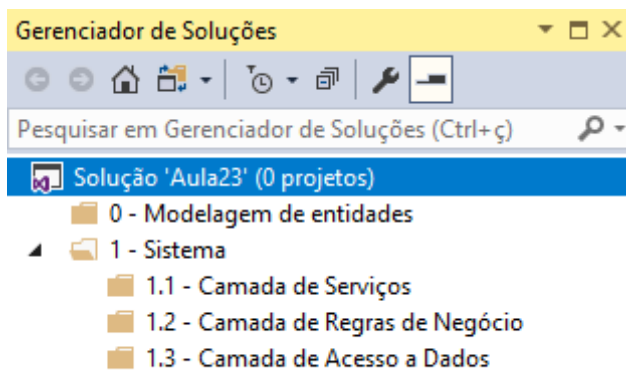
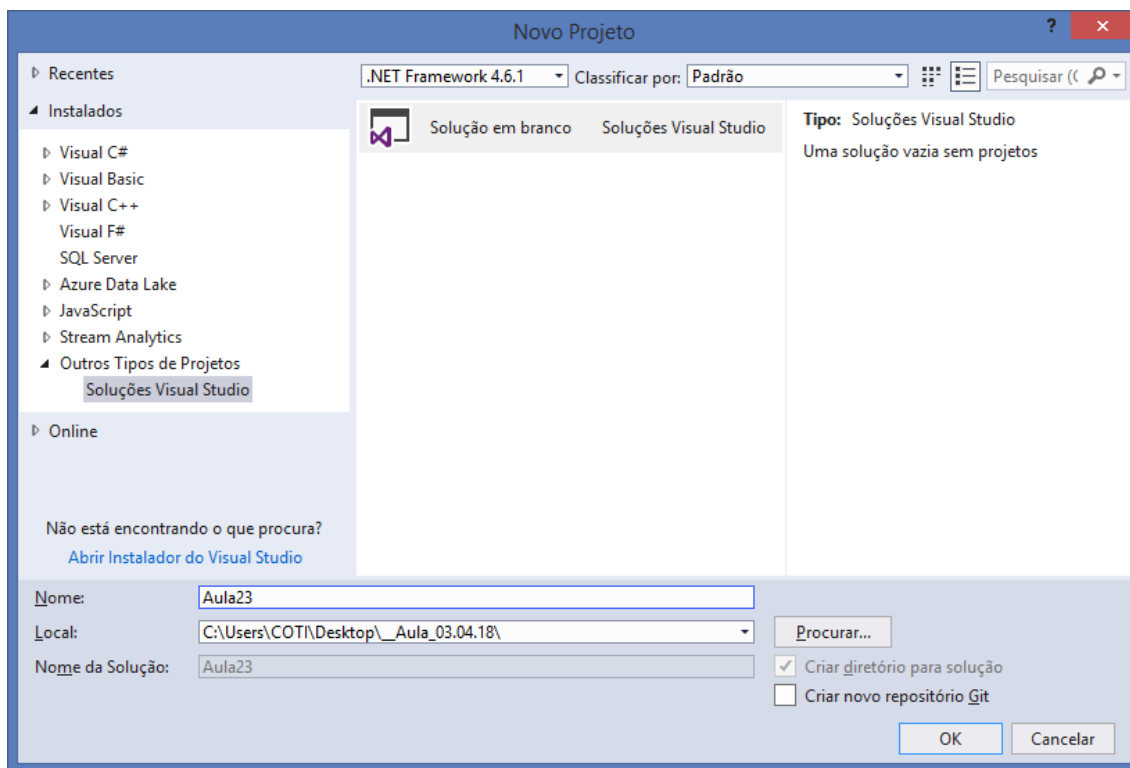
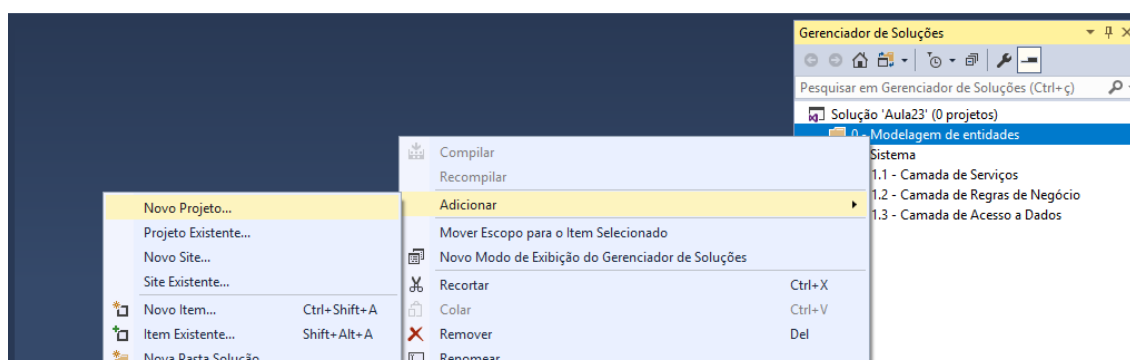
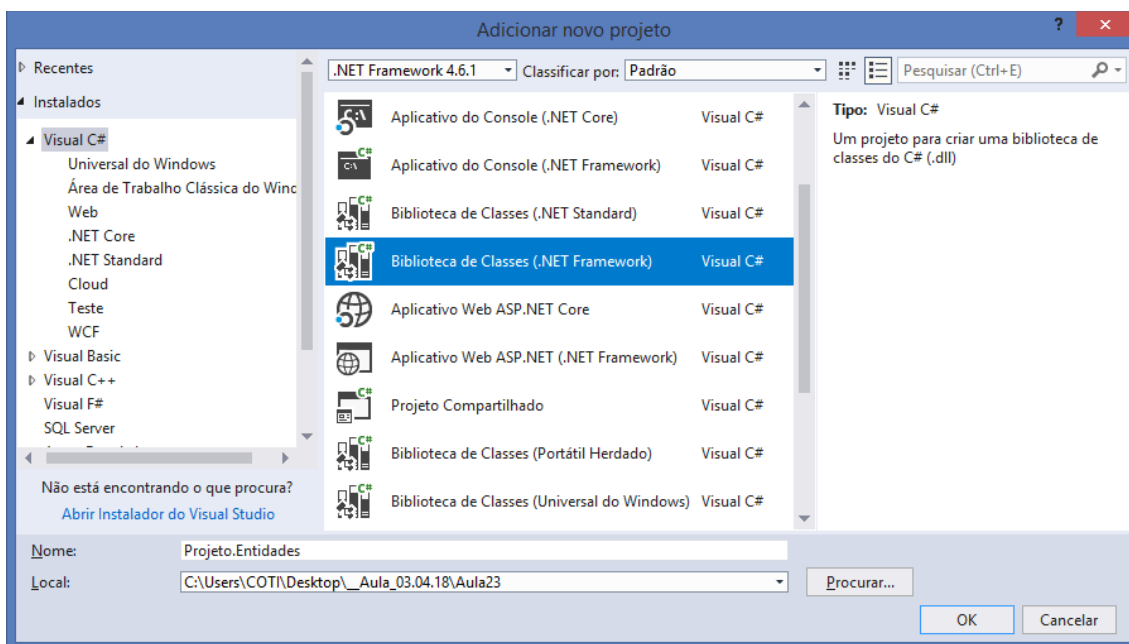


Criando um nova solution em branco:



0 - Modelagem de entidades





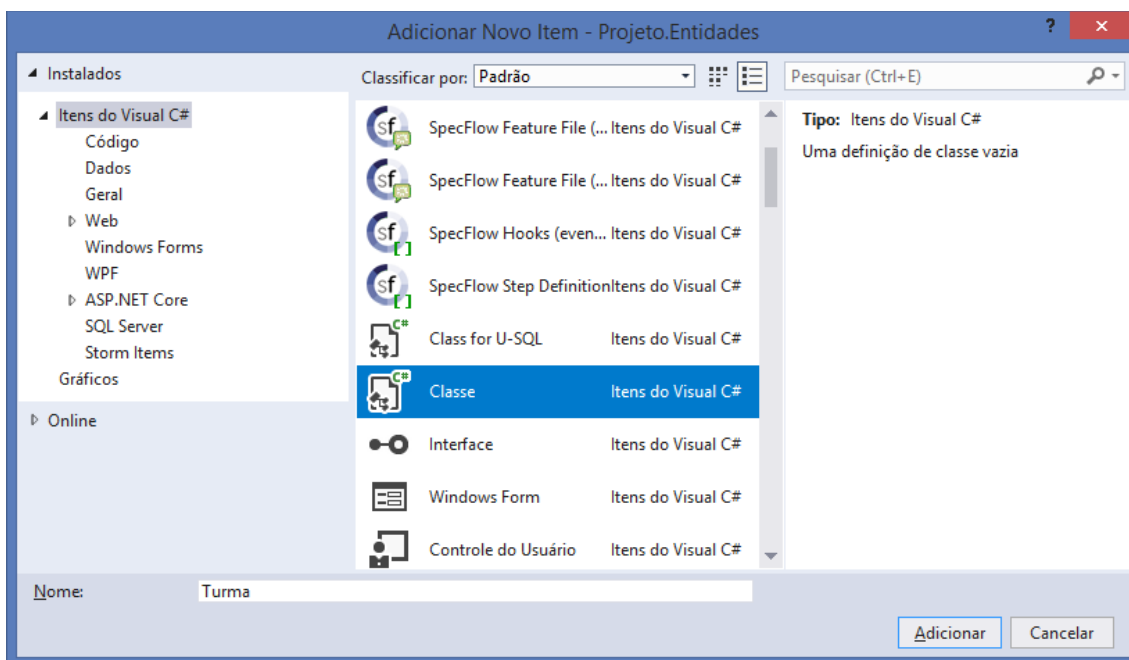
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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; }
    }
}
```

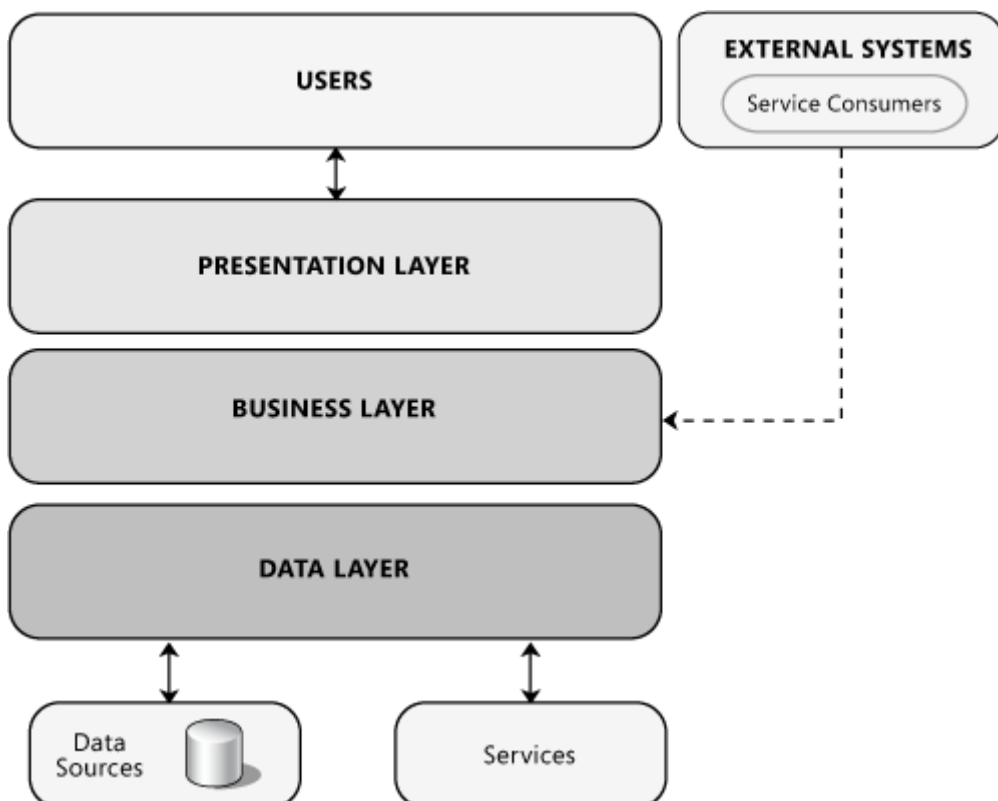
```
public Aluno()
{
    //construtor default..
}

