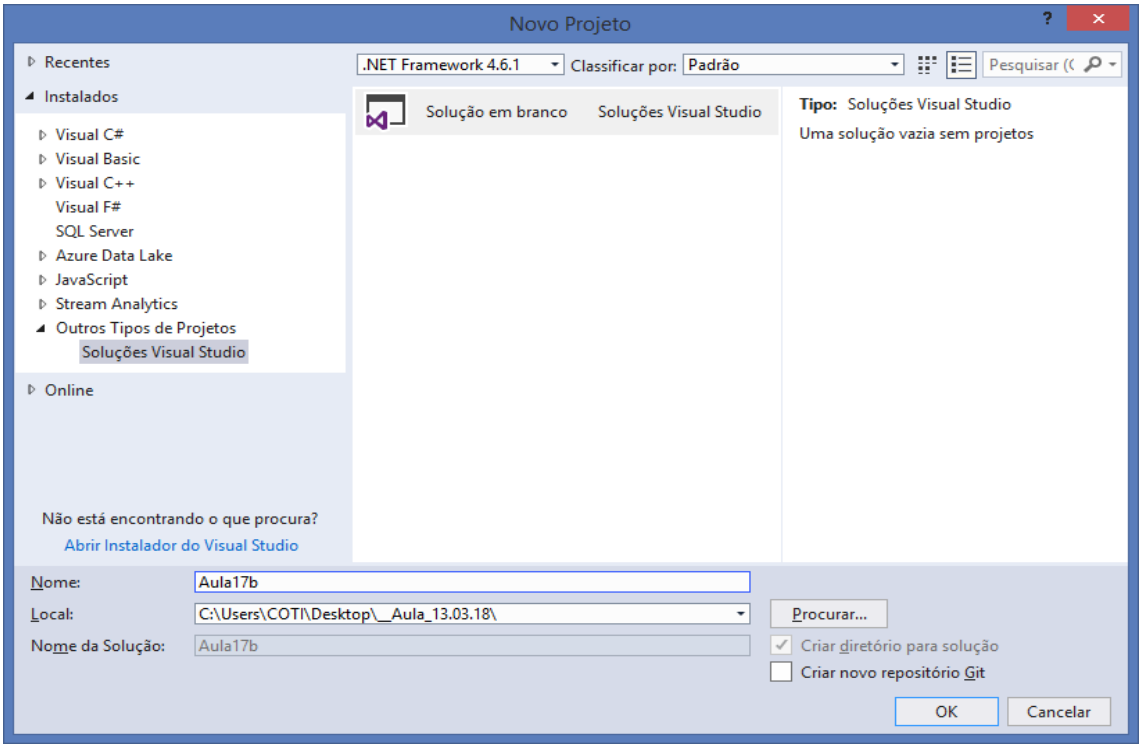
Nova solution em branco:



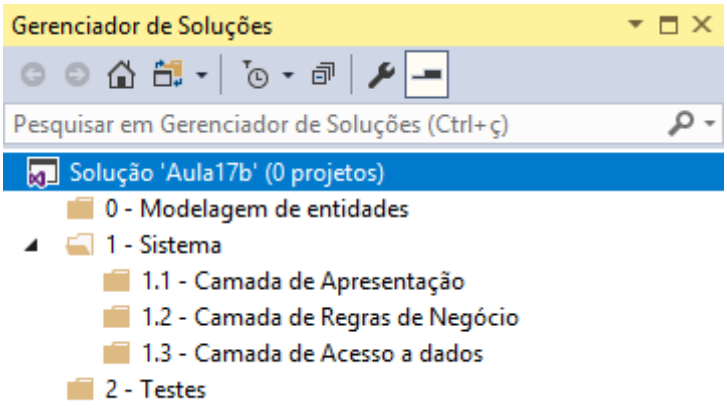
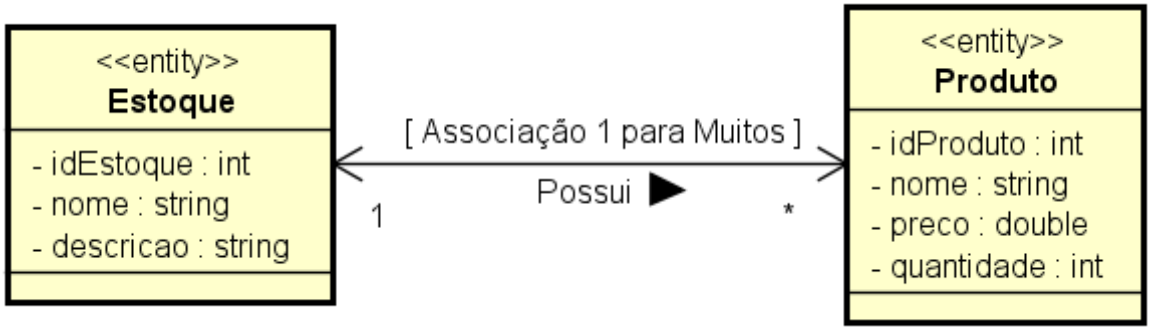


