Aplicação

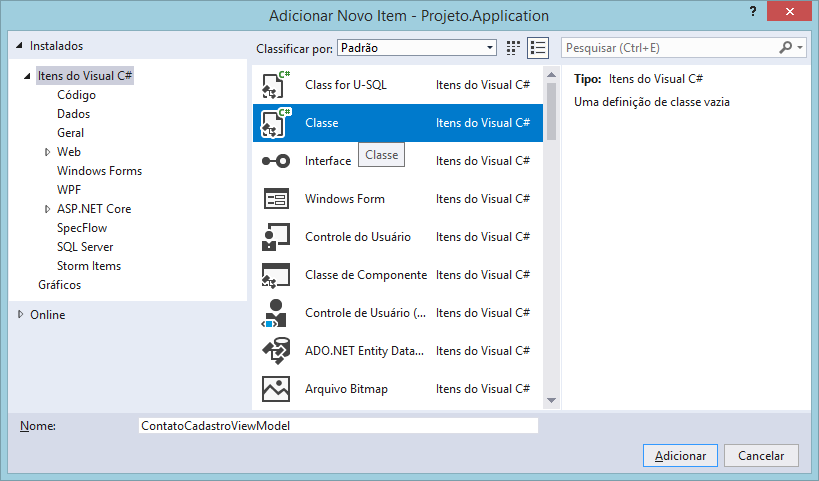
Consiste de uma "camada" entre

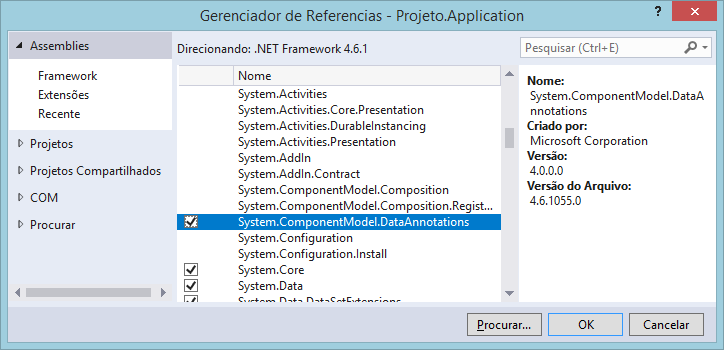
o Dominio e a Apresentação do sistema.

ViewModels

Classes utilizadas para definir os dados de

entrada / saida para cada operação da aplicação.





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations; //importando

namespace Projeto.Application.ViewModels

{

public class ContatoCadastroViewModel

{

[Required(ErrorMessage = "Campo obrigatório.")]

public string Nome { get; set; }

[EmailAddress(ErrorMessage = "Email inválido.")]

[Required(ErrorMessage = "Campo obrigatório.")]

public string Email { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Telefone { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Application.ViewModels

{

public class ContatoEdicaoViewModel

{

[Required(ErrorMessage = "Campo obrigatório.")]

public int IdContato { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Nome { get; set; }

[EmailAddress(ErrorMessage = "Email inválido.")]

[Required(ErrorMessage = "Campo obrigatório.")]

public string Email { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Telefone { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Application.ViewModels

{

public class ContatoConsultaViewModel

{

public int IdContato { get; set; }

public string Nome { get; set; }

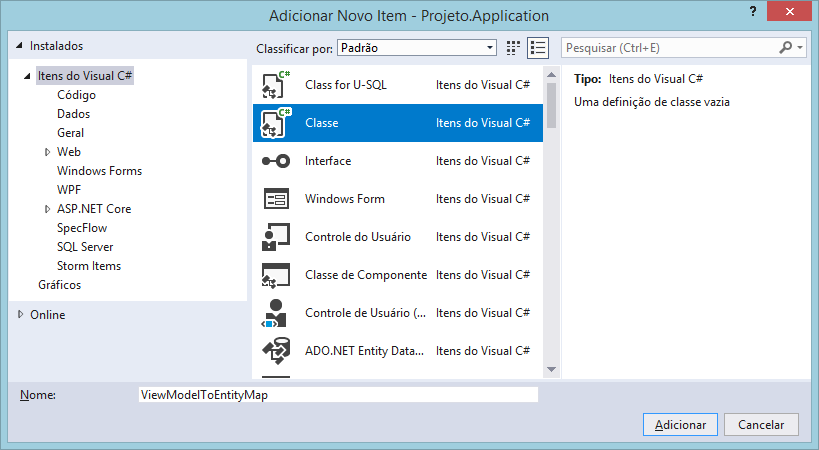
public string Email { get; set; }

public string Telefone { get; set; }

}

}

**Mapeando as trocas de dados para   
Contato atraves do AutoMapper:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Mappings

{

public class ViewModelToEntityMap : Profile

{

//ctor + 2x[tab]

public ViewModelToEntityMap()

