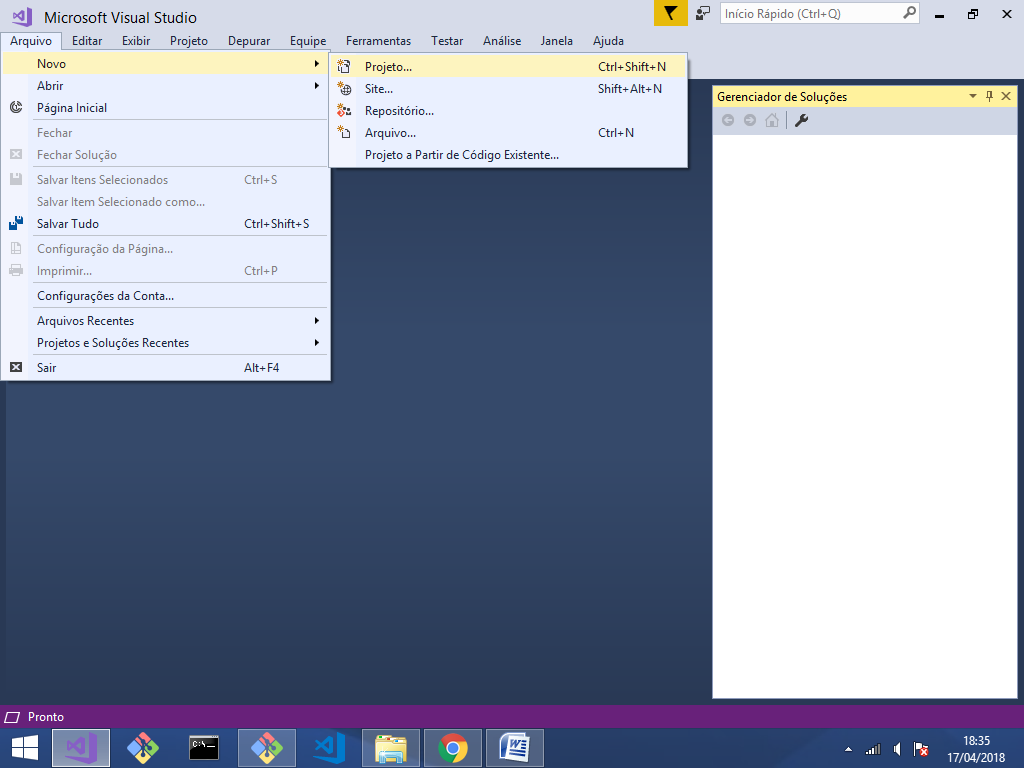
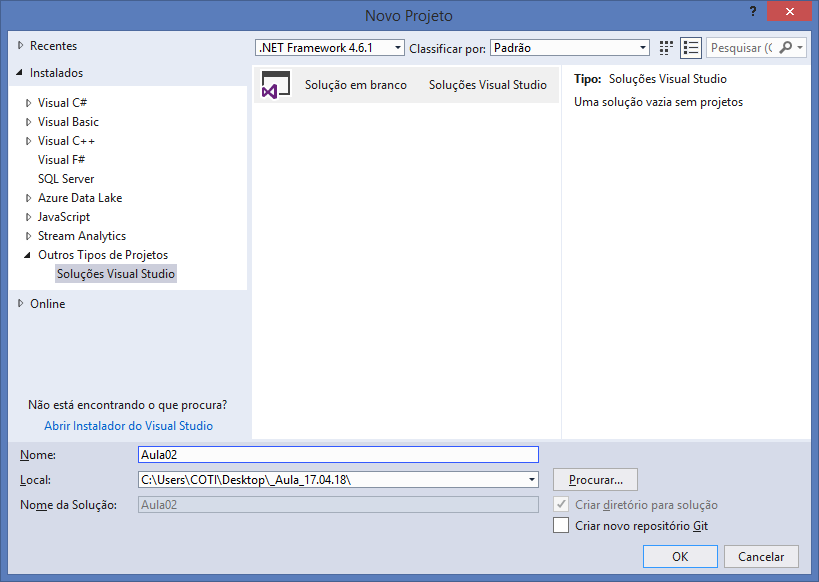
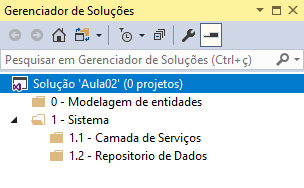
**Abrindo o visual studio:**

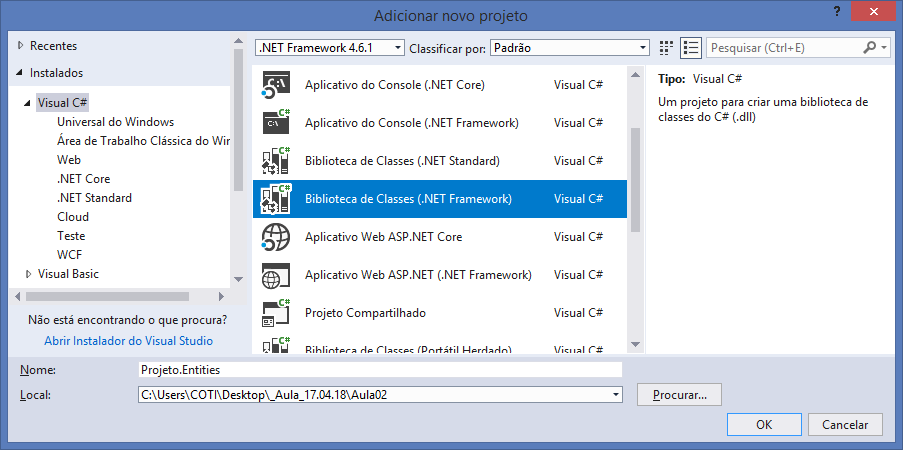






0 - Modelagem de entidades

Class Library (Biblioteca de Classes)



