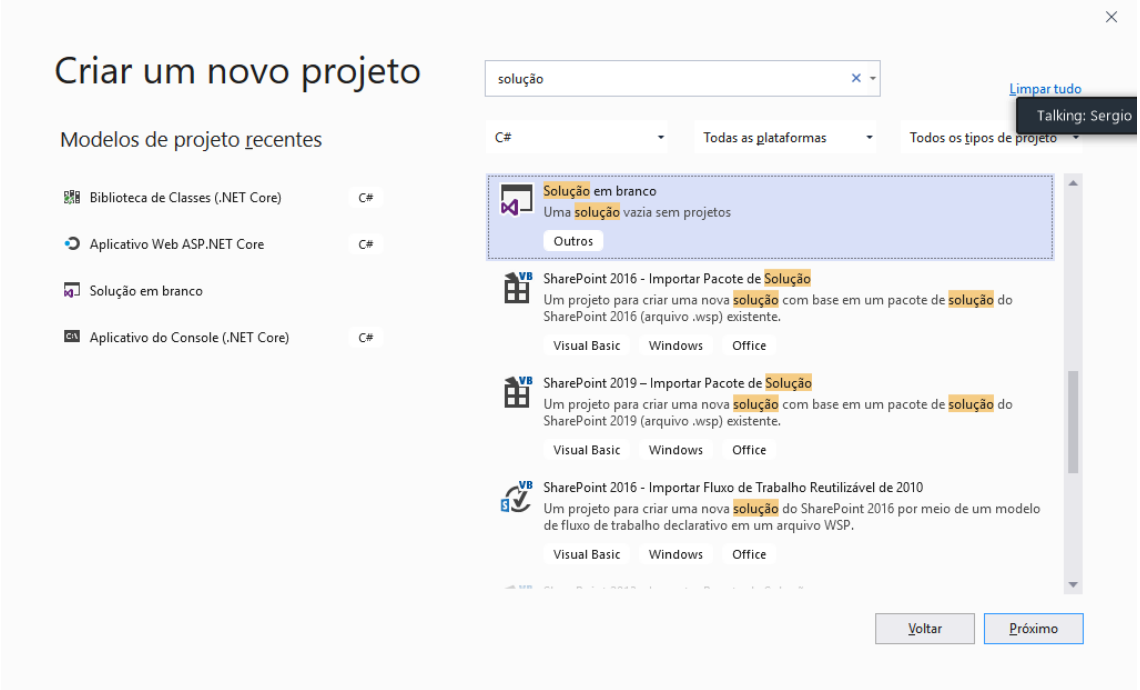
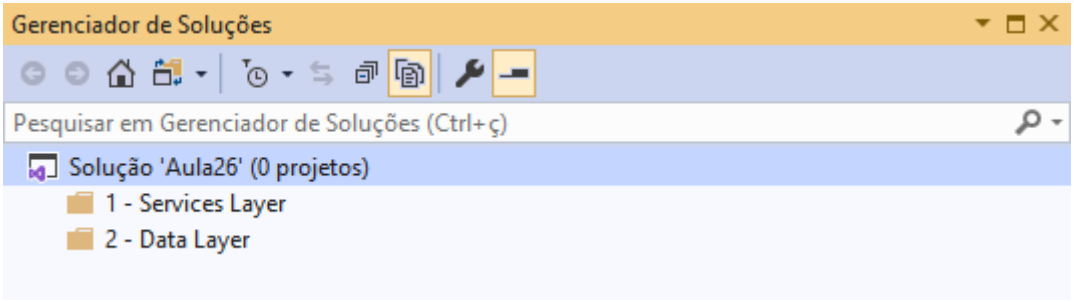
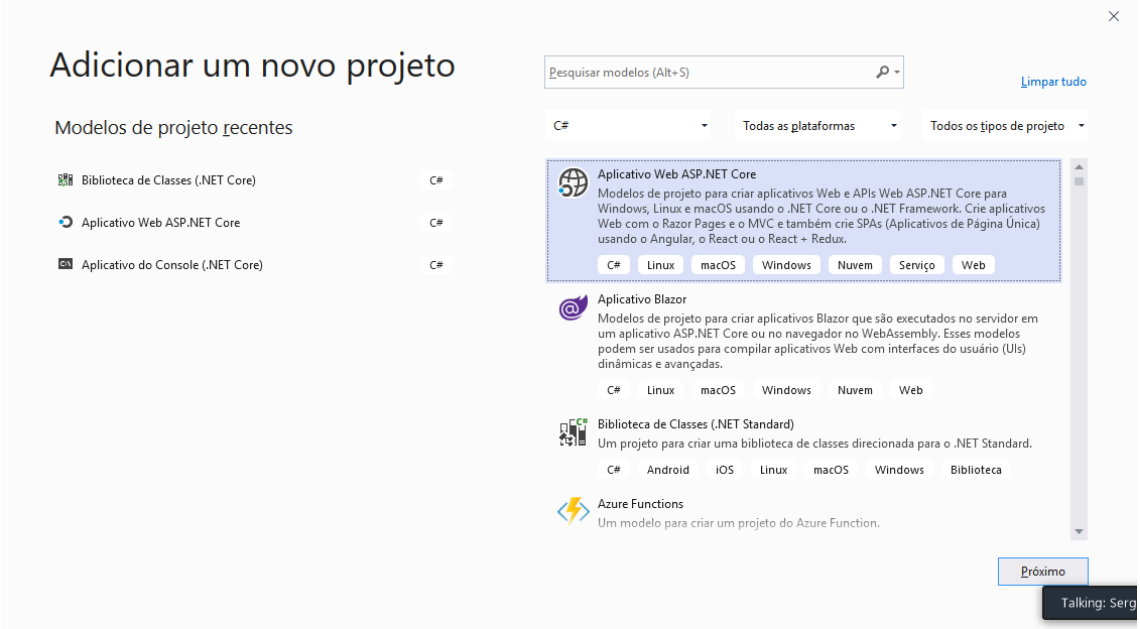
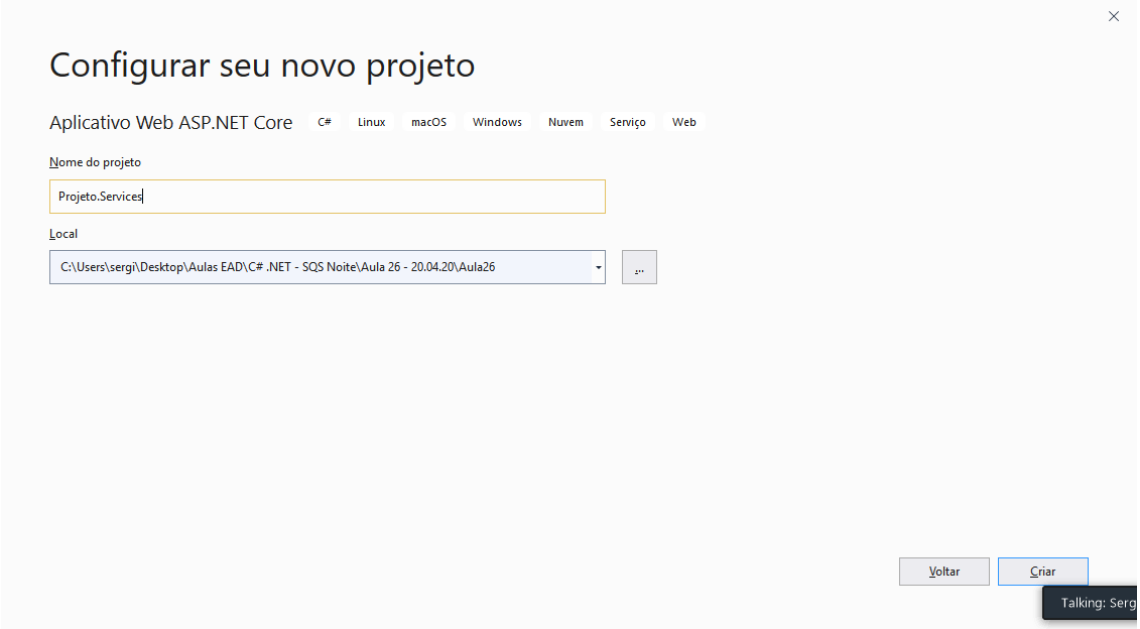
Criando uma nova solution em branco:

