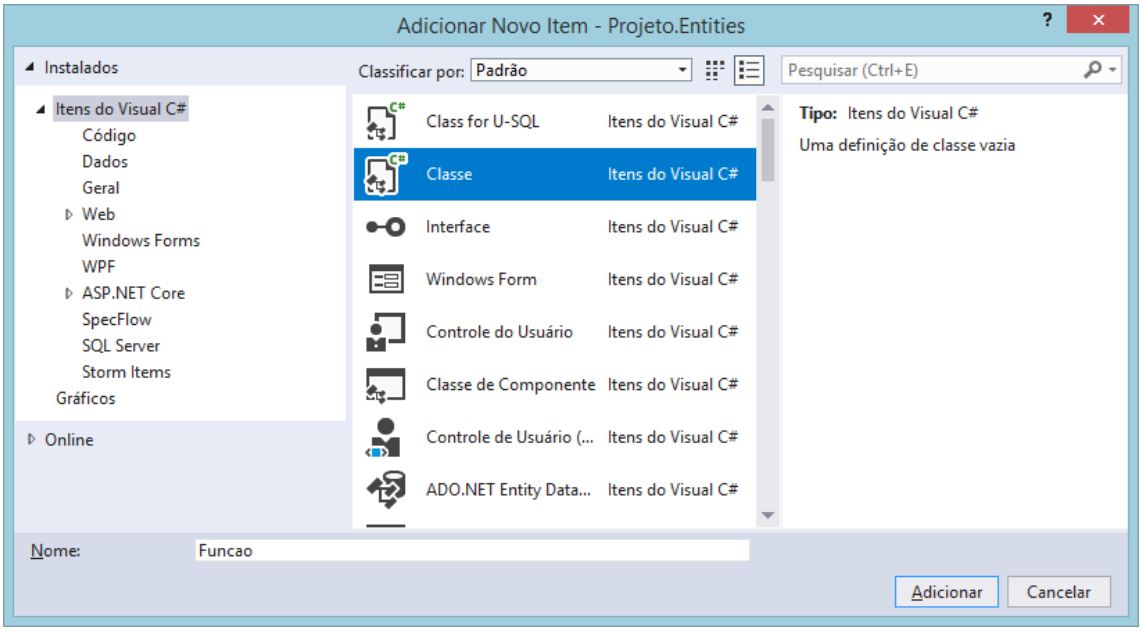
Adicionando uma nova entidade no projeto:

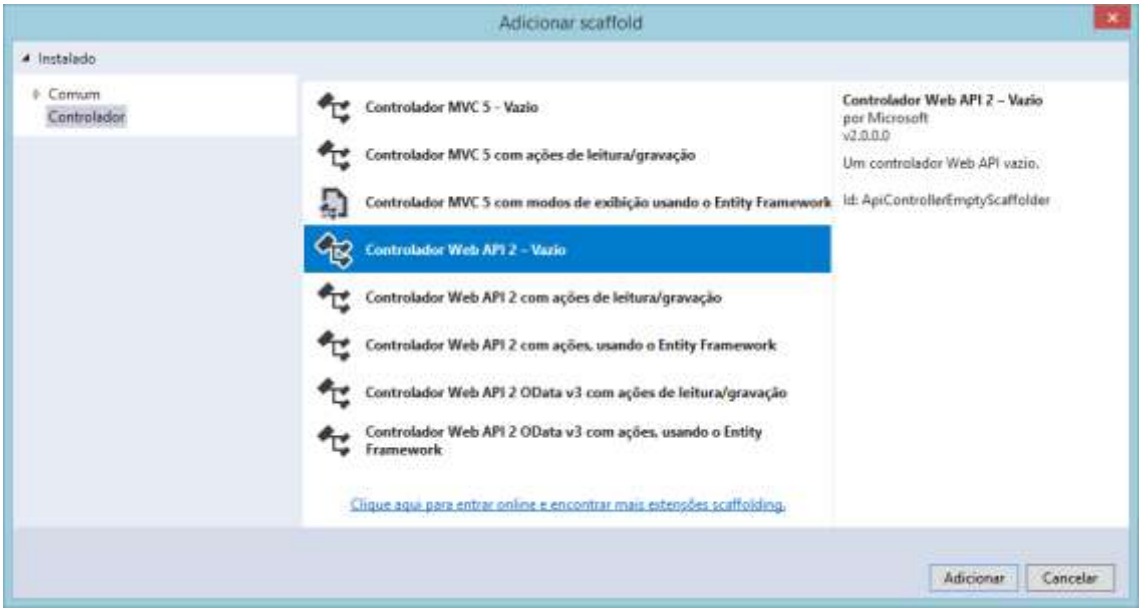
Primeiro: Iremos criar as classes de entidade e seus relacionamentos:

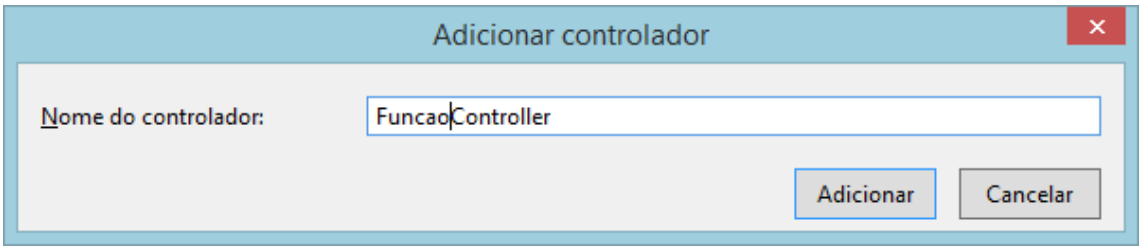



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.Entities
{
```

```
    public class Funcao
    {
        public int IdFuncao { get; set; }
        public string Nome { get; set; }

        //Relacionamento TER-MUITOS
        public List<Funcionario> Funcionarios { get; set; }
    }
}

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

namespace Projeto.Entities
{
    public class Funcionario
    {
        public int IdFuncionario { get; set; }
        public string Nome { get; set; }
        public decimal Salario { get; set; }
        public DateTime DataAdmissao { get; set; }

        //Relacionamento -> TEM MUITOS Dependentes
        public List<Dependente> Dependentes { get; set; }

        //Relacionamento -> TEM MUITAS Funções
        public List<Funcao> Funcoes { get; set; }
    }
}

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

namespace Projeto.Entities
{
    public class Dependente
    {
        public int IdDependente { get; set; }
        public string Nome { get; set; }
        public DateTime DataNascimento { get; set; }

        //Relacionamento -> TEM 1 Funcionário
        public Funcionario Funcionario { get; set; }
    }
}
```

Criando um novo controlador na API para Função
ENDPOINT: **/api/Funcao**






```
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/Funcao")]
    public class FuncaoController : ApiController
    {

    }
}
```

Criando as classes de modelo:

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

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

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

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

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

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

namespace Projeto.Services.Models
{
    public class FuncaoConsultaViewModel
    {
        public int IdFuncao { get; set; }
        public string Nome { get; set; }
    }
}
```

Realizando os mapeamentos do AutoMapper:

/Mappings/EntityToViewModelMap.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AutoMapper; //importando
using Projeto.Services.Models; //importando
using Projeto.Entities; //importando

namespace Projeto.Services.Mappings
{
    //Regra 1) HERDAR Profile
    public class EntityToViewModelMap : Profile
    {
        //Regra 2) Construtor -> ctor + 2x[tab]
        public EntityToViewModelMap()
        {
            CreateMap<Dependente, DependenteConsultaViewModel>();
            CreateMap<Funcionario, FuncionarioConsultaViewModel>();
            CreateMap<Funcao, FuncaoConsultaViewModel>();
        }
    }
}
```

/Mappings/ViewModelToEntityMap.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using AutoMapper; //importando
using Projeto.Services.Models; //importando
using Projeto.Entities; //importando

namespace Projeto.Services.Mappings
{
    //Regra 1) HERDAR Profile
    public class ViewModelToEntityMap : Profile
    {
        //Regra 2) Construtor -> ctor + 2x[tab]
        public ViewModelToEntityMap()
        {
            CreateMap<DependenteCadastroViewModel, Dependente>();
            CreateMap<DependenteEdicaoViewModel, Dependente>();

            CreateMap<FuncionarioCadastroViewModel, Funcionario>();
            CreateMap<FuncionarioEdicaoViewModel, Funcionario>();

            CreateMap<FuncaoCadastroViewModel, Funcao>();
            CreateMap<FuncaoEdicaoViewModel, Funcao>();
        }
    }
}
```