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Entities

{

public class Produto

{

//prop + 2x[tab]

public int IdProduto { get; set; }

public string Nome { get; set; }

public decimal Preco { get; set; }

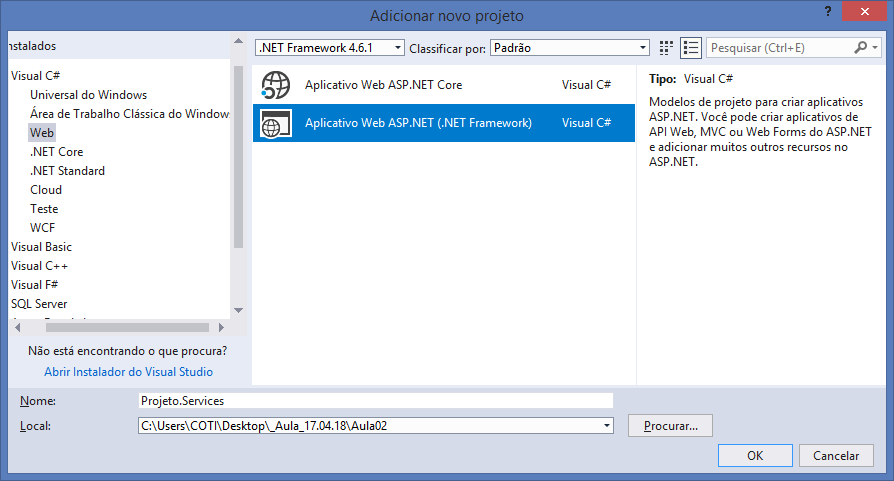
public int Quantidade { get; set; }

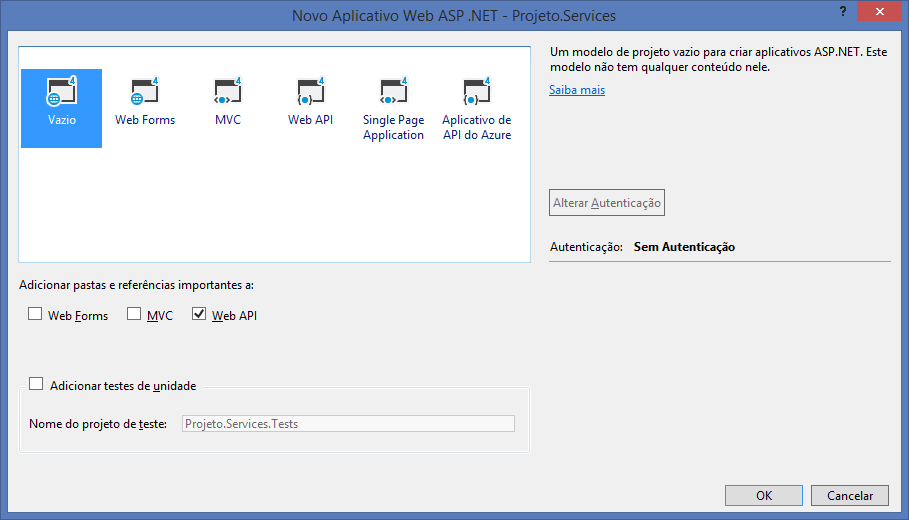
}

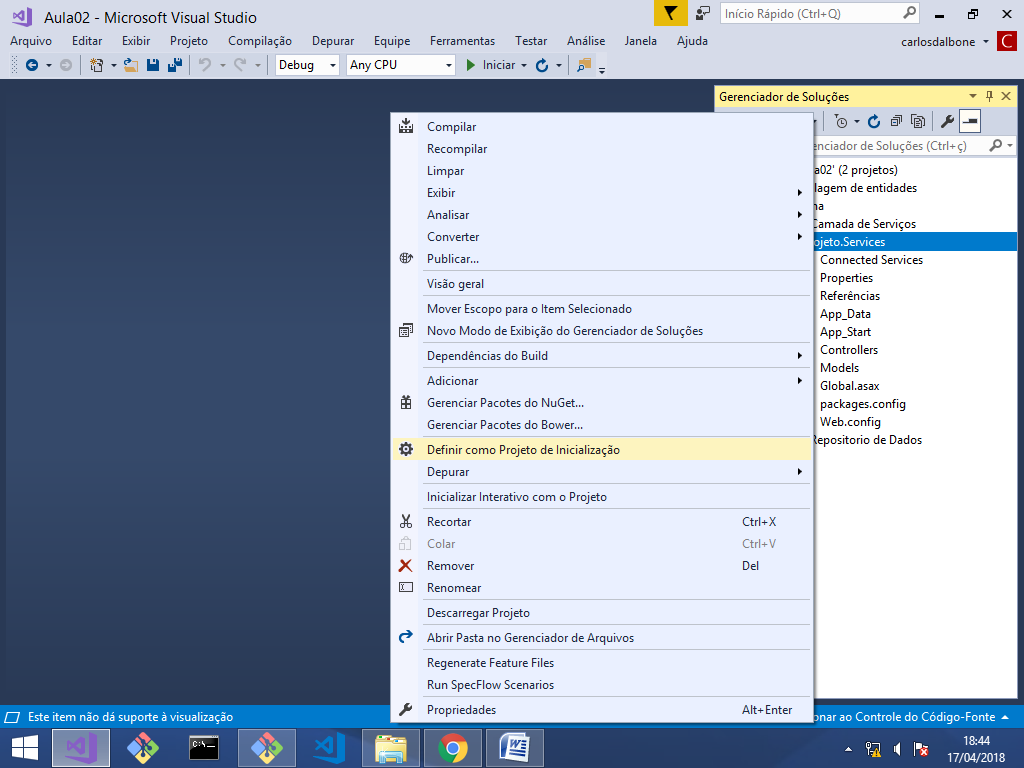
}

Arquitetura do projeto:

* Asp.Net WebApi
* Entity Framework
* Simple Injector

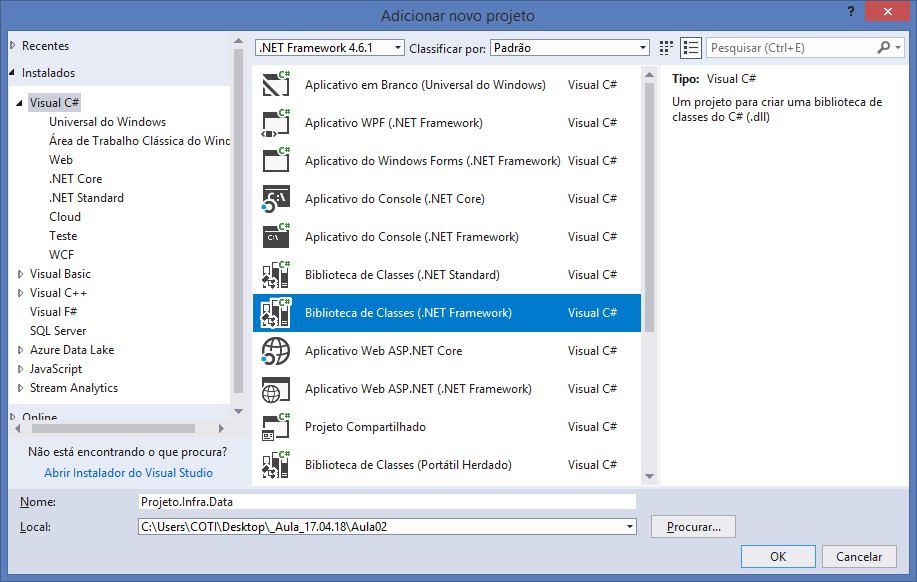






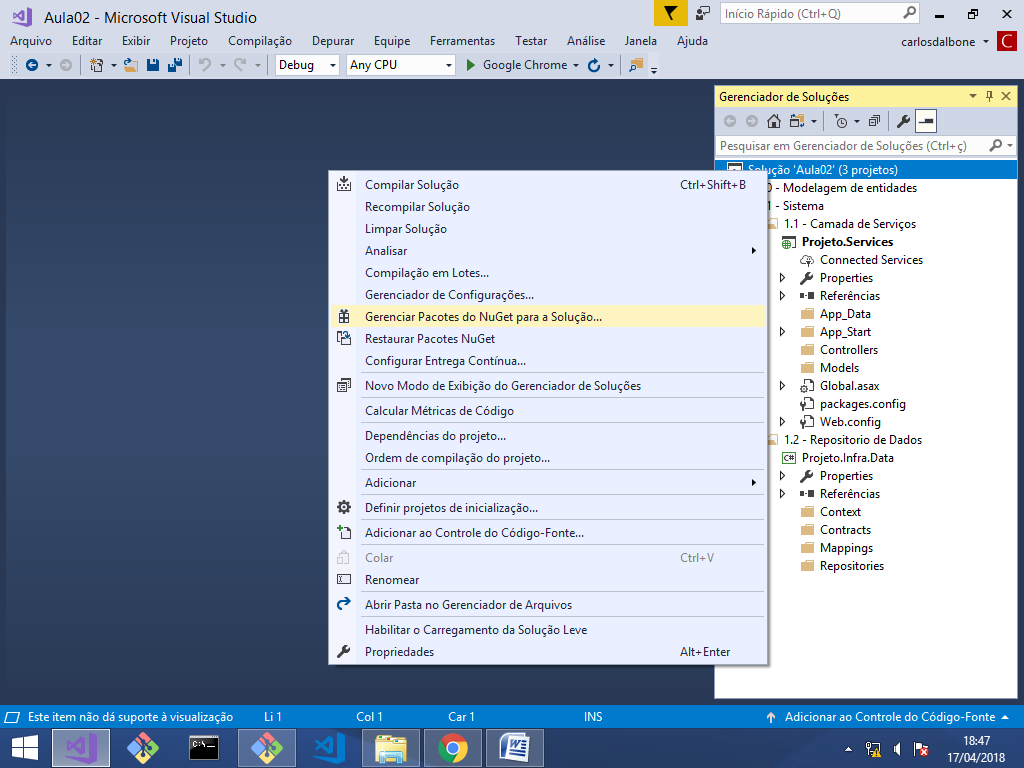
1.2 - Repositorio de dados

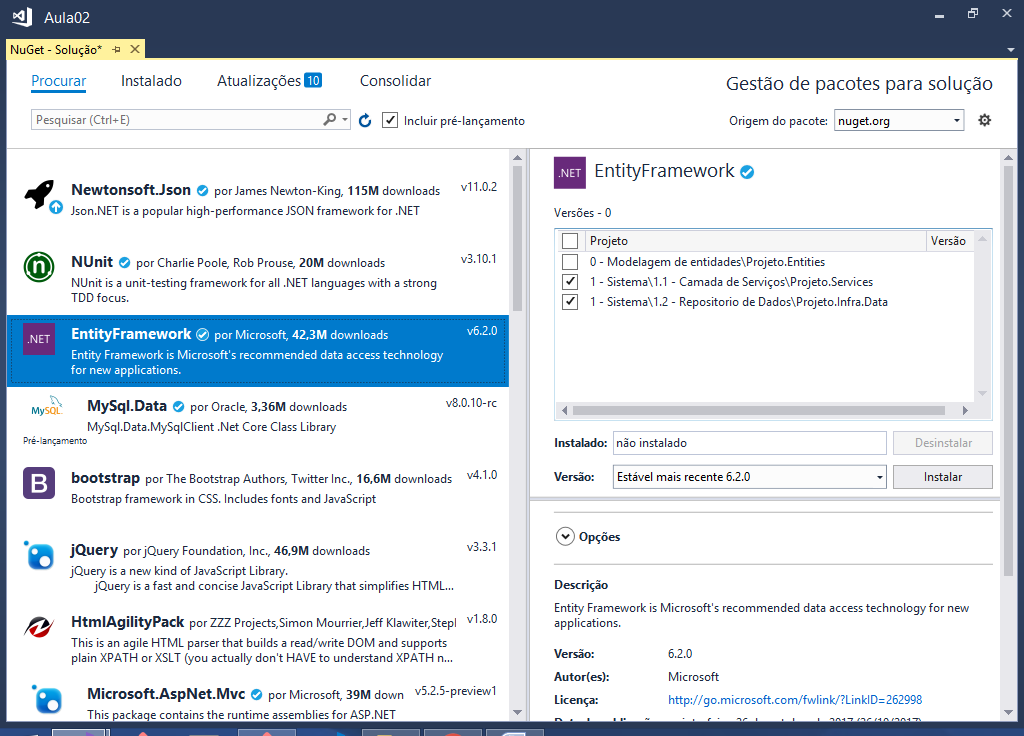
Infra estrutura



Instalando o EntityFramework

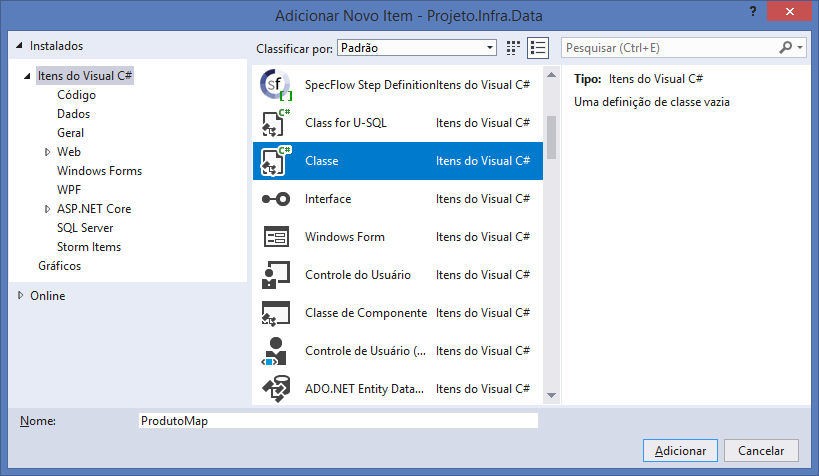
Gerenciador de pacotes do NuGet





ORM - Mapeamento Objeto Relacional

Mapear as classes de entidade para que sejam interpretadas pelo EntityFramewok 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; //mapeamento..