//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}";
}
}
}
```

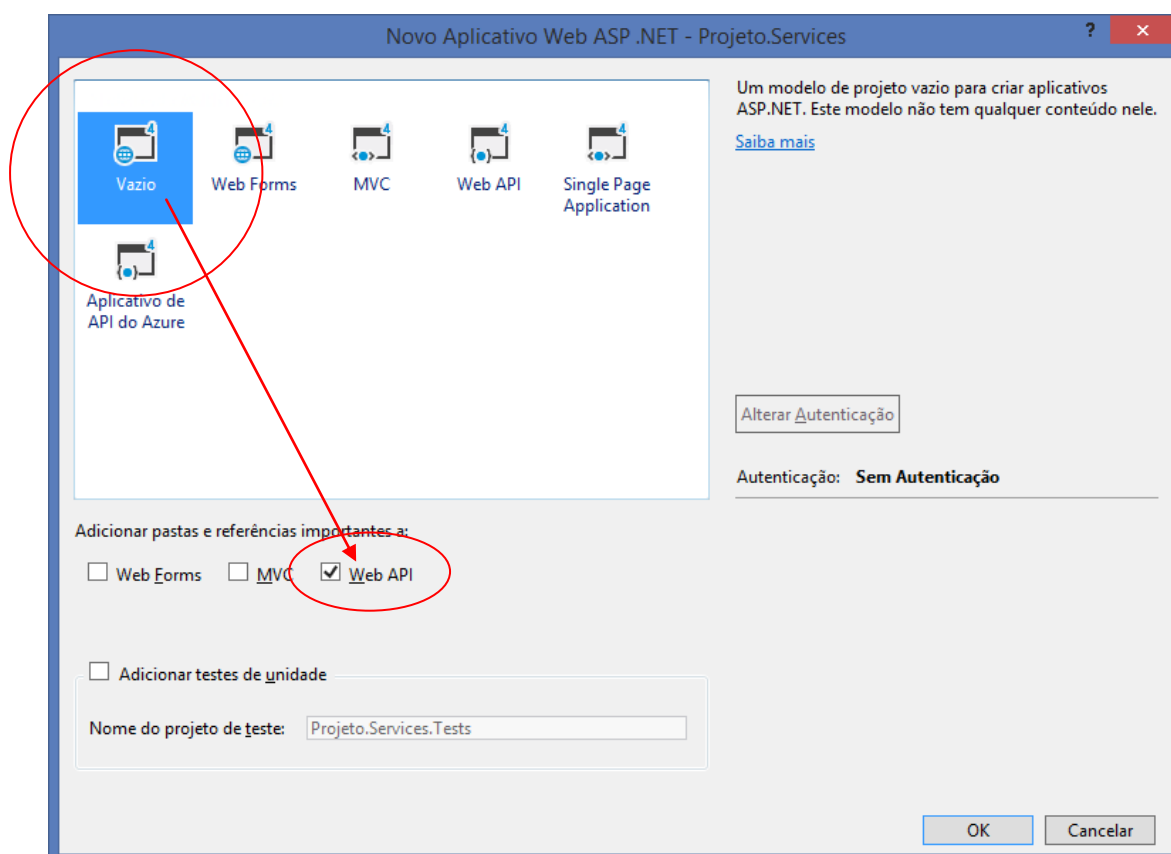
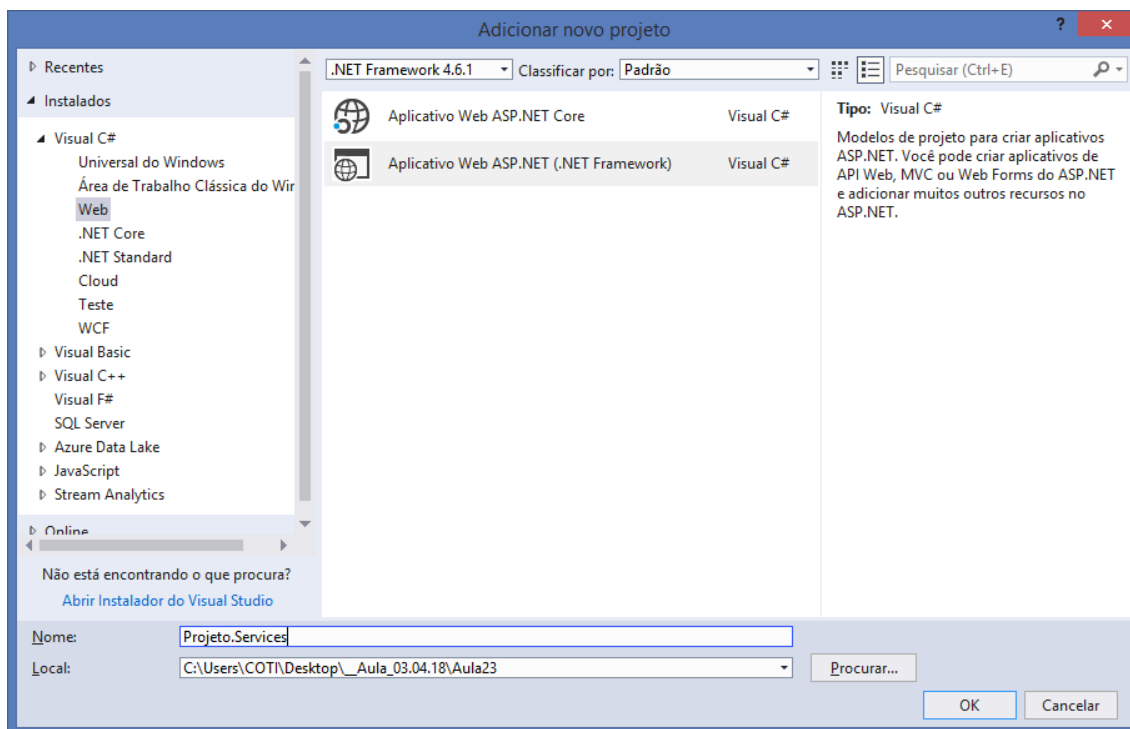
Arquitetura do projeto

Desenvolvimento baseado em camadas

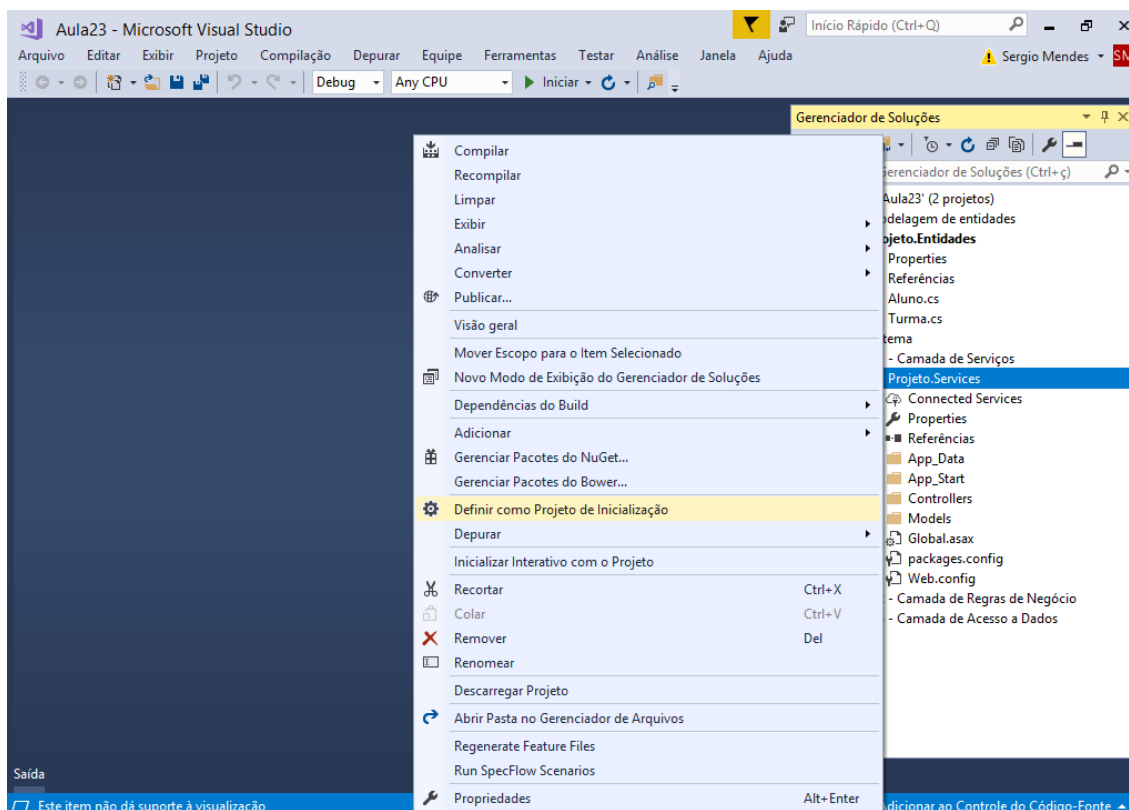


1.1 - Camada de Serviços

Projeto Asp.Net WebApplication (.NET Framework)



Definindo o projeto de inicialização da Solution:

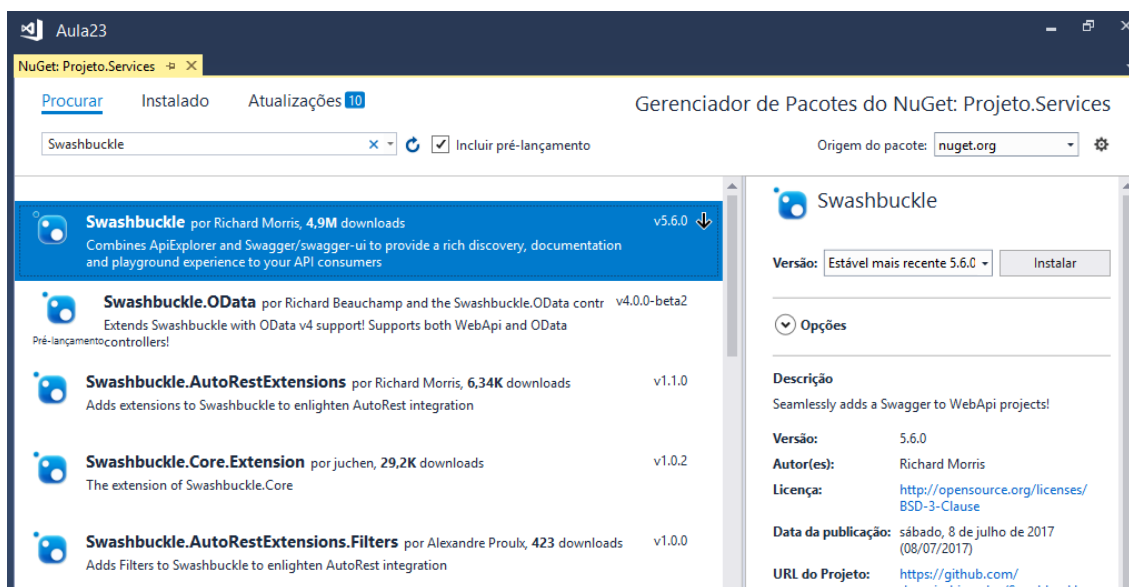


Swagger

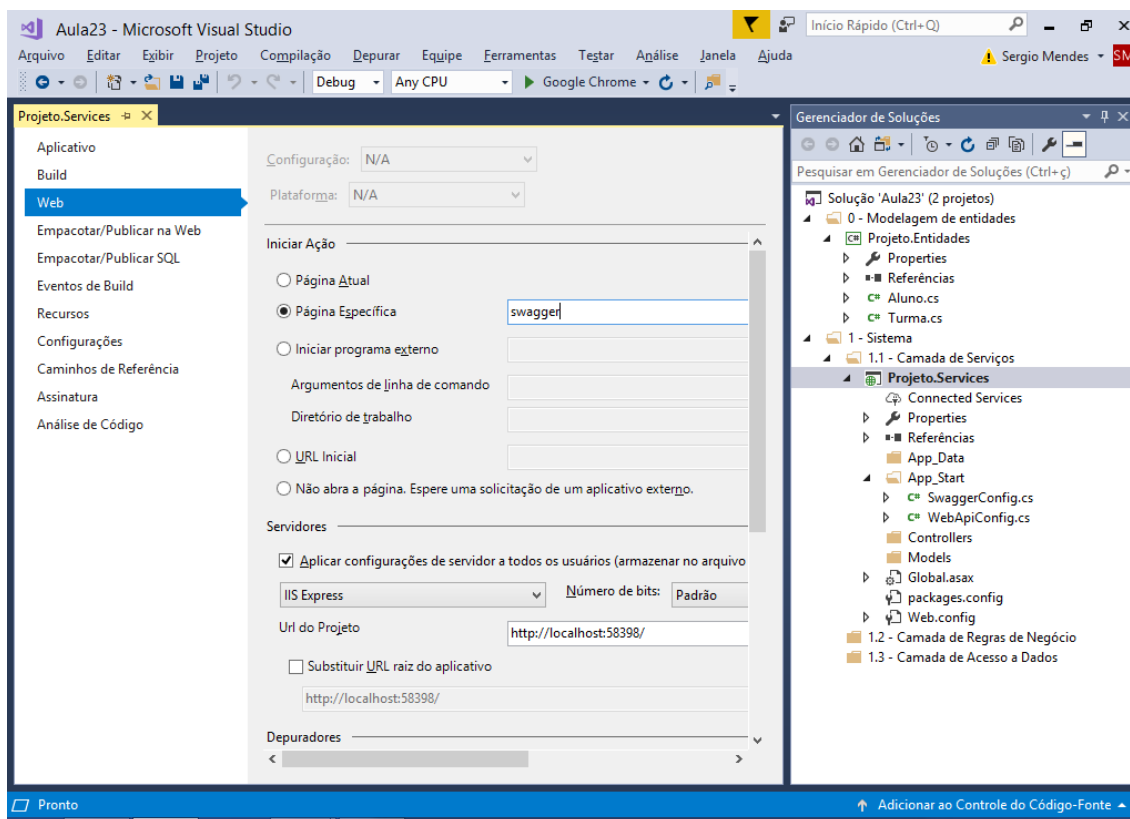
Framework para geração de documentação de serviços em projetos do tipo WebApi.

- Instalando:

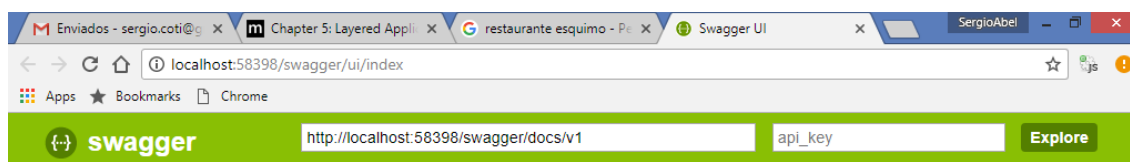
Gerenciar pacotes do Nuget



Configurando a página inicial do projeto:



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

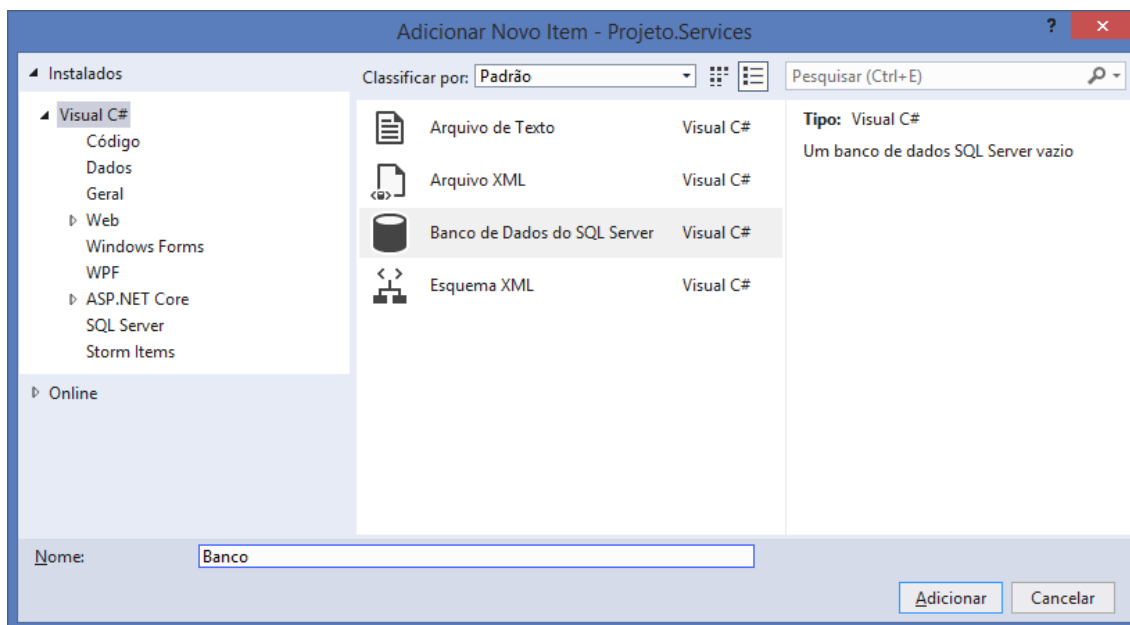


Projeto.Services

[BASE URL: , API VERSION: v1]

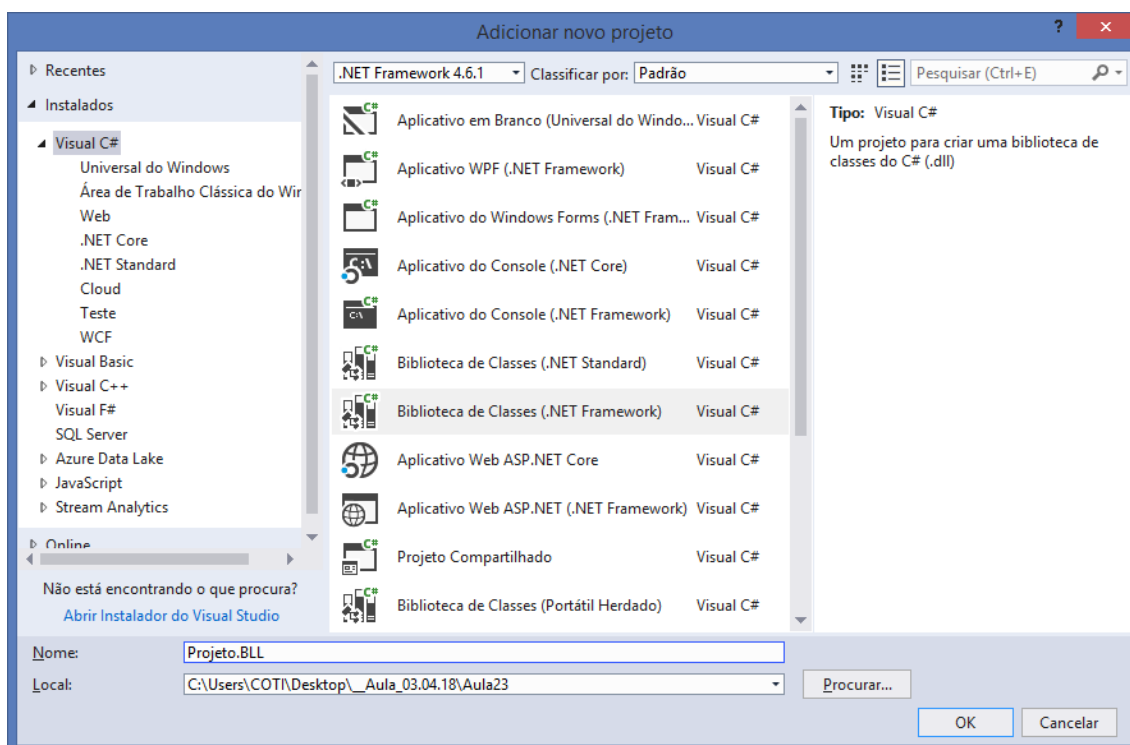
Criando a base de dados:

MDF - Master Database File



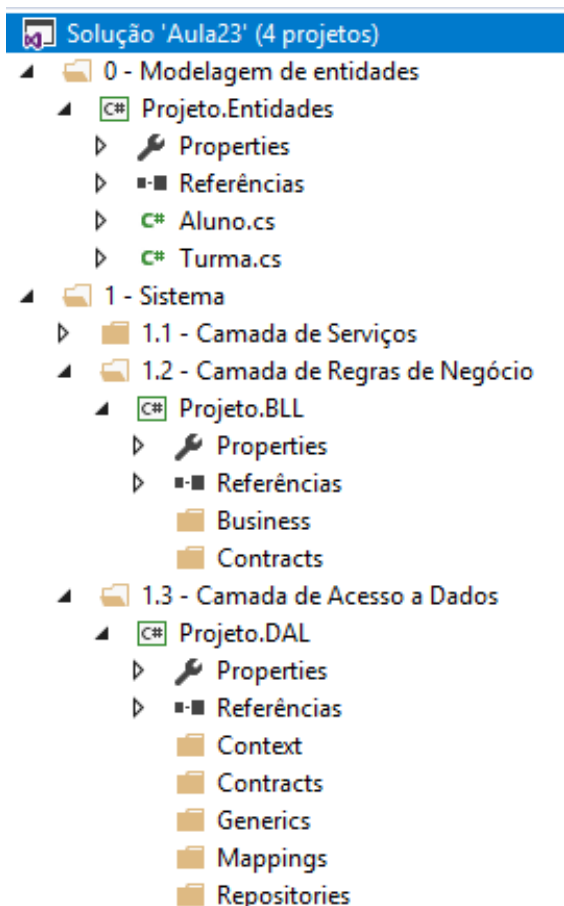
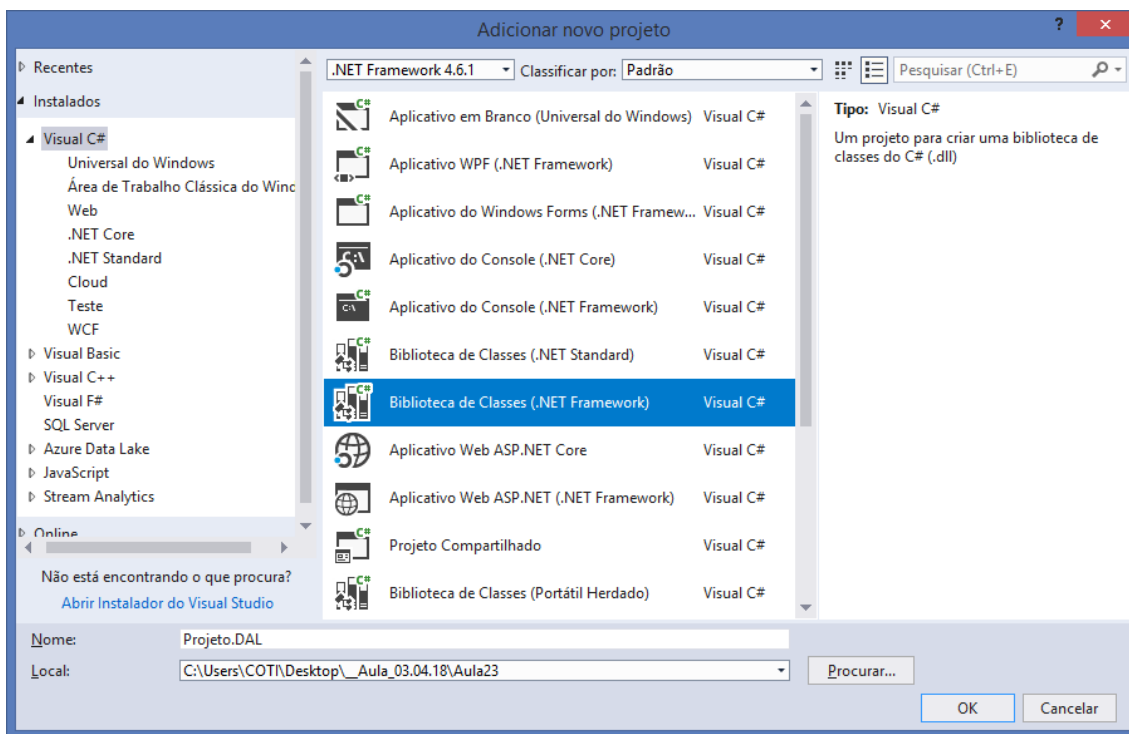
1.2 - Camada de Regras de Negócio

BLL - Business Logic Layer



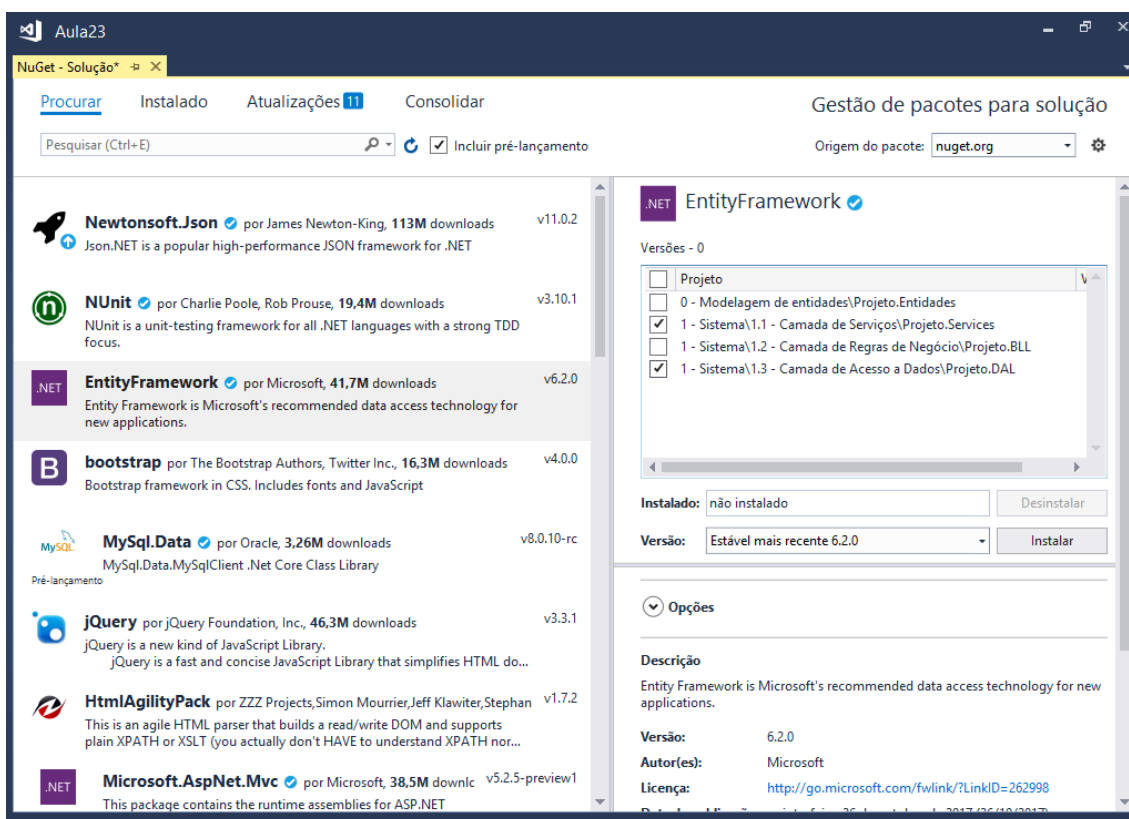
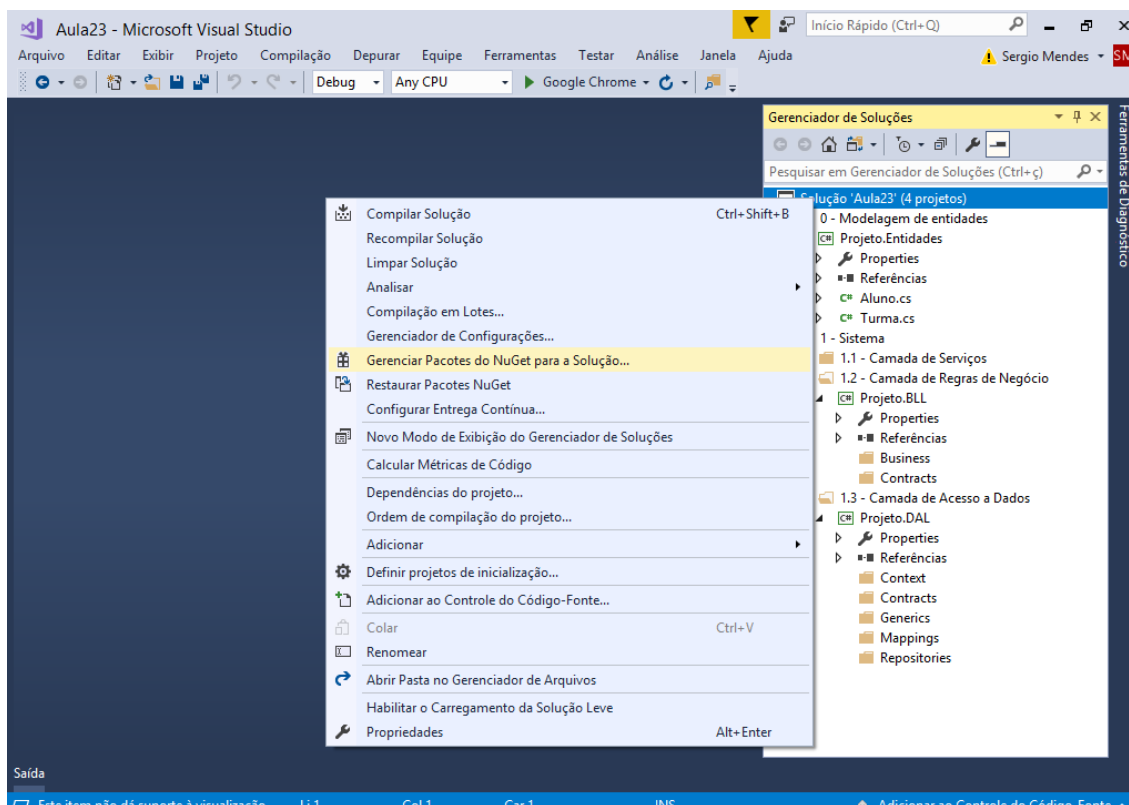
1.3 - Camada de Acesso a dados

DAL - Data Access Layer



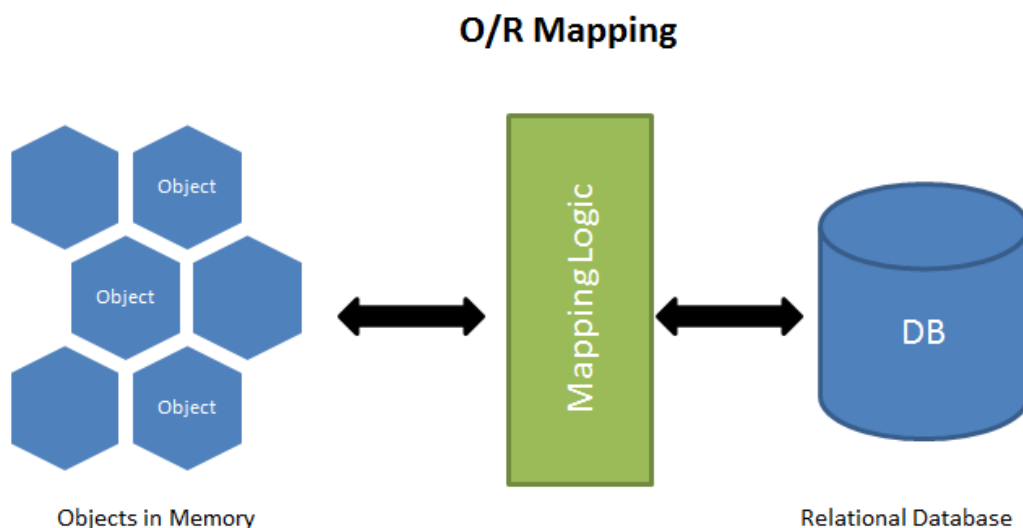
Instalando o EntityFramework

Gerenciador de pacotes do Nuget

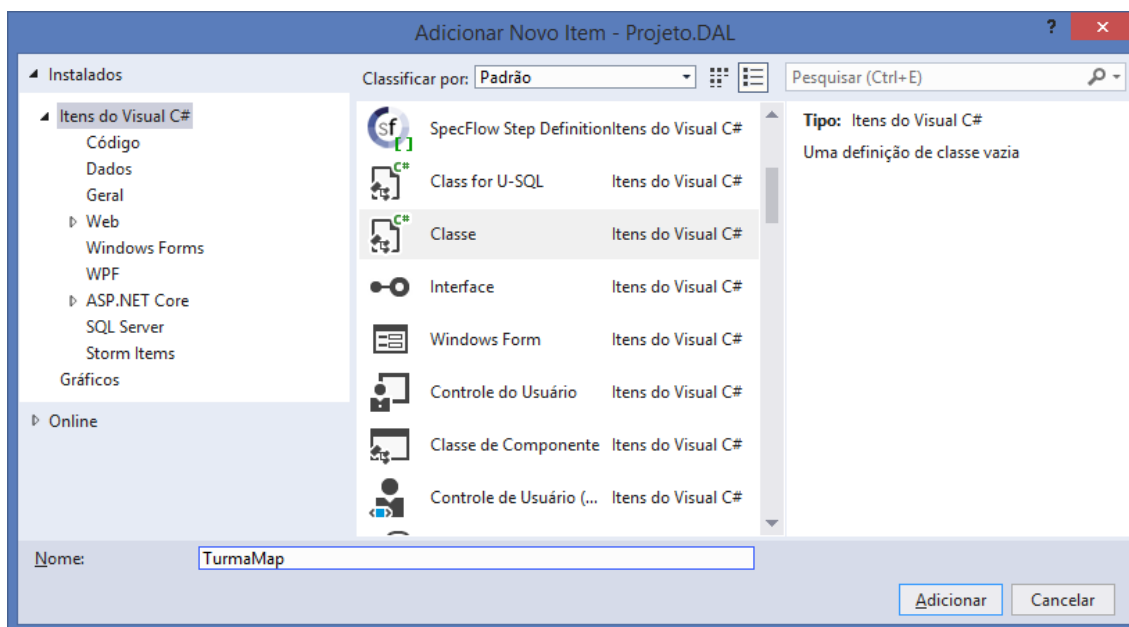


ORM - Mapeamento Objeto Relacional

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");

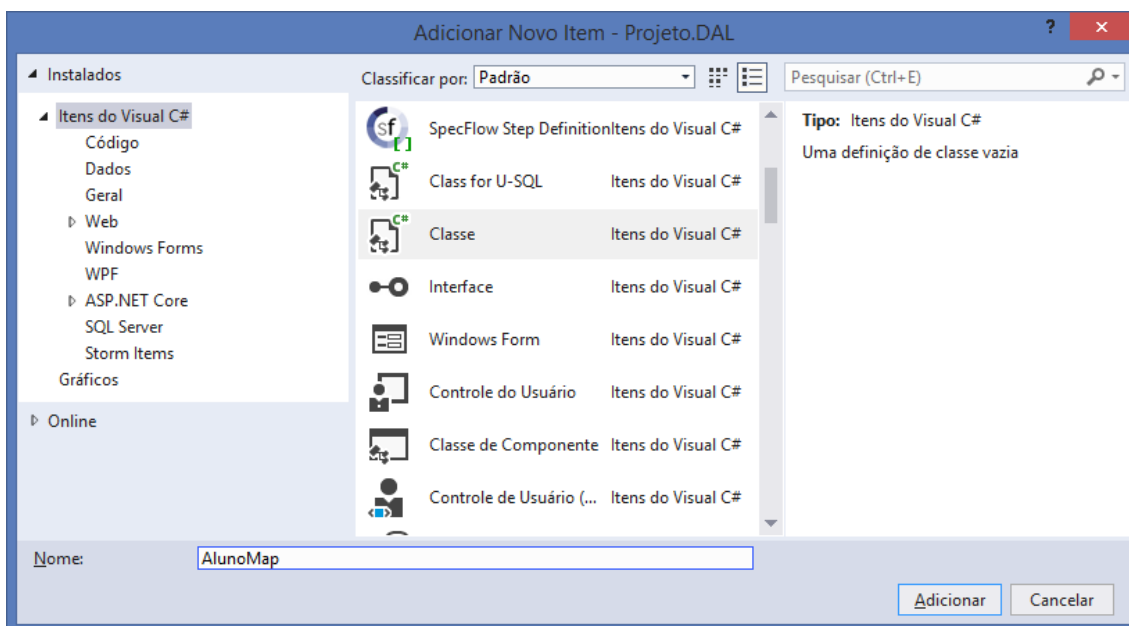
        //chave primária..
        HasKey(t => t.IdTurma);

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

        Property(t => tCurso)
            .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);

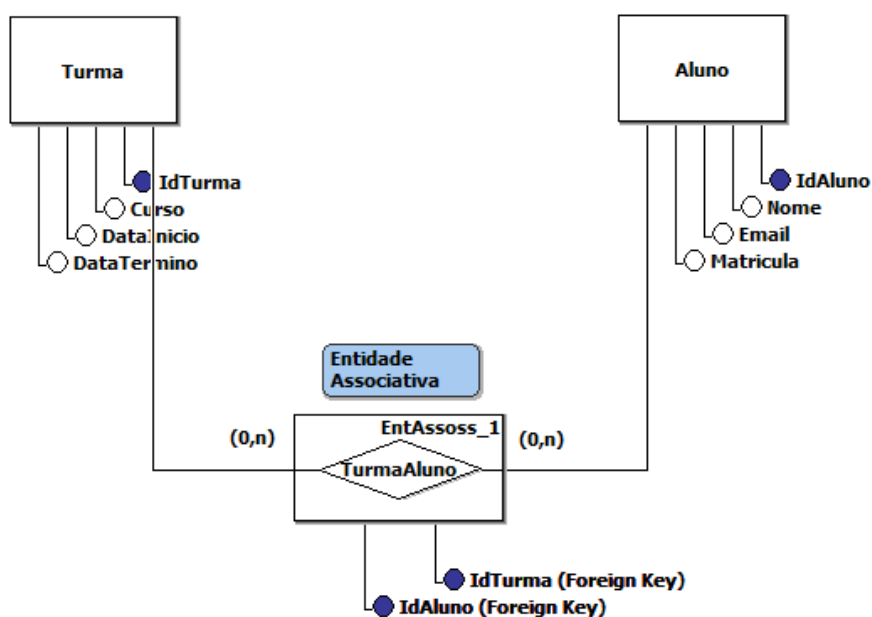
            //demais campos..
            Property(a => a.IdAluno)
                .HasColumnName("IdAluno");

            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");

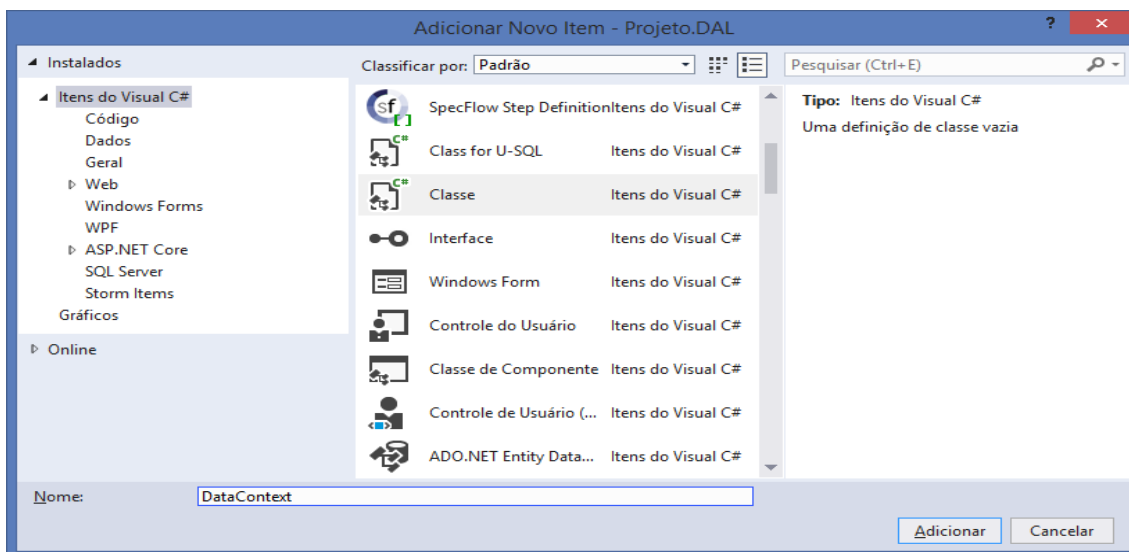
            Property(t => tCurso)
                .HasColumnName("Curso")
                .HasMaxLength(50)
                .IsRequired();

            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; }
    }
}
```

