Curso Asp.net Core e Angular

# Criar um projeto com versão específica:

C:\Projetos>**dotnet new globaljson --sdk-version 2.2.202**

Irá criar um arquivo global.json com o seguinte conteúdo:

{

"sdk": {

"version": "2.2.202"

}

}

C:\Projetos>**dotnet --version** irá mostrar a versão do global.json

C:\>**dotnet --version** irá mostrar a versão global do dotnet instalada

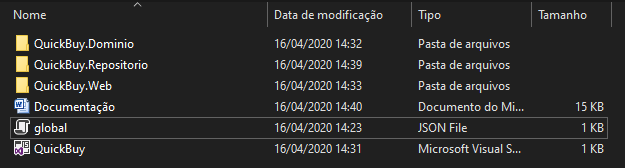
## Criar projetos de classes

dotnet new classlib --name QuickBuy.Dominio

dotnet new classlib --name QuickBuy.Repositorio

## Criar projeto de angular

**dotnet new angular --name QuickBuy.Web**



## Referenciar os projetos na solução

dotnet sln add "quickbuy.Dominio/QuickBuy.Dominio.csproj"

dotnet sln add "quickbuy.Repositorio/QuickBuy.Repositorio.csproj"

dotnet sln add "quickbuy.Web/QuickBuy.Web.csproj"

# Instalar o Node.Js

Permite desenvolver soluções compiláveis utilizando a linguagem javascript e no lado do servidor. O Node.js faz o javascript ser uma tecnologia de servidor assim como é o ASP.NET Core e o Java só para citar como exemplo.

# Instalar o Angular CLI

npm install -g @angular/cli

Startup.cs

Program.cs

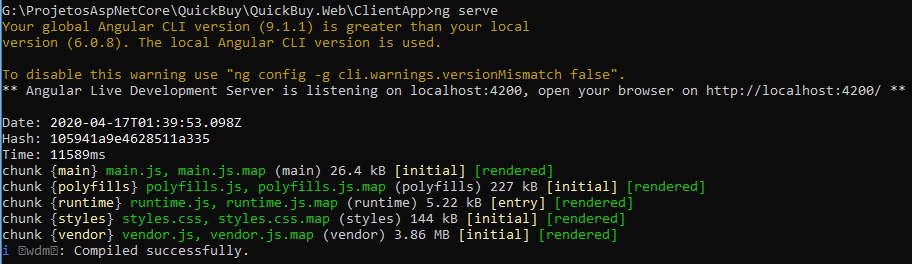
# Executar servidor do Angular CLI, sem depender do processo do asp.net core

1. No prompt de comando do Windows:

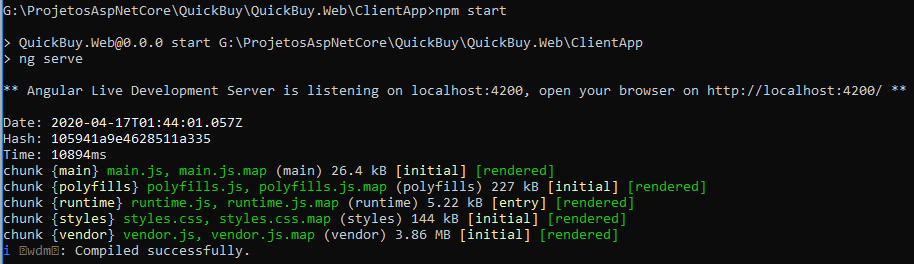
Entrar na pasta do ClientApp:



Publicar aplicativo do angular em um servidor local:



Usar o comando **npm start**



Vai buildar a solução e chamar o ng server.