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

namespace Projeto.Infra.Data.Mappings

{

//Classe de mapeamento para a entidade 'Produto'..

public class ProdutoMap : EntityTypeConfiguration<Produto>

{

//construtor..

public ProdutoMap()

{

//nome da tabela..

ToTable("Produto");

//chave primária..

HasKey(p => p.IdProduto);

//demais campos da tabela..

Property(p => p.IdProduto)

.HasColumnName("IdProduto");

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")

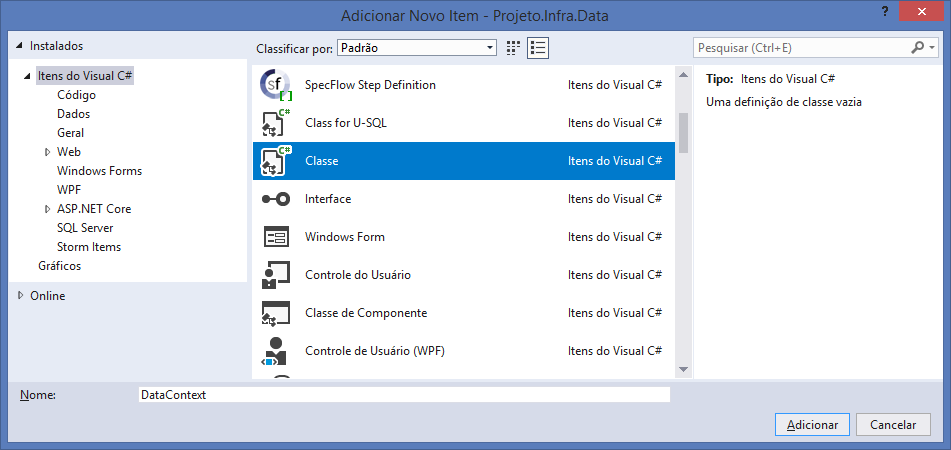
.IsRequired();

}

}

}

**Classe para conexão com o banco de   
dados através do EntityFramework**



using Projeto.Entities;

using Projeto.Infra.Data.Mappings;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Infra.Data.Context

{

//Regra 1) Herdar DbContext..

public class DataContext : DbContext

{

//Regra 2) Construtor que envia para o DbContext o

//caminho da connectionstring do banco de dados

public DataContext()

: base(ConfigurationManager.ConnectionStrings

["aula"].ConnectionString)

{

}

//Regra 3) Sobrescrever o método OnModelCreating da classe DbContext..

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

//adicionando cada classe mapeada pelo entityframework..

modelBuilder.Configurations.Add(new ProdutoMap());

}

//Regra 4) Declarar uma propriedade DbSet para cada entidade..

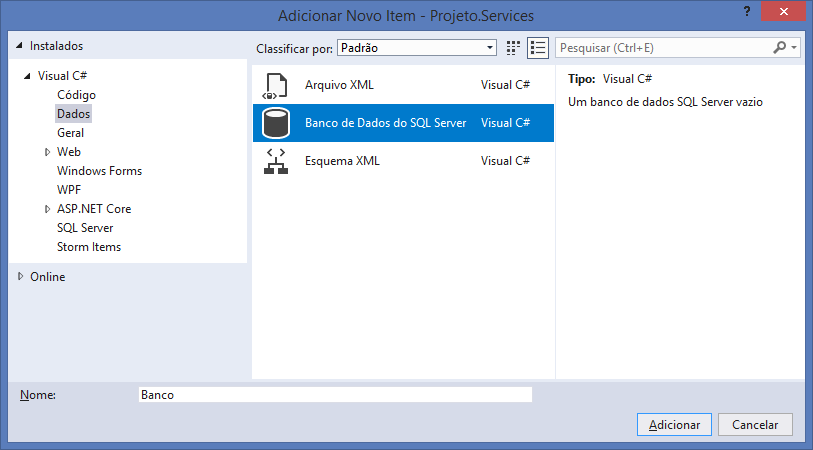
public DbSet<Produto> Produto { get; set; }

}

}

**Criando uma base de dados:**

MDF - Master Database File



\Web.config.xml

Mapeando a connectionstring:

<!-- mapeamento da connectionstring.. -->

<connectionStrings>

<add

name="aula"

connectionString="Data Source=(LocalDB)\MSSQLLocalDB;

AttachDbFilename=C:\Users\COTI\Desktop\

\_Aula\_17.04.18\Aula02\Projeto.Services\App\_Data\

Banco.mdf;Integrated Security=True"