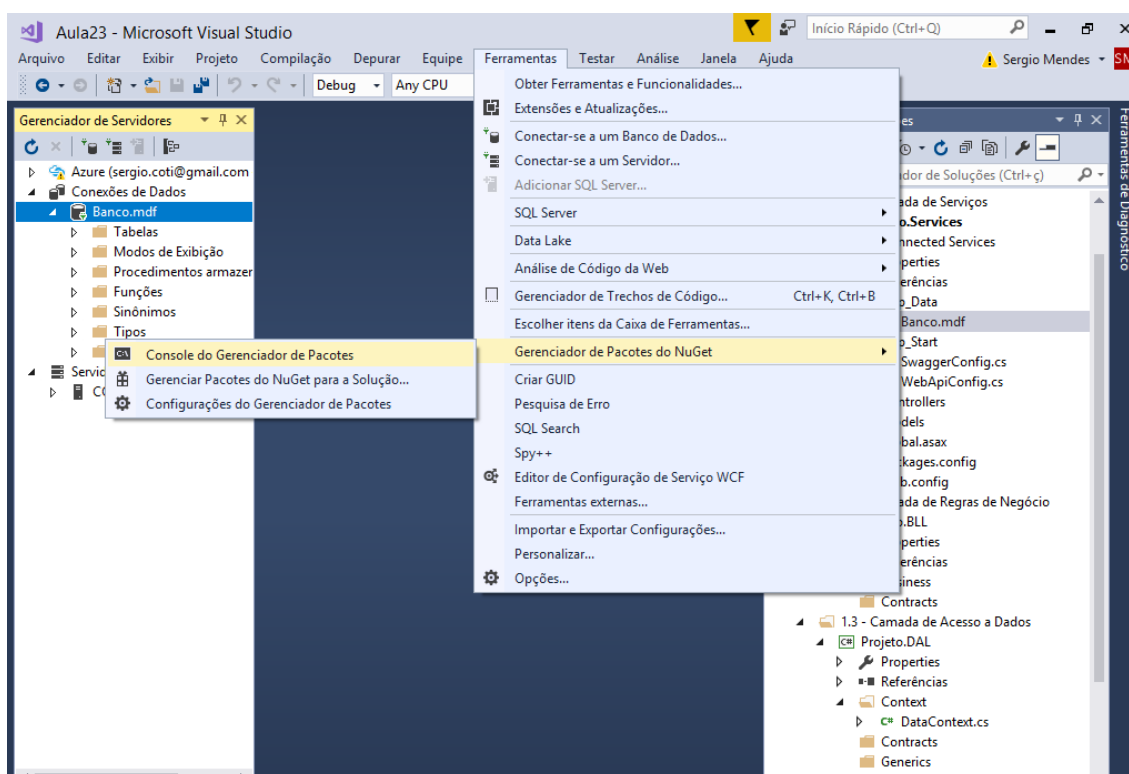
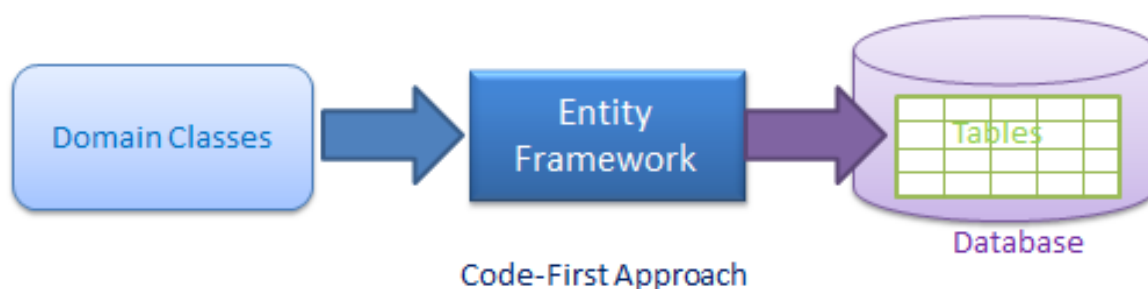

\Web.config.xml

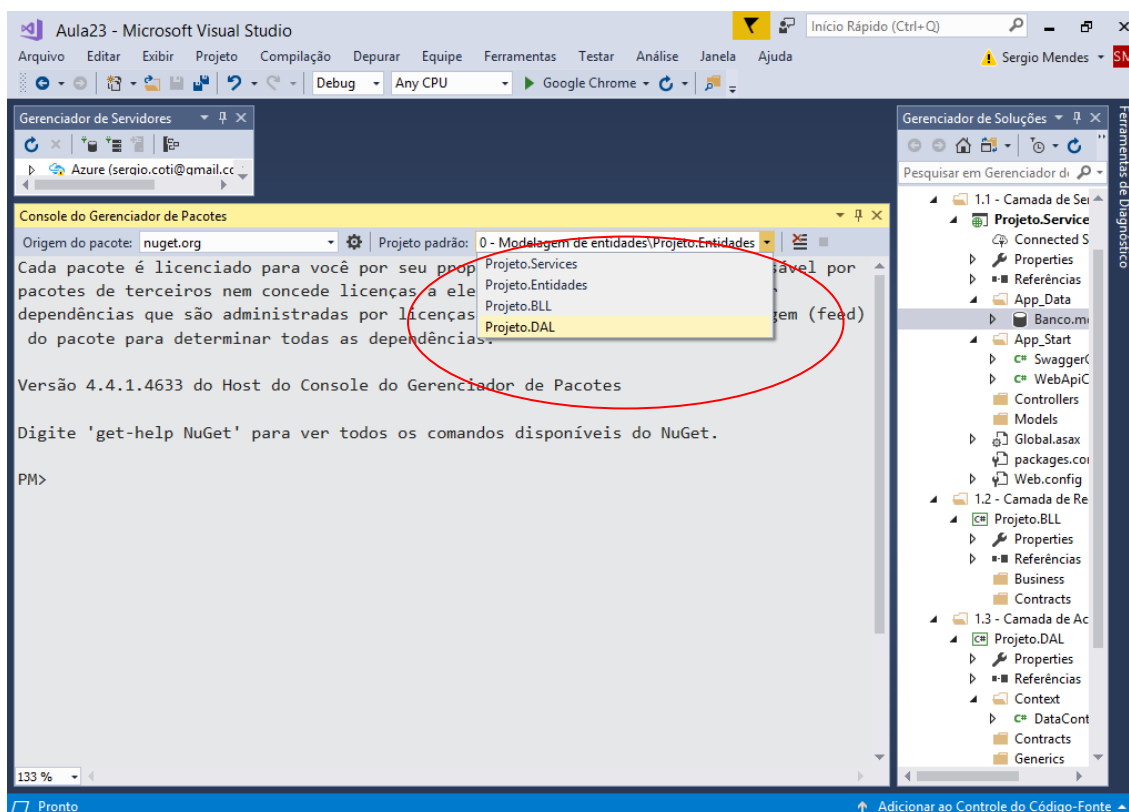
Maapeamento da connectionstring

```
<!-- 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>
```

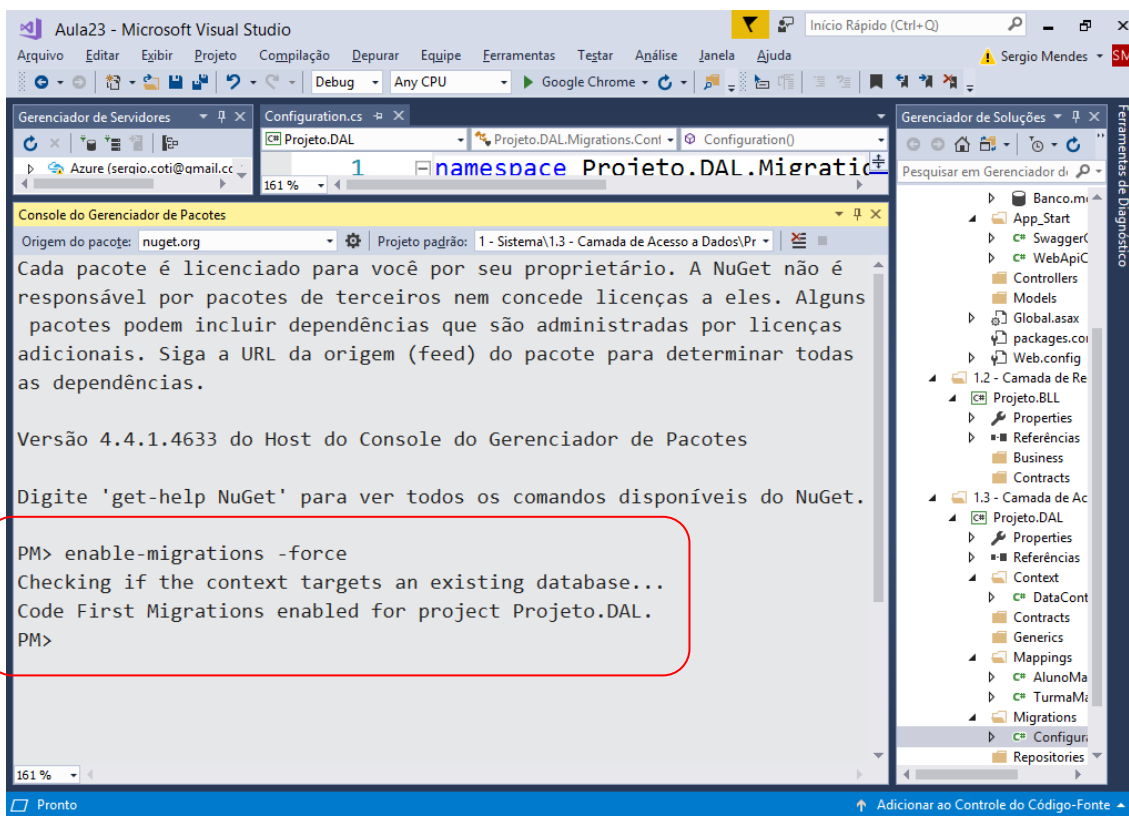
Migrations (CodeFirst)

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:

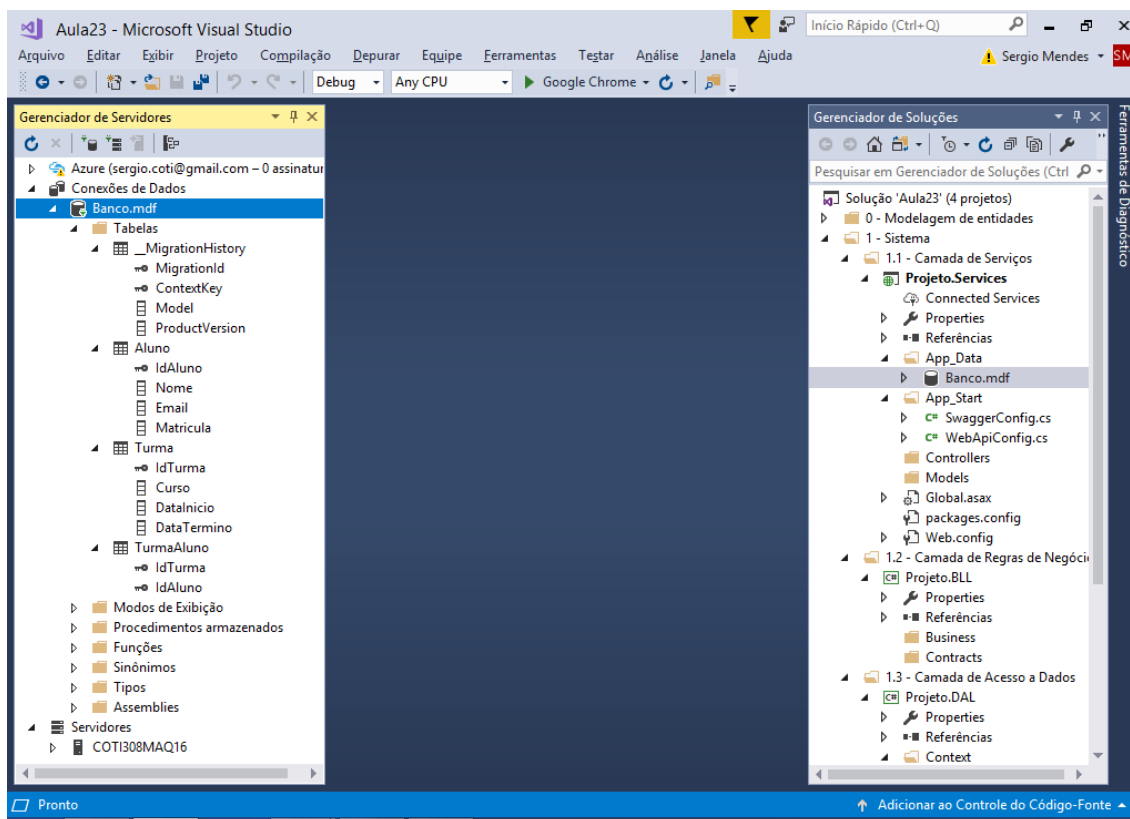
PM> update-database -verbose

```
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])
)

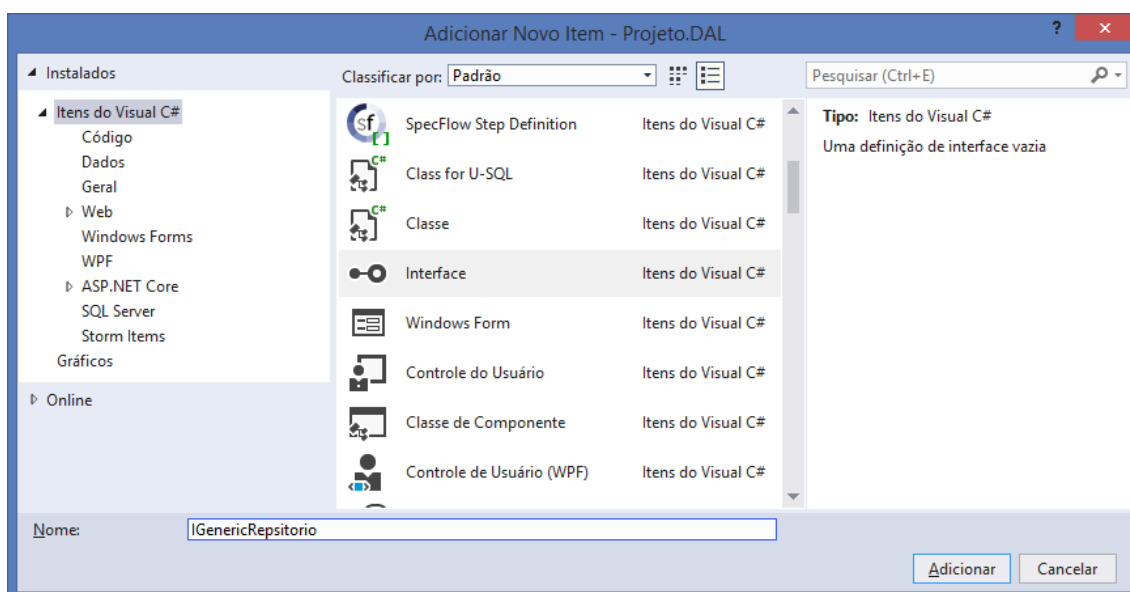
CREATE TABLE [dbo].[Turma] (
    [IdTurma] [int] NOT NULL IDENTITY,
    [Curso] [nvarchar](50) NOT NULL,
    [DataInicio] [datetime] NOT NULL,
    [DataTermino] [datetime] NOT NULL,
    CONSTRAINT [PK_dbo.Turma] PRIMARY KEY ([IdTurma])
)

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

No banco de dados:



Criando interfaces para cada classe que será programada no repositório:



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

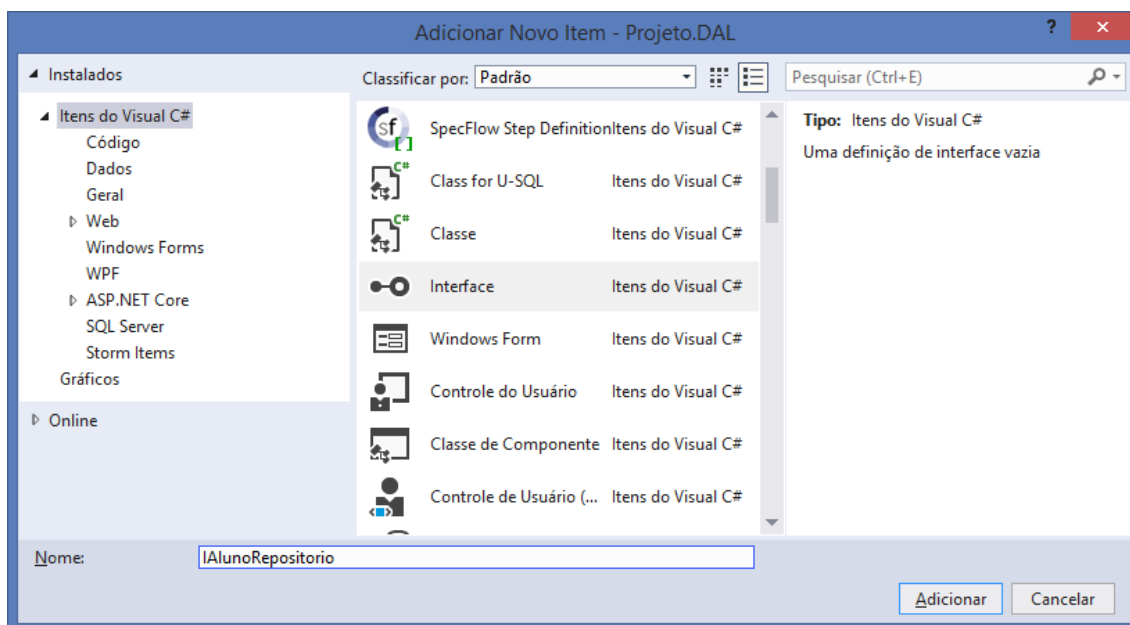
```
namespace Projeto.DAL.Contracts
{
    // <T> Tipo Genérico
    public interface IGenericRepositorio<T>
        where T : class
    {
        void Insert(T obj);

        void Update(T obj);

        void Delete(T obj);

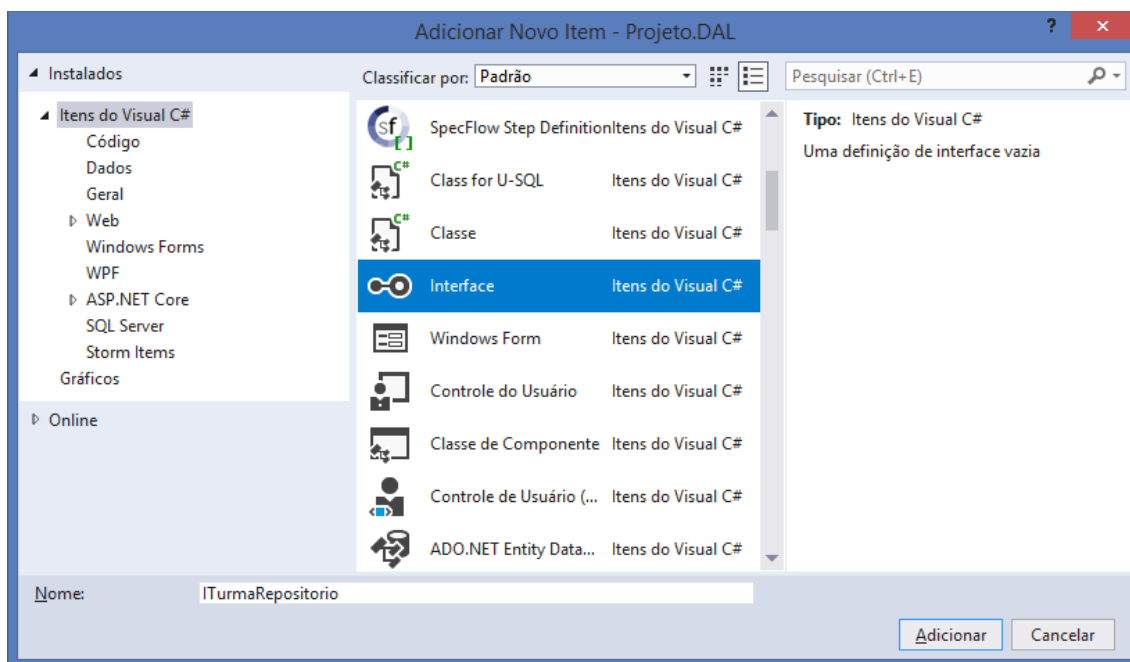
        List<T> FindAll();

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



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

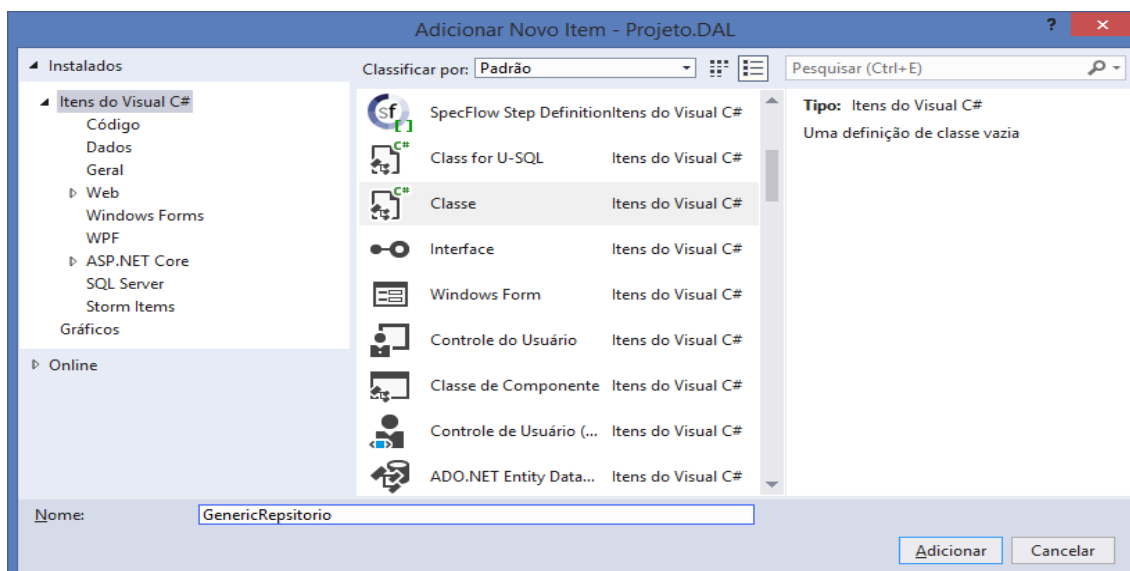
namespace Projeto.DAL.Contracts
{
    public interface IAlunoRepositorio : IGenericRepositorio<Aluno>
    {
        List<Aluno> FindByNome(string nome);
    }
}
```



```
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);
    }
}
```

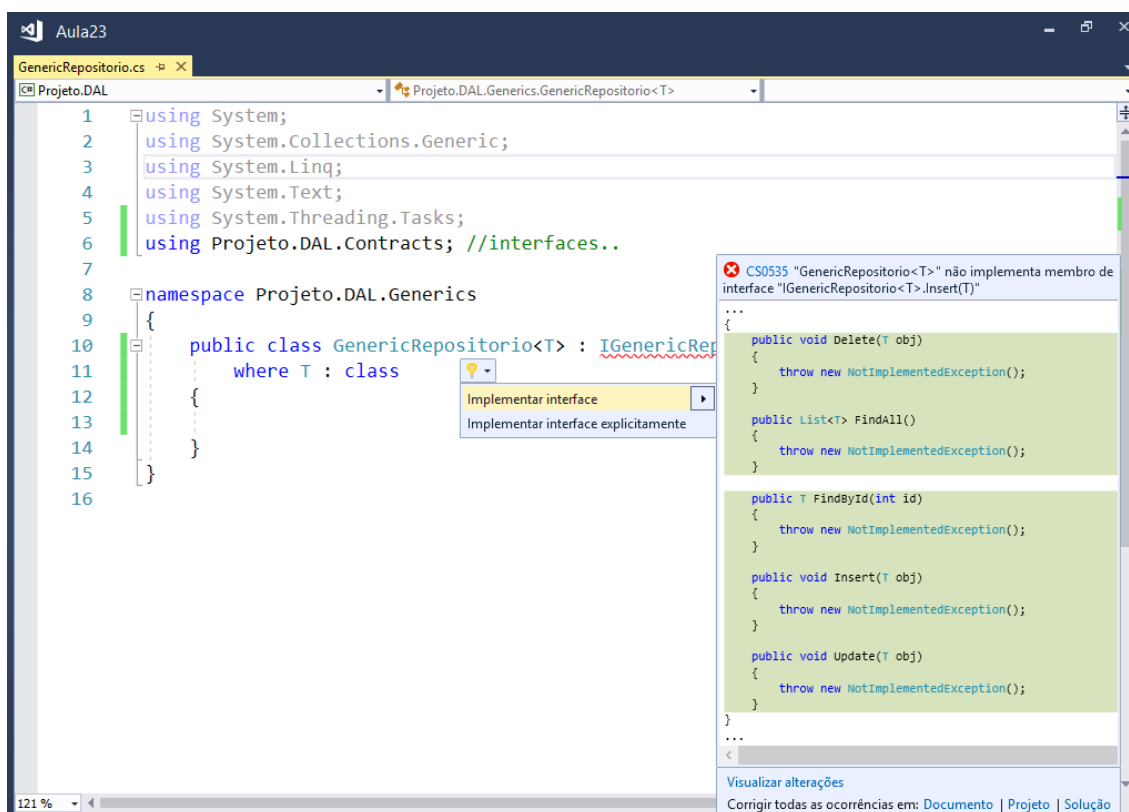
Implementando as interfaces:



```
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();
            }
        }
    }
}
```

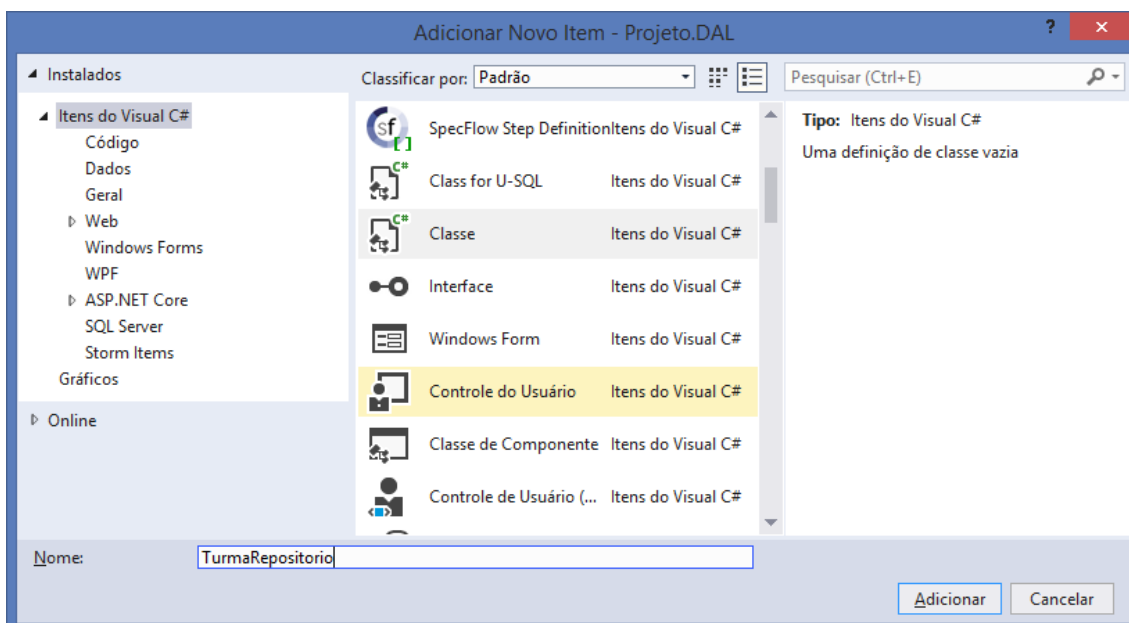


```
public void Delete(T obj)
{
    using (DataContext d = new DataContext())
    {
        d.Entry(obj).State = EntityState.Deleted;
        d.SaveChanges();
    }
}

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);
    }
}
}
```

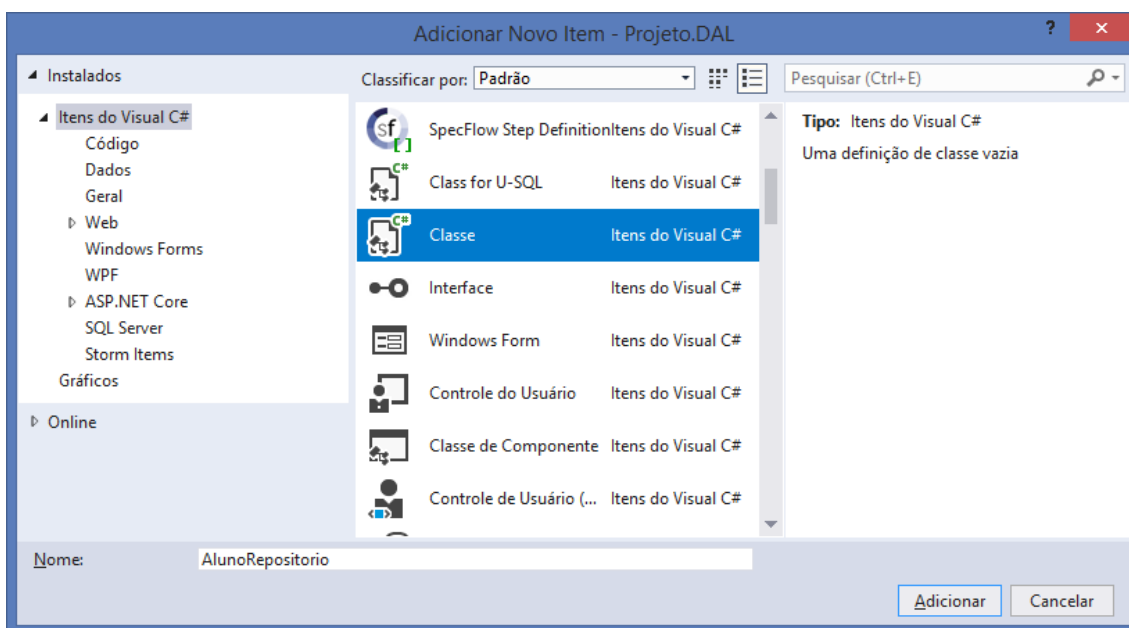
Implementando as demais interfaces:



```
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())
            {

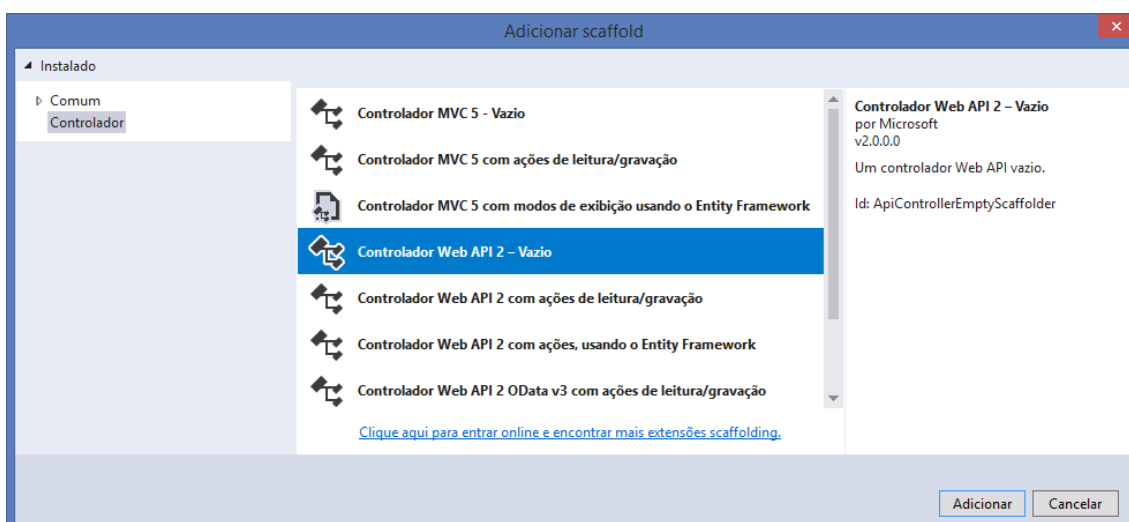
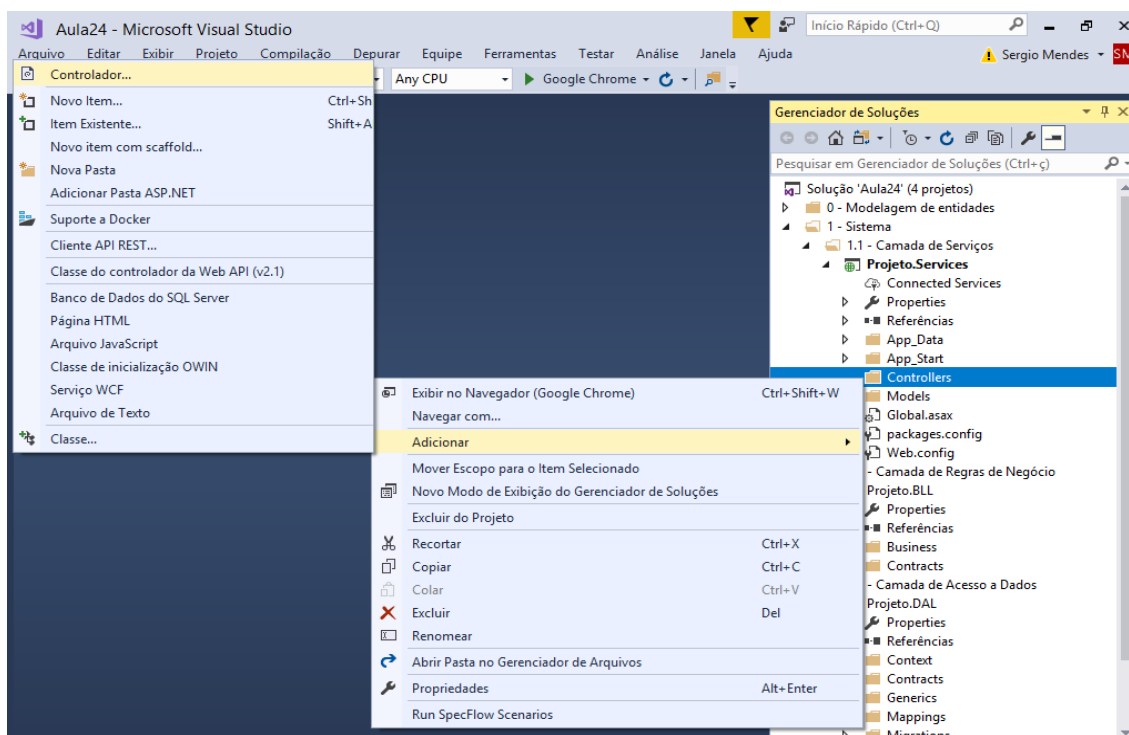
```

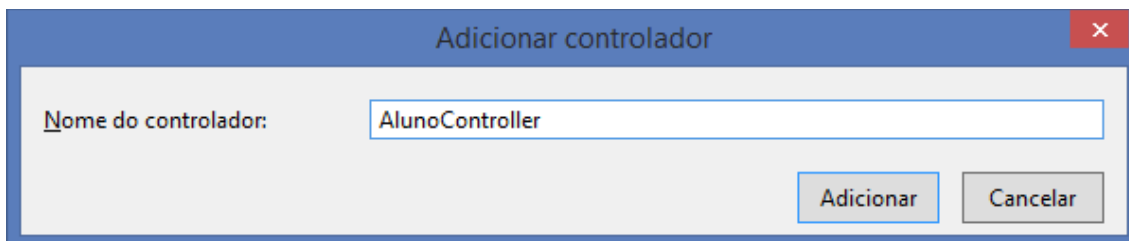
```