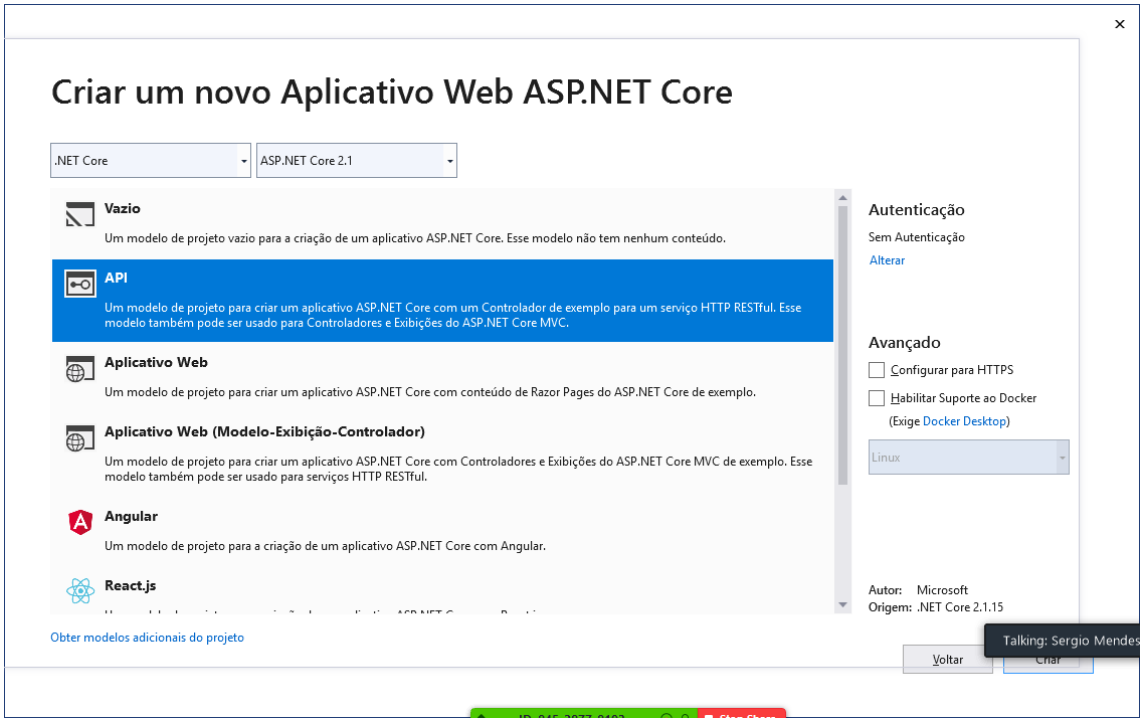


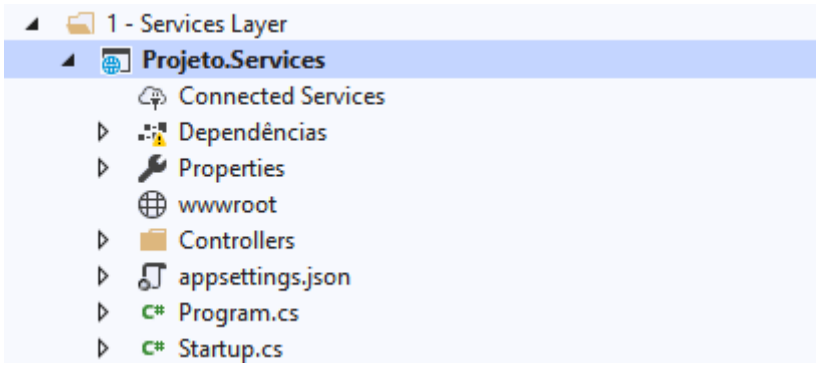


1 – Services Layer
Projeto Asp.Net CORE API

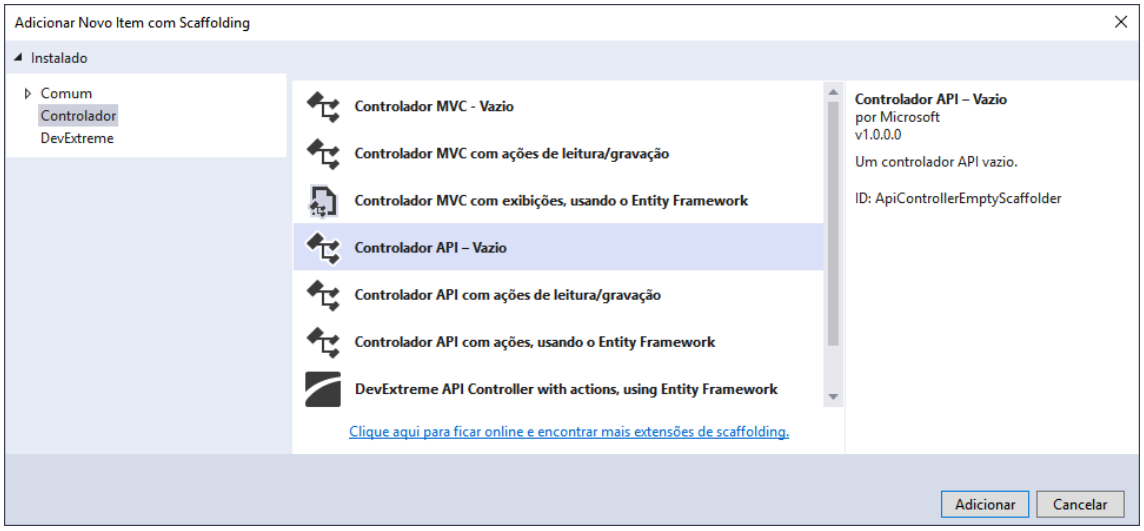


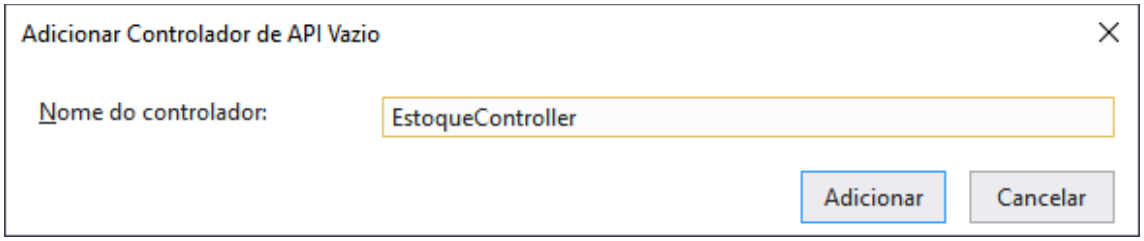

Configurar seu novo projeto
Aplicativo Web ASP.NET Core
C#
Linux
macOS
Windows
Nuvem
Serviço
Web
Nome do projeto
Projeto.Services
Local
C:\Users\sergi\Desktop\Aulas EAD\C# .NET - SQS Noite\Aula 26 - 20.04.20\Aula26
Voltar
Criar
Talking: Serg


Criar um novo Aplicativo Web ASP.NET Core
.NET Core
ASP.NET Core 2.1
Vazio
Um modelo de projeto vazio para a criação de um aplicativo ASP.NET Core. Esse modelo não tem nenhum conteúdo.
API
Um modelo de projeto para criar um aplicativo ASP.NET Core com um Controlador de exemplo para um serviço HTTP RESTful. Esse modelo também pode ser usado para Controladores e Exibições do ASP.NET Core MVC.
Aplicativo Web
Um modelo de projeto para criar um aplicativo ASP.NET Core com conteúdo de Razor Pages do ASP.NET Core de exemplo.
Aplicativo Web (Modelo-Exibição-Controlador)
Um modelo de projeto para criar um aplicativo ASP.NET Core com Controladores e Exibições do ASP.NET Core MVC de exemplo. Esse modelo também pode ser usado para serviços HTTP RESTful.