{

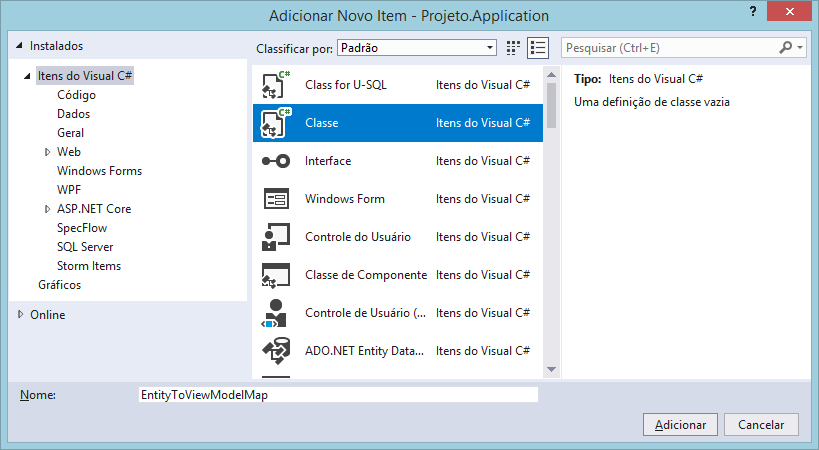
CreateMap<ContatoCadastroViewModel, Contato>();

CreateMap<ContatoEdicaoViewModel, Contato>();

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Mappings

{

public class EntityToViewModelMap : Profile

{

//ctor + 2x[tab]

public EntityToViewModelMap()

{

CreateMap<Contato, ContatoConsultaViewModel>();

}

}

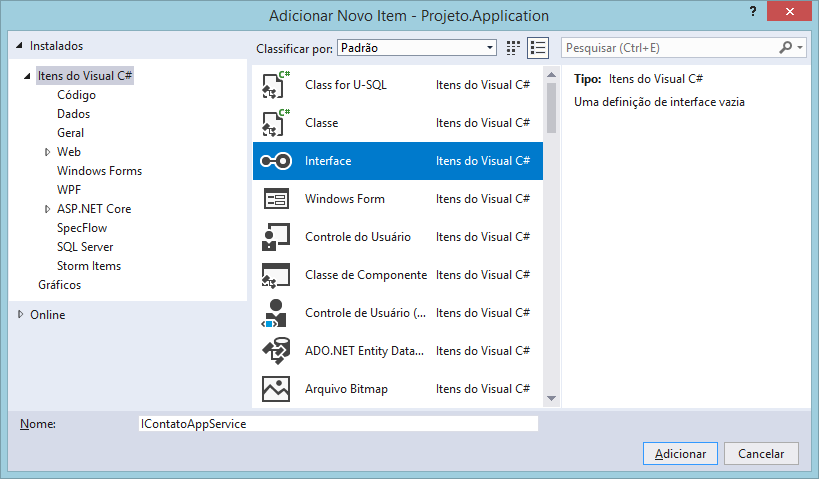
}

Application Services

Classes da "camada" de aplicação que irão entregar

para a Apresentação as operações de cada entidade.

**Primeiro: Criando o contrato:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Contracts

{

public interface IContatoAppService

{

void Cadastrar(ContatoCadastroViewModel model);

void Atualizar(ContatoEdicaoViewModel model);

void Excluir(int idContato);

List<ContatoConsultaViewModel> ConsultarTodos();

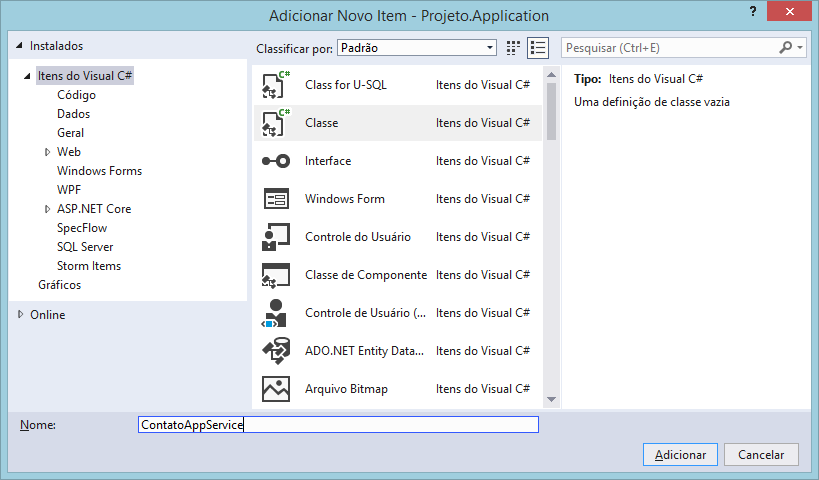
ContatoConsultaViewModel ConsultarPorId(int idContato);

ContatoConsultaViewModel ConsultarPorEmail(string email);

}

}

**Primeiro: Implementando o contrato:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Domain.Contracts.Services;

using Projeto.Application.ViewModels;

using Projeto.Application.Contracts;

namespace Projeto.Application.Services

{

public class ContatoAppService : IContatoAppService

{

private readonly IContatoDomainService domainService;

public ContatoAppService(IContatoDomainService domainService)

{

this.domainService = domainService;

}

public void Cadastrar(ContatoCadastroViewModel model)

{

var contato = Mapper.Map<Contato>(model);

domainService.Cadastrar(contato);

}

public void Atualizar(ContatoEdicaoViewModel model)

{

var contato = Mapper.Map<Contato>(model);

domainService.Atualizar(contato);

}

public void Excluir(int idContato)

{

var contato = domainService.ConsultarPorId(idContato);

domainService.Excluir(contato);

}

public List<ContatoConsultaViewModel> ConsultarTodos()

{

var lista = domainService.ConsultarTodos();

return Mapper.Map<List<ContatoConsultaViewModel>>(lista);

