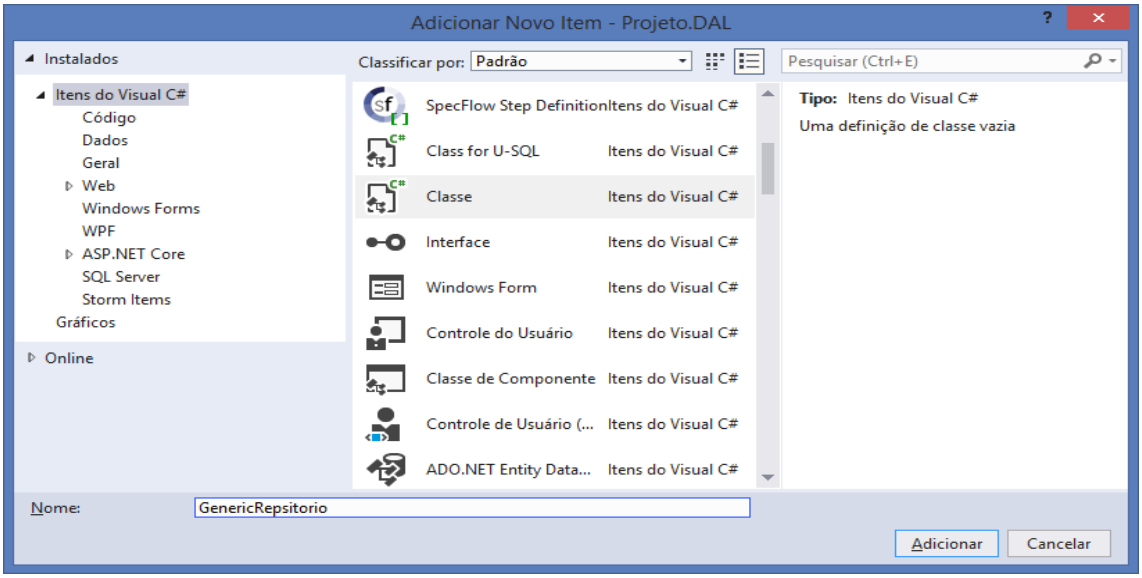



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entidades;

namespace Projeto.DAL.Contracts
{
    public interface IAlunoRepositorio : IGenericRepositorio<Aluno>
    {
        List<Aluno> FindByNome(string nome);
    }
}
```



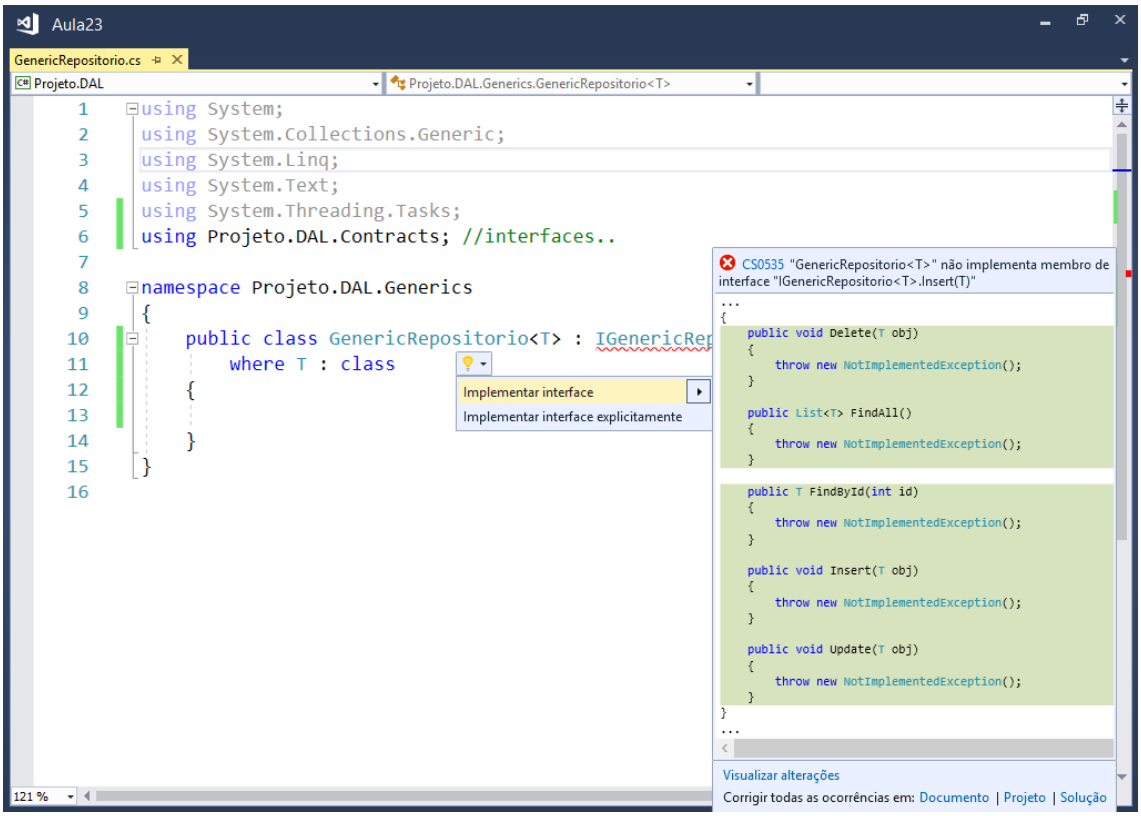

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entidades;

namespace Projeto.DAL.Contracts
{
    public interface ITurmaRepositorio : IGenericRepositorio<Turma>
    {
        List<Turma> FindByDataInicio(DateTime dataDe, DateTime dataAte);
    }
}
```