Angular
Um modelo de projeto para a criação de um aplicativo ASP.NET Core com Angular.
React.js
Autenticação
Sem Autenticação
Alterar
Avançado
Configurar para HTTPS
Habilitar Suporte ao Docker
(Exige Docker Desktop)
Linux
Autor: Microsoft
Origem: .NET Core 2.1.15
Obter modelos adicionais do projeto
Voltar
Criar
Talking: Sergio Mendes

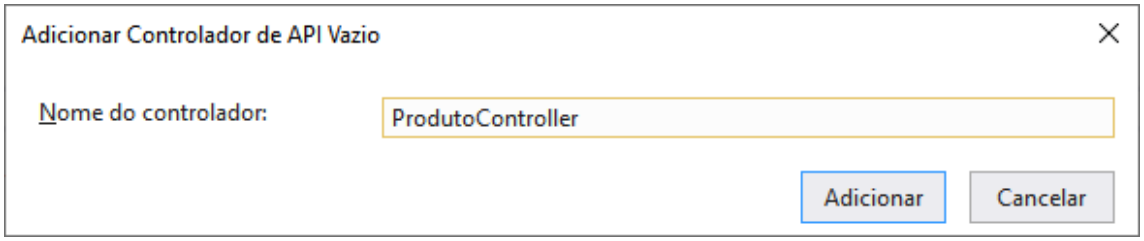

1 - Services Layer
Projeto.Services
Connected Services
Dependências
Properties
wwwroot
Controllers
appsettings.json
Program.cs
Startup.cs

Criando os controllers da API:
ENDPOINTS de serviço (endereços)






```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Projeto.Services.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EstoqueController : ControllerBase
    {

    }
}
```



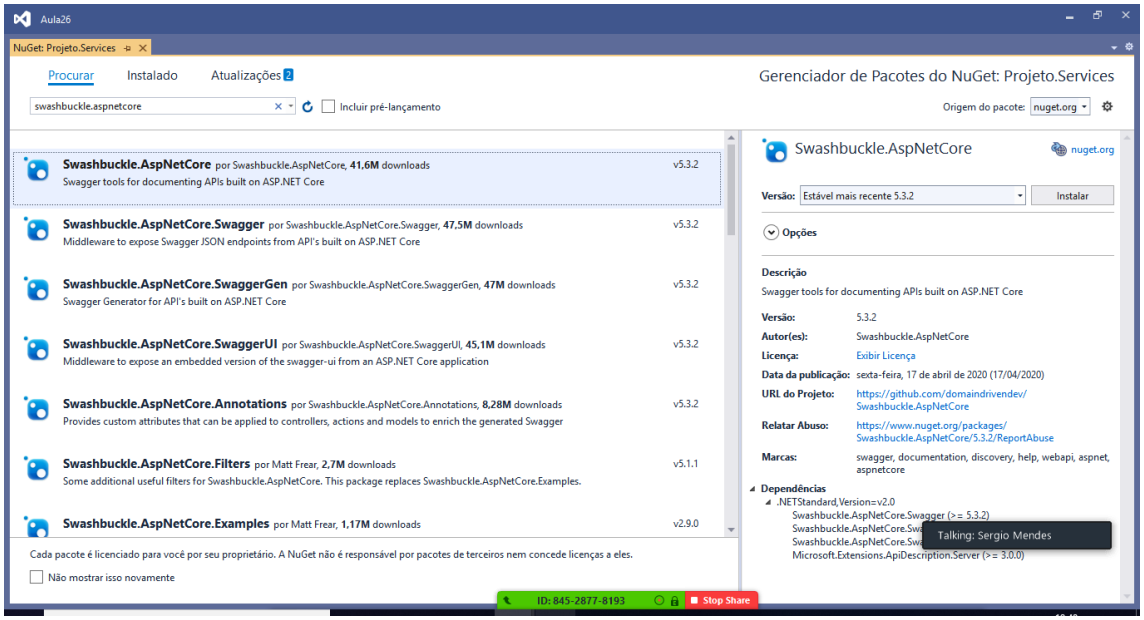
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Projeto.Services.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EstoqueController : ControllerBase
    {
        [HttpPost]
        public IActionResult Post()
        {
            return Ok();
        }