1. Arquivo startup.cs:

app.UseSpa(spa =>

{

// To learn more about options for serving an Angular SPA from ASP.NET Core,

// see https://go.microsoft.com/fwlink/?linkid=864501

spa.Options.SourcePath = "ClientApp";

if (env.IsDevelopment())

{

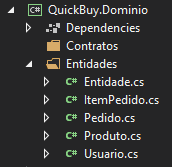
//spa.UseAngularCliServer(npmScript: "start");

spa.UseProxyToSpaDevelopmentServer("http://localhost:4200/");

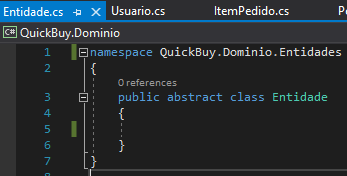
}

});

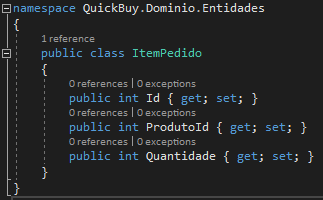
# Criar as classes:



Entidade.cs



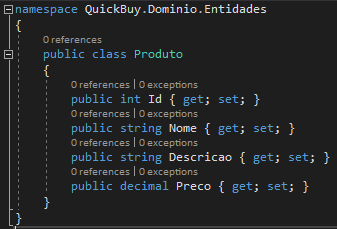
ItemPedido



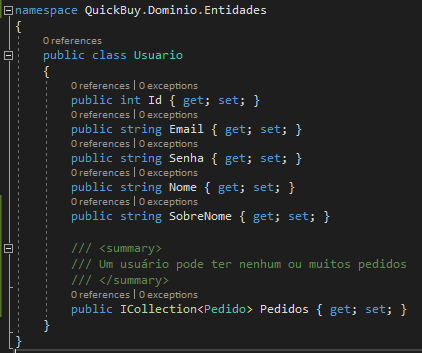
Pedido.cs

# 

Produto.cs



Usuario.cs



# Classe Pedido.cs

Adicionar mais propriedades:

public DateTime DataPrevisaoEntrega { get; set; }

public string CEP { get; set; }

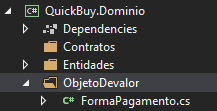
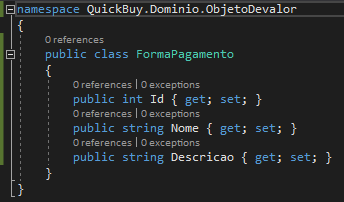
public string Estado { get; set; }

public string Cidade { get; set; }

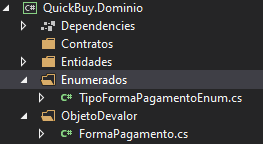
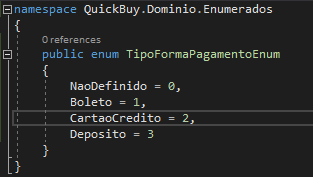
public string EnderecoCompleto { get; set; }

public int NumeroEndereco { get; set; }

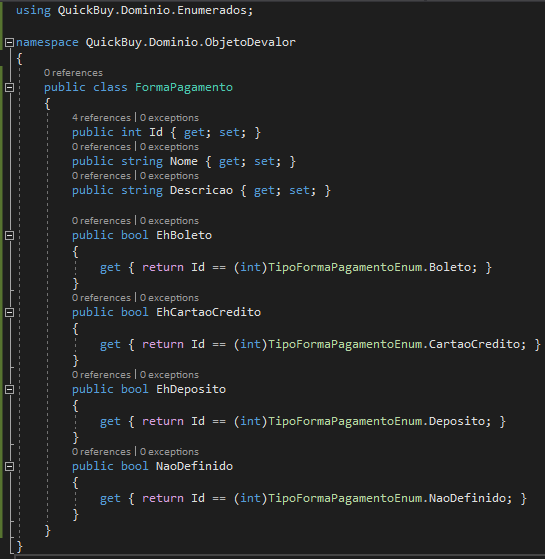
# Criar Pasta “ObjetoDeValor” e adicionar a classe FormaPagamento.cs:

# Criar pasta “Enumerados” e adicionar a classe TipoFormaPagamentoEnum.cs:

# Modificar a classe FormaPagamento.cs



# Modificar a classe Pedido:

using QuickBuy.Dominio.ObjetoDeValor;

using System;

using System.Collections.Generic;

namespace QuickBuy.Dominio.Entidades

{

public class Pedido

{

public int Id { get; set; }

public DateTime DataPedido { get; set; }

public int UsuarioId { get; set; }

public DateTime DataPrevisaoEntrega { get; set; }

public string CEP { get; set; }

public string Estado { get; set; }

public string Cidade { get; set; }

public string EnderecoCompleto { get; set; }

public int NumeroEndereco { get; set; }

public int FormaPagamentoId { get; set; }

public FormaPagamento FormaPagamento { get; set; }

/// <summary>

/// Pedido deve ter um item de pedido ou muitos itens de pedidos

/// </summary>

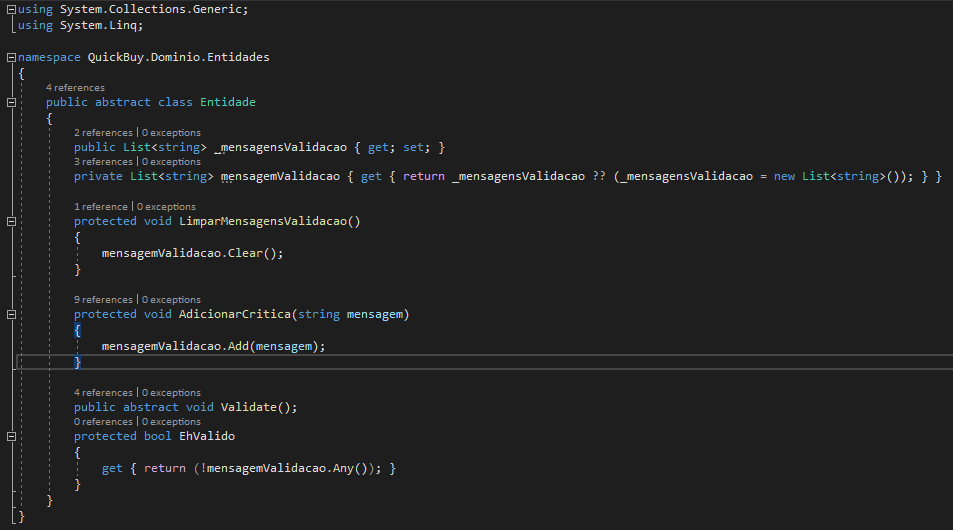
public ICollection<ItemPedido> ItensPedido { get; set; }

}

}

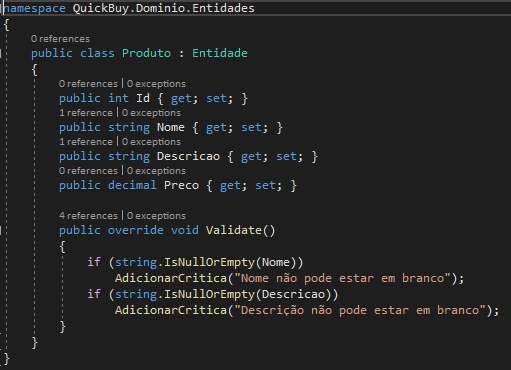
# Implementar Validação nos Dados

## Entidade.cs



As classes Produto, Pedido, Itens de Pedido e Usuario devem herdar a classe Entidade.