Implementando as interfaces:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.DAL.Contracts; //interfaces..

namespace Projeto.DAL.Generics
{
    public class GenericRepositorio<T> : IGenericRepositorio<T>
        where T : class
    {

    }
}
```

Implementando os métodos da interface:




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.DAL.Contracts; //interfaces..

namespace Projeto.DAL.Generics
{
    public class GenericRepositorio<T> : IGenericRepositorio<T>
        where T : class
    {
```

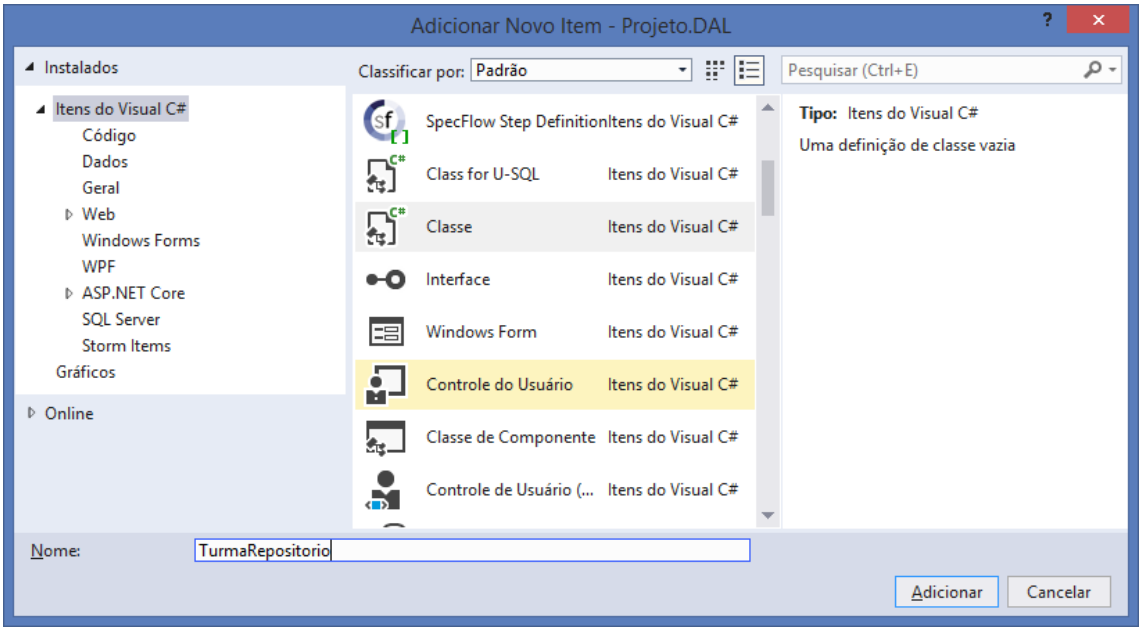
```
        public void Insert(T obj)
        {
            throw new NotImplementedException();
        }

        public void Update(T obj)
        {
            throw new NotImplementedException();
        }

        public void Delete(T obj)
        {
            throw new NotImplementedException();
        }

        public List<T> FindAll()
        {
            throw new NotImplementedException();
        }

        public T FindById(int id)
        {
            throw new NotImplementedException();
        }
    }
}
```

Programando os métodos acima:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.DAL.Contracts; //interfaces..
using Projeto.DAL.Context; //classe DataContext..
using System.Data.Entity; //entityframework..

