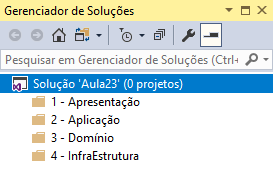


**Organização do projeto:**



DDD - Domain Driven Design

Design de projeto orientado a Domínio

É um padrão criado por Eric Evans que define um conjunto de boas práticas para desenvolvimento de projetos onde o foco é o dominio do sistema.

**Dominio:**

Entende-se por dominio todo o conjunto de entidades, serviços e regras de negócio inerentes a uma área de conhecimento.

Exemplo:

Domínio: **Agenda Pessoal**

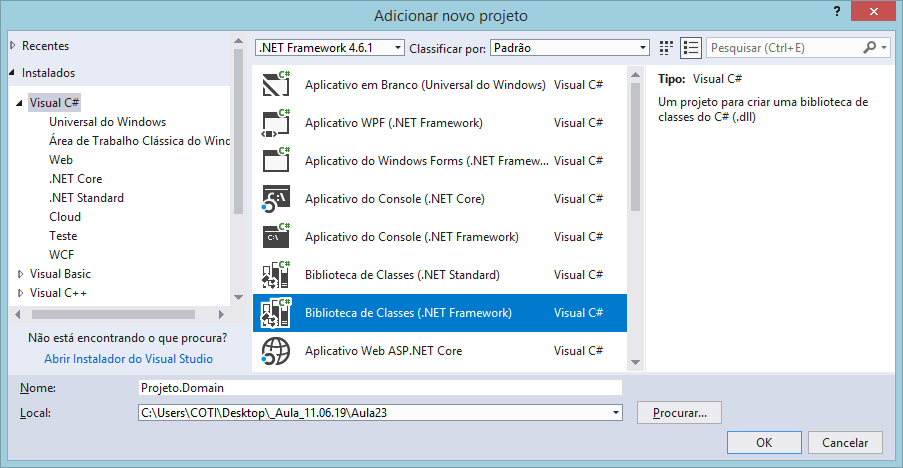
* Entidades:

Contato (IdContato, Nome, Email, Telefone)

Endereco (IdEndereco, Logradouro, Bairro, Cidade, Estado, Cep)

Compromisso (IdCompromisso, Data/Hora, Descricao, Status)

Passo 1) Criar o domínio:



No padrão DDD, todo projeto deve conter um ou mais domínios e também as seguintes partes:

**Apresentação**

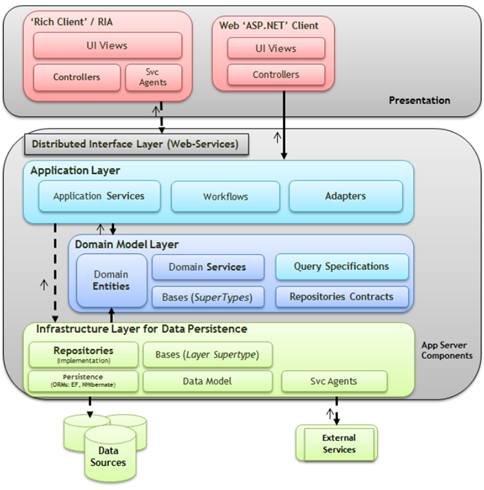
Parte do projeto que irá disponibilizar a interface do sistema com o usuário final. (WebAPI, MVC, etc..)

**Aplicação**

Parte do projeto que interliga a apresentação com o dominio

**InfraEstrutura**

Parte que dá sustentação ao dominio, onde implementamos repositorio de dados, logs, e qualquer tipo de projeto que provenha recursos ao dominio



Domain Entities

Classes que irão compor as entidades do dominio.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Domain.Entities

{

public class Contato

{

public int IdContato { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public string Telefone { get; set; }

public Endereco Endereco { get; set; }

public List<Compromisso> Compromissos { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Domain.Entities

{

public class Compromisso

{

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; }

public Contato Contato { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Domain.Entities