}

public ContatoConsultaViewModel ConsultarPorId(int idContato)

{

var contato = domainService.ConsultarPorId(idContato);

return Mapper.Map<ContatoConsultaViewModel>(contato);

}

public ContatoConsultaViewModel ConsultarPorEmail(string email)

{

var contato = domainService.ConsultarPorEmail(email);

return Mapper.Map<ContatoConsultaViewModel>(contato);

}

}

}

**Criando as classes ViewModel para endereco:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Application.ViewModels

{

public class EnderecoCadastroViewModel

{

[Required(ErrorMessage = "Campo obrigatório.")]

public string Logradouro { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Bairro { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Cidade { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Estado { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Cep { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Application.ViewModels

{

public class EnderecoEdicaoViewModel

{

[Required(ErrorMessage = "Campo obrigatório.")]

public int IdEndereco { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Logradouro { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Bairro { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Cidade { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Estado { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Cep { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Application.ViewModels

{

public class EnderecoConsultaViewModel

{

public int IdEndereco { get; set; }

public string Logradouro { get; set; }

public string Bairro { get; set; }

public string Cidade { get; set; }

public string Estado { get; set; }

public string Cep { get; set; }

}

}

**Mapeamentos do AutoMapper:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Mappings

{

public class ViewModelToEntityMap : Profile

{

//ctor + 2x[tab]

public ViewModelToEntityMap()

{

CreateMap<ContatoCadastroViewModel, Contato>();

CreateMap<ContatoEdicaoViewModel, Contato>();

CreateMap<EnderecoCadastroViewModel, Endereco>();

CreateMap<EnderecoEdicaoViewModel, Endereco>();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Mappings

{

public class EntityToViewModelMap : Profile

{

//ctor + 2x[tab]

public EntityToViewModelMap()

{

CreateMap<Contato, ContatoConsultaViewModel>();

CreateMap<Endereco, EnderecoConsultaViewModel>();

}

}

}

**Criando os serviços da aplicação:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Contracts

{

public interface IEnderecoAppService

{

void Cadastrar(EnderecoCadastroViewModel model);

void Atualizar(EnderecoEdicaoViewModel model);

void Excluir(int idEndereco);

List<EnderecoConsultaViewModel> ConsultarTodos();

EnderecoConsultaViewModel ConsultarPorId(int idEndereco);

}

}

**Implementando a interface:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Domain.Contracts.Services;

using Projeto.Application.ViewModels;

using Projeto.Application.Contracts;

namespace Projeto.Application.Services

{

public class EnderecoAppService : IEnderecoAppService

{

//atributo

private readonly IEnderecoDomainService domainService;

//construtor para injeção de dependência

public EnderecoAppService(IEnderecoDomainService domainService)

{

this.domainService = domainService;

}

public void Cadastrar(EnderecoCadastroViewModel model)

{

var endereco = Mapper.Map<Endereco>(model);

domainService.Cadastrar(endereco);

}

public void Atualizar(EnderecoEdicaoViewModel model)

{

var endereco = Mapper.Map<Endereco>(model);

domainService.Atualizar(endereco);

}

public void Excluir(int idEndereco)

{

var endereco = domainService.ConsultarPorId(idEndereco);

domainService.Excluir(endereco);

}

public List<EnderecoConsultaViewModel> ConsultarTodos()

{

var lista = domainService.ConsultarTodos();

return Mapper.Map<List<EnderecoConsultaViewModel>>(lista);

}

public EnderecoConsultaViewModel ConsultarPorId(int idEndereco)

{

var endereco = domainService.ConsultarPorId(idEndereco);

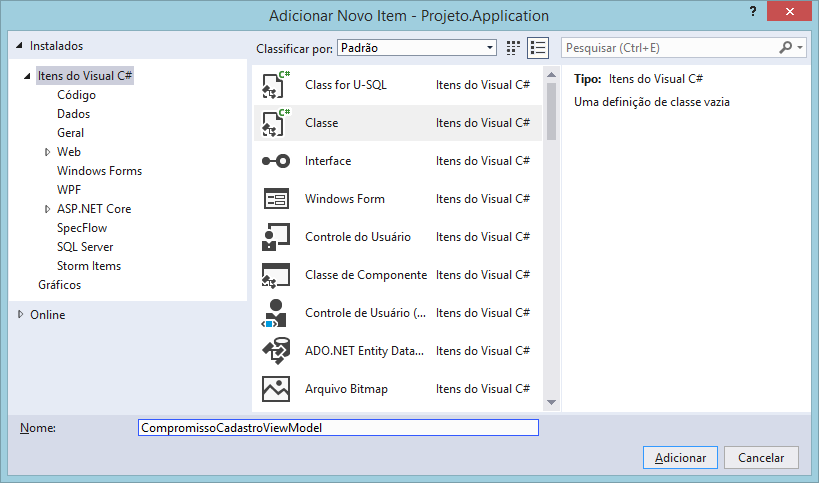
return Mapper.Map<EnderecoConsultaViewModel>(endereco);

}

}

}

**Criando as viewmodels para compromisso:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Application.ViewModels

{

public class CompromissoCadastroViewModel

{

[Required(ErrorMessage = "Campo obrigatório.")]

public DateTime DataHora { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Descricao { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Status { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public int IdContato { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Application.ViewModels

{

public class CompromissoEdicaoViewModel

{

[Required(ErrorMessage = "Campo obrigatório.")]