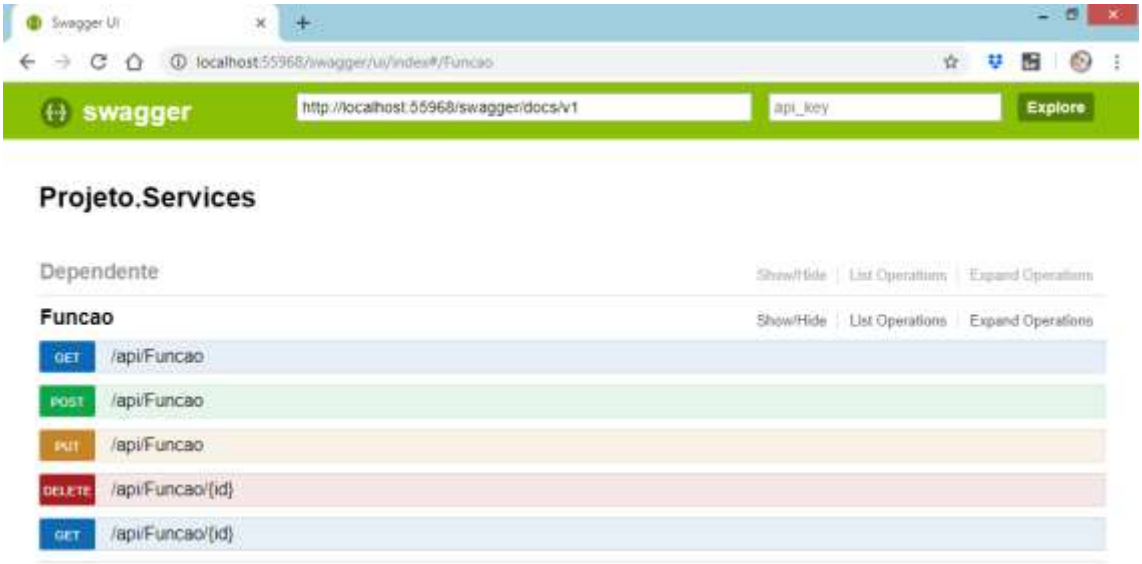
Criando o controller:
/Controllers/FuncaoController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Projeto.Entities; //importando
using Projeto.Services.Models; //importando
using AutoMapper; //importando

namespace Projeto.Services.Controllers
{
    [RoutePrefix("api/Funcao")]
    public class FuncaoController : ApiController
    {
        [HttpPost]
        public HttpResponseMessage Post(FuncaoCadastroViewModel model)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    //transferir os dados da model para entidade
                    var funcao = Mapper.Map<Funcao>(model);
                    //TODO..

                    return Request.CreateResponse(HttpStatusCode.OK,
                        $"Função {model.Nome}, cadastrado com sucesso.");
                }
                catch (Exception e)
                {
                    //erro HTTP 500 -> INTERNAL SERVER ERROR
                    return Request.CreateResponse
                        (HttpStatusCode.InternalServerError,
                        "Erro interno de servidor: " + e.Message);
                }
            }
            else
            {
                //erro HTTP 400 -> BAD REQUEST
                return Request.CreateResponse(HttpStatusCode.BadRequest,
                                    "Ocorreram erros de validação.");
            }
        }

        [HttpPut]
        public HttpResponseMessage Put(FuncaoEdicaoViewModel model)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    //transferir os dados da model para entidade
                    var funcao = Mapper.Map<Funcao>(model);
                    //TODO..
```

```
                return Request.CreateResponse(HttpStatusCode.OK,
                    $"Função {model.Nome}, atualizado com sucesso.");
            }
            catch (Exception e)
            {
                //erro HTTP 500 -> INTERNAL SERVER ERROR
                return Request.CreateResponse
                    (HttpStatusCode.InternalServerError,
                    "Erro interno de servidor: " + e.Message);
            }
        }
        else
        {
            //erro HTTP 400 -> BAD REQUEST
            return Request.CreateResponse(HttpStatusCode.BadRequest,
                                "Ocorreram erros de validação.");
        }
    }

    [HttpDelete]
    public HttpResponseMessage Delete(int id)
    {
        try
        {
            //TODO..

            return Request.CreateResponse(HttpStatusCode.OK,
                        "Função excluído com sucesso.");
        }
        catch (Exception e)
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }

    [HttpGet]
    public HttpResponseMessage GetAll()
    {
        try
        {
            //TODO..
            return Request.CreateResponse(HttpStatusCode.OK);
        }
        catch (Exception e)
        {
            return Request.CreateResponse(HttpStatusCode.InternalServerError,
                "Erro interno de servidor: " + e.Message);
        }
    }

    [HttpGet]
    public HttpResponseMessage GetById(int id)
    {
        try
        {
            //TODO..
            return Request.CreateResponse(HttpStatusCode.OK);
        }
```