        [HttpPut]
        public IActionResult Put()
        {
            return Ok();
        }

        [HttpDelete("{id}")]
        public IActionResult Delete(int id)
        {
            return Ok();
        }

        [HttpGet]
        public IActionResult GetAll()
        {
            return Ok();
        }

        [HttpGet("{id}")]
        public IActionResult GetById(int id)
        {
            return Ok();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Projeto.Services.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProdutoController : ControllerBase
    {
```

```
        [HttpPost]
        public IActionResult Post()
        {
            return Ok();
        }

        [HttpPut]
        public IActionResult Put()
        {
            return Ok();
        }

        [HttpDelete("{id}")]
        public IActionResult Delete(int id)
        {
            return Ok();
        }

        [HttpGet]
        public IActionResult GetAll()
        {
            return Ok();
        }

        [HttpGet("{id}")]
        public IActionResult GetById(int id)
        {
            return Ok();
        }
    }
}
```

Instalando o Swagger (Swashbuckle.AspNet Core)

Gerando a documentação dos serviços da API



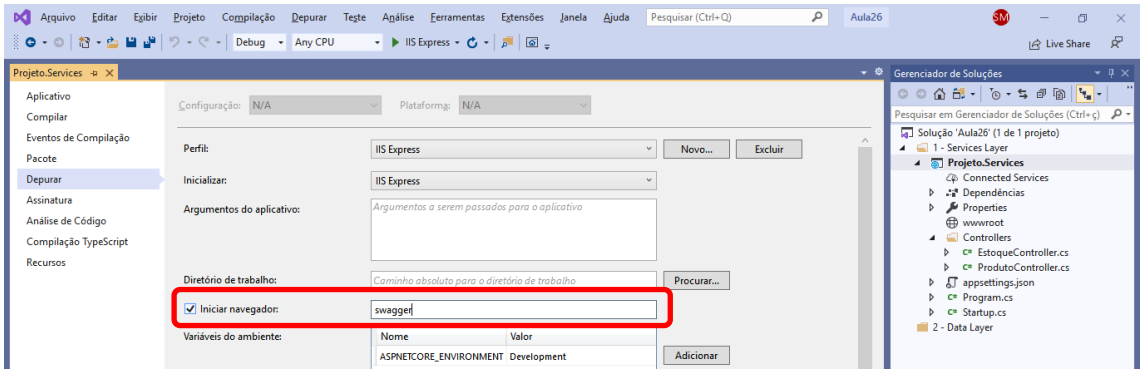
Startup.cs

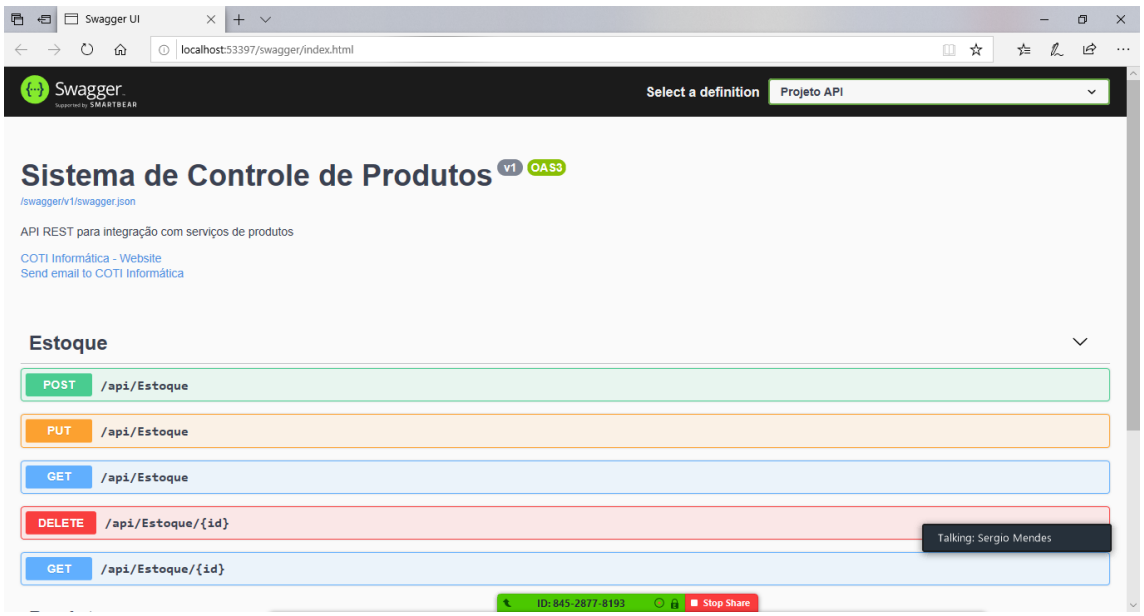
Configurando o Swagger para gerar a documentação do projeto API.

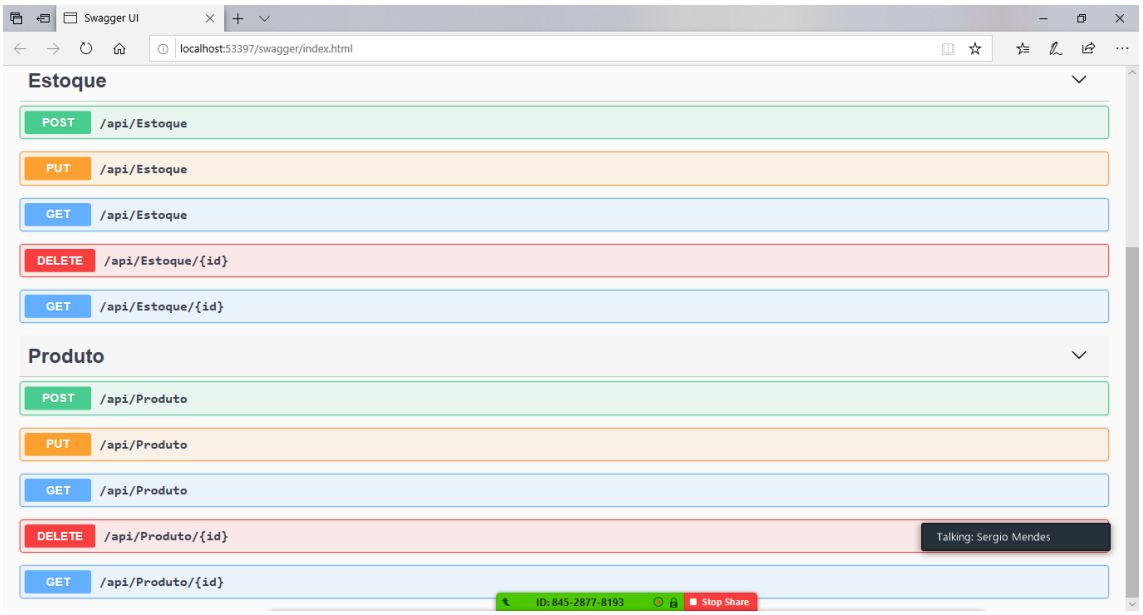
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using Microsoft.OpenApi.Models;

namespace Projeto.Services
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime.
        // Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc().SetCompatibilityVersion
                    (CompatibilityVersion.Version_2_1);

            #region Swagger

            services.AddSwaggerGen(
               c =>
               {
                   c.SwaggerDoc("v1", new OpenApiInfo
                   {
                       Title = "Sistema de Controle de Produtos",
                       Description = "API REST para integração
                                      com serviços de produtos",
                       Version = "v1",
                       Contact = new OpenApiContact
                       {
                           Name = "COTI Informática",
                           Url = new Uri("http://www.cotiinformatica.com.br/"),
                           Email = "contato@cotiinformatica.com.br"
                       }
                   });
               }
               );

            #endregion
        }
```

```
        // This method gets called by the runtime.
        // Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            #region Swagger

            app.UseSwagger();
            app.UseSwaggerUI(c =>
            {
                c.SwaggerEndpoint("/swagger/v1/swagger.json", "Projeto API");
            });

            #endregion

            app.UseMvc();
        }
    }
}
```