Diagrama de Classes:

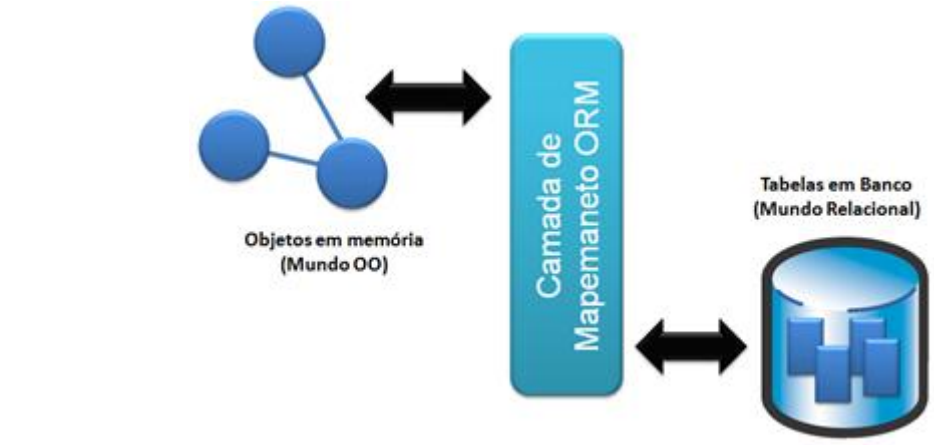


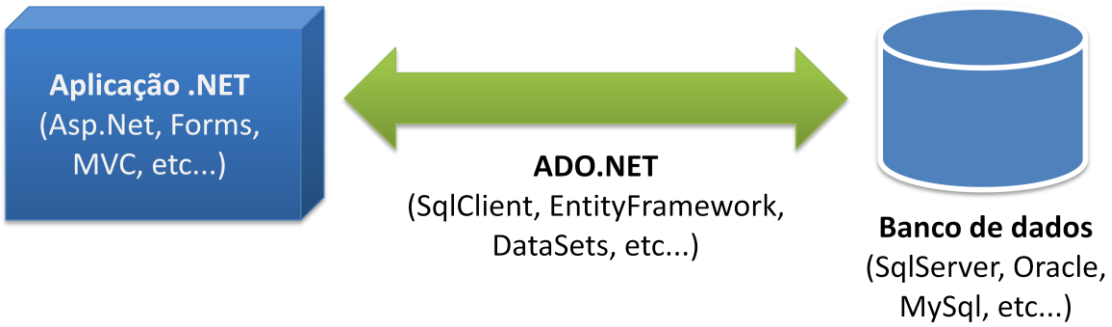
ADO.NET Entity Framework

ADO.NET (ActiveX Data Objects) consiste em um conjunto de bibliotecas definidas pelo .NET framework (localizadas no namespace **System.Data**) que pode ser utilizado para manipular e persistir informações armazenadas numa base de dados remota.

O Microsoft® **ADO.NET Entity Framework** é um framework do tipo ORM (Object/Relational Mapping) que permite aos desenvolvedores trabalhar com dados relacionais como objetos de domínio específico, eliminando a necessidade de maior parte dos códigos de acesso de dados que os desenvolvedores geralmente precisam escrever. Com o Entity Framework, os desenvolvedores podem lançar consultas usando expressões LAMBDA, e depois recuperar e manipular dados como objetos fortemente tipificados.