public int IdCompromisso { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public DateTime DataHora { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Descricao { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public string Status { get; set; }

[Required(ErrorMessage = "Campo obrigatório.")]

public int IdContato { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Application.ViewModels

{

public class CompromissoConsultaViewModel

{

public int IdCompromisso { get; set; }

public DateTime DataHora { get; set; }

public string Descricao { get; set; }

public string Status { get; set; }

public int IdContato { get; set; }

}

}

**Mapeando no AutoMapper:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Mappings

{

public class ViewModelToEntityMap : Profile

{

//ctor + 2x[tab]

public ViewModelToEntityMap()

{

CreateMap<ContatoCadastroViewModel, Contato>();

CreateMap<ContatoEdicaoViewModel, Contato>();

CreateMap<EnderecoCadastroViewModel, Endereco>();

CreateMap<EnderecoEdicaoViewModel, Endereco>();

CreateMap<CompromissoCadastroViewModel, Compromisso>();

CreateMap<CompromissoEdicaoViewModel, Compromisso>();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Mappings

{

public class EntityToViewModelMap : Profile

{

//ctor + 2x[tab]

public EntityToViewModelMap()

{

CreateMap<Contato, ContatoConsultaViewModel>();

CreateMap<Endereco, EnderecoConsultaViewModel>();

CreateMap<Compromisso, CompromissoConsultaViewModel>();

}

}

}

**Criando os contratos da Aplicação para Compromisso:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Application.ViewModels;

namespace Projeto.Application.Contracts

{

public interface ICompromissoAppService

{

void Cadastrar(CompromissoCadastroViewModel model);

void Atualizar(CompromissoEdicaoViewModel model);

void Excluir(int idCompromisso);

List<CompromissoConsultaViewModel> ConsultarTodos();

List<CompromissoConsultaViewModel> ConsultarPorDatas

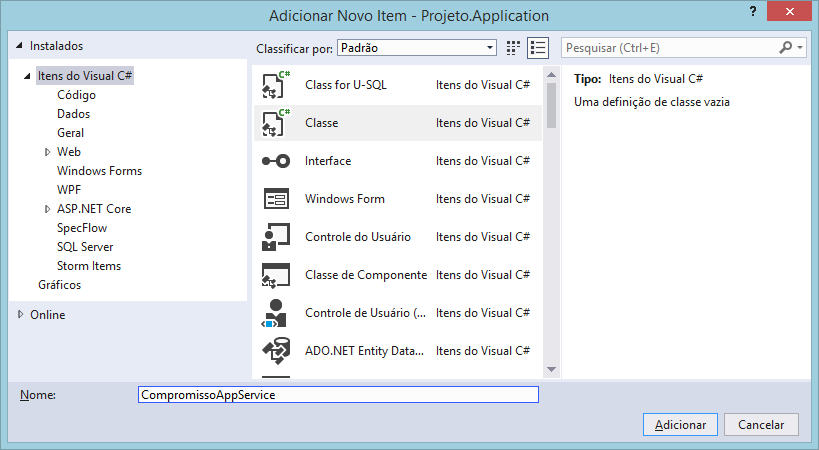
(DateTime dataInicio, DateTime dataFim);

CompromissoConsultaViewModel ConsultarPorId(int idCompromisso);

}

}

**Implementando:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutoMapper;

using Projeto.Domain.Entities;

using Projeto.Domain.Contracts.Services;

using Projeto.Application.ViewModels;

using Projeto.Application.Contracts;

namespace Projeto.Application.Services

{

public class CompromissoAppService : ICompromissoAppService

{

//atributo

private readonly ICompromissoDomainService domainService;

//construtor para injeção de dependência

public CompromissoAppService(ICompromissoDomainService domainService)

{

this.domainService = domainService;

}

public void Cadastrar(CompromissoCadastroViewModel model)

{

var compromisso = Mapper.Map<Compromisso>(model);

domainService.Cadastrar(compromisso);

}

public void Atualizar(CompromissoEdicaoViewModel model)

{

var compromisso = Mapper.Map<Compromisso>(model);

domainService.Atualizar(compromisso);

}

public void Excluir(int idCompromisso)

{

var compromisso = domainService.ConsultarPorId(idCompromisso);

domainService.Excluir(compromisso);

}

public List<CompromissoConsultaViewModel> ConsultarTodos()

{

var lista = domainService.ConsultarTodos();

return Mapper.Map<List<CompromissoConsultaViewModel>>(lista);

}

public List<CompromissoConsultaViewModel> ConsultarPorDatas

(DateTime dataInicio, DateTime dataFim)

{

var lista = domainService.ConsultarPorDatas(dataInicio, dataFim);

return Mapper.Map<List<CompromissoConsultaViewModel>>(lista);

}

public CompromissoConsultaViewModel ConsultarPorId(int idCompromisso)

{

var compromisso = domainService.ConsultarPorId(idCompromisso);

return Mapper.Map<CompromissoConsultaViewModel>(compromisso);