Configurando a página inicial do projeto para o Swagger:

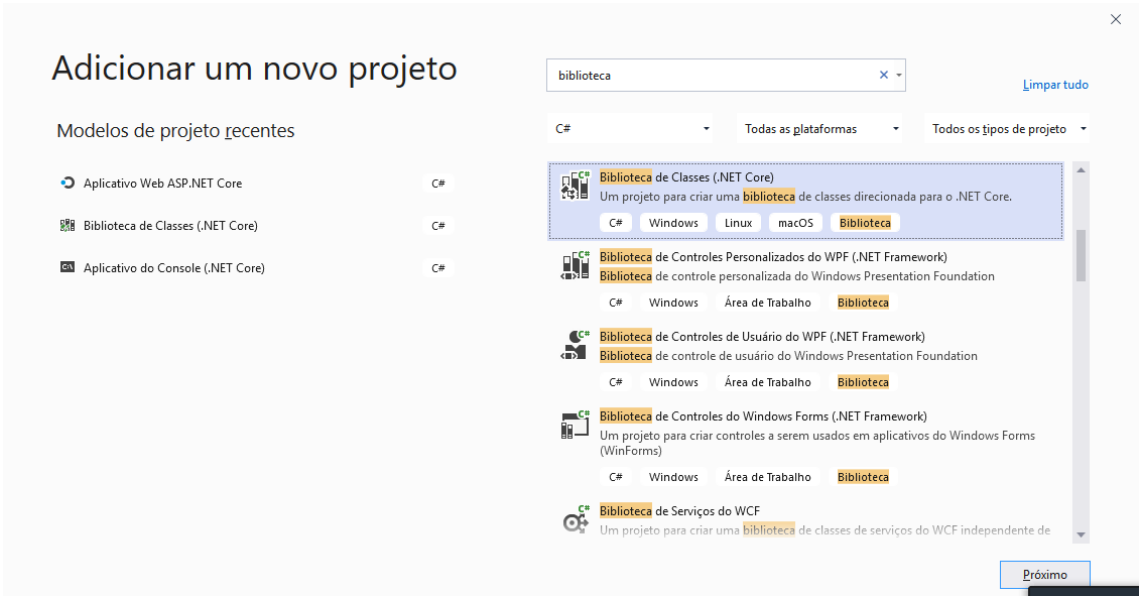


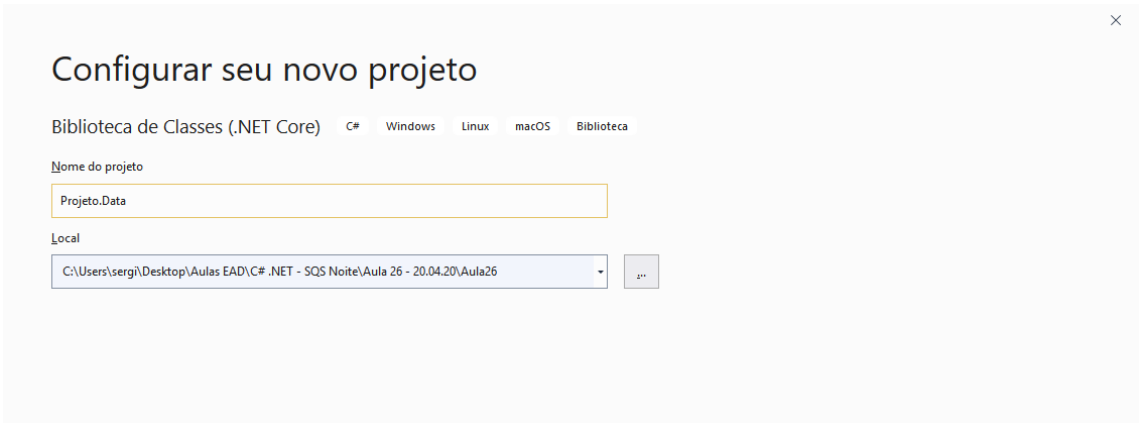


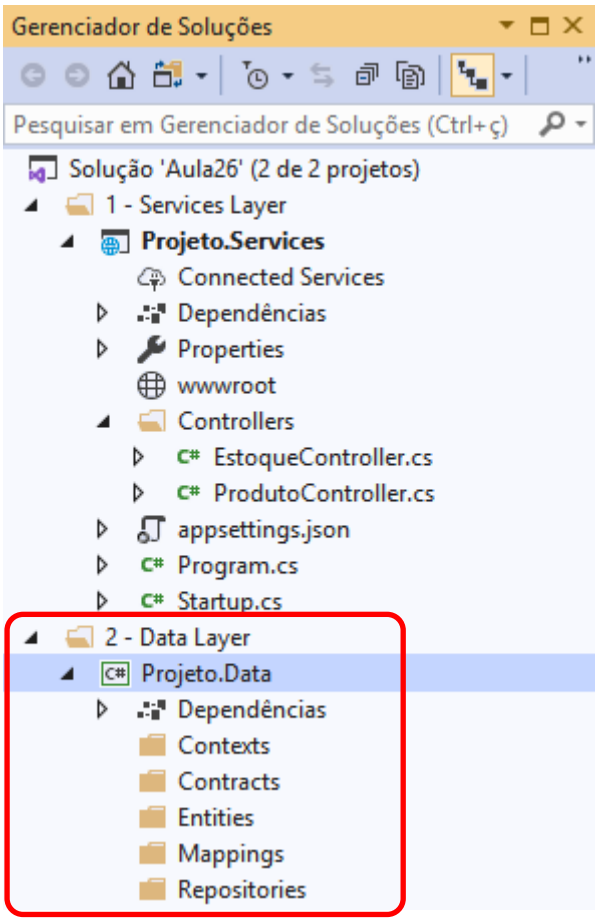


2 – Data Layer (Biblioteca de Classes .NET CORE)

Projeto voltado para desenvolvimento da camada de acesso a dados.