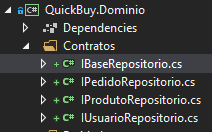
## Classe Produto:



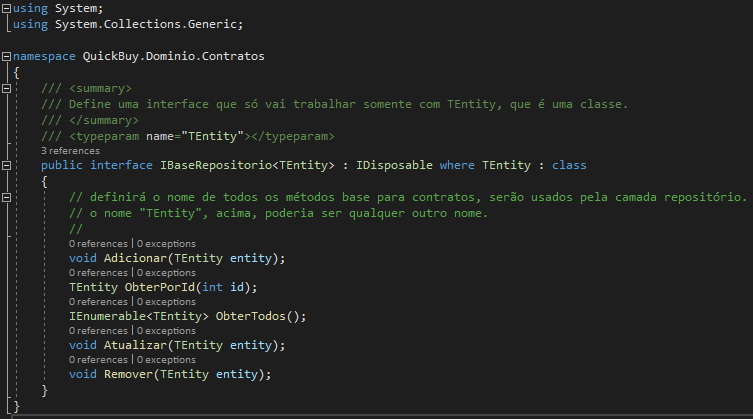
# Projeto Repositório

Comunicação com o Banco de Dados

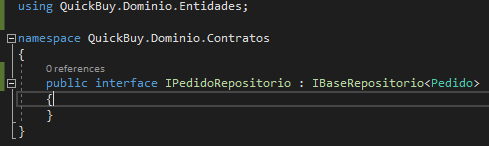
## Criar as interfaces (contratos):



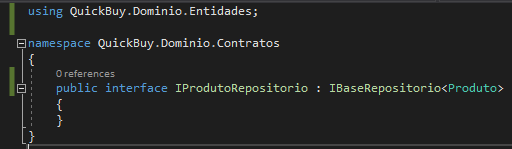
### Interface Base: IBaseRepositorio



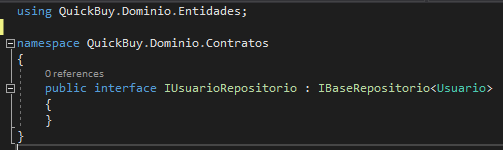
### Interface IPedidoRepositorio:



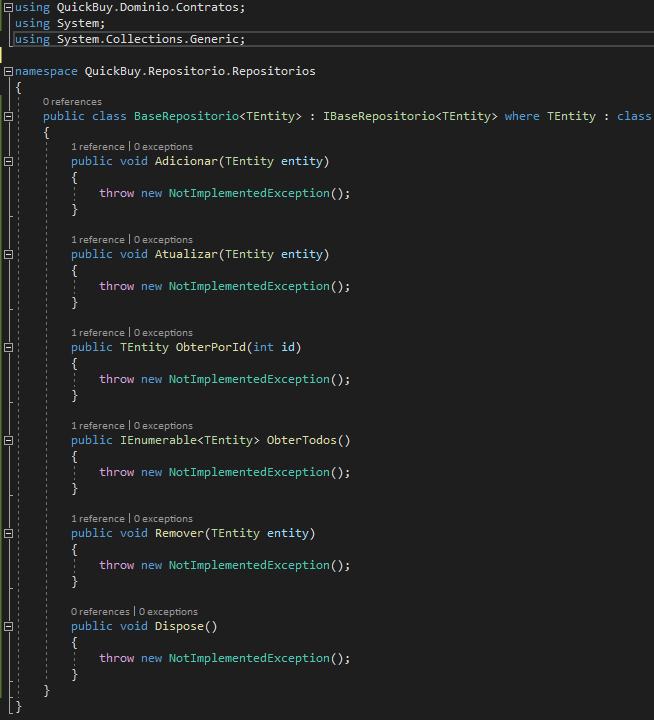
### Interface IProdutoRepositorio:



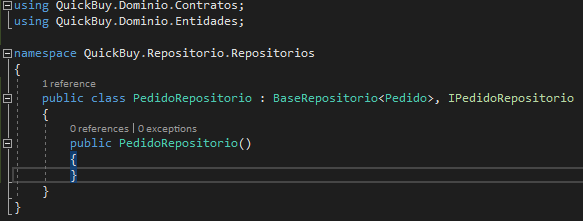
### Interface IUsuarioRepositorio:



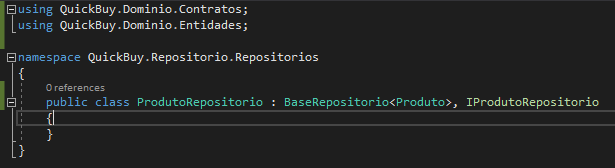
### Adicionar pasta “Repositorios” à QuickBuy.Dominio e criar classe BaseRepositorio.cs dentro.