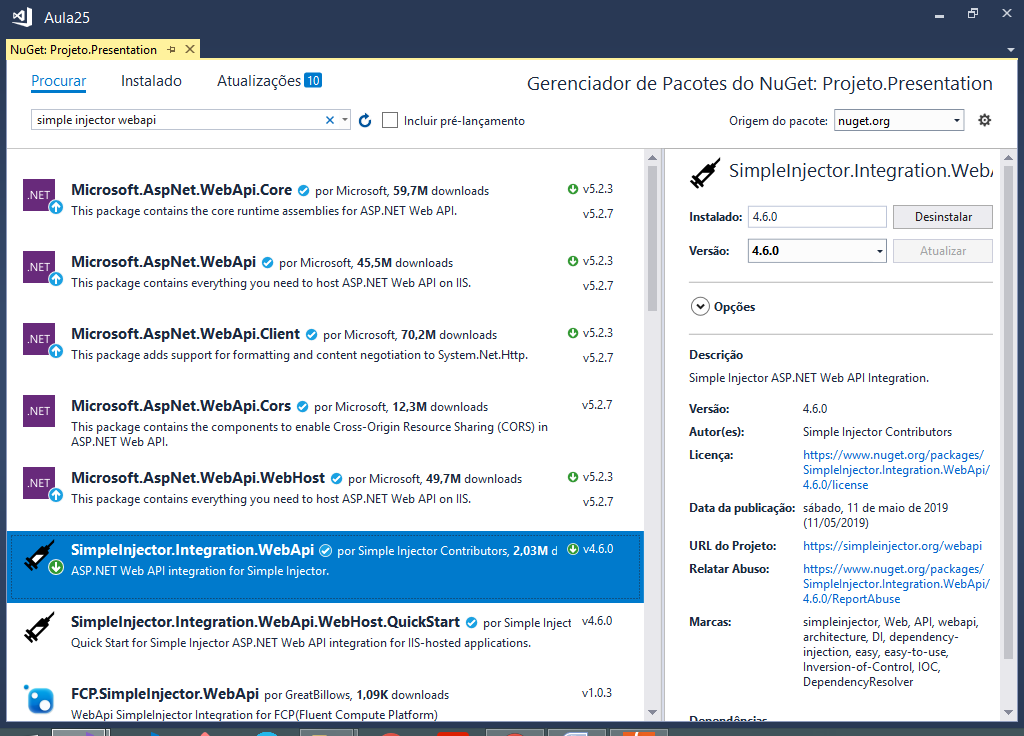
}

}

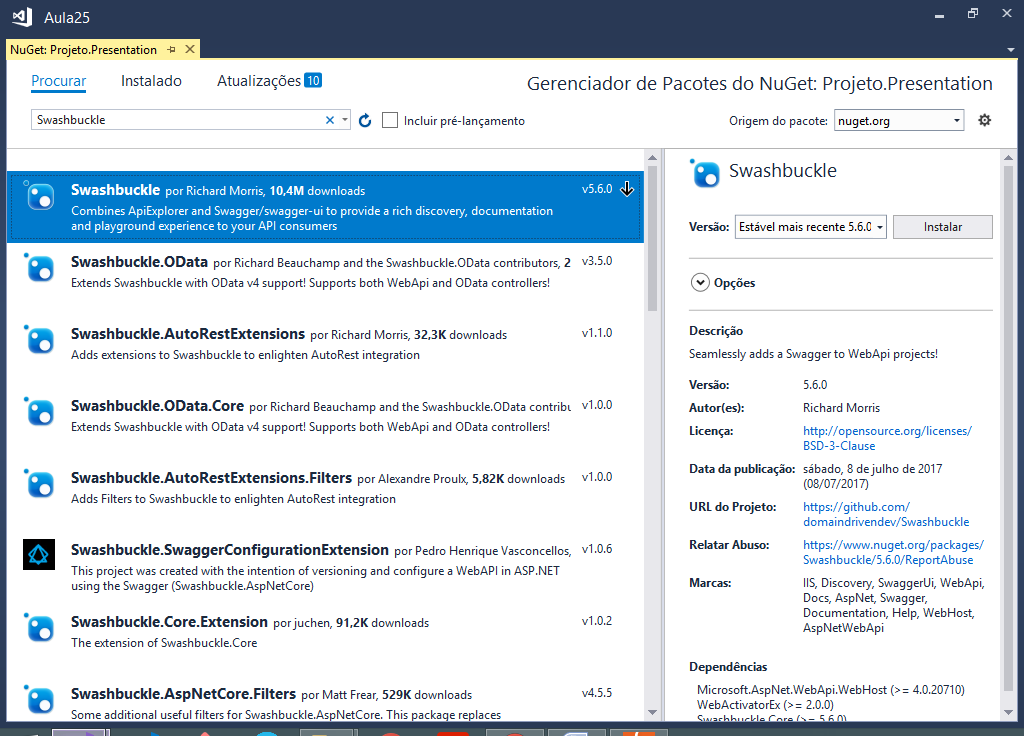
}

**Instalando o Simple Injector no projeto Asp.Net WebApi**

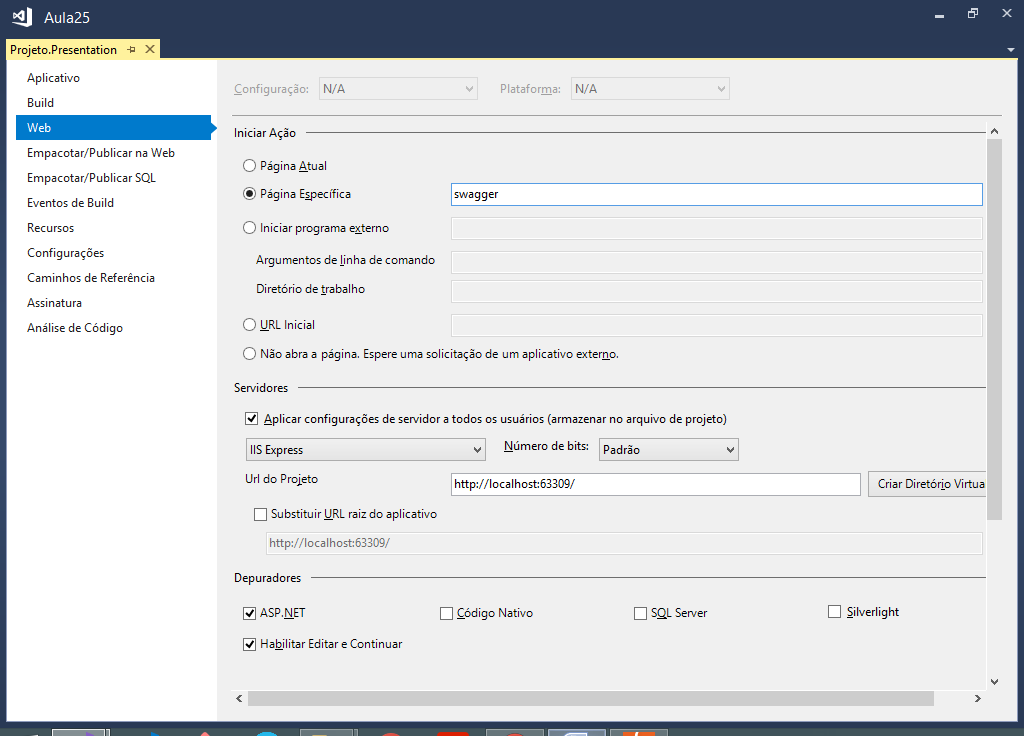
Gerenciador de pacotes do NuGet



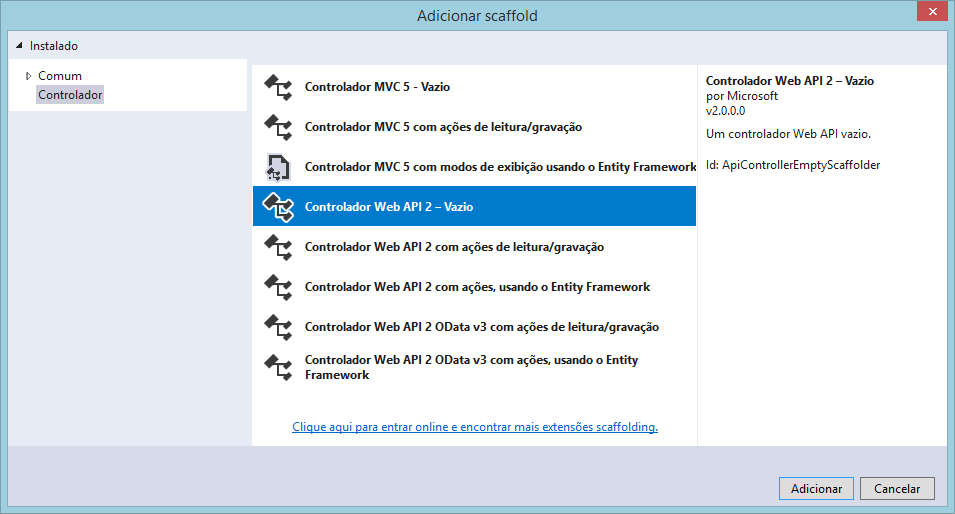
Instalando o Swagger (**SWASHBUCKLE**)

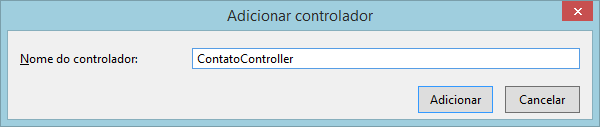


Configurando a página inicial do projeto:



**Criando os controllers da API:**





using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Projeto.Presentation.Controllers

{

[RoutePrefix("api/Contato")] //ENDPOINT

public class ContatoController : ApiController

{

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Projeto.Presentation.Controllers

{

[RoutePrefix("api/Endereco")]

public class EnderecoController : ApiController

{

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Projeto.Presentation.Controllers

{

[RoutePrefix("api/Compromisso")]