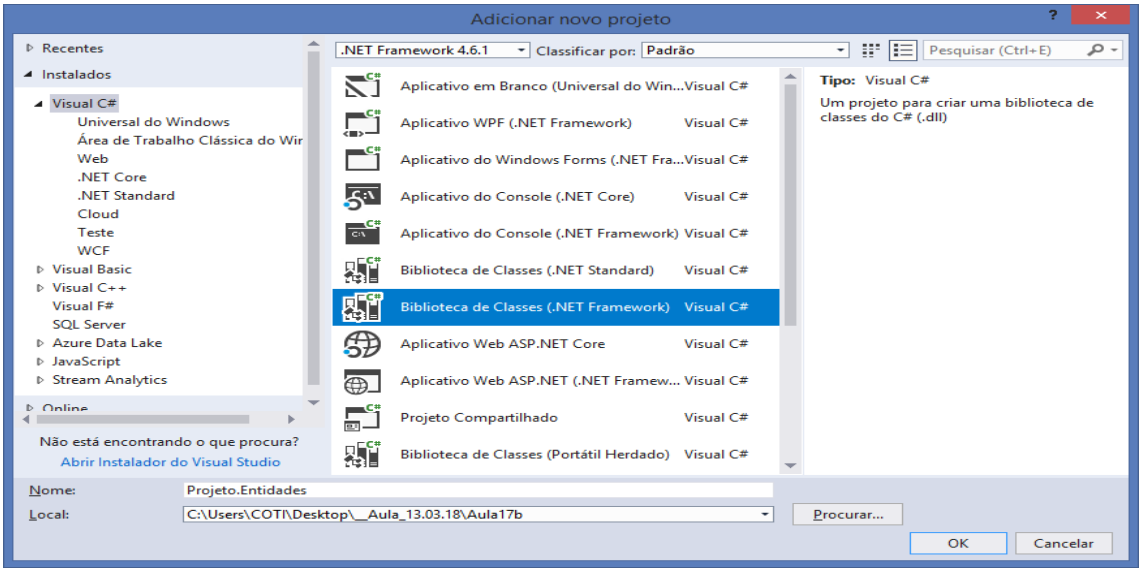
A implementação do ORM do Entity Framework fornece serviços como rastreamento de alterações, resolução de identidades, lazy loading e tradução de consultas para que os desenvolvedores possam se concentrar na lógica de negócios de seus aplicativos em vez dos princípios básicos de acesso a dados.





0 - Modelagem de entidades

Class Library .NET Framework




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

namespace Projeto.Entidades
{
    public class Produto
    {
        public int IdProduto { get; set; }
        public string Nome { get; set; }
        public decimal Preco { get; set; }
        public int Quantidade { get; set; }

        public Estoque Estoque { get; set; }
    }
}
```

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

namespace Projeto.Entidades
{
    public class Estoque
    {
        public int IdEstoque { get; set; }
        public string Nome { get; set; }
        public string Descricao { get; set; }

        public List<Produto> Produtos { get; set; }
    }
}
```

Para que o EntityFramework possa mapear as classes de entidade, é necessário que as propriedades set e get sejam declaradas com operador **virtual**

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

namespace Projeto.Entidades
{
    public class Estoque
    {
        public virtual int IdEstoque { get; set; }
        public virtual string Nome { get; set; }
        public virtual string Descricao { get; set; }

        public virtual List<Produto> Produtos { get; set; }
    }
}
```

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

namespace Projeto.Entidades
{
    public class Produto
    {
        public virtual int IdProduto { get; set; }
        public virtual string Nome { get; set; }
        public virtual decimal Preco { get; set; }
        public virtual int Quantidade { get; set; }

        public virtual Estoque Estoque { get; set; }
    }
}
```