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

```

Criando os controllers na camada de serviços para construir a API:





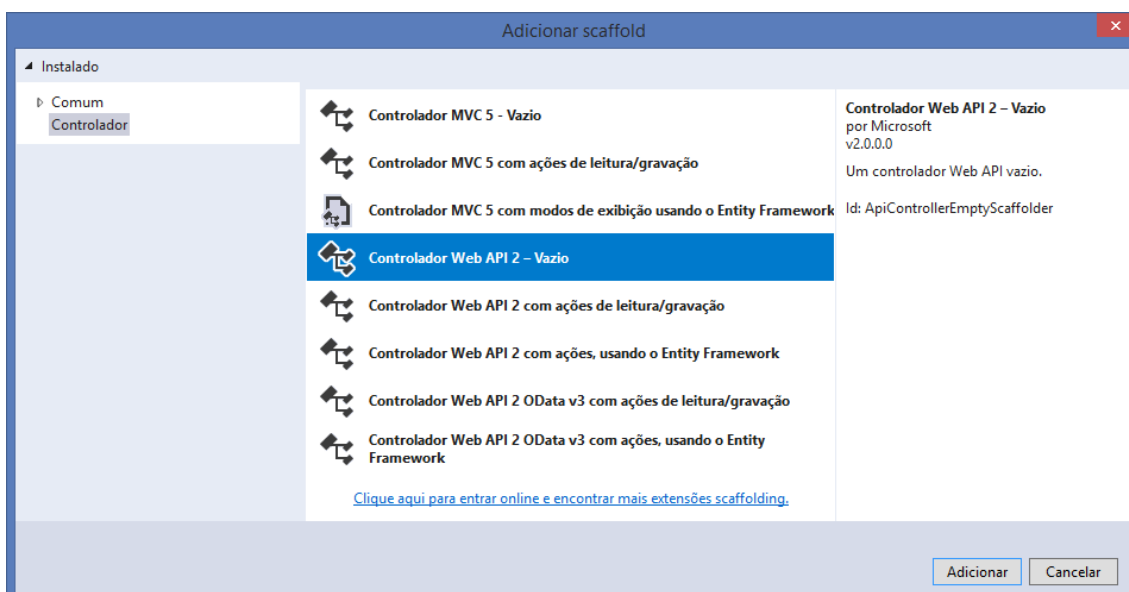
Adicionar controlador

Nome do controlador:

Adicionar Cancelar

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/aluno")]
    public class AlunoController : ApiController
    {
    }
}
```



Adicionar scaffold

Instalado

Comum

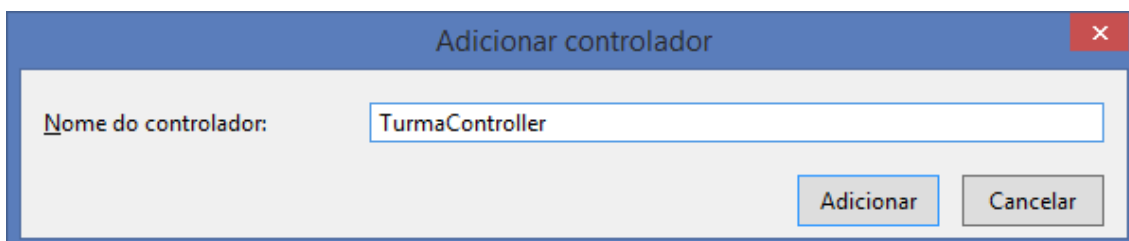
Controlador

- Controlador MVC 5 - Vazio
- Controlador MVC 5 com ações de leitura/gravação
- Controlador MVC 5 com modos de exibição usando o Entity Framework
- Controlador Web API 2 - Vazio**
- Controlador Web API 2 com ações de leitura/gravação
- Controlador Web API 2 com ações, usando o Entity Framework
- Controlador Web API 2 OData v3 com ações de leitura/gravação
- Controlador Web API 2 OData v3 com ações, usando o Entity Framework

Controlador Web API 2 - Vazio por Microsoft v2.0.0.0
Um controlador Web API vazio.
Id: ApiControllerEmptyScaffolder

[Clique aqui para entrar online e encontrar mais extensões scaffolding.](#)

Adicionar Cancelar



Adicionar controlador

Nome do controlador:

Adicionar Cancelar

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
```

```
using System.Web.Http;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/turma")]
    public class TurmaController : ApiController
    {
    }
}
```

Criando as classes de modelo para os serviços de Aluno e Turma:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models
{
    public class AlunoCadastroViewModel
    {
        [Required(ErrorMessage = "Campo obrigatório")]
        public string Nome { get; set; }

        [EmailAddress(ErrorMessage = "Email inválido")]
        [Required(ErrorMessage = "Campo obrigatório")]
        public string Email { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models
{
    public class AlunoEdicaoViewModel
    {
        [Required(ErrorMessage = "Campo obrigatório")]
        public int IdAluno { get; set; }

        [Required(ErrorMessage = "Campo obrigatório")]
        public string Nome { get; set; }

        [EmailAddress(ErrorMessage = "Email inválido")]
        [Required(ErrorMessage = "Campo obrigatório")]
        public string Email { get; set; }
    }
}

using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Projeto.Services.Models
{
    public class AlunoConsultaViewModel
    {
        public int IdAluno { get; set; }
        public string Nome { get; set; }
        public string Email { get; set; }
        public string Matricula { get; set; }
    }
}
```

```
-----

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models
{
    public class TurmaCadastroViewModel
    {
        [Required(ErrorMessage = "Campo obrigatório")]
        public string Curso { get; set; }

        [Required(ErrorMessage = "Campo obrigatório")]
        public DateTime DataInicio { get; set; }

        [Required(ErrorMessage = "Campo obrigatório")]
        public DateTime DataTermino { get; set; }
    }
}
```

```
-----

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models
{
    public class TurmaEdicaoViewModel
    {
        [Required(ErrorMessage = "Campo obrigatório")]
        public int IdTurma { get; set; }

        [Required(ErrorMessage = "Campo obrigatório")]
        public string Curso { get; set; }

        [Required(ErrorMessage = "Campo obrigatório")]
        public DateTime DataInicio { get; set; }
    }
}
```

```
[Required(ErrorMessage = "Campo obrigatório")]
public DateTime DataTermino { get; set; }
}
}
```

```
-----

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Projeto.Services.Models
{
    public class TurmaConsultaViewModel
    {
        public int IdTurma { get; set; }
        public string Curso { get; set; }
        public DateTime DataInicio { get; set; }
        public DateTime DataTermino { get; set; }
    }
}
```

Criando os métodos da classe AlunoController: /Controllers/AlunoController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Projeto.Services.Models;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/aluno")]
    public class AlunoController : ApiController
    {
        [HttpPost]
        [Route("cadastrar")] //URL: /api/aluno/cadastrar
        public HttpResponseMessage Post(AlunoCadastroViewModel model)
        {
            try
            {
                //TODO..
                return Request.CreateResponse(HttpStatusCode.OK);
            }
            catch (Exception e)
            {
                return Request.CreateResponse
                    (HttpStatusCode.InternalServerError, e.Message);
            }
        }

        [HttpPut]
        [Route("atualizar")] //URL: /api/aluno/atualizar
        public HttpResponseMessage Put(AlunoEdicaoViewModel model)
        {

```

```

        try
        {
            //TODO..
            return Request.CreateResponse(HttpStatusCode.OK);
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }

    [HttpDelete]
    [Route("excluir")] //URL: /api/aluno/excluir?id={0}
    public HttpResponseMessage Delete(int id)
    {
        try
        {
            //TODO..
            return Request.CreateResponse(HttpStatusCode.OK);
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }

    [HttpGet]
    [Route("consultar")] //URL: /api/aluno/consultar
    public HttpResponseMessage GetAll()
    {
        try
        {
            //TODO..
            return Request.CreateResponse(HttpStatusCode.OK);
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }

    [HttpGet]
    [Route("obter")] //URL: /api/aluno/obter?id={0}
    public HttpResponseMessage GetById(int id)
    {
        try
        {
            return Request.CreateResponse(HttpStatusCode.OK);
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }
}
}
}

```


Criando os métodos da classe TurmaController:

/Controllers/TurmaController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Projeto.Services.Models;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/turma")]
    public class TurmaController : ApiController
    {
        [HttpPost]
        [Route("cadastrar")] //URL: /api/turma/cadastrar
        public HttpResponseMessage Post(TurmaCadastroViewModel model)
        {
            try
            {
                //TODO..
                return Request.CreateResponse(HttpStatusCode.OK);
            }
            catch (Exception e)
            {
                return Request.CreateResponse
                    (HttpStatusCode.InternalServerError, e.Message);
            }
        }

        [HttpPut]
        [Route("atualizar")] //URL: /api/turma/atualizar
        public HttpResponseMessage Put(TurmaEdicaoViewModel model)
        {
            try
            {
                //TODO..
                return Request.CreateResponse(HttpStatusCode.OK);
            }
            catch (Exception e)
            {
                return Request.CreateResponse
                    (HttpStatusCode.InternalServerError, e.Message);
            }
        }

        [HttpDelete]
        [Route("excluir")] //URL: /api/turma/excluir?id={0}
        public HttpResponseMessage Delete(int id)
        {
            try
            {
                //TODO..
                return Request.CreateResponse(HttpStatusCode.OK);
            }
            catch (Exception e)
            {
            }
        }
    }
}
```

```

        return Request.CreateResponse
            (HttpStatusCode.InternalServerError, e.Message);
    }

    [HttpGet]
    [Route("consultar")] //URL: /api/turma/consultar
    public HttpResponseMessage GetAll()
    {
        try
        {
            //TODO..
            return Request.CreateResponse(HttpStatusCode.OK);
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }

    [HttpGet]
    [Route("obter")] //URL: /api/turma/obter?id={0}
    public HttpResponseMessage GetById(int id)
    {
        try
        {
            //TODO..
            return Request.CreateResponse(HttpStatusCode.OK);
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }
}

```

Global.asax

Incluindo a configuração para que o projeto tenha permissão para ser acessado por outras aplicações externas (**Access-Control-Allow-Origin**)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Routing;

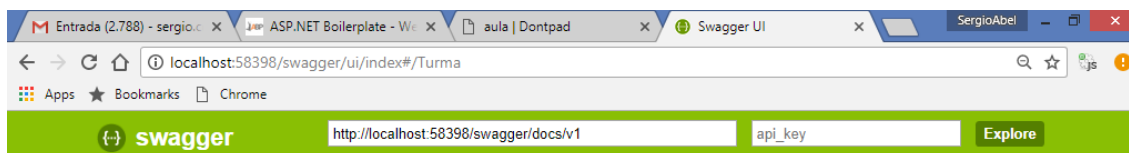
namespace Projeto.Services
{
    public class WebApiApplication : System.Web.HttpApplication
    {
```

```
protected void Application_Start()
{
    GlobalConfiguration.Configure(WebApiConfig.Register);
}

protected void Application_BeginRequest(object sender, EventArgs e)
{
    HttpContext.Current.Response.AddHeader
        ("Access-Control-Allow-Origin", "*");

    if (HttpContext.Current.Request.HttpMethod == "OPTIONS")
    {
        HttpContext.Current.Response.AddHeader
            ("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");
        HttpContext.Current.Response.AddHeader
            ("Access-Control-Allow-Headers", "Content-Type, Accept,
            Authorization");
        HttpContext.Current.Response.AddHeader
            ("Access-Control-Max-Age", "1728000");
        HttpContext.Current.Response.End();
    }
}
}
```

Executando o projeto: Swagger (Documentação da API)



Projeto.Services

Aluno

Show/Hide | List Operations | Expand Operations

POST	/api/aluno/cadastrar
PUT	/api/aluno/atualizar
DELETE	/api/aluno/excluir
GET	/api/aluno/consultar
GET	/api/aluno/obter

Turma

Show/Hide | List Operations | Expand Operations

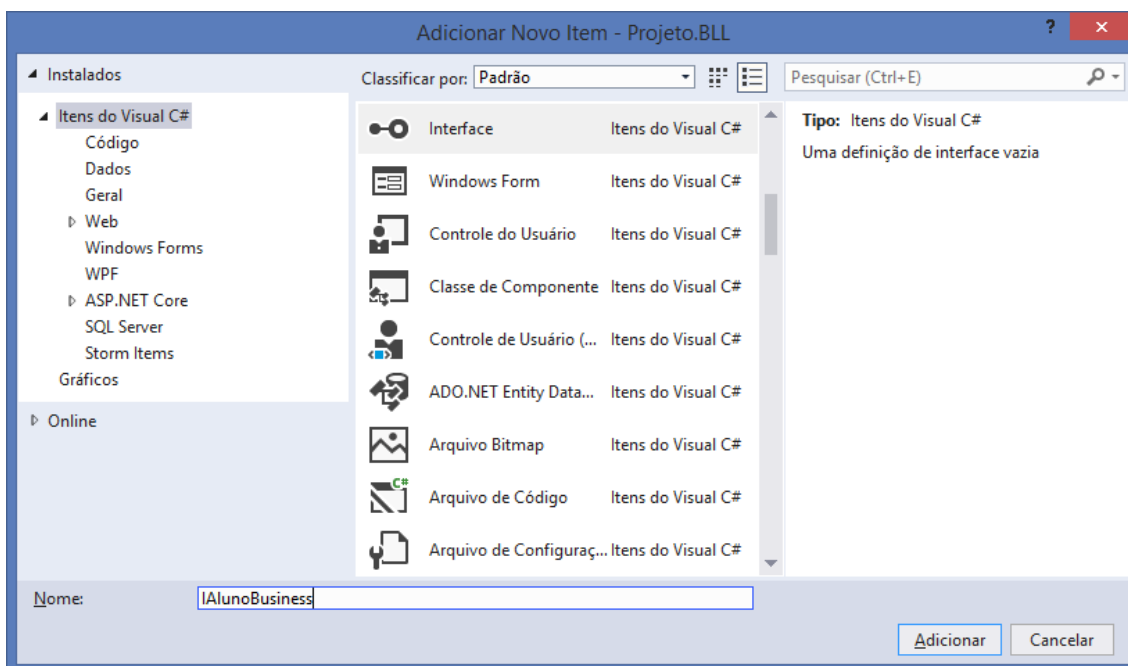
POST	/api/turma/cadastrar
PUT	/api/turma/atualizar
DELETE	/api/turma/excluir
GET	/api/turma/consultar
GET	/api/turma/obter

Business - Camada de Regras de Negócio

BLL - Business Logic Layer

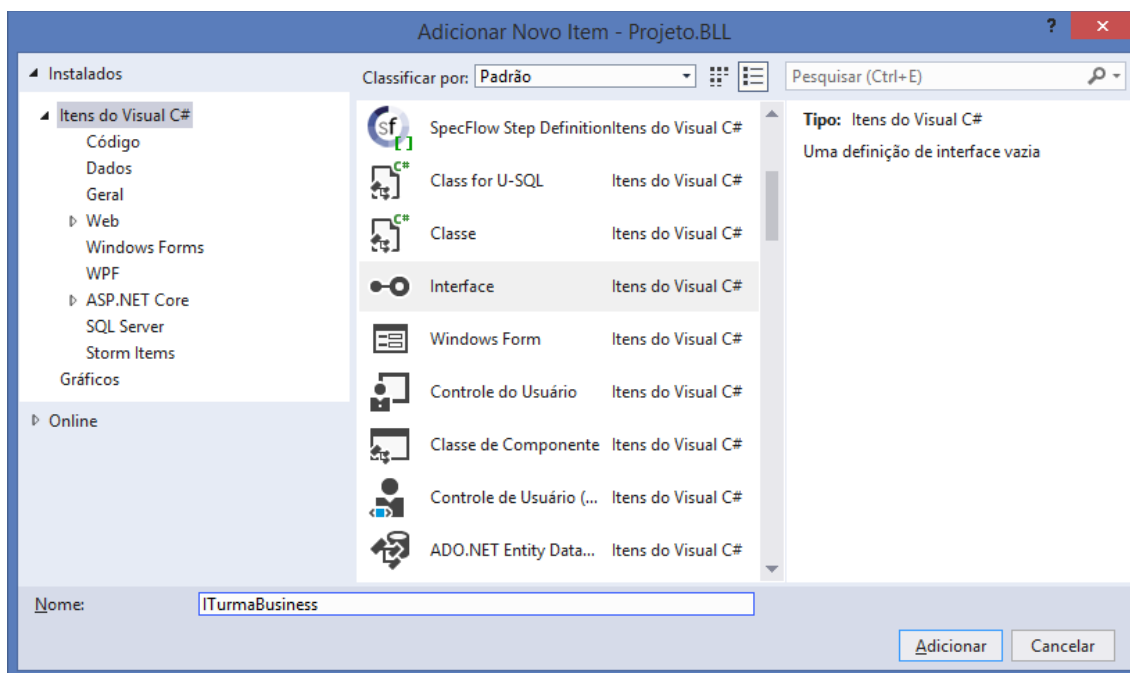
Primeiro Passo:

Criar as interfaces para as classes de regras de negócio:



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

namespace Projeto.BLL.Contracts
{
    public interface IALunoBusiness
    {
        void Cadastrar(Aluno a);
        void Atualizar(Aluno a);
        void Excluir(int idAluno);
        List<Aluno> ConsultarTodos();
        Aluno ObterPorId(int idAluno);
    }
}
```



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

namespace Projeto.BLL.Contracts
{
    public interface ITurmaBusiness
    {
        void Cadastrar(Turma t);

        void Atualizar(Turma t);

        void Excluir(int idTurma);

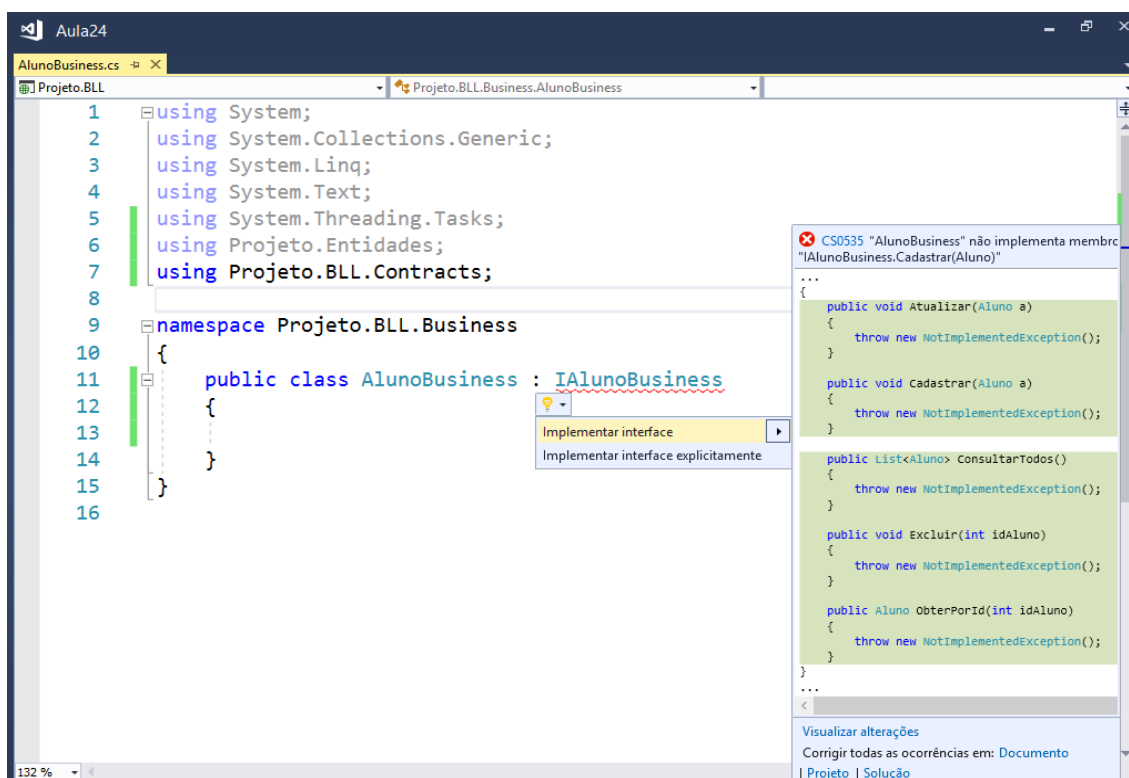
        List<Turma> ConsultarTodos();