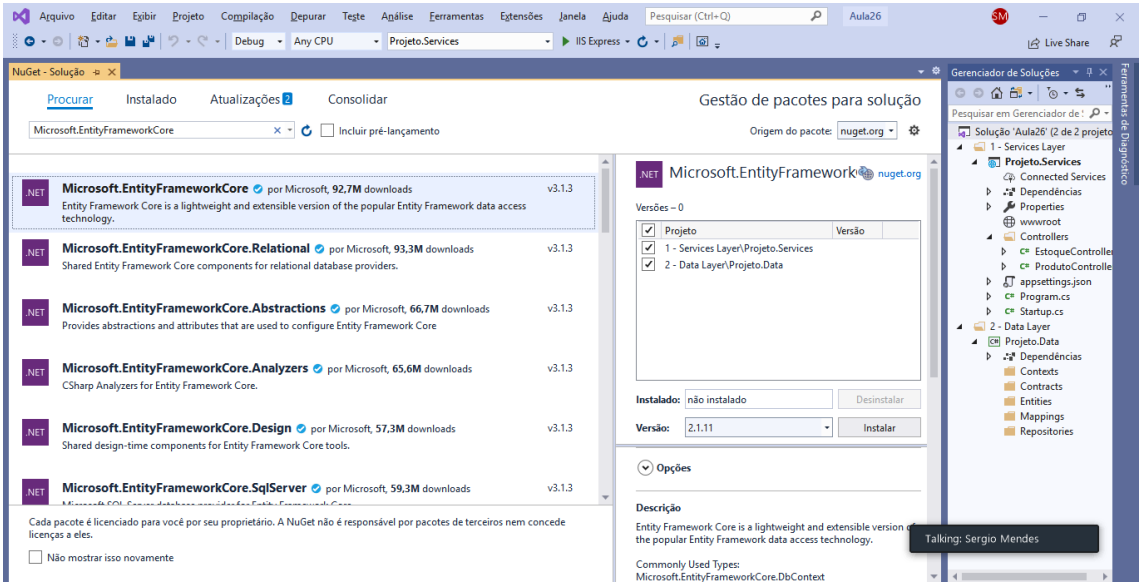




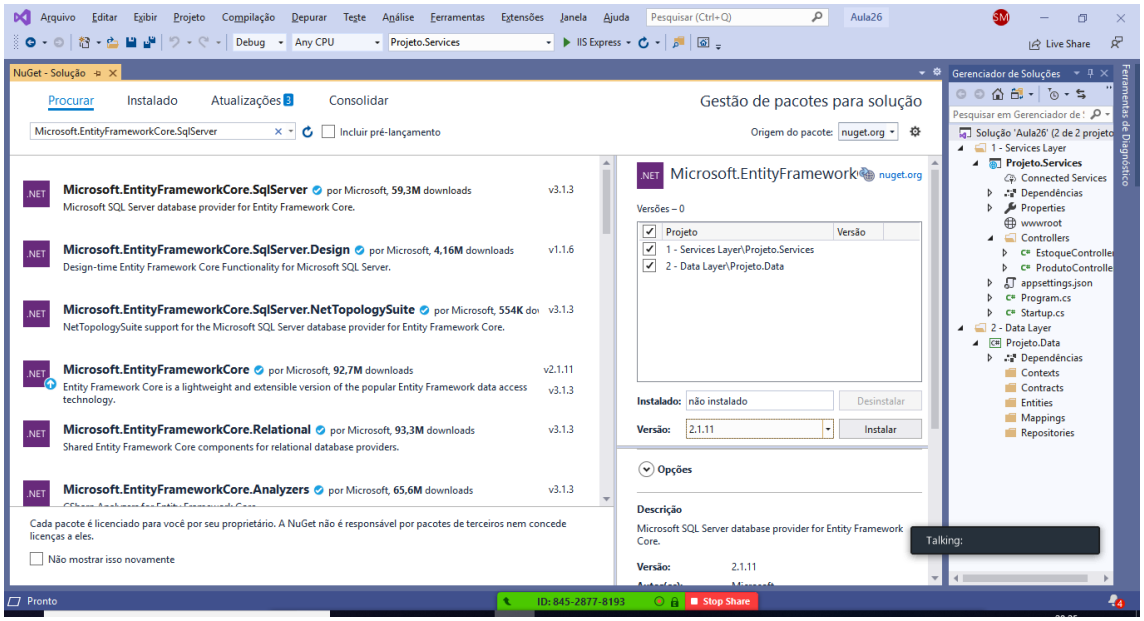
Instalando o EntityFramework

O EF deverá ser instalado tanto na camada de acesso a dados do projeto quanto na camada de apresentação (MVC ou API), pois deverá ser configurado na classe Startup.cs