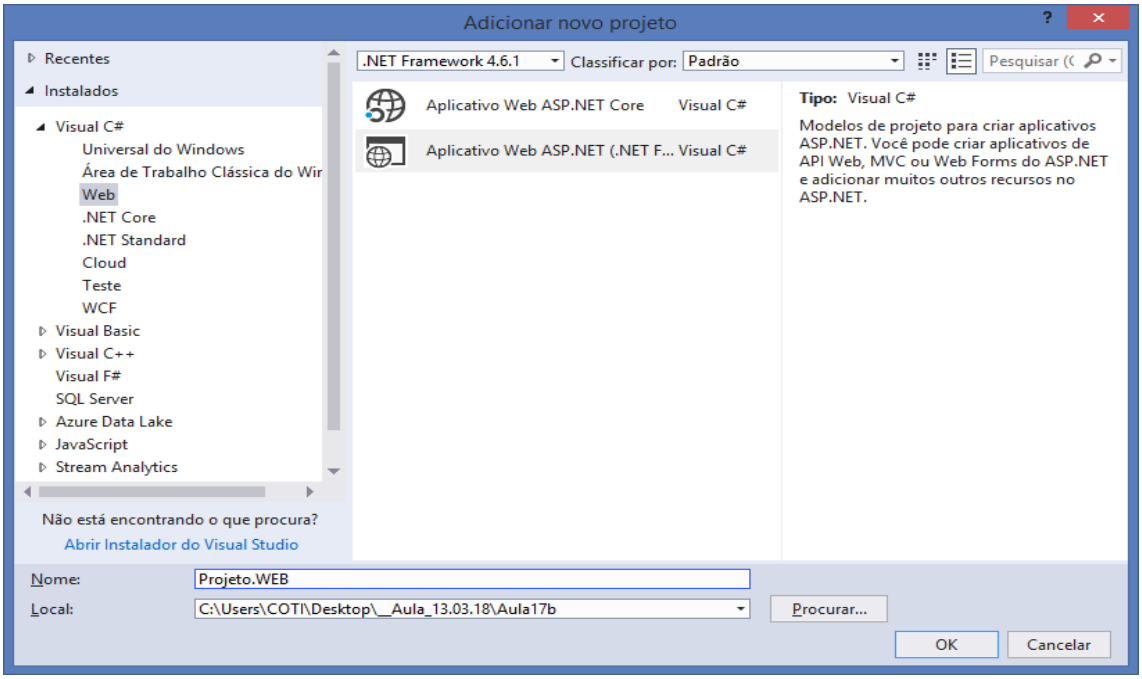
Regra: Se uma entidade no banco de dados conter foreign key, será necessário declarar na classe uma propriedade para esta foreign key

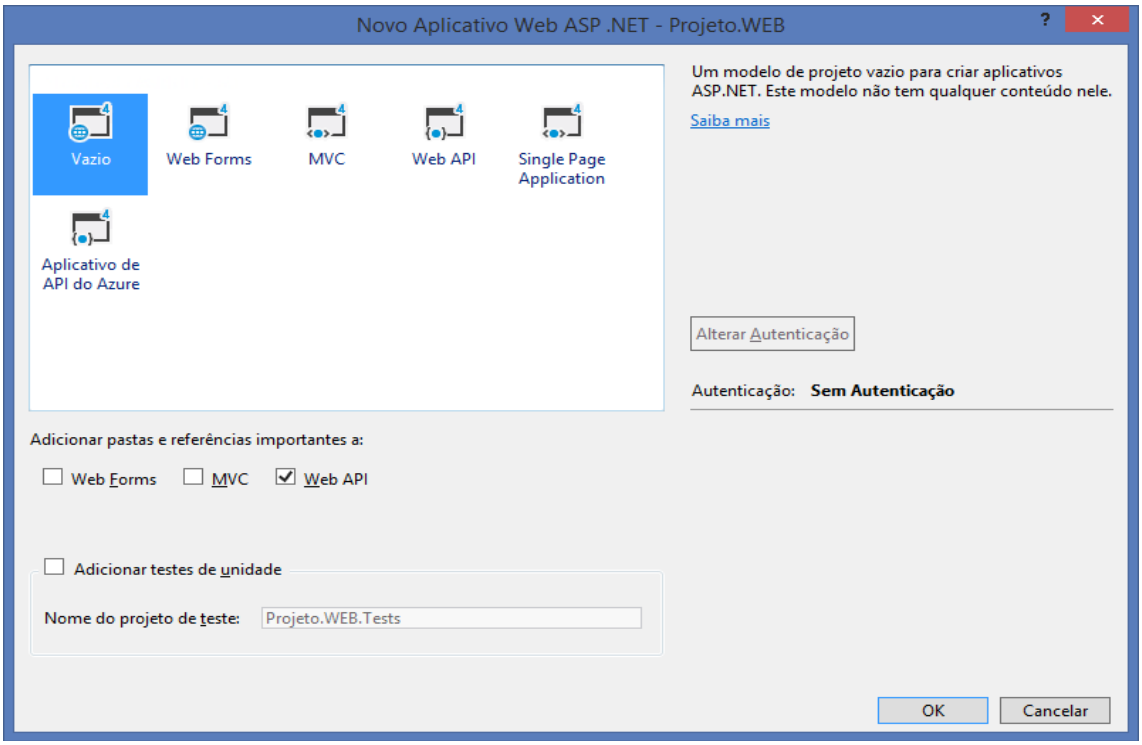
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.Entidades
{
    public class Produto
    {
        public virtual int IdProduto { get; set; }
        public virtual string Nome { get; set; }
```