        Turma ObterPorId(int idTurma);
    }
}
```

Implementando os métodos de cada interface:

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

namespace Projeto.BLL.Business
{
    public class AlunoBusiness : IAlunoBusiness
    {
    }
}
```



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

namespace Projeto.BLL.Business
{
    public class AlunoBusiness : IAlunoBusiness
    {
        public void Cadastrar(Aluno a)
        {
            throw new NotImplementedException();
        }
    }
}
```

```

    }

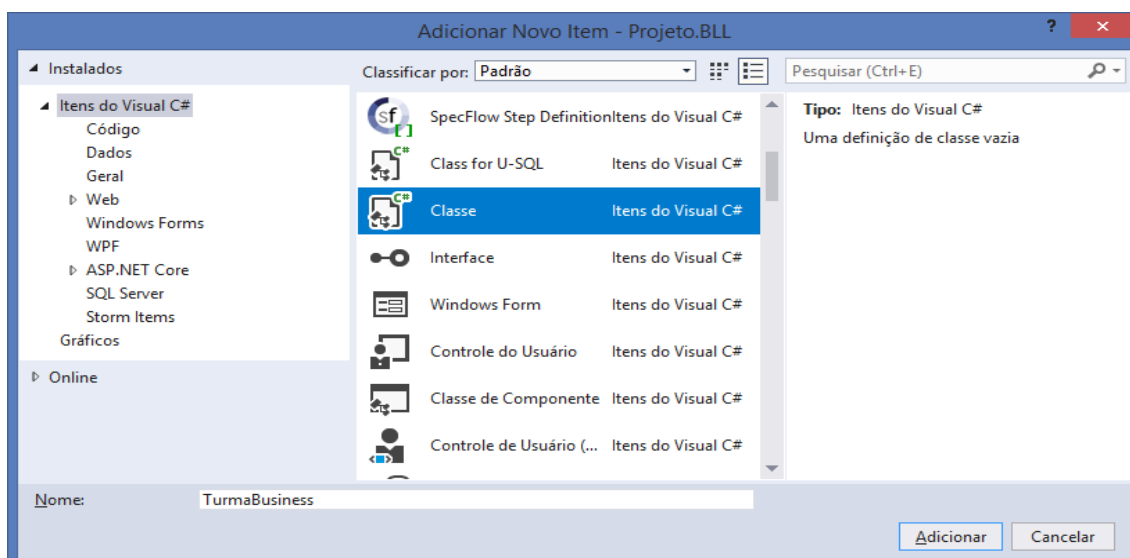
    public void Atualizar(Aluno a)
    {
        throw new NotImplementedException();
    }

    public void Excluir(int idAluno)
    {
        throw new NotImplementedException();
    }

    public List<Aluno> ConsultarTodos()
    {
        throw new NotImplementedException();
    }

    public Aluno ObterPorId(int idAluno)
    {
        throw new NotImplementedException();
    }
}
}

```



```

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

namespace Projeto.BLL.Business
{
    public class TurmaBusiness : ITurmaBusiness
    {
    }
}

```

Implementando os métodos da interface:

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

namespace Projeto.BLL.Business
{
    public class TurmaBusiness : ITurmaBusiness
    {
        public void Cadastrar(Turma t)
        {
            throw new NotImplementedException();
        }

        public void Atualizar(Turma t)
        {
            throw new NotImplementedException();
        }

        public void Excluir(int idTurma)
        {
            throw new NotImplementedException();
        }

        public List<Turma> ConsultarTodos()
        {
            throw new NotImplementedException();
        }

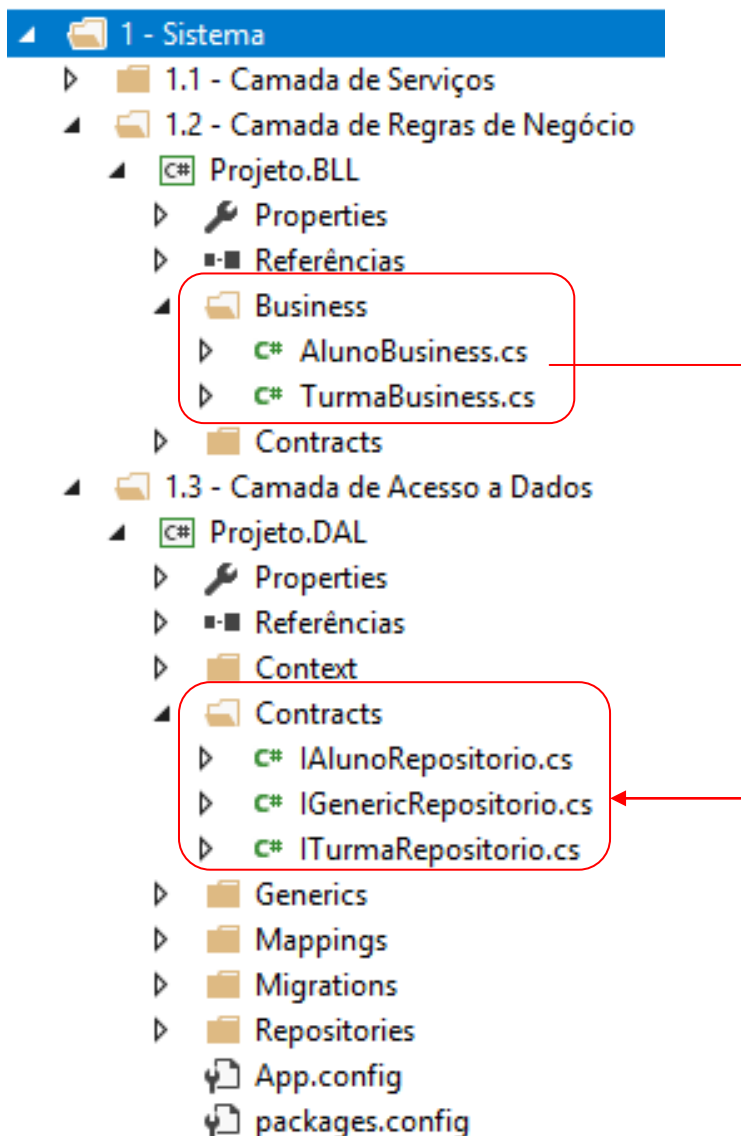
        public Turma ObterPorId(int idTurma)
        {
            throw new NotImplementedException();
        }
    }
}
```

Injeção de Dependência

Princípio da programação orientada a objetos que tem como objetivo "desacoplar" ainda mais a dependência entre os módulos de um sistema de forma a permitir que cada módulo de um projeto conheça o mínimo possível a respeito do conteúdo de outro módulo.

Em um projeto que utiliza o conceito de **DI (Dependency Injection)**, cada camada irá acessar os métodos de outro módulo através não de suas classes mas sim de suas interfaces.

Exemplo: A camada Business irá acessar os métodos da camada DAL por meio de suas interfaces:



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

namespace Projeto.BLL.Business
{
    public class AlunoBusiness : IAlunoBusiness
    {
        //atributo..
        private IAlunoRepositorio repositorio;
```

```
public void Cadastrar(Aluno a)
{
    //gerando um numero de matricula..
    Random r = new Random();
    a.Matricula = DateTime.Now.Year + "-"
        + r.Next(1000, 999999999).ToString();

    repositorio.Insert(a);
}

public void Atualizar(Aluno a)
{
    repositorio.Update(a); //atualizando..
}

public void Excluir(int idAluno)
{
    Aluno a = repositorio.FindById(idAluno);
    repositorio.Delete(a); //excluindo..
}

public List<Aluno> ConsultarTodos()
{
    return repositorio.FindAll();
}

public Aluno ObterPorId(int idAluno)
{
    return repositorio.FindById(idAluno);
}
}
}
```

```
-----

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

namespace Projeto.BLL.Business
{
    public class TurmaBusiness : ITurmaBusiness
    {
        private ITurmaRepositorio repositorio;

        public void Cadastrar(Turma t)
        {
            repositorio.Insert(t);
        }

        public void Atualizar(Turma t)
        {
            repositorio.Update(t);
        }
    }
}
```

```

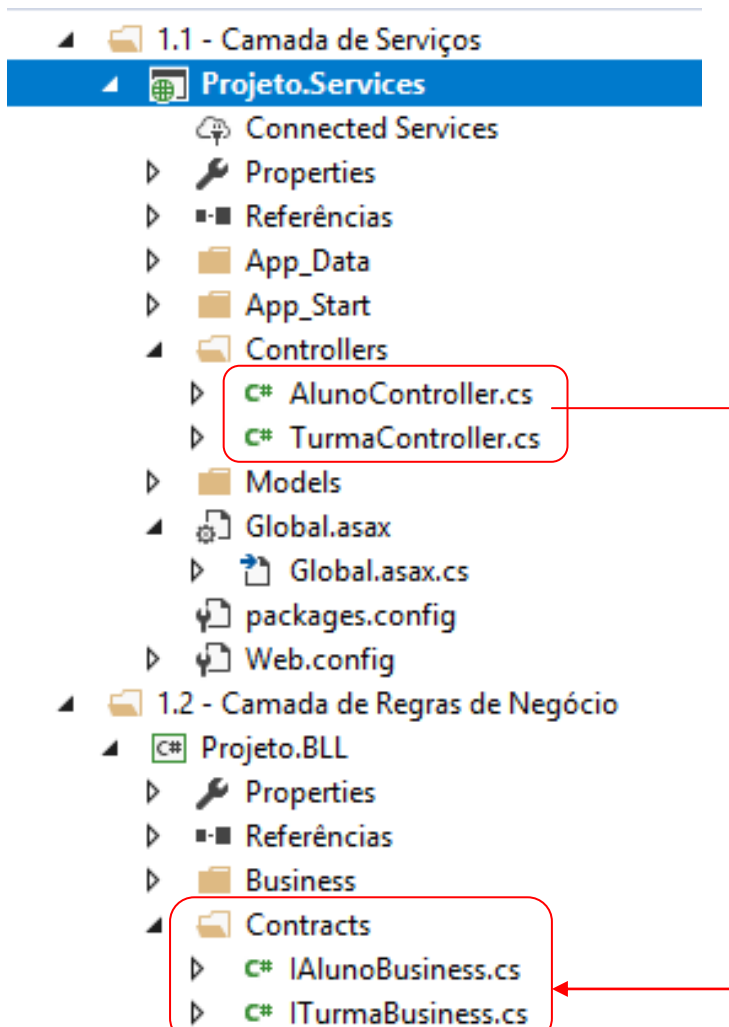
public void Excluir(int idTurma)
{
    Turma t = repositorio.FindById(idTurma);
    repositorio.Delete(t);
}

public List<Turma> ConsultarTodos()
{
    return repositorio.FindAll();
}

public Turma ObterPorId(int idTurma)
{
    return repositorio.FindById(idTurma);
}
}
}

```

O projeto WebApi irá acessar as interfaces da camada BLL



using System;

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Projeto.Services.Models;
using Projeto.BLL.Contracts;
using Projeto.Entidades;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/aluno")]
    public class AlunoController : ApiController
    {
        //atributo..
        private IAlunoBusiness business;

        [HttpPost]
        [Route("cadastrar")] //URL: /api/aluno/cadastrar
        public HttpResponseMessage Post(AlunoCadastroViewModel model)
        {
            try
            {
                if(ModelState.IsValid) //se passou nas regras de validação..
                {
                    Aluno a = new Aluno(); //entidade..
                    a.Nome = model.Nome;
                    a.Email = model.Email;

                    business.Cadastrar(a);

                    return Request.CreateResponse(HttpStatusCode.OK,
                        "Aluno cadastrado com sucesso.");
                }
                else
                {
                    return Request.CreateResponse //Erro HTTP 400
                        (HttpStatusCode.BadRequest, ModelState);
                }
            }
            catch(Exception e)
            {
                return Request.CreateResponse //Erro HTTP 500
                    (HttpStatusCode.InternalServerError, e.Message);
            }
        }

        [HttpPut]
        [Route("atualizar")] //URL: /api/aluno/atualizar
        public HttpResponseMessage Put(AlunoEdicaoViewModel model)
        {
            try
            {
                if(ModelState.IsValid)
                {
                    //buscar o aluno pelo id..
                    Aluno a = business.ObterPorId(model.IdAluno);

                    a.Nome = model.Nome;
                    a.Email = model.Email;
                }
            }
        }
    }
}
```

```

        business.Atualizar(a); //atualizando..

        return Request.CreateResponse(HttpStatusCode.OK,
            "Aluno atualizado com sucesso.");
    }
    else
    {
        return Request.CreateResponse
            (HttpStatusCode.BadRequest, ModelState);
    }
}
catch (Exception e)
{
    return Request.CreateResponse
        (HttpStatusCode.InternalServerError, e.Message);
}
}

[HttpDelete]
[Route("excluir")] //URL: /api/aluno/excluir?id={0}
public HttpResponseMessage Delete(int id)
{
    try
    {
        business.Excluir(id);

        return Request.CreateResponse(HttpStatusCode.OK,
            "Aluno excluído com sucesso.");
    }
    catch (Exception e)
    {
        return Request.CreateResponse
            (HttpStatusCode.InternalServerError, e.Message);
    }
}

[HttpGet]
[Route("consultar")] //URL: /api/aluno/consultar
public HttpResponseMessage GetAll()
{
    try
    {
        List<AlunoConsultaViewModel> lista
            = new List<AlunoConsultaViewModel>();

        foreach (Aluno a in business.ConsultarTodos())
        {
            AlunoConsultaViewModel model = new AlunoConsultaViewModel();
            model.IdAluno = a.IdAluno;
            model.Nome = a.Nome;
            model.Email = a.Email;
            model.Matricula = a.Matricula;

            lista.Add(model); //adicionar na lista..
        }

        return Request.CreateResponse(HttpStatusCode.OK, lista);
    }
}

```

```

        catch (Exception e)
        {
            return Request.CreateResponse(
                HttpStatusCode.InternalServerError, e.Message);
        }
    }

    [HttpGet]
    [Route("obter")] //URL: /api/aluno/obter?id={0}
    public HttpResponseMessage GetById(int id)
    {
        try
        {
            Aluno a = business.ObterPorId(id);

            if(a != null) //se o aluno foi encontrado..
            {
                AlunoConsultaViewModel model = new AlunoConsultaViewModel();
                model.IdAluno = a.IdAluno;
                model.Nome = a.Nome;
                model.Email = a.Email;
                model.Matricula = a.Matricula;

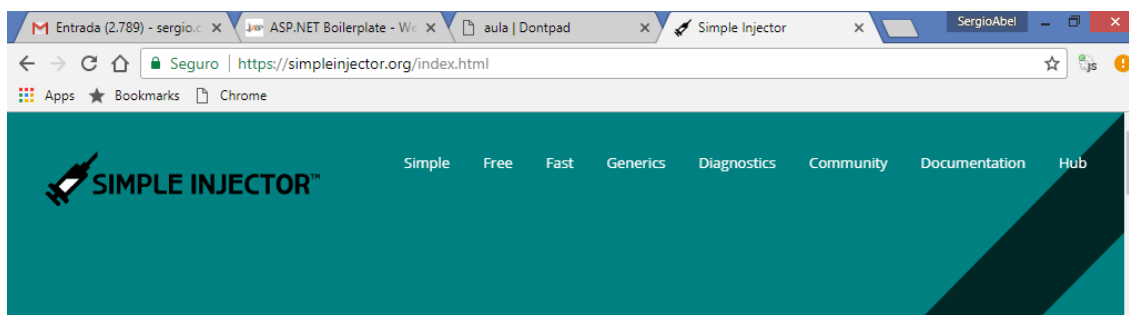
                return Request.CreateResponse(HttpStatusCode.OK, model);
            }
            else
            {
                return Request.CreateResponse(HttpStatusCode.BadRequest,
                    "Aluno não encontrado");
            }
        }
        catch (Exception e)
        {
            return Request.CreateResponse(
                HttpStatusCode.InternalServerError, e.Message);
        }
    }
}

```

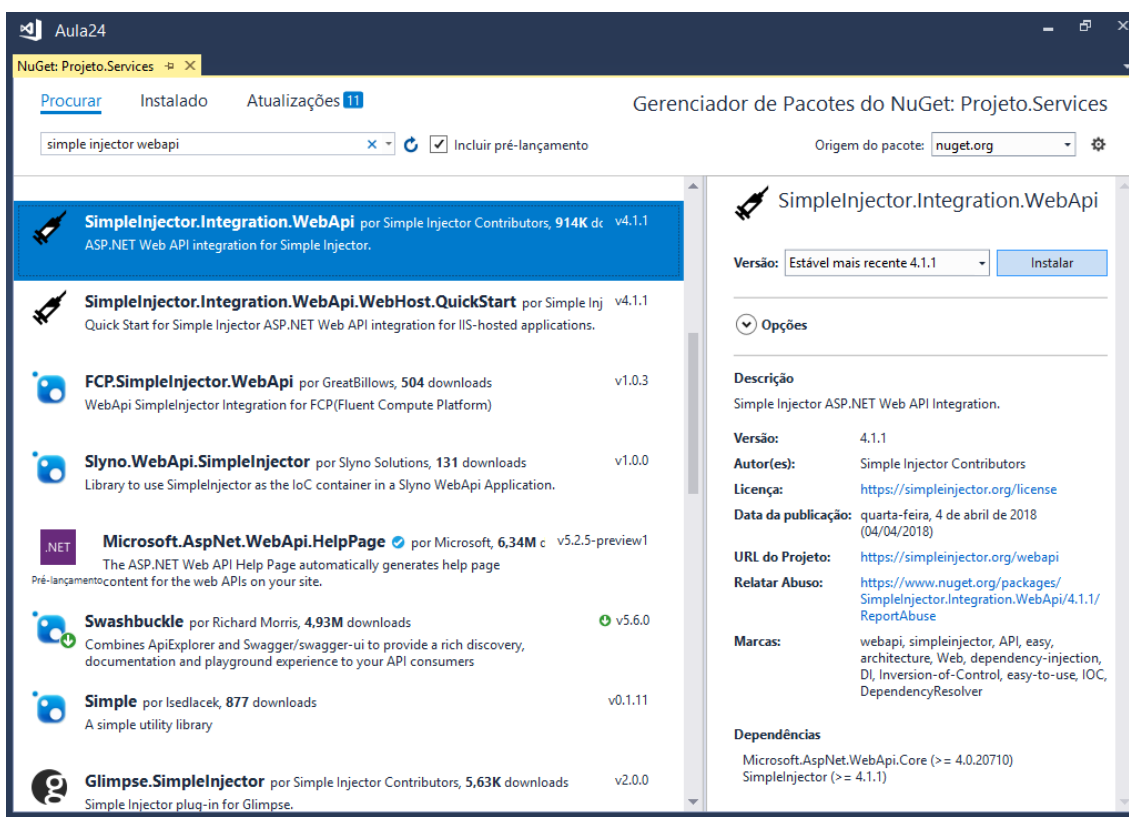
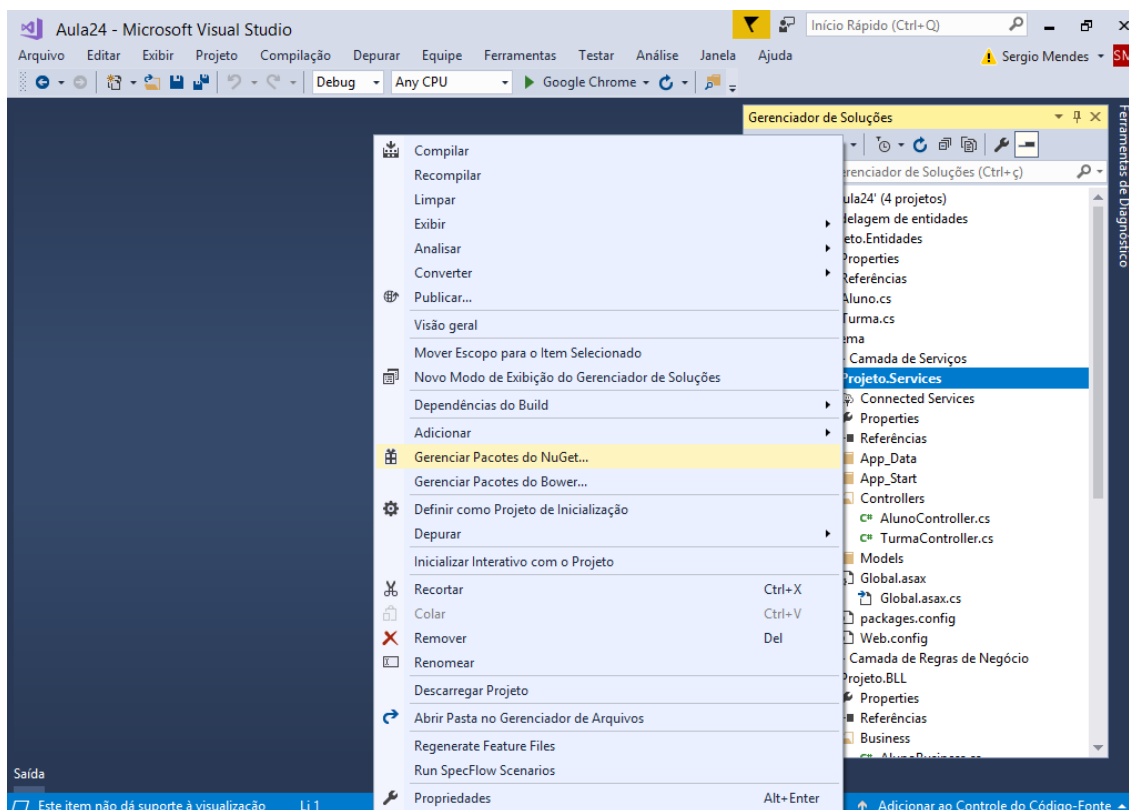
Para que a injeção de dependencia funcione, precisamos instalar um framework que será responsavel por inicializar as interfaces necessarias para que uma camada possa ser executada.