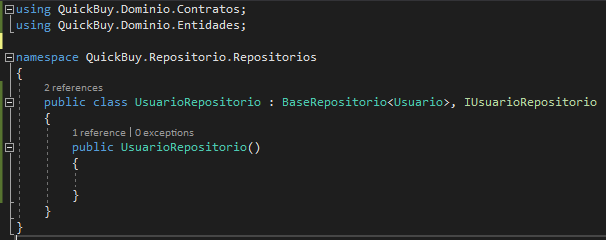
### Acrescentar classe “PedidoRepositorio”



### Acrescentar classe “ProdutoRepositorio”



### Acrescentar classe “UsuarioRepositorio”

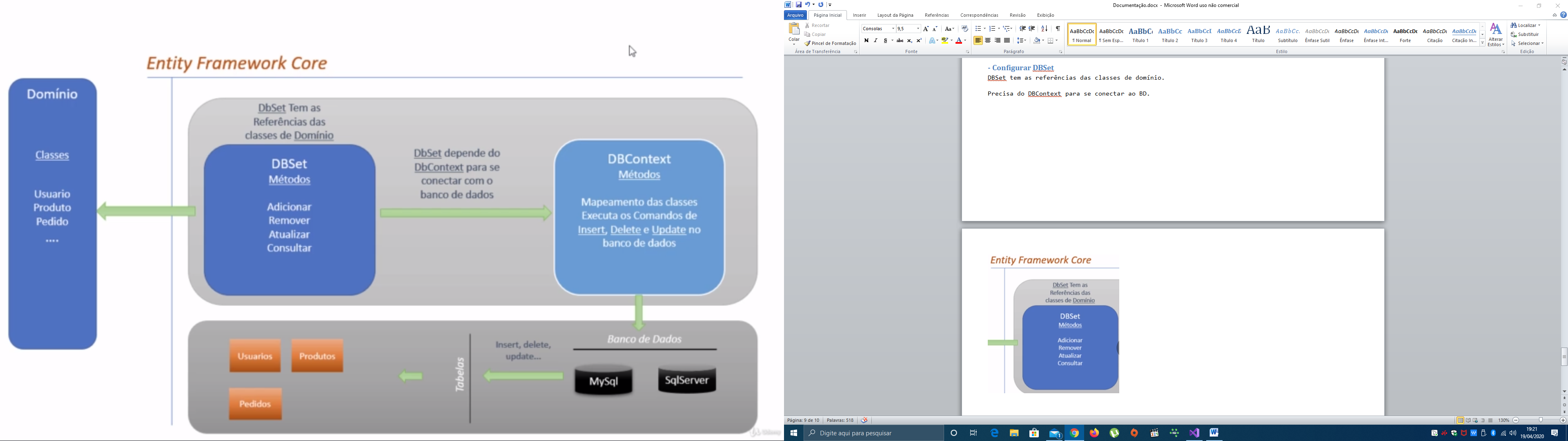


## Entity Framework Core (conectar ao banco de dados)

Ferramenta de ORM (Object Relational Mapping – Mapeamento Objeto Relacional)

### - Configurar DBSet

DBSet tem as referências das classes de domínio.



### Instalar Entity Framework Core

Clicar com o botão direito sobre o projeto “QuickBuy.Repositorio” e escolher Manage NuGet Packages...

- Instalar **Pomelo.EntityFrameworkCore.MySql**

- Instalar **Microsoft.EntityFrameworkCore.Proxies**