```
        public virtual decimal Preco { get; set; }
        public virtual int Quantidade { get; set; }
        public virtual int IdEstoque { get; set; }

        public virtual Estoque Estoque { get; set; }
    }
}
```

1.1 - Camada de Apresentação

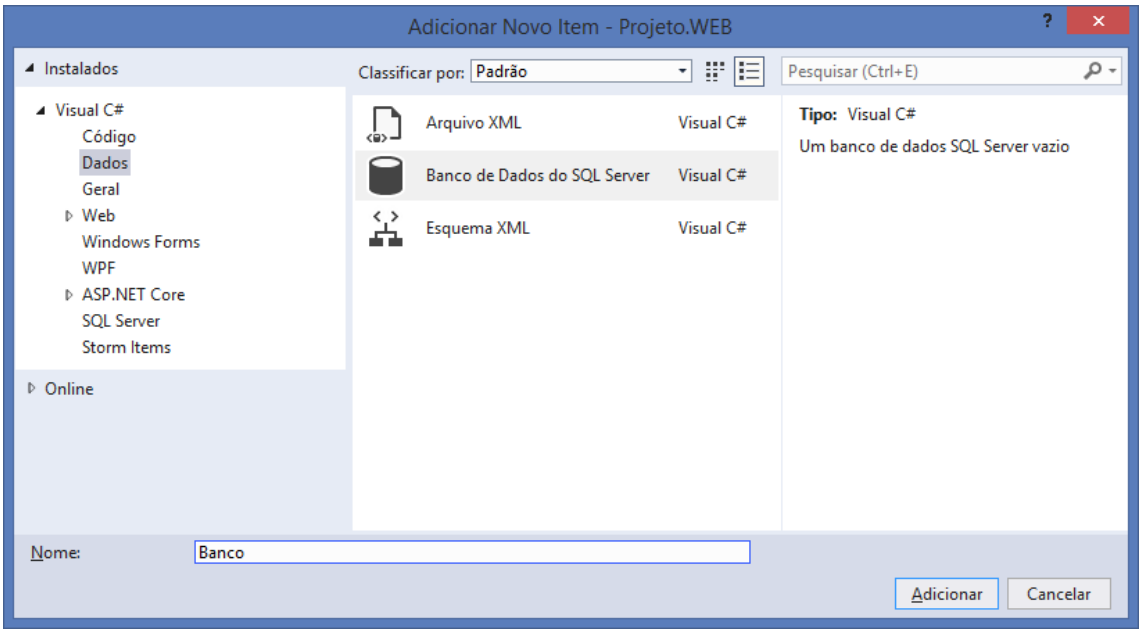
Asp.Net WebApi





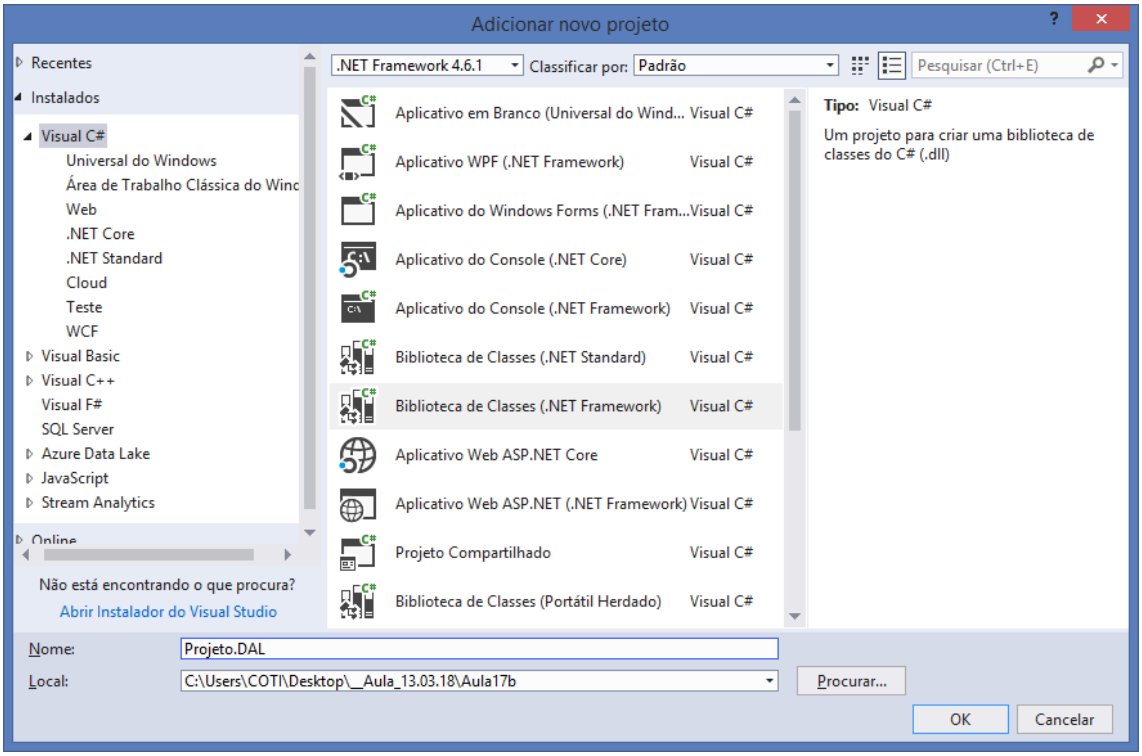