Simple Injector

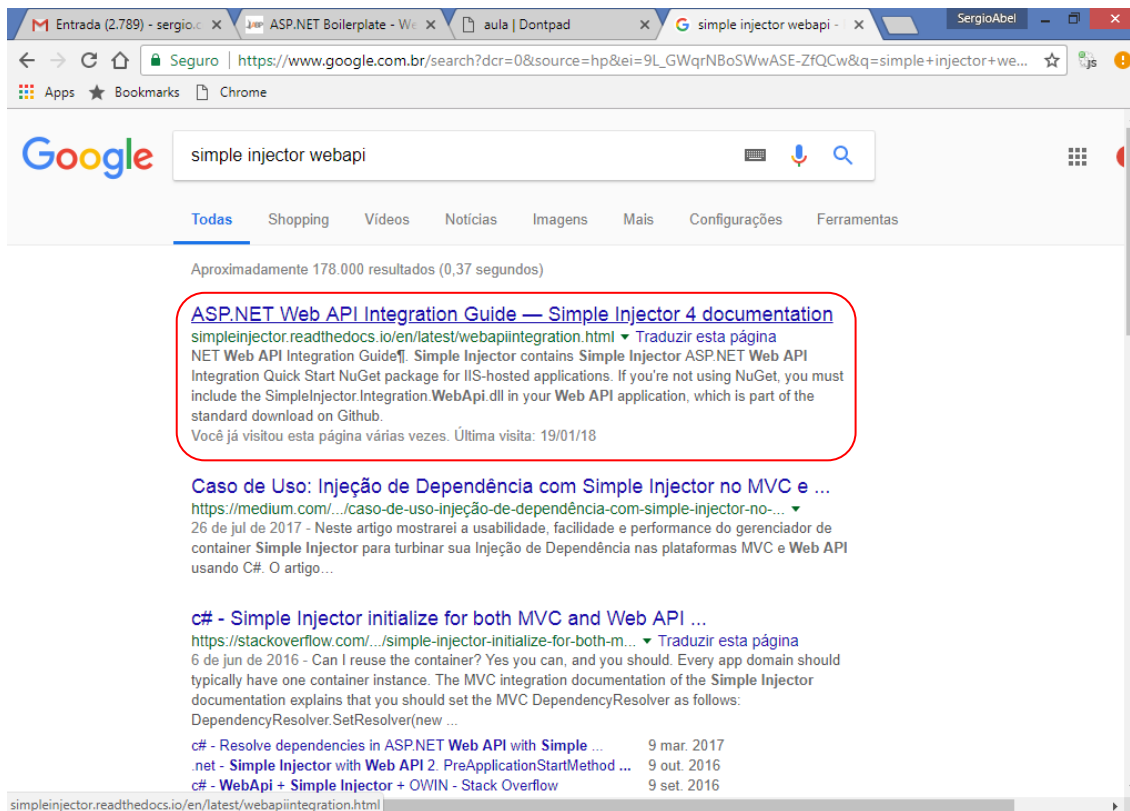
<https://simpleinjector.org/index.html>



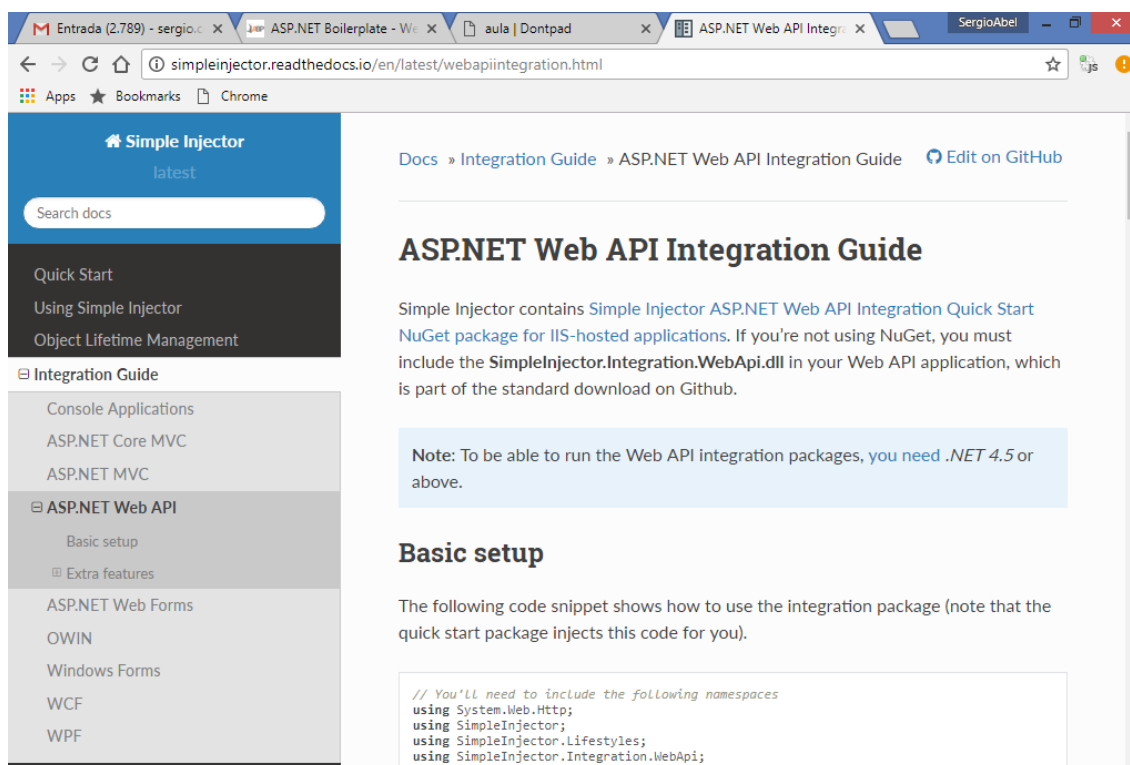
Instalando:



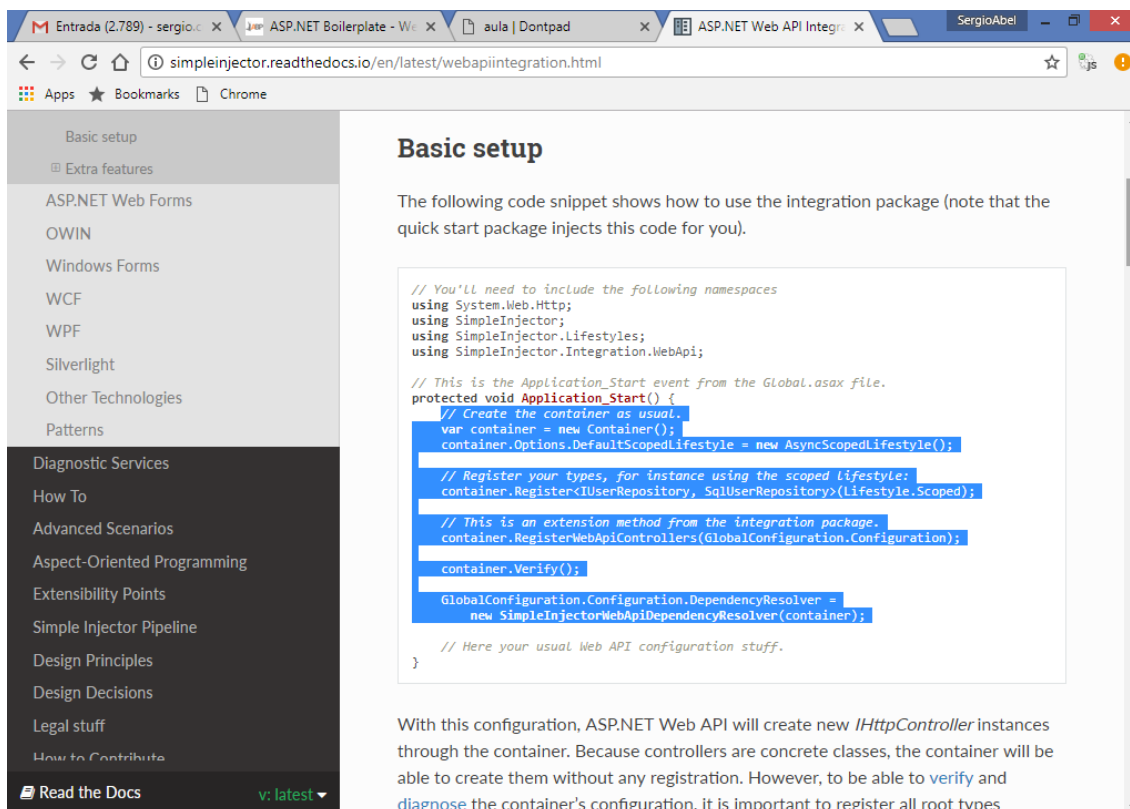
Mapear as interfaces do sistema bem como a classe que implementa cada interface:



<http://simpleinjector.readthedocs.io/en/latest/webapiintegration.html>



Copie o código abaixo:



The screenshot shows a web browser window with the URL `simpleinjector.readthedocs.io/en/latest/webapiintegration.html`. The page title is "Basic setup". The content explains how to use the integration package and provides a code snippet for the `Application_Start` event in the `Global.asax` file.

```
// You'll need to include the following namespaces
using System.Web.Http;
using SimpleInjector;
using SimpleInjector.Lifestyles;
using SimpleInjector.Integration.WebApi;

// This is the Application_Start event from the Global.asax file.
protected void Application_Start() {
    // Create the container as usual.
    var container = new Container();
    container.Options.DefaultScopedLifestyle = new AsyncScopedLifestyle();

    // Register your types, for instance using the scoped lifestyle:
    container.Register<IUserRepository, SqlUserRepository>(Lifestyle.Scoped);

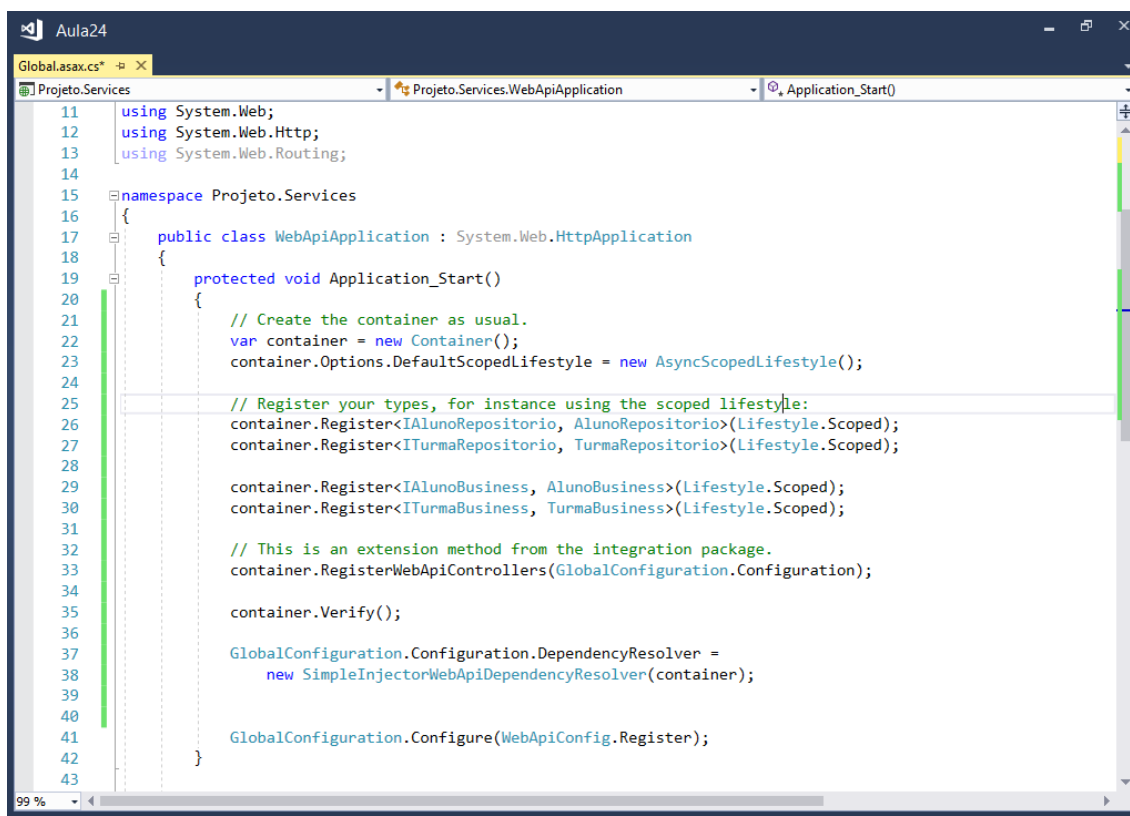
    // This is an extension method from the integration package.
    container.RegisterWebApiControllers(GlobalConfiguration.Configuration);

    container.Verify();

    GlobalConfiguration.Configuration.DependencyResolver =
        new SimpleInjectorWebApiDependencyResolver(container);

    // Here your usual Web API configuration stuff.
}
```

With this configuration, ASP.NET Web API will create new `IHttpController` instances through the container. Because controllers are concrete classes, the container will be able to create them without any registration. However, to be able to **verify** and **diagnose** the container's configuration, it is important to register all root types



The screenshot shows a Visual Studio code editor with the file `Global.asax.cs` open. The code implements the `Application_Start` method, creating a `Container` and registering various types and controllers.

```
11 using System.Web;
12 using System.Web.Http;
13 using System.Web.Routing;
14
15 namespace Projeto.Services
16 {
17     public class WebApiApplication : System.Web.HttpApplication
18     {
19         protected void Application_Start()
20         {
21             // Create the container as usual.
22             var container = new Container();
23             container.Options.DefaultScopedLifestyle = new AsyncScopedLifestyle();
24
25             // Register your types, for instance using the scoped lifestyle:
26             container.Register<IALunoRepositorio, AlunoRepositorio>(Lifestyle.Scoped);
27             container.Register<ITurmaRepositorio, TurmaRepositorio>(Lifestyle.Scoped);
28
29             container.Register<IALunoBusiness, AlunoBusiness>(Lifestyle.Scoped);
30             container.Register<ITurmaBusiness, TurmaBusiness>(Lifestyle.Scoped);
31
32             // This is an extension method from the integration package.
33             container.RegisterWebApiControllers(GlobalConfiguration.Configuration);
34
35             container.Verify();
36
37             GlobalConfiguration.Configuration.DependencyResolver =
38                 new SimpleInjectorWebApiDependencyResolver(container);
39
40             GlobalConfiguration.Configure(WebApiConfig.Register);
41         }
42     }
43 }
```

```
using Projeto.BLL.Business;
using Projeto.BLL.Contracts;
using Projeto.DAL.Contracts;
using Projeto.DAL.Repositories;
using SimpleInjector;
using SimpleInjector.Integration.WebApi;
using SimpleInjector.Lifestyles;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Routing;

namespace Projeto.Services
{
    public class WebApiApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            // Create the container as usual.
            var container = new Container();
            container.Options.DefaultScopedLifestyle
                = new AsyncScopedLifestyle();

            // Register your types, for instance using the scoped lifestyle:
            container.Register<IALunoRepositorio,
                AlunoRepositorio>(Lifestyle.Scoped);
            container.Register<ITurmaRepositorio,
                TurmaRepositorio>(Lifestyle.Scoped);

            container.Register<IALunoBusiness, AlunoBusiness>(Lifestyle.Scoped);
            container.Register<ITurmaBusiness, TurmaBusiness>(Lifestyle.Scoped);

            // This is an extension method from the integration package.
            container.RegisterWebApiControllers
                (GlobalConfiguration.Configuration);

            container.Verify();

            GlobalConfiguration.Configuration.DependencyResolver =
                new SimpleInjectorWebApiDependencyResolver(container);

            GlobalConfiguration.Configure(WebApiConfig.Register);
        }

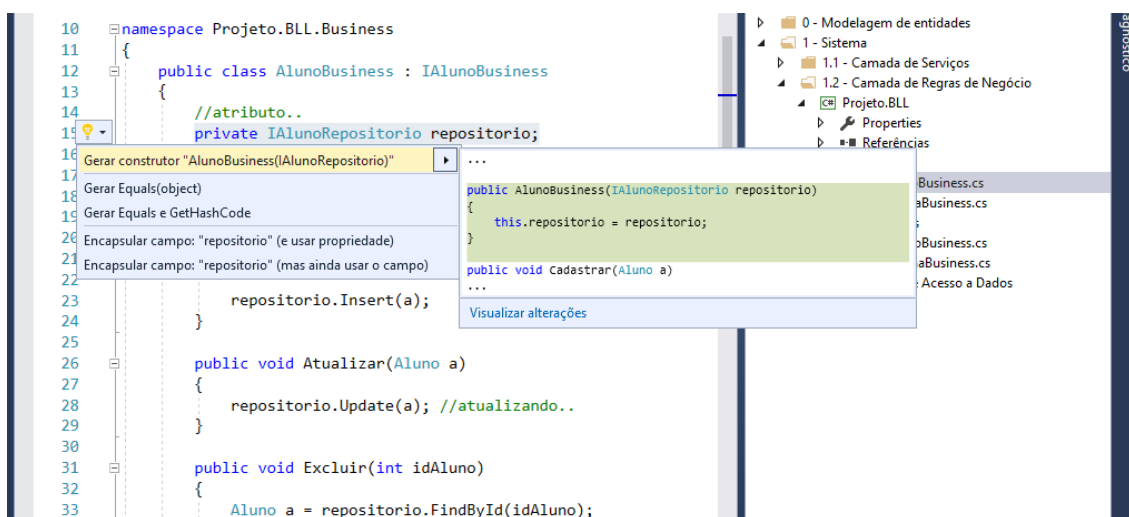
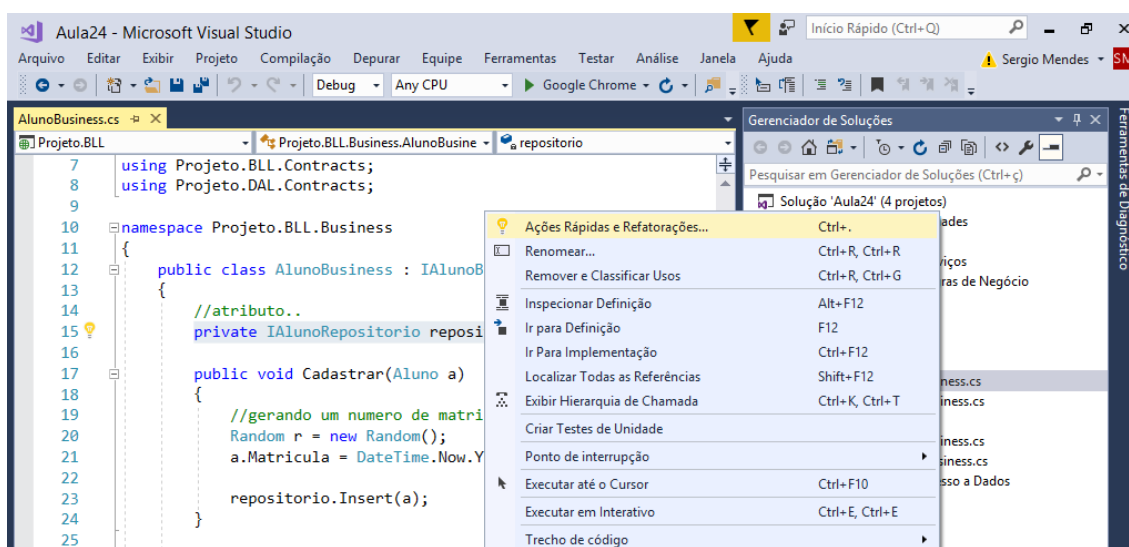
        protected void Application_BeginRequest(object sender, EventArgs e)
        {
            HttpContext.Current.Response.AddHeader
                ("Access-Control-Allow-Origin", "*");
        }
    }
}
```

```

if (HttpContext.Current.Request.HttpMethod == "OPTIONS")
{
    HttpContext.Current.Response.AddHeader
        ("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");
    HttpContext.Current.Response.AddHeader
        ("Access-Control-Allow-Headers", "Content-Type,
        Accept, Authorization");
    HttpContext.Current.Response.AddHeader
        ("Access-Control-Max-Age", "1728000");
    HttpContext.Current.Response.End();
}
}
}
}