{

public class Endereco

{

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; }

public Contato Contato { get; set; }

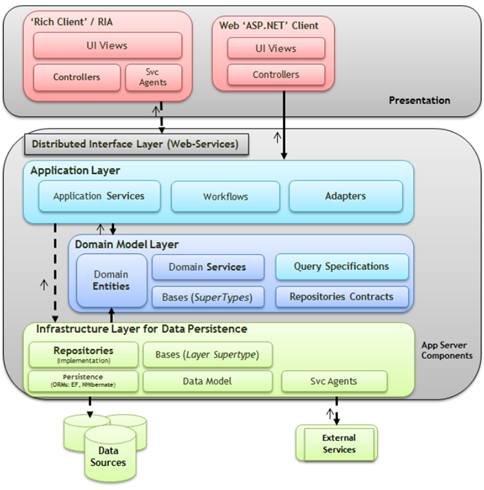
}

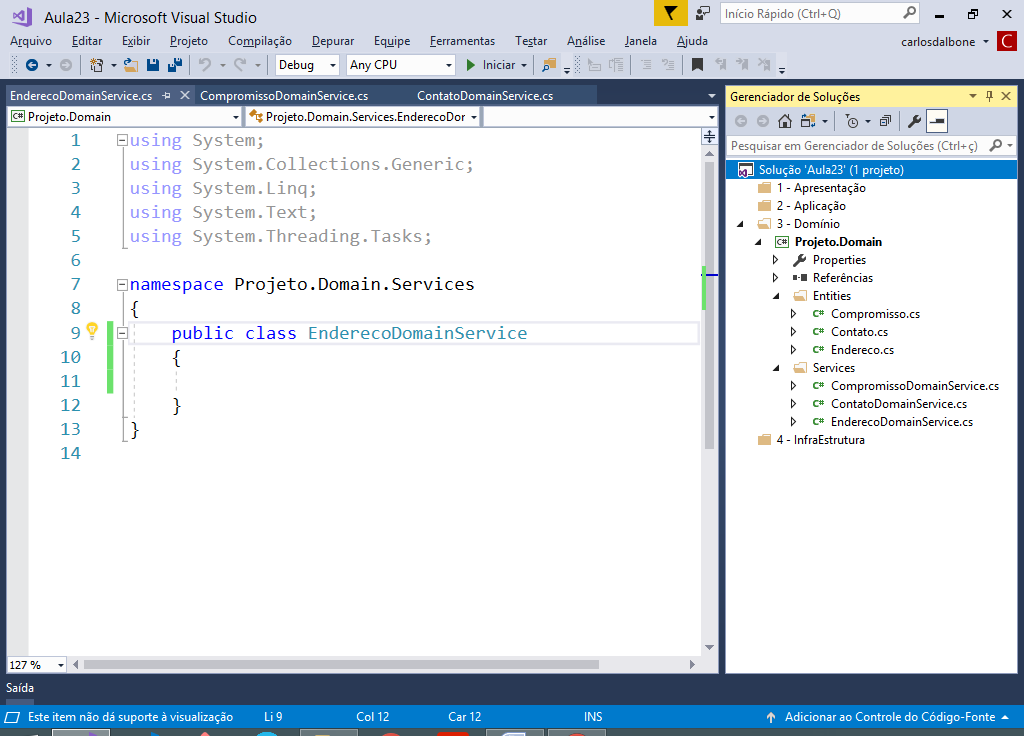
}

--------------------------

Domain Services

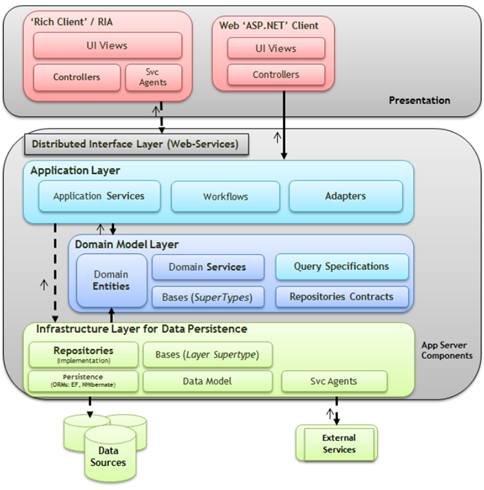
São classes que deverão prover os serviços e operações realizadas pelo dominio para cada Domain Entity, implementando suas regras de negócio.





Bases (Super Types)

Para o DDD, todo serviço pode ser abstraido atraves de um tipo generico.