Criando a base de dados:
MDF - Master Database File

/App_Data/Banco.mdf



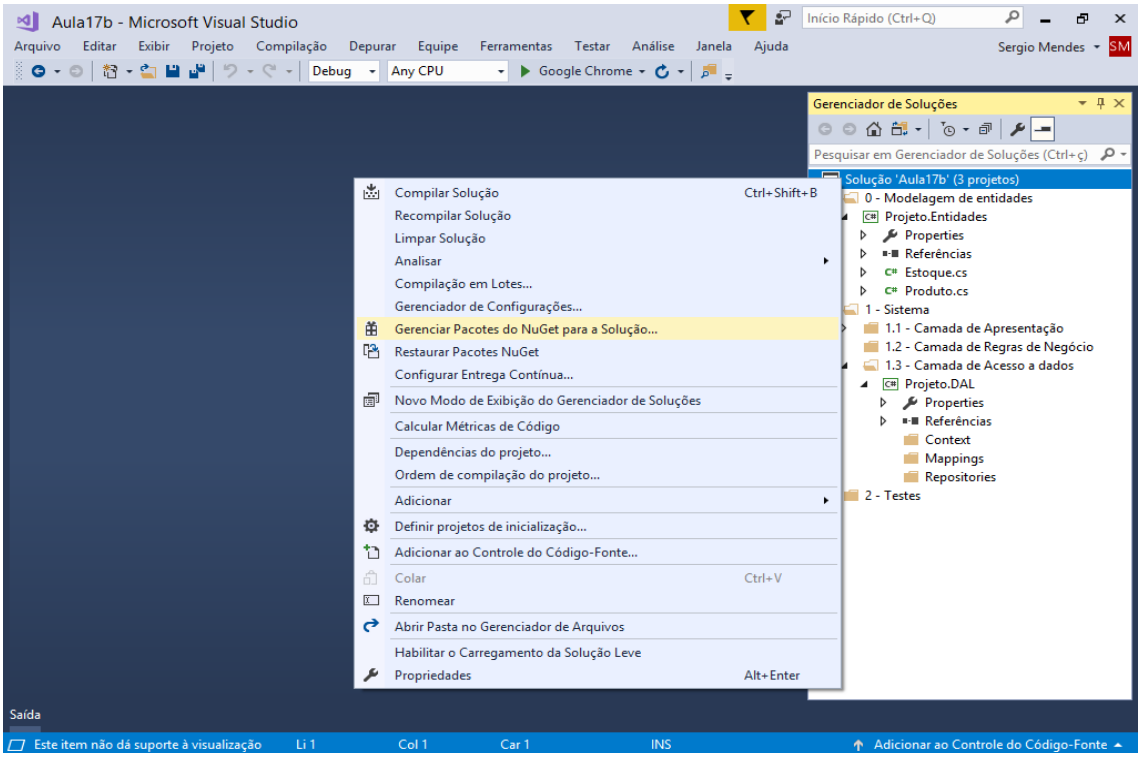
1.3 - Camada de Acesso a Dados

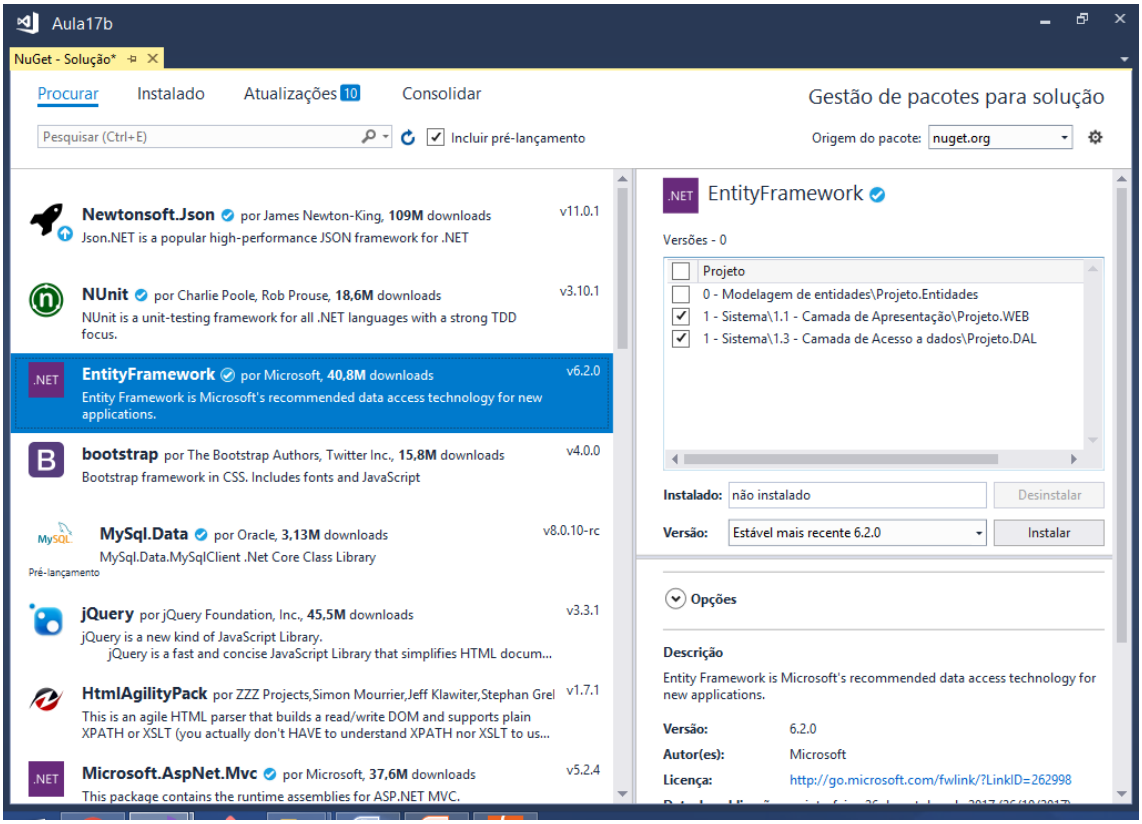
DAL - Data Access Layer



Instalando o EntityFramework

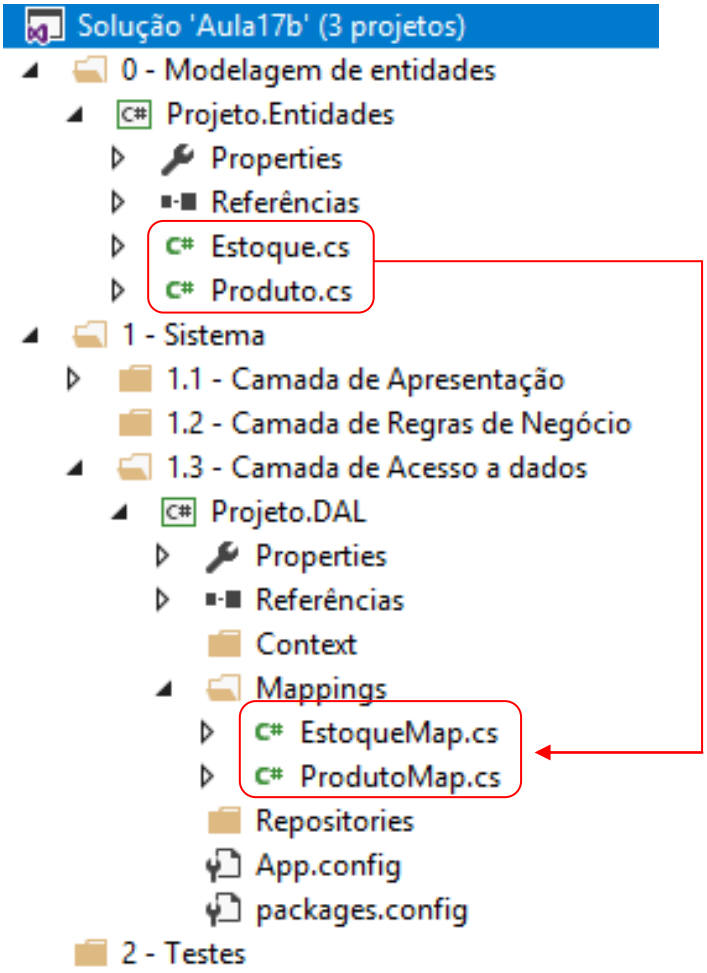