```

Para que o SimpleInjector possa inicializar as interfaces, cada classe deverá declarar um construtor recebendo a interface como argumento:



using System;

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

namespace Projeto.BLL.Business
{
    public class AlunoBusiness : IAlunoBusiness
    {
        //atributo..
        private IAlunoRepositorio repositorio;

        public AlunoBusiness(IAlunoRepositorio repositorio)
        {
            this.repositorio = repositorio;
        }

        public void Cadastrar(Aluno a)
        {
            //gerando um numero de matricula..
            Random r = new Random();
            a.Matricula = DateTime.Now.Year + "-"
                + r.Next(1000, 999999999).ToString();

            repositorio.Insert(a);
        }

        public void Atualizar(Aluno a)
        {
            repositorio.Update(a); //atualizando..
        }

        public void Excluir(int idAluno)
        {
            Aluno a = repositorio.FindById(idAluno);
            repositorio.Delete(a); //excluindo..
        }

        public List<Aluno> ConsultarTodos()
        {
            return repositorio.FindAll();
        }

        public Aluno ObterPorId(int idAluno)
        {
            return repositorio.FindById(idAluno);
        }
    }
}
```

```
-----

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

```
using Projeto.Entidades;
using Projeto.BLL.Contracts;
using Projeto.DAL.Contracts;

namespace Projeto.BLL.Business
{
    public class TurmaBusiness : ITurmaBusiness
    {
        private ITurmaRepositorio repositorio;

        public TurmaBusiness(ITurmaRepositorio repositorio)
        {
            this.repositorio = repositorio;
        }

        public void Cadastrar(Turma t)
        {
            repositorio.Insert(t);
        }

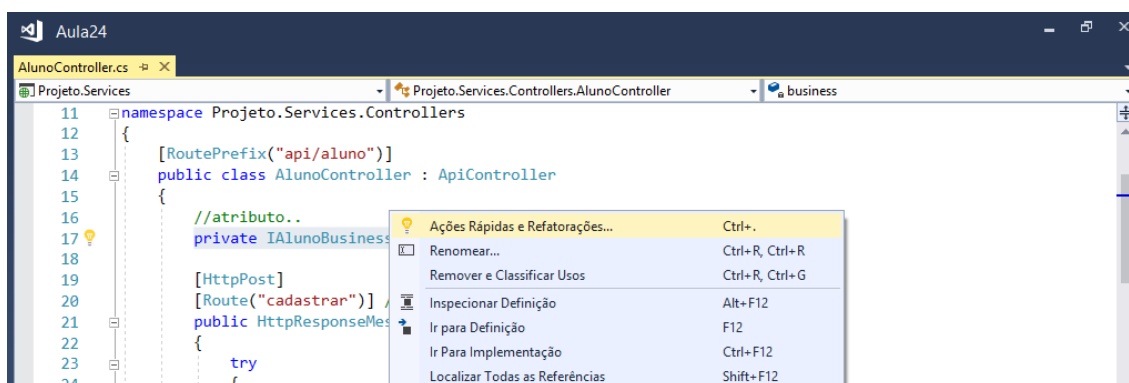
        public void Atualizar(Turma t)
        {
            repositorio.Update(t);
        }

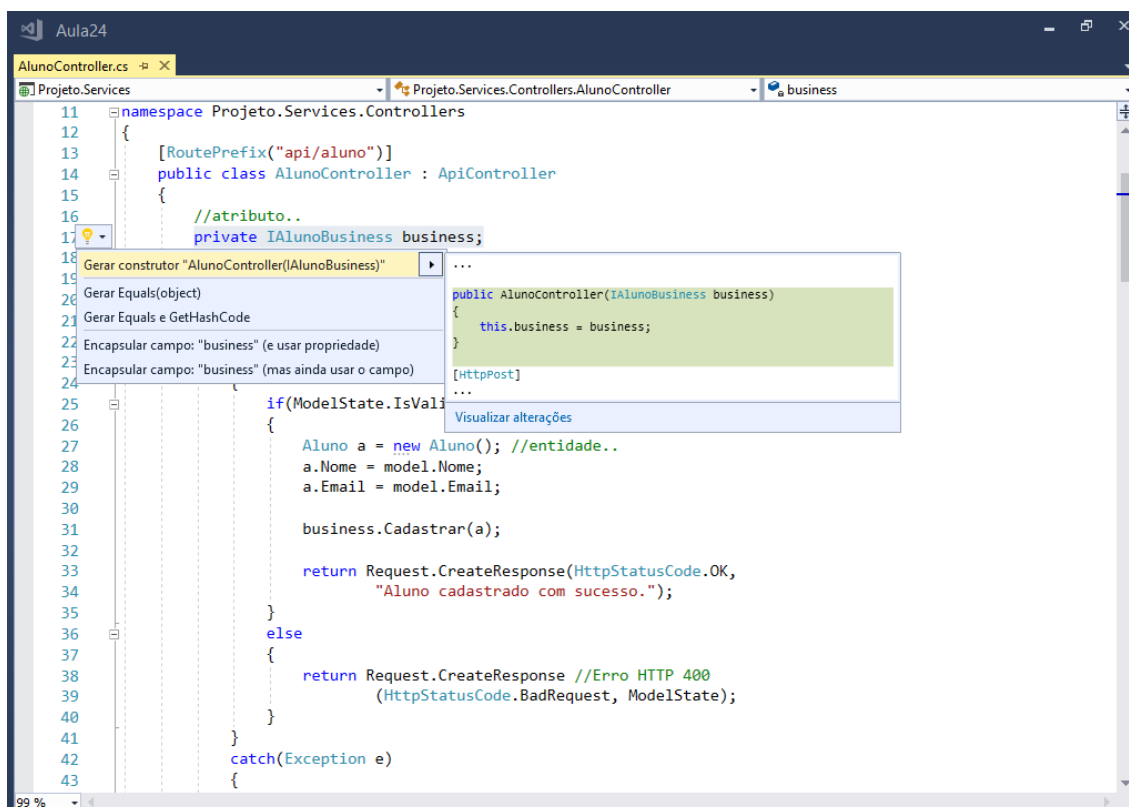
        public void Excluir(int idTurma)
        {
            Turma t = repositorio.FindById(idTurma);
            repositorio.Delete(t);
        }

        public List<Turma> ConsultarTodos()
        {
            return repositorio.FindAll();
        }

        public Turma ObterPorId(int idTurma)
        {
            return repositorio.FindById(idTurma);
        }
    }
}
```

No projeto WebApi:





```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Projeto.Services.Models;
using Projeto.BLL.Contracts;
using Projeto.Entidades;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/aluno")]
    public class AlunoController : ApiController
    {
        //atributo..
        private IAlunoBusiness business;

        public AlunoController(IAlunoBusiness business)
        {
            this.business = business;
        }

        [HttpPost]
        [Route("cadastrar")] //URL: /api/aluno/cadastrar
        public HttpResponseMessage Post(AlunoCadastroViewModel model)
        {
            try
            {
                if (ModelState.IsValid) //se passou nas regras de validação..
                {
                    Aluno a = new Aluno(); //entidade..
                    a.Nome = model.Nome;
                    a.Email = model.Email;

                    business.Cadastrar(a);

                    return Request.CreateResponse(HttpStatusCode.OK,
                        "Aluno cadastrado com sucesso.");
                }
                else
                {
                    return Request.CreateResponse //Erro HTTP 400
                        (HttpStatusCode.BadRequest, ModelState);
                }
            }
            catch (Exception e)
            {
            }
        }
    }
}
```

```

        Aluno a = new Aluno(); //entidade..
        a.Nome = model.Nome;
        a.Email = model.Email;

        business.Cadastrar(a);

        return Request.CreateResponse(HttpStatusCode.OK,
            "Aluno cadastrado com sucesso.");
    }
    else
    {
        return Request.CreateResponse //Erro HTTP 400
            (HttpStatusCode.BadRequest, ModelState);
    }
}
catch(Exception e)
{
    return Request.CreateResponse //Erro HTTP 500
        (HttpStatusCode.InternalServerError, e.Message);
}
}

[HttpPut]
[Route("atualizar")] //URL: /api/aluno/atualizar
public HttpResponseMessage Put(AlunoEdicaoViewModel model)
{
    try
    {
        if(ModelState.IsValid)
        {
            //buscar o aluno pelo id..
            Aluno a = business.ObterPorId(model.IdAluno);

            a.Nome = model.Nome;
            a.Email = model.Email;

            business.Atualizar(a); //atualizando..

            return Request.CreateResponse(HttpStatusCode.OK,
                "Aluno atualizado com sucesso.");
        }
        else
        {
            return Request.CreateResponse
                (HttpStatusCode.BadRequest, ModelState);
        }
    }
    catch (Exception e)
    {
        return Request.CreateResponse
            (HttpStatusCode.InternalServerError, e.Message);
    }
}

[HttpDelete]
[Route("excluir")] //URL: /api/aluno/excluir?id={0}
public HttpResponseMessage Delete(int id)
{

```

```
try
{
    business.Excluir(id);

    return Request.CreateResponse(HttpStatusCode.OK,
        "Aluno excluído com sucesso.");
}
catch (Exception e)
{
    return Request.CreateResponse
        (HttpStatusCode.InternalServerError, e.Message);
}
}

[HttpGet]
[Route("consultar")] //URL: /api/aluno/consultar
public HttpResponseMessage GetAll()
{
    try
    {
        List<AlunoConsultaViewModel> lista
            = new List<AlunoConsultaViewModel>();

        foreach (Aluno a in business.ConsultarTodos())
        {
            AlunoConsultaViewModel model = new AlunoConsultaViewModel();
            model.IdAluno = a.IdAluno;
            model.Nome = a.Nome;
            model.Email = a.Email;
            model.Matricula = a.Matricula;

            lista.Add(model); //adicionar na lista..
        }

        return Request.CreateResponse(HttpStatusCode.OK, lista);
    }
    catch (Exception e)
    {
        return Request.CreateResponse
            (HttpStatusCode.InternalServerError, e.Message);
    }
}

[HttpGet]
[Route("obter")] //URL: /api/aluno/obter?id={0}
public HttpResponseMessage GetById(int id)
{
    try
    {
        Aluno a = business.ObterPorId(id);

        if (a != null) //se o aluno foi encontrado..
        {
            AlunoConsultaViewModel model = new AlunoConsultaViewModel();
            model.IdAluno = a.IdAluno;
            model.Nome = a.Nome;
            model.Email = a.Email;
            model.Matricula = a.Matricula;
        }
    }
}
```

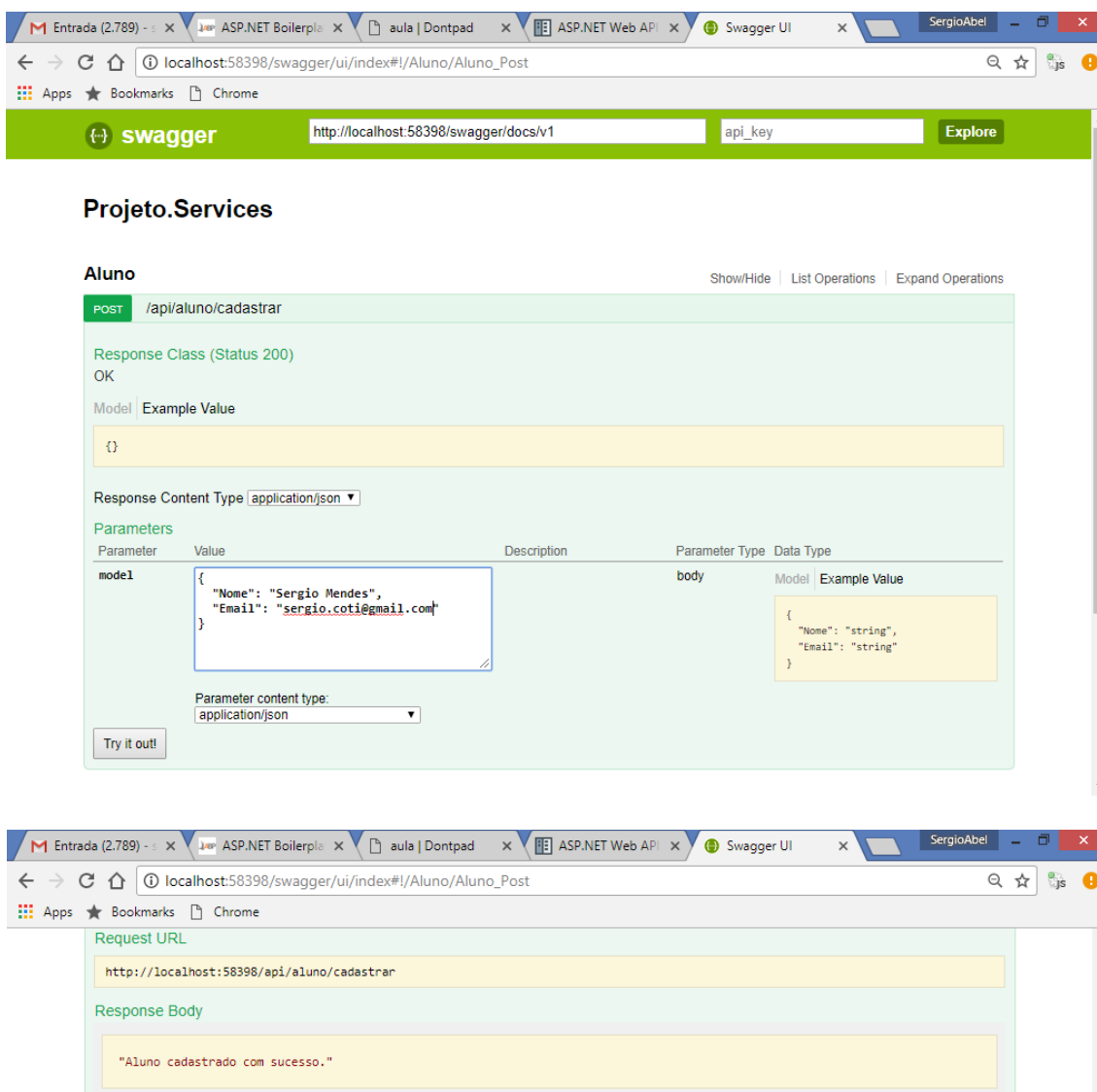


```

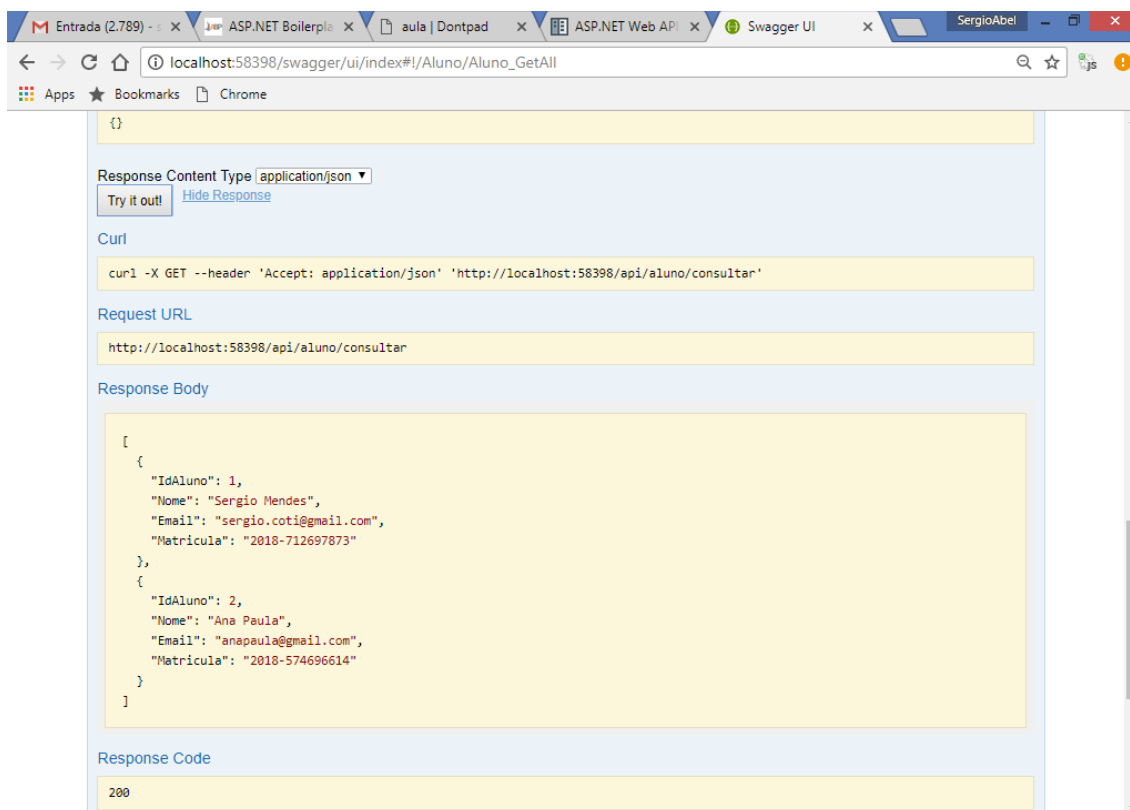
        return Request.CreateResponse(HttpStatusCode.OK, model);
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.BadRequest,
            "Aluno não encontrado");
    }
}
catch (Exception e)
{
    return Request.CreateResponse
        (HttpStatusCode.InternalServerError, e.Message);
}
}
}
}

```

Testando:



The screenshot shows the Swagger UI interface in a web browser. The top bar indicates the URL is `localhost:58398/swagger/ui/index#/Aluno/Aluno_Post`. The main content area displays the **Projeto.Services** API with the **Aluno** endpoint. The endpoint is a **POST** request to `/api/aluno/cadastrar`. The response class is **Status 200 OK**. The response content type is `application/json`. The parameters section shows a **body** parameter with a JSON example: `{ "Nome": "Sergio Mendes", "Email": "sergio.coti@gmail.com" }`. The response body section shows the JSON response: `"Aluno cadastrado com sucesso."`.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Projeto.Services.Models;
using Projeto.Entidades;
using Projeto.BLL.Contracts;

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/turma")]
    public class TurmaController : ApiController
    {
        //atributo..
        private ITurmaBusiness business;

        public TurmaController(ITurmaBusiness business)
        {
            this.business = business;
        }

        [HttpPost]
        [Route("cadastrar")] //URL: /api/turma/cadastrar
        public HttpResponseMessage Post(TurmaCadastroViewModel model)
        {
            try
            {
                if (ModelState.IsValid)
                {

```

```

        Turma t = new Turma();
        tCurso = model.Curso;
        t.DataInicio = model.DataInicio;
        t.DataTermino = model.DataTermino;

        business.Cadastrar(t);

        return Request.CreateResponse(HttpStatusCode.OK,
            "Turma cadastrada com sucesso");
    }
    else
    {
        return Request.CreateResponse
            (HttpStatusCode.BadRequest, ModelState);
    }
}
catch (Exception e)
{
    return Request.CreateResponse
        (HttpStatusCode.InternalServerError, e.Message);
}
}

[HttpPut]
[Route("atualizar")] //URL: /api/turma/atualizar
public HttpResponseMessage Put(TurmaEdicaoViewModel model)
{
    try
    {
        if (ModelState.IsValid)
        {
            Turma t = business.ObterPorId(model.IdTurma);

            tCurso = model.Curso;
            t.DataInicio = model.DataInicio;
            t.DataTermino = model.DataTermino;

            business.Atualizar(t);

            return Request.CreateResponse(HttpStatusCode.OK,
                "Turma atualizada com sucesso");
        }
        else
        {
            return Request.CreateResponse
                (HttpStatusCode.BadRequest, ModelState);
        }
    }
    catch (Exception e)
    {
        return Request.CreateResponse
            (HttpStatusCode.InternalServerError, e.Message);
    }
}

[HttpDelete]
[Route("excluir")] //URL: /api/turma/excluir?id={0}
public HttpResponseMessage Delete(int id)
{

```

```

        try
        {
            business.Excluir(id);

            return Request.CreateResponse(HttpStatusCode.OK,
                "Turma excluída com sucesso.");
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }

    [HttpGet]
    [Route("consultar")] //URL: /api/turma/consultar
    public HttpResponseMessage GetAll()
    {
        try
        {
            List<TurmaConsultaViewModel> lista
                = new List<TurmaConsultaViewModel>();

            foreach (Turma t in business.ConsultarTodos())
            {
                TurmaConsultaViewModel model = new TurmaConsultaViewModel();
                model.IdTurma = t.IdTurma;
                modelCurso = t.Curso;
                model.DataInicio = t.DataInicio;
                model.DataTermino = t.DataTermino;

                lista.Add(model); //adicionar na lista..
            }

            return Request.CreateResponse(HttpStatusCode.OK, lista);
        }
        catch (Exception e)
        {
            return Request.CreateResponse
                (HttpStatusCode.InternalServerError, e.Message);
        }
    }

    [HttpGet]
    [Route("obter")] //URL: /api/turma/obter?id={0}
    public HttpResponseMessage GetById(int id)
    {
        try
        {
            Turma t = business.ObterPorId(id);

            if (t != null) //se o aluno foi encontrado..
            {
                TurmaConsultaViewModel model = new TurmaConsultaViewModel();
                model.IdTurma = t.IdTurma;
                modelCurso = t.Curso;
                model.DataInicio = t.DataInicio;
                model.DataTermino = t.DataTermino;
            }
        }
    }

```

```
        return Request.CreateResponse(HttpStatusCode.OK, model);
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.BadRequest,
            "Turma não encontrada");
    }
}
catch (Exception e)
{
    return Request.CreateResponse
        (HttpStatusCode.InternalServerError, e.Message);
}
}
}
```