```
            catch (Exception e)
            {
                return Request.CreateResponse(HttpStatusCode.InternalServerError,
                    "Erro interno de servidor: " + e.Message);
            }
        }
    }
}
```

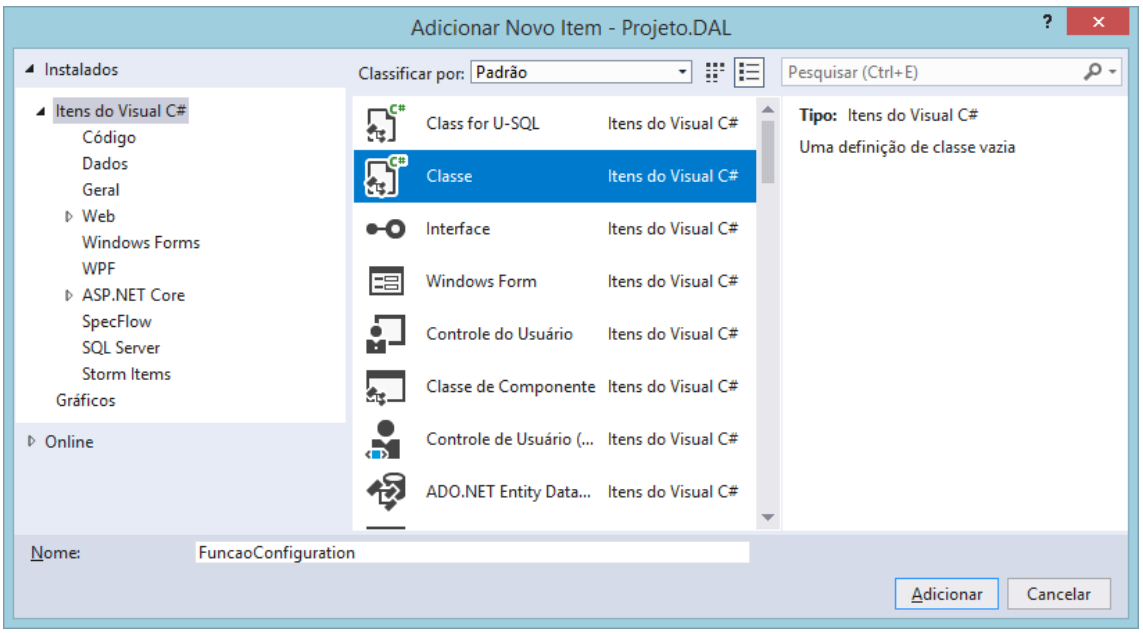
Executando a API:



Mapeando a entidade Funcao no EntityFramework
/Mappings/FuncaoMapping.cs

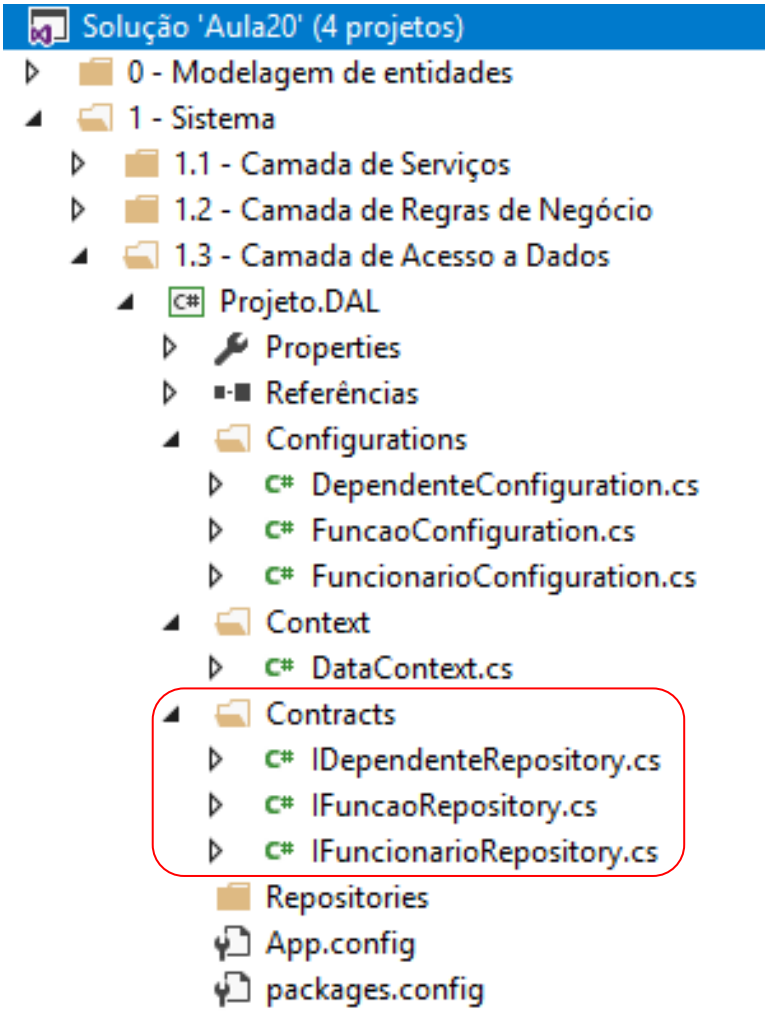
ORM - Object Relational Mapping

Mapeamento Objeto / Relacional



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

namespace Projeto.DAL.Configurations
{
    //classe de mapeamento para a entidade Funcao
    public class FuncaoConfiguration
        : EntityTypeConfiguration<Funcao>
    {
        //construtor -> ctor + 2x[tab]
        public FuncaoConfiguration()
        {
            //nome da tabela
            ToTable("FUNCAO");

            //chave primária
            HasKey(f => new { f.IdFuncao });

            //mapear os campos
            Property(f => f.IdFuncao)
                .HasColumnName("IDFUNCAO");

            Property(f => f.Nome)
                .HasColumnName("NOME")
                .HasMaxLength(150)
                .IsRequired();

            //mapeamento do relacionamento..
            //muitos para muitos
            HasMany(f => f.Funcionarios)
                .WithMany(f => f.Funcoes)
                .Map(map =>
                {
                    //nome da tabela associativa
                    map.ToTable("FUNCAOFUNCIONARIO");

                    //chave estrangeira para a entidade 'Funcao'
                    map.MapLeftKey("IDFUNCAO");

                    //chave estrangeira para a entidade 'Funcionario'
                    map.MapRightKey("IDFUNCIONARIO");
                });
        }

    }

}
```