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Domain.Contracts.Services

{

public interface IBaseDomainService<TEntity, TKey>

where TEntity : class

{

void Cadastrar(TEntity obj);

void Atualizar(TEntity obj);

void Excluir(TEntity obj);

List<TEntity> ConsultarTodos();

TEntity ConsultarPorId(TKey id);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

namespace Projeto.Domain.Contracts.Services

{

public interface IContatoDomainService

: IBaseDomainService<Contato, int>

{

Contato ConsultarPorEmail(string email);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

namespace Projeto.Domain.Contracts.Services

{

public interface ICompromissoDomainService

: IBaseDomainService<Compromisso, int>

{

List<Compromisso> ConsultarPorDatas

(DateTime dataInicio, DateTime dataFim);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

namespace Projeto.Domain.Contracts.Services

{

public interface IEnderecoDomainService

: IBaseDomainService<Endereco, int>

{

}

}

**Implementando as interfaces:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Contracts.Services;

namespace Projeto.Domain.Services

{

public abstract class BaseDomainService<TEntity, TKey>

: IBaseDomainService<TEntity, TKey>

where TEntity : class

{

public void Cadastrar(TEntity obj)

{

throw new NotImplementedException();

}

public void Atualizar(TEntity obj)

{

throw new NotImplementedException();

}

public void Excluir(TEntity obj)

{

throw new NotImplementedException();

}

public List<TEntity> ConsultarTodos()

{

throw new NotImplementedException();

}

public TEntity ConsultarPorId(TKey id)

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

using Projeto.Domain.Contracts.Services;

namespace Projeto.Domain.Services

{

public class CompromissoDomainService

: BaseDomainService<Compromisso, int>, ICompromissoDomainService

{

public List<Compromisso> ConsultarPorDatas

(DateTime dataInicio, DateTime dataFim)

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

using Projeto.Domain.Contracts.Services;

namespace Projeto.Domain.Services

{

public class ContatoDomainService

: BaseDomainService<Contato, int>, IContatoDomainService

{

public Contato ConsultarPorEmail(string email)

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

using Projeto.Domain.Contracts.Services;

namespace Projeto.Domain.Services

{

public class EnderecoDomainService

: BaseDomainService<Endereco, int>, IEnderecoDomainService

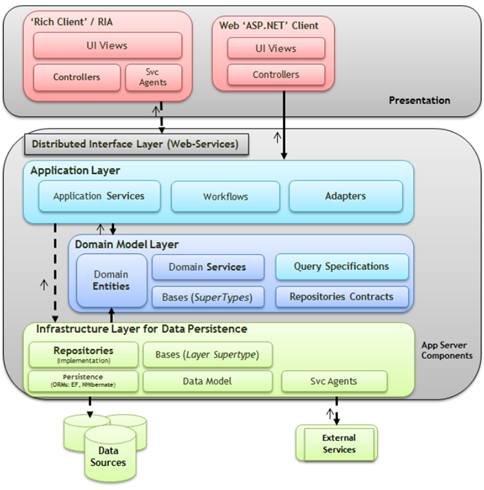
{

}

}

Repository Contracts

Nesta parte do dominio serão criadas as interfaces para implementar as classes do repositorio de dados



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Domain.Contracts.Repositories

{

public interface IBaseRepository<TEntity, TKey>

where TEntity : class

{

void Insert(TEntity obj);

void Update(TEntity obj);

void Delete(TEntity obj);

List<TEntity> GetAll();

TEntity GetById(TKey id);

}

}

----------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

namespace Projeto.Domain.Contracts.Repositories

{

public interface ICompromissoRepository

: IBaseRepository<Compromisso, int>

{

List<Compromisso> GetByDatas

(DateTime dataInicio, DateTime dataFim);

}

}

----------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

namespace Projeto.Domain.Contracts.Repositories

{

public interface IContatoRepository

: IBaseRepository<Contato, int>

{

Contato GetByEmail(string email);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Domain.Entities;