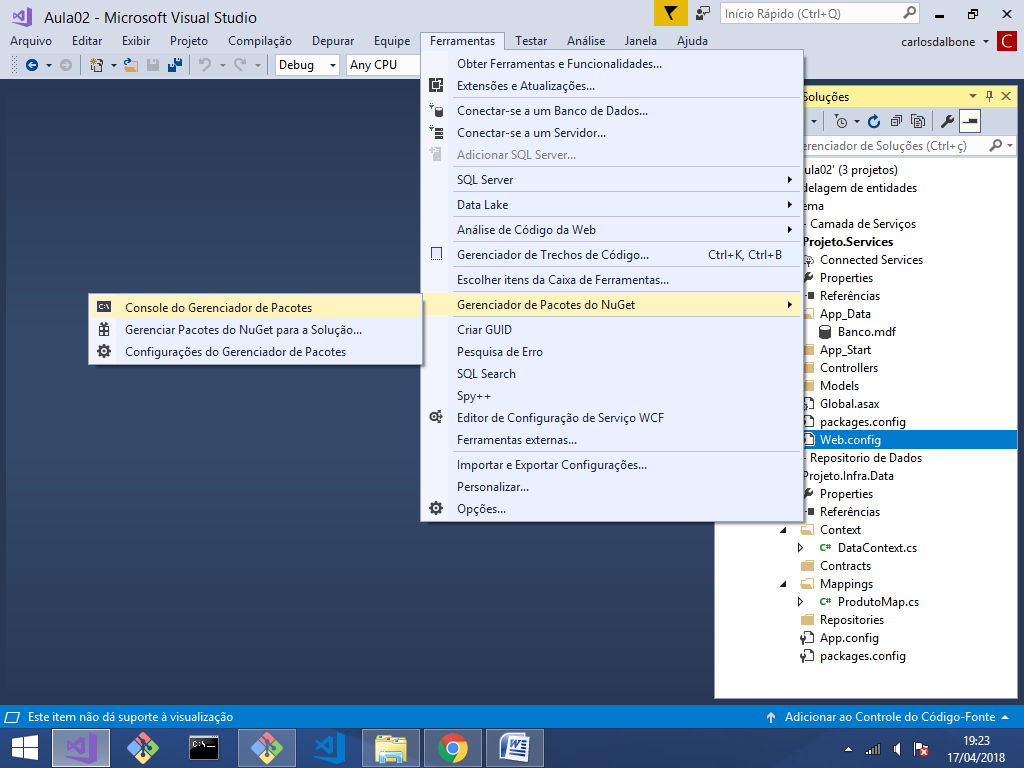
/>

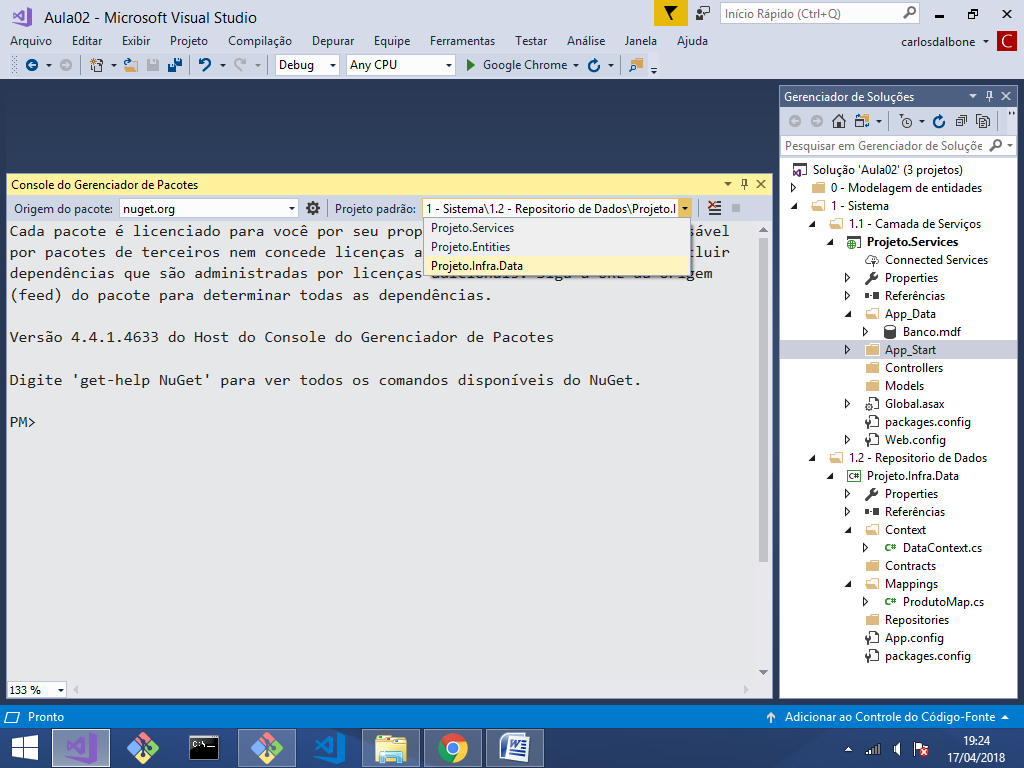
</connectionStrings>

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

Migrations

CodeFirst





**PM> enable-migrations -force**

PM> enable-migrations -force

Checking if the context targets an existing database...

Code First Migrations enabled for project Projeto.Infra.Data.

PM>

namespace Projeto.Infra.Data.Migrations

{

using System;

using System.Data.Entity;

using System.Data.Entity.Migrations;

using System.Linq;

internal sealed class Configuration : DbMigrationsConfiguration

<Projeto.Infra.Data.Context.DataContext>

{

public Configuration()

{

//habilitar CREATE e ALTER..

AutomaticMigrationsEnabled = true;

//habilitar DROP..

AutomaticMigrationDataLossAllowed = true;

}

protected override void Seed

(Projeto.Infra.Data.Context.DataContext context)

{

// This method will be called after migrating to the latest version.

// You can use the DbSet<T>.AddOrUpdate() helper extension method

// to avoid creating duplicate seed data.

}

}

}

Gerando as tabelas no banco de dados:

**PM> update-database -verbose**

CREATE TABLE [dbo].[Produto] (

[IdProduto] [int] NOT NULL IDENTITY,

[Nome] [nvarchar](50) NOT NULL,

[Preco] [decimal](18, 2) NOT NULL,

[Quantidade] [int] NOT NULL,

CONSTRAINT [PK\_dbo.Produto] PRIMARY KEY ([IdProduto])

)