Criando interfaces (Contratos) para todas as classes de repositorio que serão implementadas no projeto DAL



Criando uma interface genérica:

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

namespace Projeto.DAL.Contracts
{
    //<T> Tipo de dado genérico
    public interface IBaseRepository<T> where T : class
    {
        void Insert(T obj);
        void Update(T obj);
        void Remove(T obj);

        List<T> FindAll();
        T FindById(int id);
    }
}
```

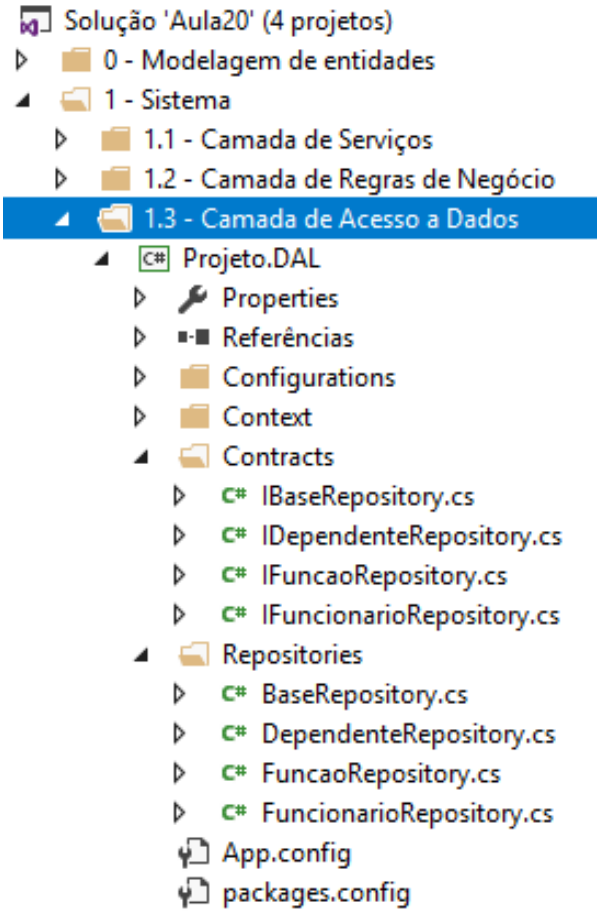
Herdando a interface genérica:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando

namespace Projeto.DAL.Contracts
{
    public interface IDependenteRepository
        : IBaseRepository<Dependente>
    {

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando

namespace Projeto.DAL.Contracts
{
    public interface IFuncaoRepository
        : IBaseRepository<Funcao>
    {

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando

namespace Projeto.DAL.Contracts
{
    public interface IFuncionarioRepository
        : IBaseRepository<Funcionario>
    {

    }
}
```

Criando as classes para implementar cada interface:



Continua...