namespace Projeto.Domain.Contracts.Repositories

{

public interface IEnderecoRepository

: IBaseRepository<Endereco, int>

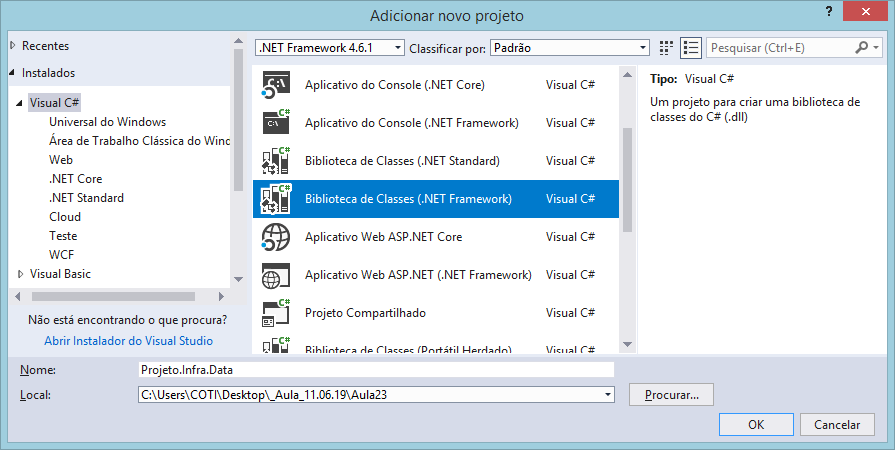
{

}

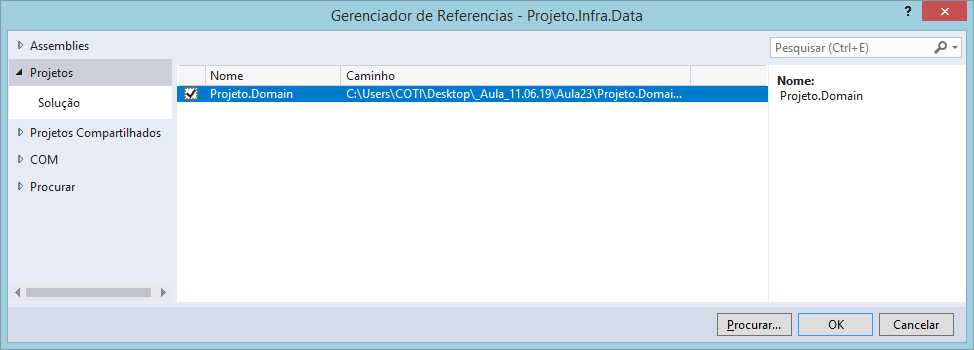
}

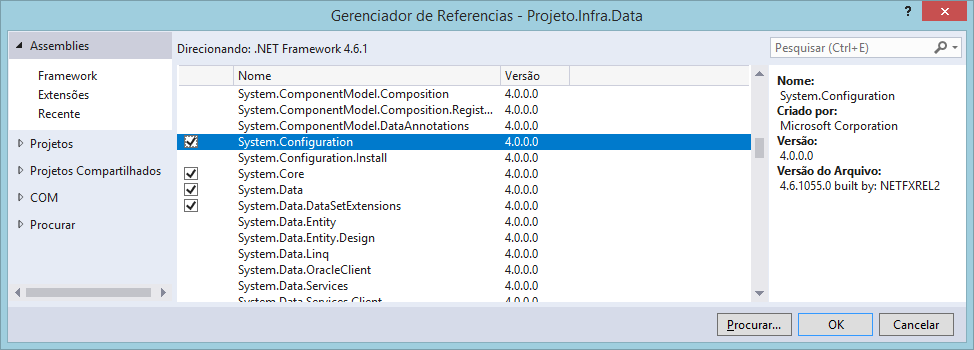
4 - InfraEstrutura

Parte do sistema DDD destinado a implementar operações de suporte do dominio. Todo projeto feito em DDD terá pelo menos 1 InfraEstrutura destinada a implementar o repositorio de dados.



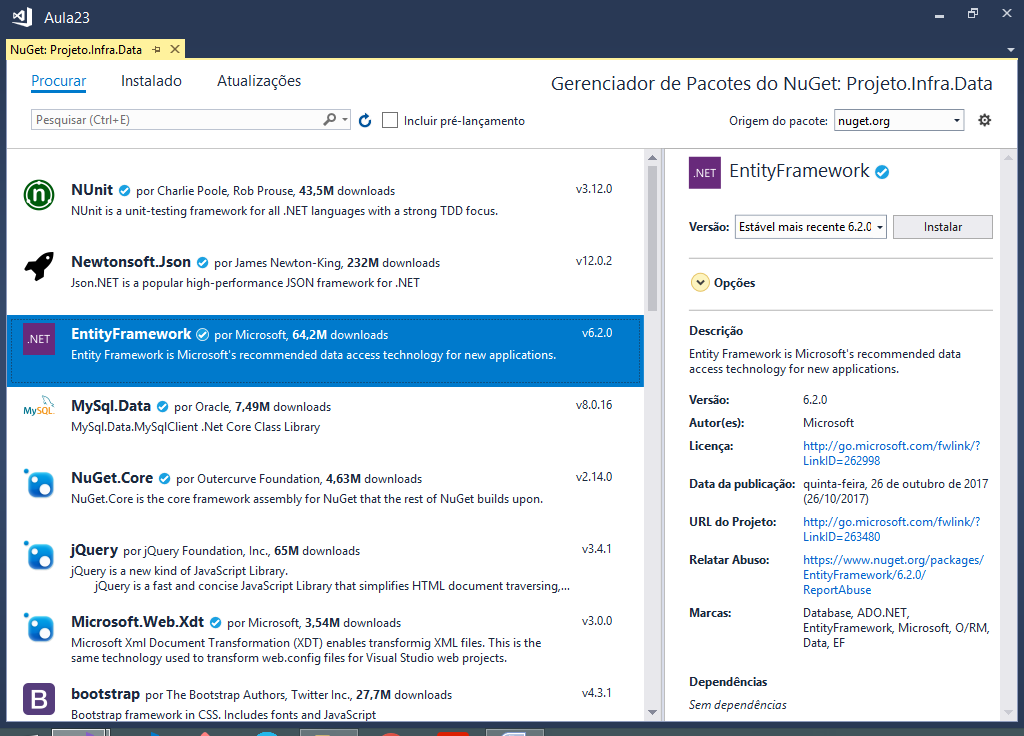
Adicionando referencias:





Instalando o EntityFramework

Gerenciador de pacotes do NuGet



ORM - Mapeamento Objeto / Relacional

Mapear as classes Domain Entities para que sejam   
interpretadas como tabelas do banco de dados.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity.ModelConfiguration; //EntityFramework

using Projeto.Domain.Entities; //classes de entidade

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Infrastructure.Annotations;

namespace Projeto.Infra.Data.Configurations

{

public class ContatoConfiguration

: EntityTypeConfiguration<Contato>

{

//construtor -> ctor + 2x[tab]

public ContatoConfiguration()

{

ToTable("Contato");

HasKey(c => new { c.IdContato });

Property(c => c.IdContato)

.HasColumnName("IdContato");

Property(c => c.Nome)

.HasColumnName("Nome")

.HasMaxLength(150)

.IsRequired();

Property(c => c.Email)

.HasColumnName("Email")

.HasMaxLength(100)

.IsRequired()

.HasColumnAnnotation(

IndexAnnotation.AnnotationName,

new IndexAnnotation(

new IndexAttribute("IX\_Email")

{ IsUnique = true }));

Property(c => c.Telefone)

.HasColumnName("Telefone")

.HasMaxLength(20)

.IsRequired();

}

}

}

--------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity.ModelConfiguration; //importando

using Projeto.Domain.Entities; //importando

namespace Projeto.Infra.Data.Configurations

{

public class CompromissoConfiguration

: EntityTypeConfiguration<Compromisso>

{

//ctor + 2x[tab]

public CompromissoConfiguration()

{

ToTable("Compromisso");

HasKey(c => new { c.IdCompromisso });

Property(c => c.IdCompromisso)

.HasColumnName("IdCompromisso");

Property(c => c.DataHora)

.HasColumnName("DataHora")

.IsRequired();

Property(c => c.Descricao)

.HasColumnName("Descricao")

.HasMaxLength(500 )

.IsRequired();

Property(c => c.Status)

.HasColumnName("Status")

.HasMaxLength(25)

.IsRequired();

Property(c => c.IdContato)

.HasColumnName("IdContato")

.IsRequired();

//mapear o relacionamento com Contato

HasRequired(c => c.Contato) //Compromisso TEM 1 Contato

.WithMany(c => c.Compromissos) //Contato TEM MUITOS Compromissos

.HasForeignKey(c => c.IdContato) //chave estrangeira

.WillCascadeOnDelete(false); //delete cascade -> false

}

}

}

--------------------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity.ModelConfiguration; //importando

using Projeto.Domain.Entities; //importando

namespace Projeto.Infra.Data.Configurations

{

public class EnderecoConfiguration

: EntityTypeConfiguration<Endereco>

{

//construtor -> ctor + 2x[tab]

public EnderecoConfiguration()

{

ToTable("Endereco");

HasKey(e => e.IdEndereco);

Property(e => e.IdEndereco)

.HasColumnName("IdEndereco")

.IsRequired();

Property(e => e.Logradouro)

.HasColumnName("Logradouro")

.HasMaxLength(150)

.IsRequired();

Property(e => e.Bairro)

.HasColumnName("Bairro")

.HasMaxLength(100)

.IsRequired();

Property(e => e.Cidade)

.HasColumnName("Cidade")

.HasMaxLength(100)

.IsRequired();

Property(e => e.Estado)

.HasColumnName("Estado")

.HasMaxLength(50)

.IsRequired();

//Mapeamento do relacionamento

HasRequired(e => e.Contato)

.WithOptional(c => c.Endereco)

.WillCascadeOnDelete(false);

}

}

}

Continua...