CREATE TABLE [dbo].[\_\_MigrationHistory] (

[MigrationId] [nvarchar](150) NOT NULL,

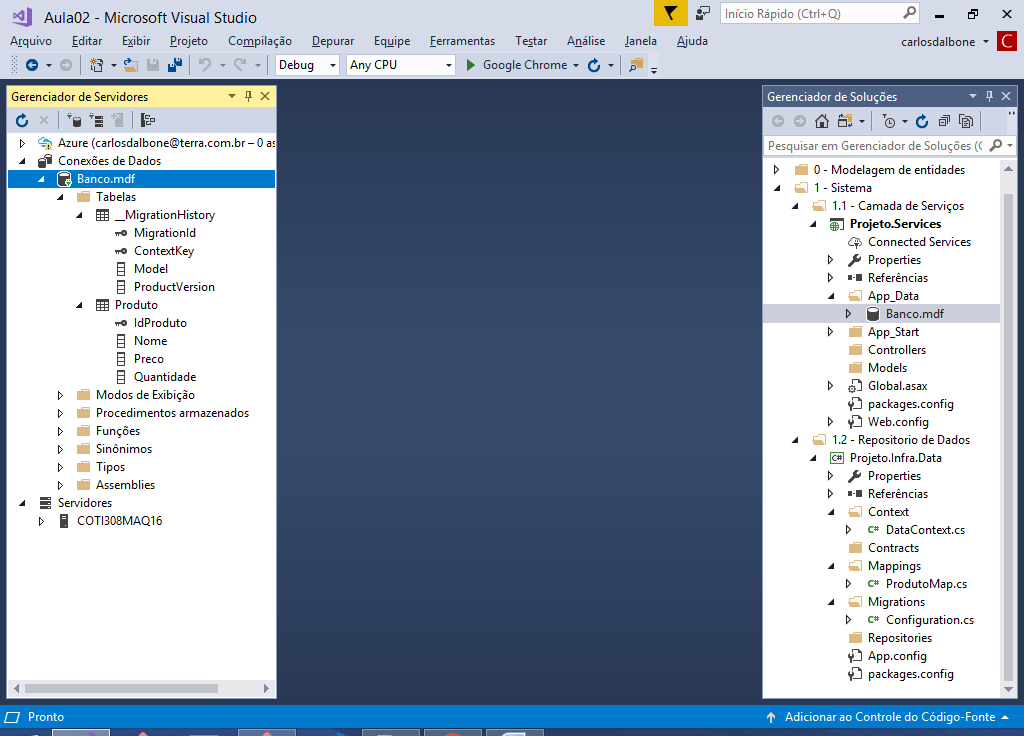
[ContextKey] [nvarchar](300) NOT NULL,

[Model] [varbinary](max) NOT NULL,

[ProductVersion] [nvarchar](32) NOT NULL,

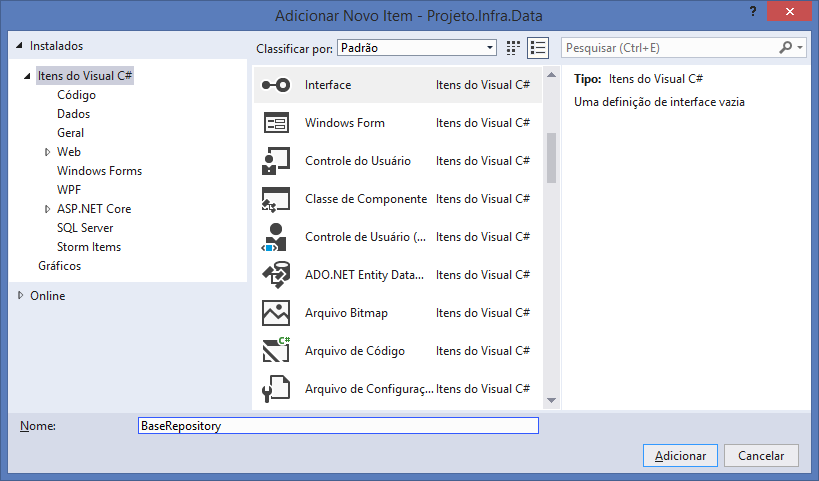
CONSTRAINT [PK\_dbo.\_\_MigrationHistory] PRIMARY KEY ([MigrationId], [ContextKey])

)



**Criando uma interface para a classe de repositorio de produto:**

\*\* Primeiro criaremos uma interface generica:



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Infra.Data.Contracts

{

public interface IBaseRepository<T>

where T : class

{

void Insert(T obj);

void Update(T obj);

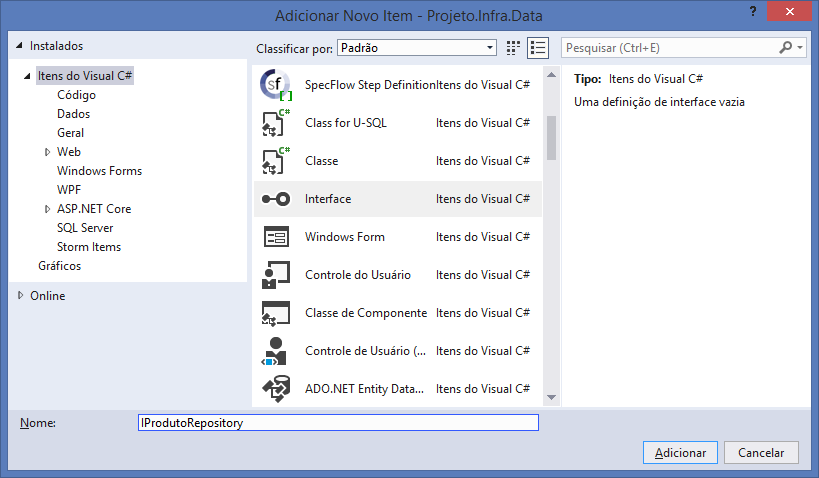
void Delete(T obj);

List<T> FindAll();

T FindById(int id);

}

}



using Projeto.Entities;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Infra.Data.Contracts

{

public interface IProdutoRepository

: IBaseRepository<Produto>

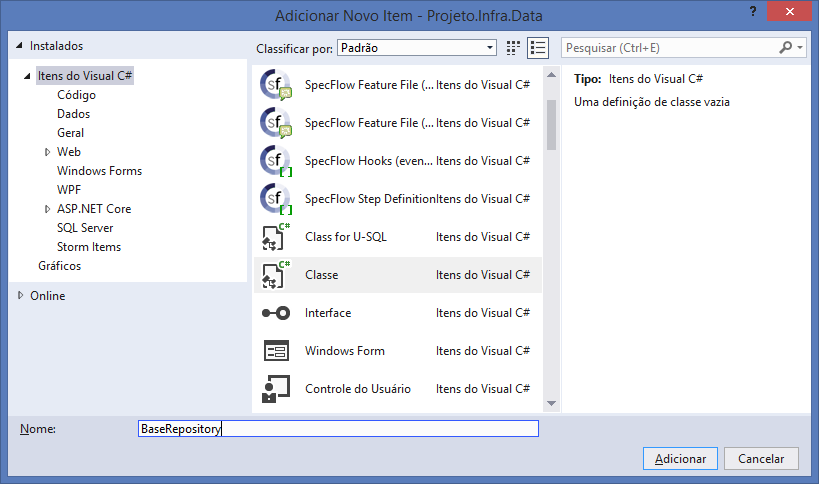
{

List<Produto> FindByNome(string nome);

}

}

**Implementando as interfaces:**



using Projeto.Infra.Data.Context;

using Projeto.Infra.Data.Contracts;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Infra.Data.Repositories

{

public abstract class BaseRepository<T> : IBaseRepository<T>

where T : class

{

public virtual void Insert(T obj)

{

using (DataContext d = new DataContext())