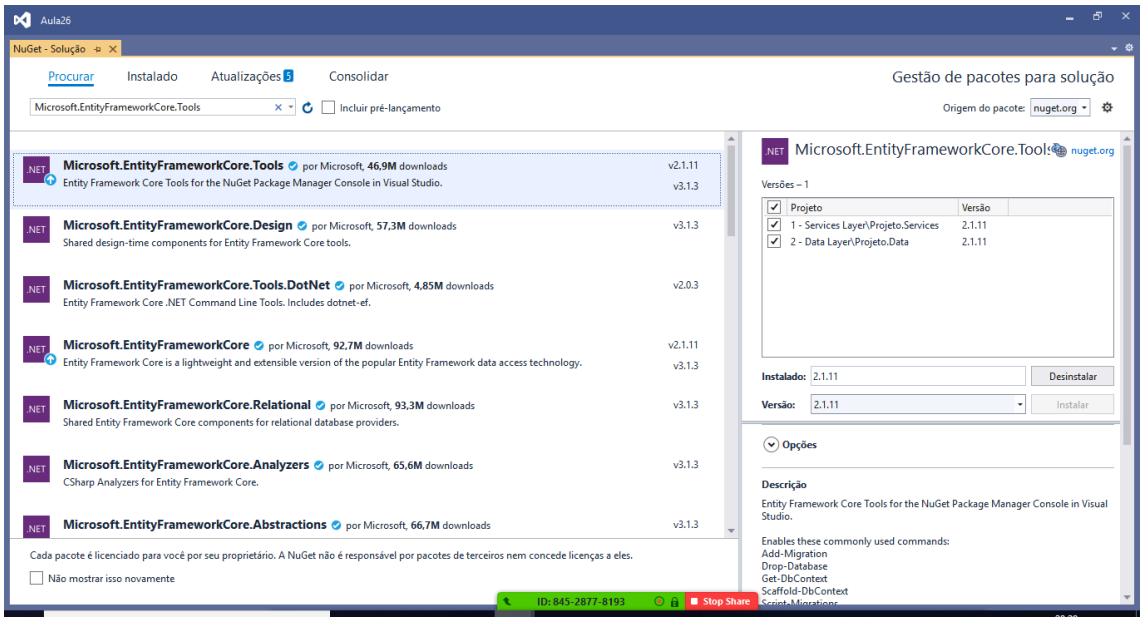
Microsoft.EntityFrameworkCore v.2.1.1



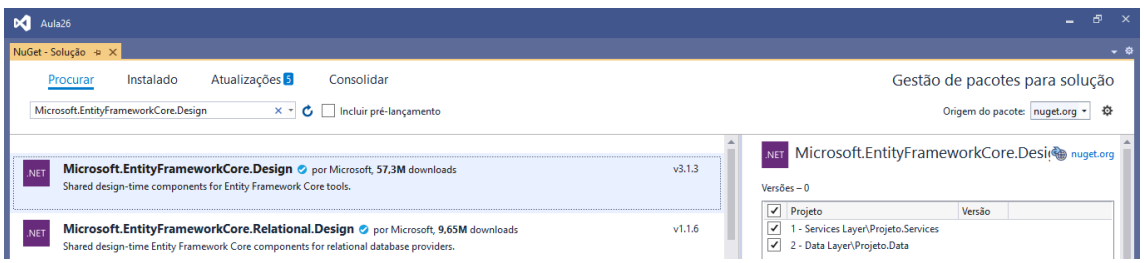
Microsoft.EntityFrameworkCore.SqlServer v.2.1.1



Microsoft.EntityFrameworkCore.Tools v.2.1.1

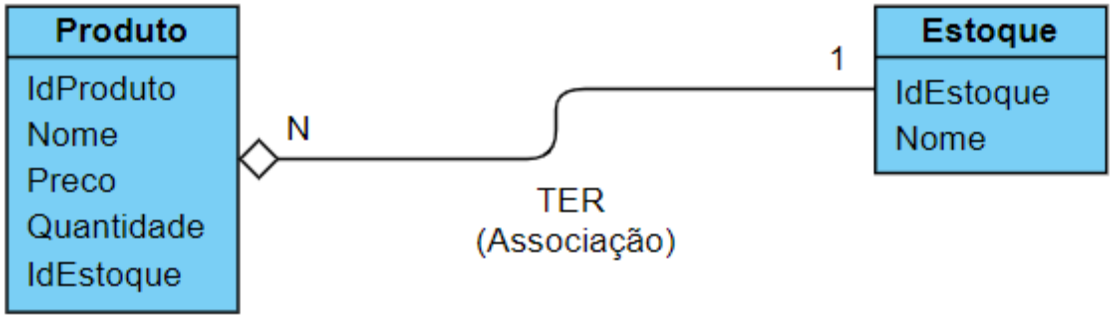


Microsoft.EntityFrameworkCore.Design v.2.1.1



Modelagem de entidades

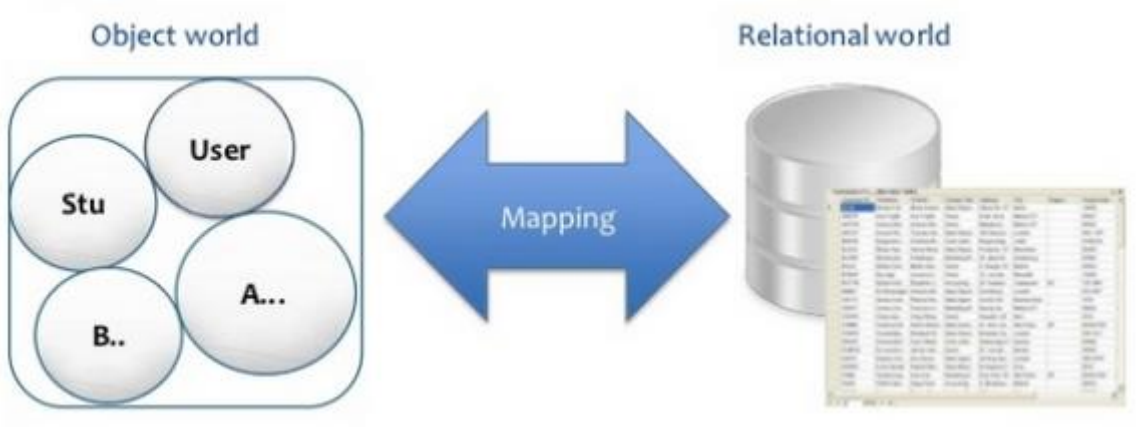
Diagrama de Classes




```
using System;
using System.Collections.Generic;
using System.Text;

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

        #region Associações

        public Estoque Estoque { get; set; }

        #endregion
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Entities
{
    public class Estoque
    {
        public int IdEstoque { get; set; }
        public string Nome { get; set; }

        #region Associações

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

        #endregion
    }
}
```