Observação: O EntityFramework deverá ser instalado no projeto DAL, mas tambem no projeto Asp.Net (ConnectionString)



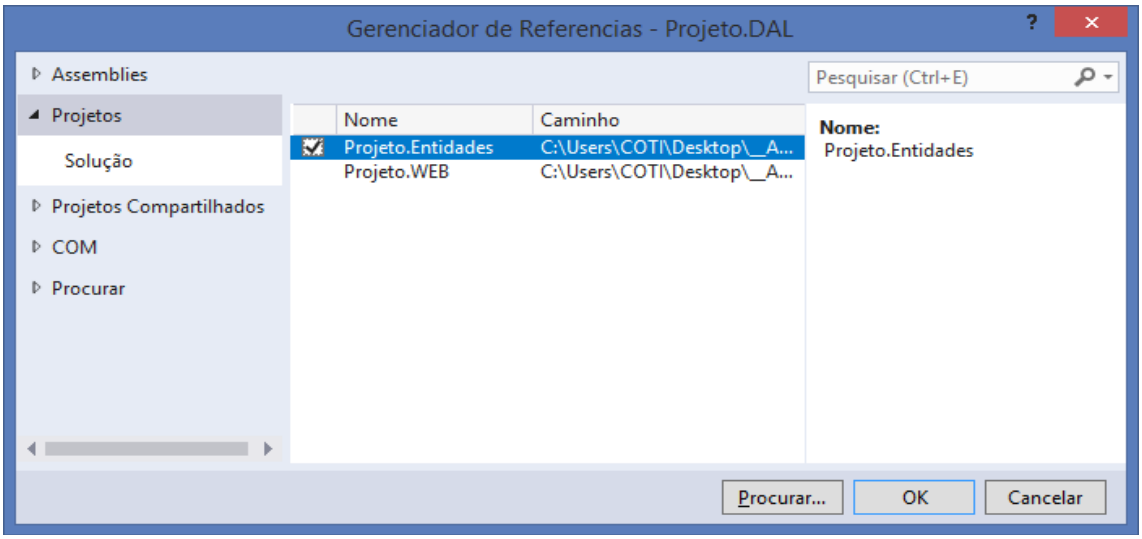


ORM - Mapeamento Objeto Relacional

Mapear cada classe de entidade para estas sejam interpretadas pelo EntityFramework como tabelas do banco de dados.



Adicionando referencia no projeto DAL para o projeto Entidades:



```
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; //mapeamento..

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

            //chave primária..
            HasKey(e => e.IdEstoque);

            //mapear os campos..
            Property(e => e.IdEstoque)
                .HasColumnName("IdEstoque")
                .IsRequired();

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

            Property(e => e.Descricao)
                .HasColumnName("Descricao")
                .HasMaxLength(250)
                .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; //mapeamento..

namespace Projeto.DAL.Mappings
{
    //classe de mapeamento para a entidade Produto
    public class ProdutoMap : EntityTypeConfiguration<Produto>
    {
        //construtor [ctor] + 2x[tab]
        public ProdutoMap()
        {
            //nome da tabela..
            ToTable("Produto");
```

```
            //chave primária..
            HasKey(p => p.IdProduto);

            //campos da tabela..
            Property(p => p.IdProduto)
                .HasColumnName("IdProduto")
                .IsRequired();

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

            Property(p => p.Preco)
                .HasColumnName("Preco")
                .HasPrecision(18,2)
                .IsRequired();

            Property(p => p.Quantidade)
                .HasColumnName("Quantidade")
                .IsRequired();

            //mapear o relacionamento
            //cardinalidade 1 para muitos..
            HasRequired(p => p.Estoque) //Produto TEM 1 Estoque
                .WithMany(e => e.Produtos) //Estoque TEM MUITOS Produtos
                .HasForeignKey(p => p.IdEstoque); //Chave Estrangeira
        }
    }
}
```

Continua...