{

d.Entry(obj).State = EntityState.Added;

d.SaveChanges();

}

}

public virtual void Update(T obj)

{

using (DataContext d = new DataContext())

{

d.Entry(obj).State = EntityState.Modified;

d.SaveChanges();

}

}

public virtual void Delete(T obj)

{

using (DataContext d = new DataContext())

{

d.Entry(obj).State = EntityState.Deleted;

d.SaveChanges();

}

}

public virtual List<T> FindAll()

{

using (DataContext d = new DataContext())

{

return d.Set<T>().ToList();

}

}

public virtual T FindById(int id)

{

using (DataContext d = new DataContext())

{

return d.Set<T>().Find(id);

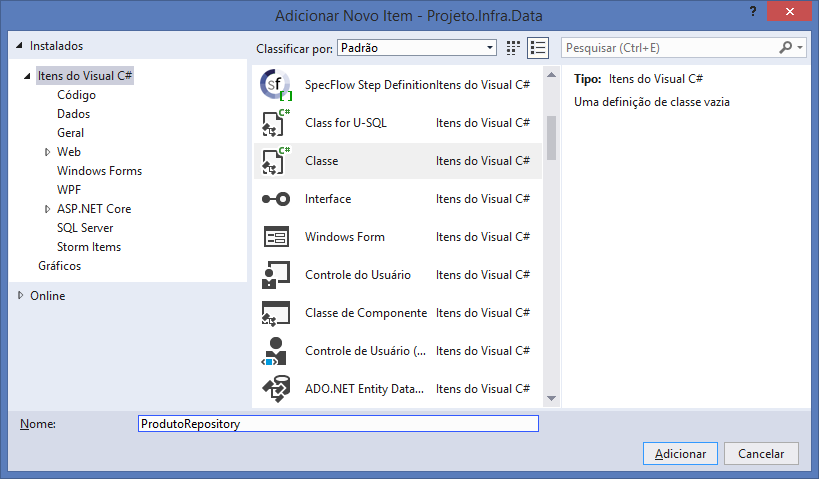
}

}

}

}

**Criando o repositorio de Produto:**



using Projeto.Entities;

using Projeto.Infra.Data.Context;

using Projeto.Infra.Data.Contracts;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Infra.Data.Repositories

{

public class ProdutoRepository

: BaseRepository<Produto>, IProdutoRepository

{

public List<Produto> FindByNome(string nome)

{

using (DataContext d = new DataContext())

{

return d.Produto

.Where(p => p.Nome.Contains(nome))

.OrderBy(p => p.Nome)

.ToList();