ORM – Object Relational Mapping
Mapeamento Objeto Relacional



/Mappings/EstoqueMapping.cs
Mapeamento da entidade **Estoque**

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Projeto.Data.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Mappings
{
     public class EstoqueMapping : IEntityTypeConfiguration<Estoque>
    {
        public void Configure(EntityTypeBuilder<Estoque> builder)
        {
            throw new NotImplementedException();
        }
    }
}
```

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Projeto.Data.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Mappings
{
    public class EstoqueMapping : IEntityTypeConfiguration<Estoque>
    {
        public void Configure(EntityTypeBuilder<Estoque> builder)
        {
            //nome da tabela no banco de dados (opcional)
            builder.ToTable("Estoque");

            //chave primária da tabela
            //para o EF, todo campo int que for definido como chave primária
```

```
            //já é criado como identity (auto-incremento)
            builder.HasKey(e => e.IdEstoque);

            //nome do campo id do estoque (opcional)
            builder.Property(e => e.IdEstoque)
                .HasColumnName("IdEstoque");

            //campo nome do estoque
            builder.Property(e => e.Nome)
                .HasColumnName("Nome")
                .HasMaxLength(150)
                .IsRequired();
        }
    }
}
```

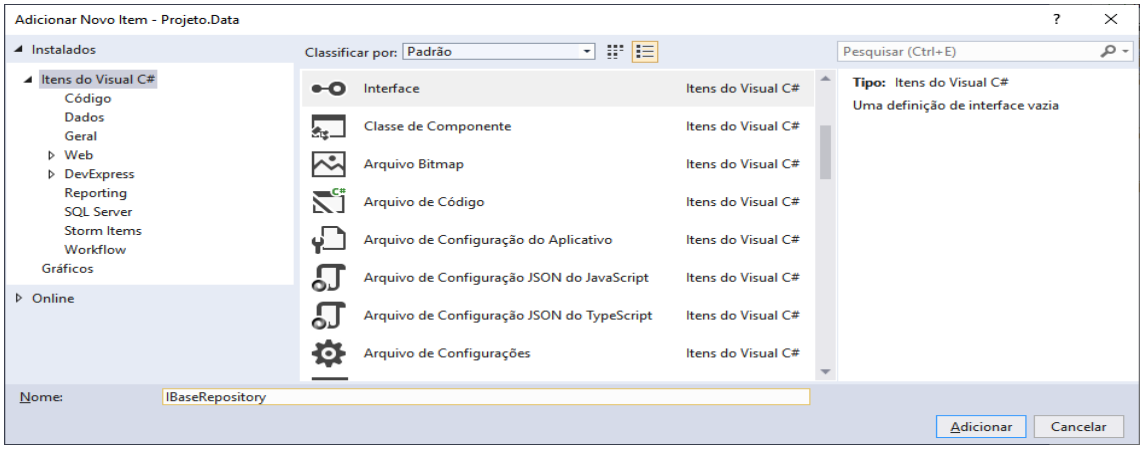
/Mappings/ProdutoMapping.cs

Mapeamento da entidade **Produto**

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using Projeto.Data.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Mappings
{
    public class ProdutoMapping : IEntityTypeConfiguration<Produto>
    {
        public void Configure(EntityTypeBuilder<Produto> builder)
        {
            //nome da tabela (opcional)
            builder.ToTable("Produto");

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

            //demais campos da entidade
            builder.Property(p => p.IdProduto)
                .HasColumnName("IdProduto");

            builder.Property(p => p.Nome)
                .HasColumnName("Nome")
                .HasMaxLength(150)
                .IsRequired();

            builder.Property(p => p.Preco)
                .HasColumnName("Preco")
                .HasColumnType("decimal(18,2)")
                .IsRequired();

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

            builder.Property(p => p.IdEstoque)
                .HasColumnName("IdEstoque")
                .IsRequired();
```

```
            #region Mapeamento dos Relacionamentos

            //Mapeamento de cardinalidade 1 para MUITOS
            builder.HasOne(p => p.Estoque) //Produto TEM 1 Estoque
                .WithMany(e => e.Produtos) //Estoque TEM MUITOS Produtos
                .HasForeignKey(p => p.IdEstoque); //Chave estrangeira

            #endregion
        }
    }
}
```

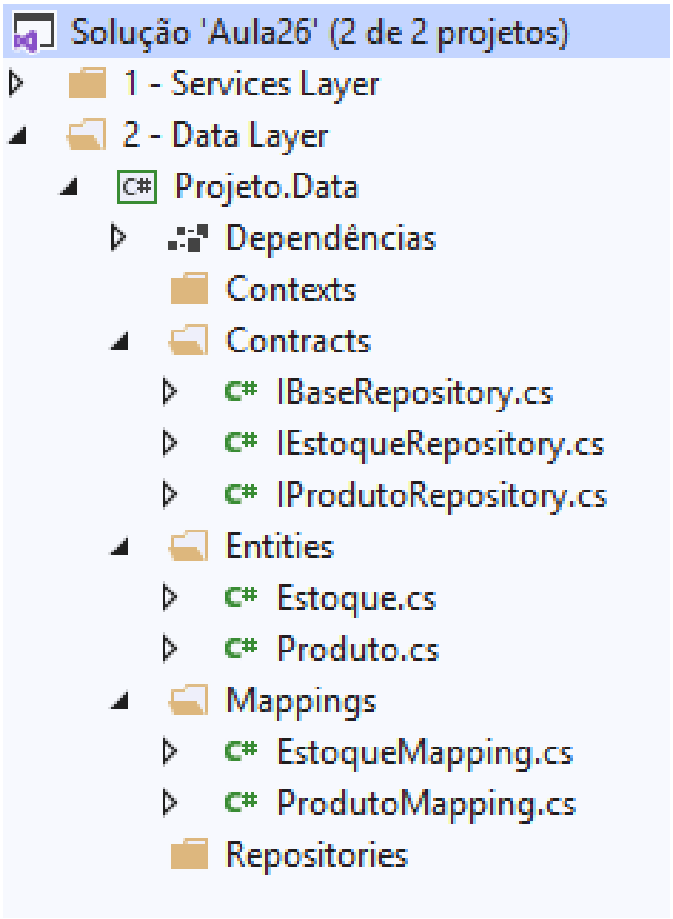
Criando as interfaces do Repositório:




```
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Contracts
{
    public interface IBaseRepository<T>
        where T : class
    {
        void Inserir(T entity);
        void Alterar(T entity);
        void Excluir(T entity);
        List<T> Consultar();
        T ObterPorId(int id);
    }
}
```

```
using Projeto.Data.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Contracts
{
    public interface IEstoqueRepository : IBaseRepository<Estoque>
    {

    }
}
```

```
using Projeto.Data.Entities;
using System;
using System.Collections.Generic;
using System.Text;

namespace Projeto.Data.Contracts
{
    public interface IProdutoRepository : IBaseRepository<Produto>
    {

    }
}
```



Continua...