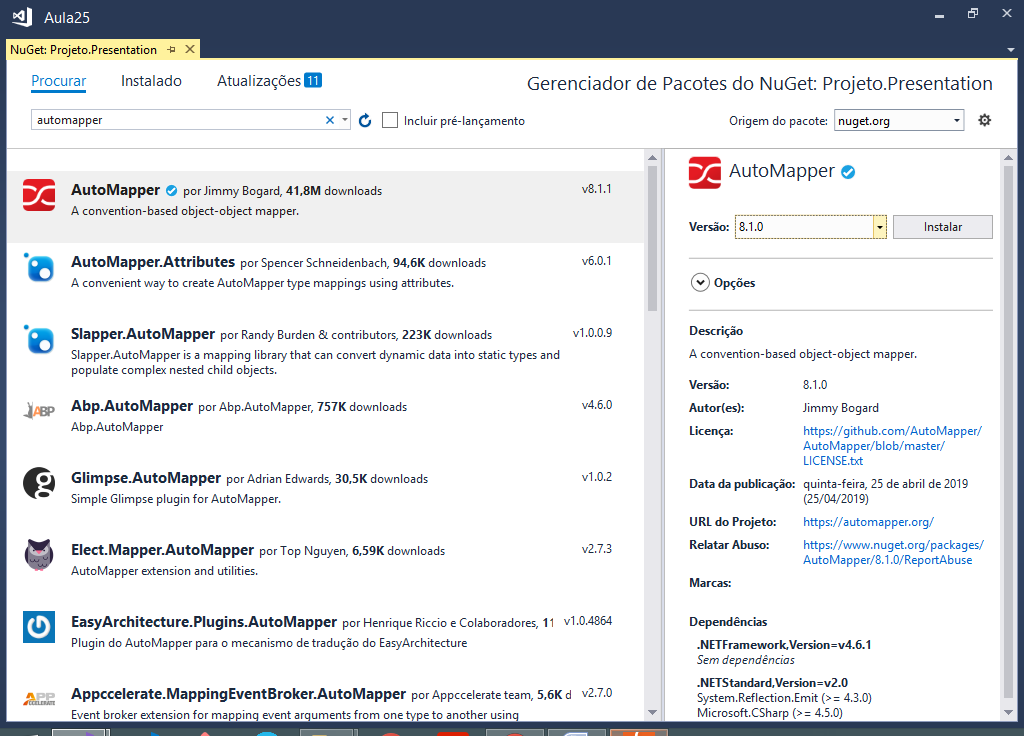
public class CompromissoController : ApiController

{

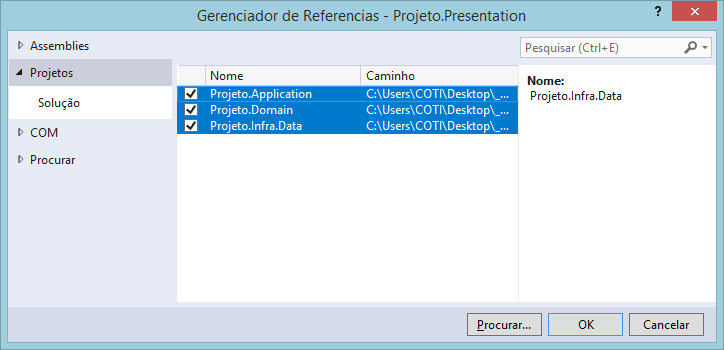
}

}

**Instalando o AutoMapper no projeto Presentation:**



**Adicionando referencias no projeto Presentation:**



Global.asax

* Configurando o AutoMapper
* Configurando o SimpleInjector

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Http;

using System.Web.Routing;

using AutoMapper;

using Projeto.Application.Mappings;