namespace Projeto.DAL.Generics
{
    public class GenericRepositorio<T> : IGenericRepositorio<T>
        where T : class
    {
        public void Insert(T obj)
        {
            using (DataContext d = new DataContext())
            {
                d.Entry(obj).State = EntityState.Added; //inserindo..
                d.SaveChanges(); //executando..
            }
        }

        public void Update(T obj)
        {
            using (DataContext d = new DataContext())
            {
                d.Entry(obj).State = EntityState.Modified;
                d.SaveChanges();
            }
        }
```

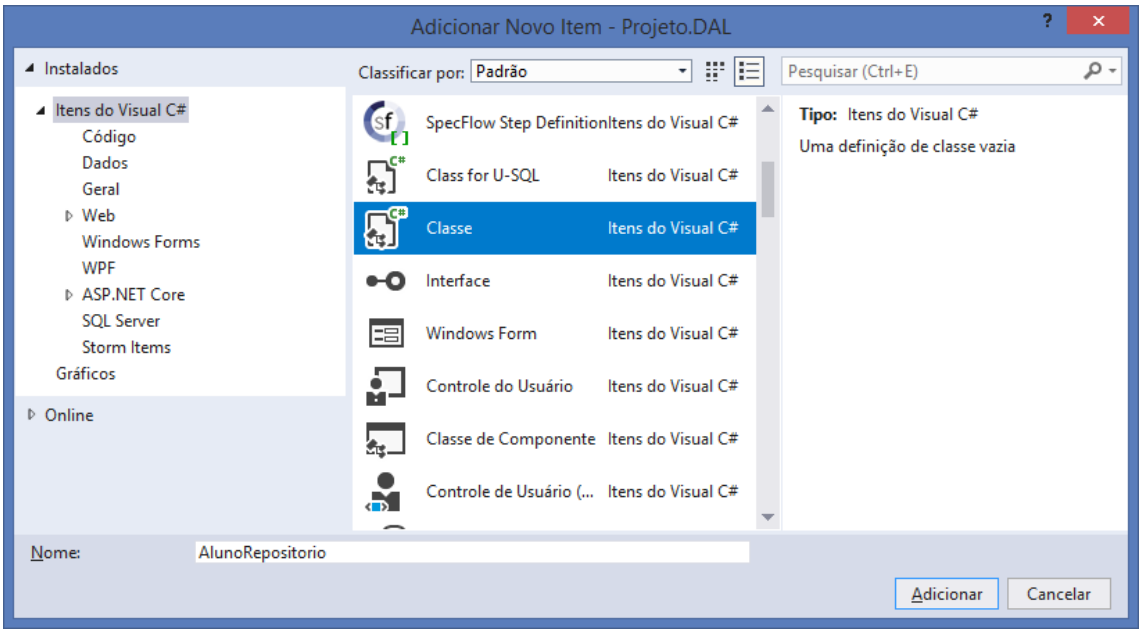
```
        public void Delete(T obj)
        {
            using (DataContext d = new DataContext())
            {
                d.Entry(obj).State = EntityState.Deleted;
                d.SaveChanges();
            }
        }

        public List<T> FindAll()
        {
            using (DataContext d = new DataContext())
            {
                return d.Set<T>().ToList();
            }
        }

        public T FindById(int id)
        {
            using (DataContext d = new DataContext())
            {
                return d.Set<T>().Find(id);
            }
        }
    }
}
```

Implementando as demais interfaces:




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entidades;
using Projeto.DAL.Contracts;
using Projeto.DAL.Context;
```

```
using Projeto.DAL.Generics;

namespace Projeto.DAL.Repositories
{
    public class TurmaRepositorio : GenericRepositorio<Turma>, ITurmaRepositorio
    {
        public List<Turma> FindByDataInicio(DateTime dataDe, DateTime dataAte)
        {
            using (DataContext d = new DataContext())
            {
                return d.Turma
                        .Where(t => t.DataInicio >= dataDe
                                 && t.DataInicio <= dataAte)
                        .OrderBy(t => t.DataInicio)
                        .ToList();
            }
        }
    }
}
```




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entidades;
using Projeto.DAL.Contracts;
using Projeto.DAL.Context;
using Projeto.DAL.Generics;

namespace Projeto.DAL.Repositories
{
    public class AlunoRepositorio : GenericRepositorio<Aluno>, IAlunoRepositorio
    {
        public List<Aluno> FindByNome(string nome)
        {
            using (DataContext d = new DataContext())
            {
```

```
                return d.Aluno
                        .Where(a => a.Nome.Contains(nome))
                        .OrderBy(a => a.Nome)
                        .ToList();
            }
        }
    }
}
```

Continua...