namespace Projeto.Presentation

{

public class WebApiApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

**//configurando o AutoMapper**

**Mapper.Initialize(cfg =>**

**{**

**cfg.AddProfile<EntityToViewModelMap>();**

**cfg.AddProfile<ViewModelToEntityMap>();**

**});**

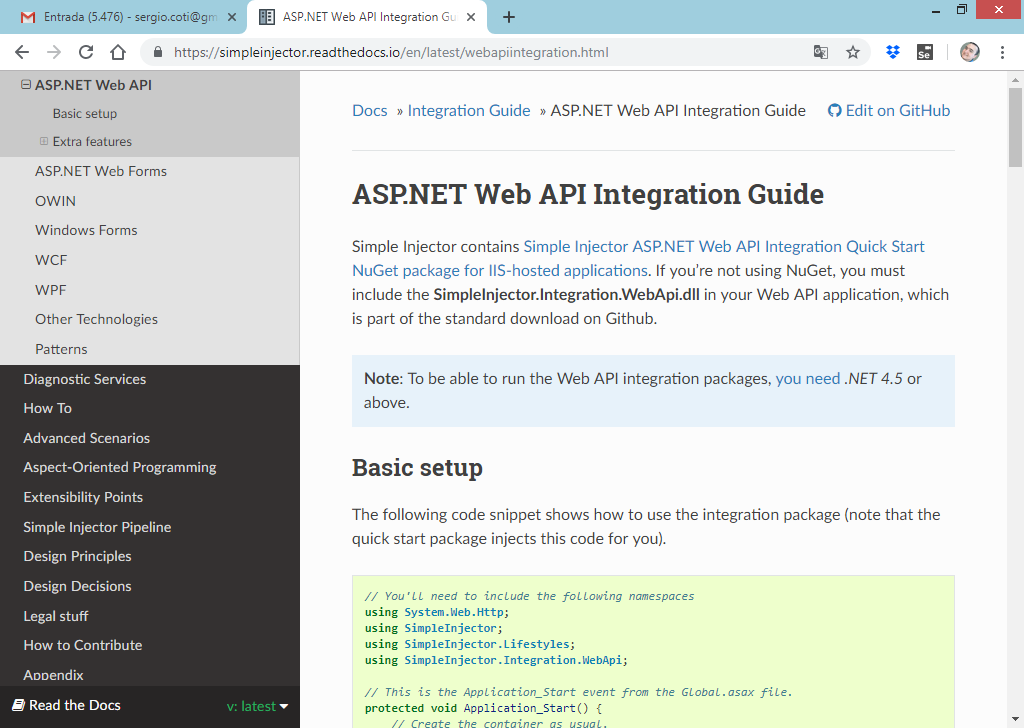
}

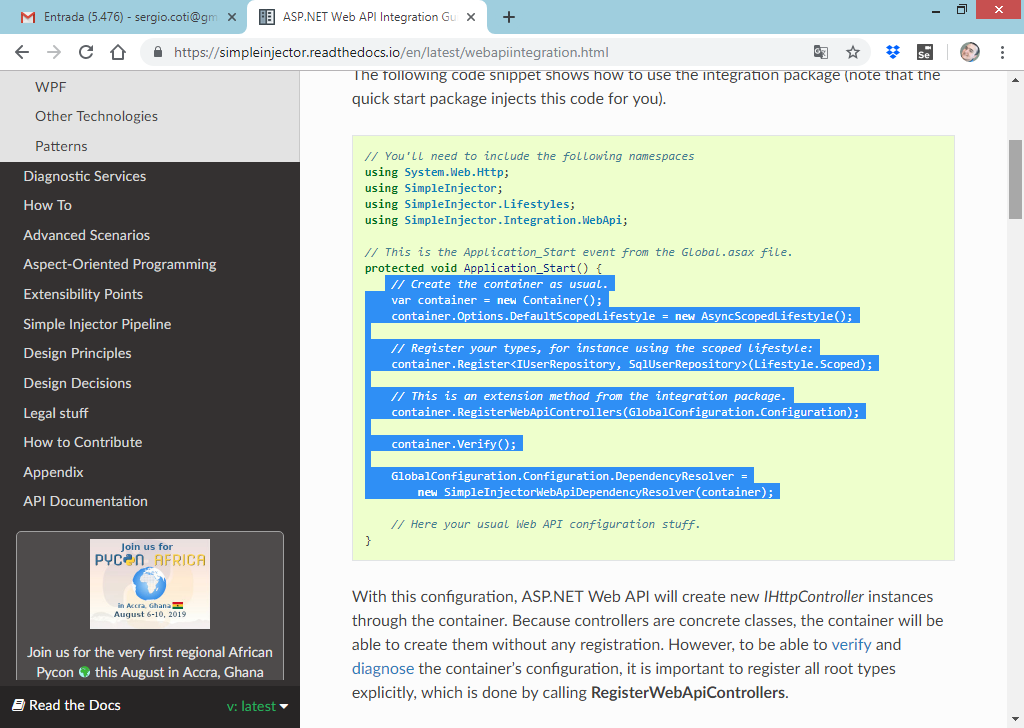
}

}

**Configurando o AutoMapper:**

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





using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Http;

using System.Web.Routing;

using AutoMapper;

using Projeto.Application.Mappings;

using SimpleInjector;

using SimpleInjector.Lifestyles;

using SimpleInjector.Integration.WebApi;

using Projeto.Domain.Contracts.Repositories;

using Projeto.Infra.Data.Repositories;

using Projeto.Domain.Contracts.Services;

using Projeto.Domain.Services;

using Projeto.Application.Contracts;

using Projeto.Application.Services;

namespace Projeto.Presentation

{

public class WebApiApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

//configurando o AutoMapper

Mapper.Initialize(cfg =>

{

cfg.AddProfile<EntityToViewModelMap>();

cfg.AddProfile<ViewModelToEntityMap>();

});

//configurando o SimpleInjector

// 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<IContatoRepository,**

**ContatoRepository>(Lifestyle.Scoped);**

**container.Register<IEnderecoRepository,**

**EnderecoRepository>(Lifestyle.Scoped);**

**container.Register<ICompromissoRepository,**

**CompromissoRepository>(Lifestyle.Scoped);**

**container.Register<IContatoDomainService,**

**ContatoDomainService>(Lifestyle.Scoped);**

**container.Register<IEnderecoDomainService,**

**EnderecoDomainService>(Lifestyle.Scoped);**

**container.Register<ICompromissoDomainService,**

**CompromissoDomainService>(Lifestyle.Scoped);**

**container.Register<IContatoAppService,**

**ContatoAppService>(Lifestyle.Scoped);**

**container.Register<IEnderecoAppService,**

**EnderecoAppService>(Lifestyle.Scoped);**

**container.Register<ICompromissoAppService,**

**CompromissoAppService>(Lifestyle.Scoped);**

// This is an extension method from the integration package.

container.RegisterWebApiControllers

(GlobalConfiguration.Configuration);

container.Verify();

GlobalConfiguration.Configuration.DependencyResolver =

new SimpleInjectorWebApiDependencyResolver(container);

}

}

}

**Implementando os serviços de contato:**

/Controllers/ContatoController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using Projeto.Application.ViewModels;

using Projeto.Application.Contracts;

namespace Projeto.Presentation.Controllers

{

[RoutePrefix("api/Contato")] //ENDPOINT

public class ContatoController : ApiController

{

//atributo

private readonly IContatoAppService appService;

//construtor para injeção de dependência

public ContatoController(IContatoAppService appService)

{

this.appService = appService;

}

[HttpPost]

public HttpResponseMessage Post(ContatoCadastroViewModel model)

{

if(ModelState.IsValid)

{

try

{

appService.Cadastrar(model);

return Request.CreateResponse

(HttpStatusCode.OK, "Contato cadastrado com sucesso.");

}

catch(Exception e)

{

return Request.CreateResponse

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

return Request.CreateResponse(HttpStatusCode.BadRequest);

}

}

}

}

**Testando:**