}

}

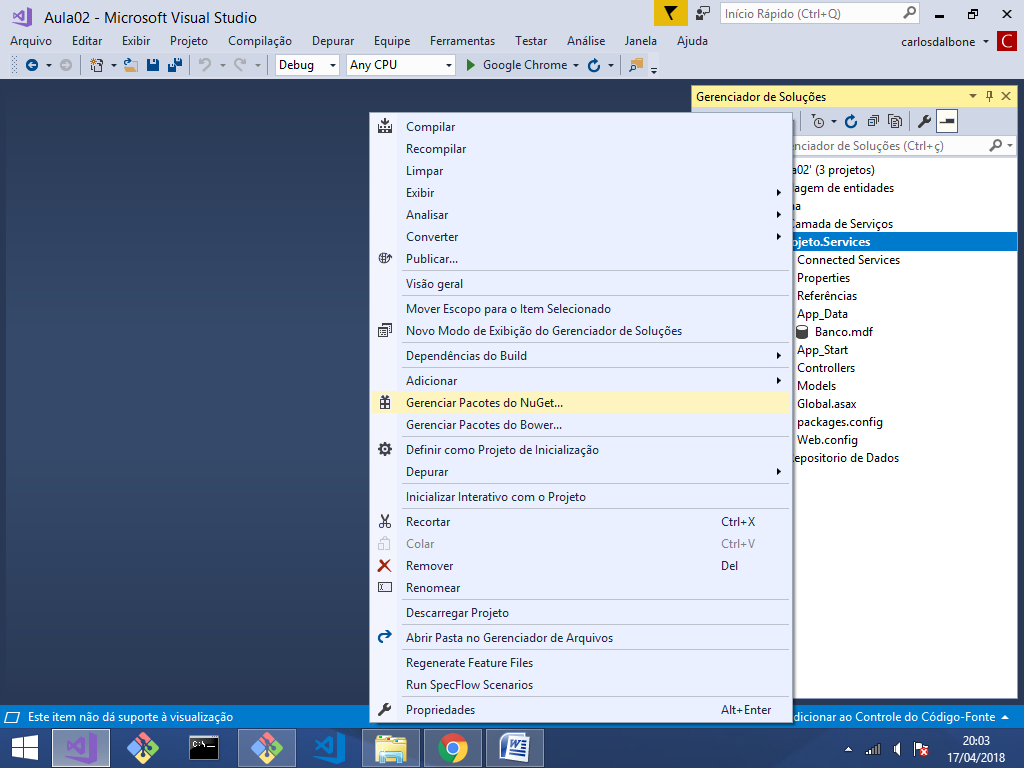
}

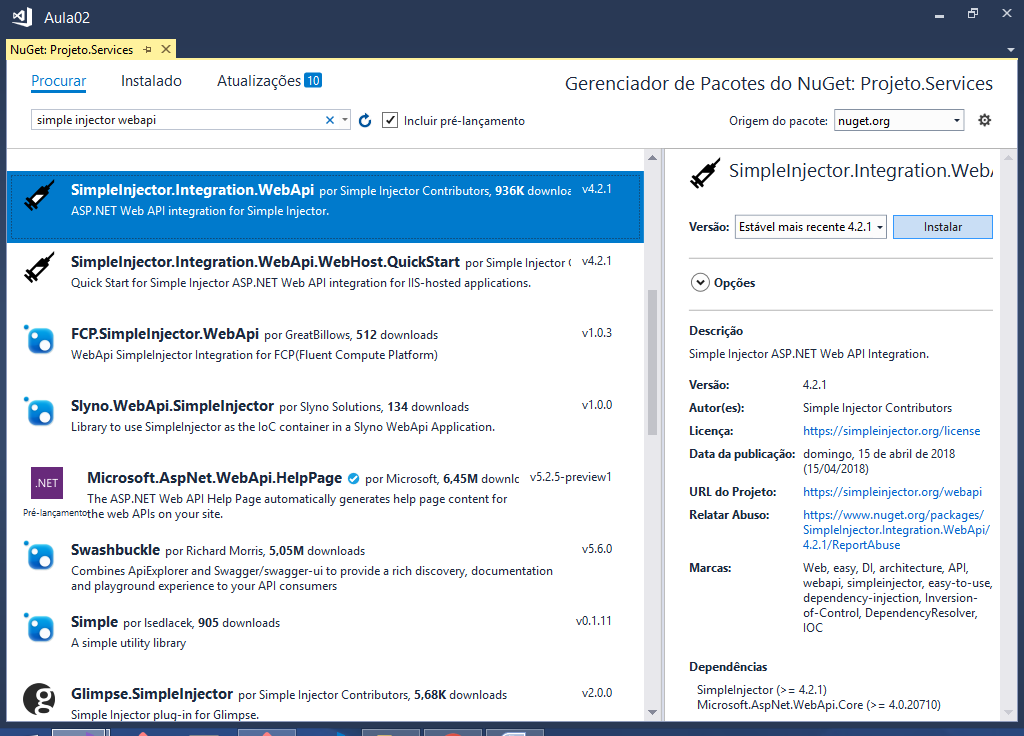
}

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

Simple Injector para WebApi

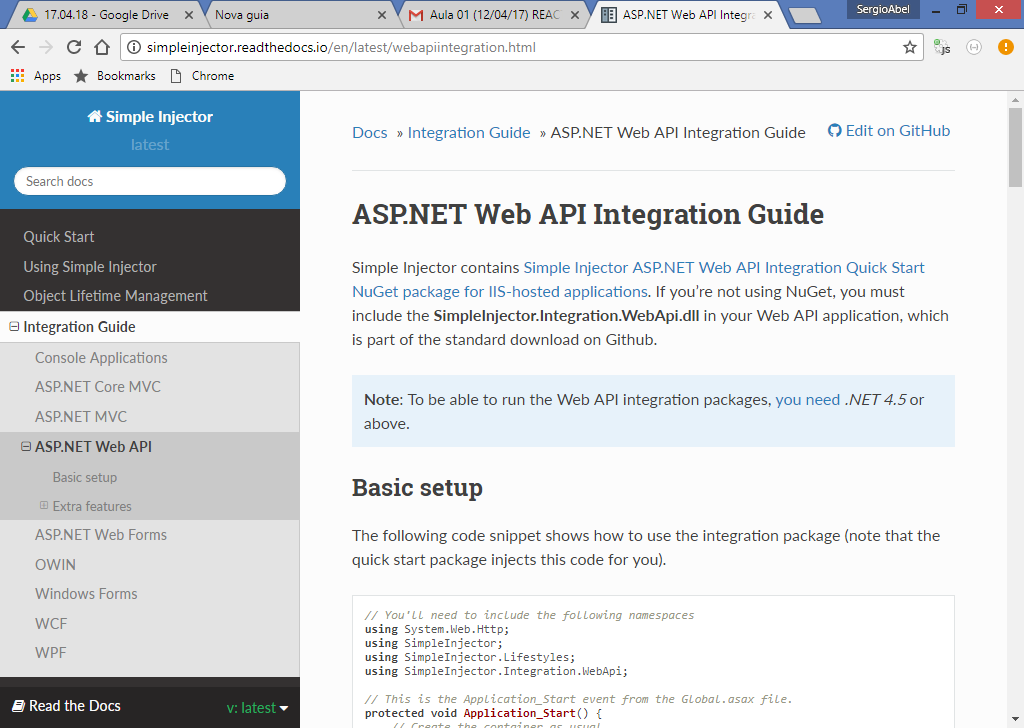
Framework para injeção de dependencia





**Configurando o Simple Injector:**

http://simpleinjector.readthedocs.io/en/latest/webapiintegration.html





using Projeto.Infra.Data.Contracts;

using Projeto.Infra.Data.Repositories;

using SimpleInjector;

using SimpleInjector.Integration.WebApi;

using SimpleInjector.Lifestyles;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Http;

using System.Web.Routing;

namespace Projeto.Services

{

public class WebApiApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

// 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<IProdutoRepository,**

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

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

container.RegisterWebApiControllers

(GlobalConfiguration.Configuration);

container.Verify();

GlobalConfiguration.Configuration.DependencyResolver =

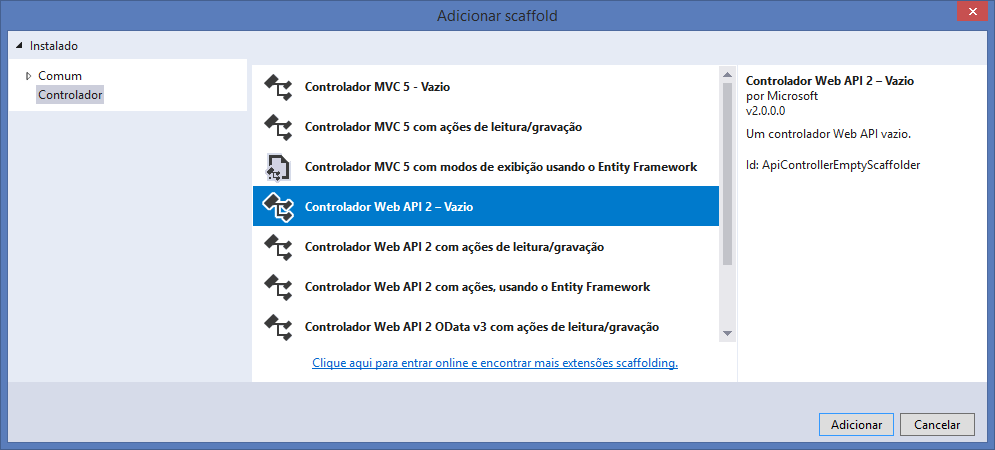
new SimpleInjectorWebApiDependencyResolver(container);

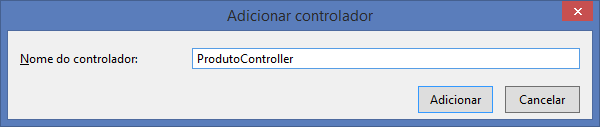
}

}

}

**Classe de controle:**





using Projeto.Infra.Data.Contracts;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Projeto.Services.Controllers

{

[RoutePrefix("api/produto")]

public class ProdutoController : ApiController

{

//atributo..

private readonly IProdutoRepository repository;

//construtor com entrada de argumentos

//utilizado pelo framework de injeção de dependência..

public ProdutoController(IProdutoRepository repository)

{

this.repository = repository;

}

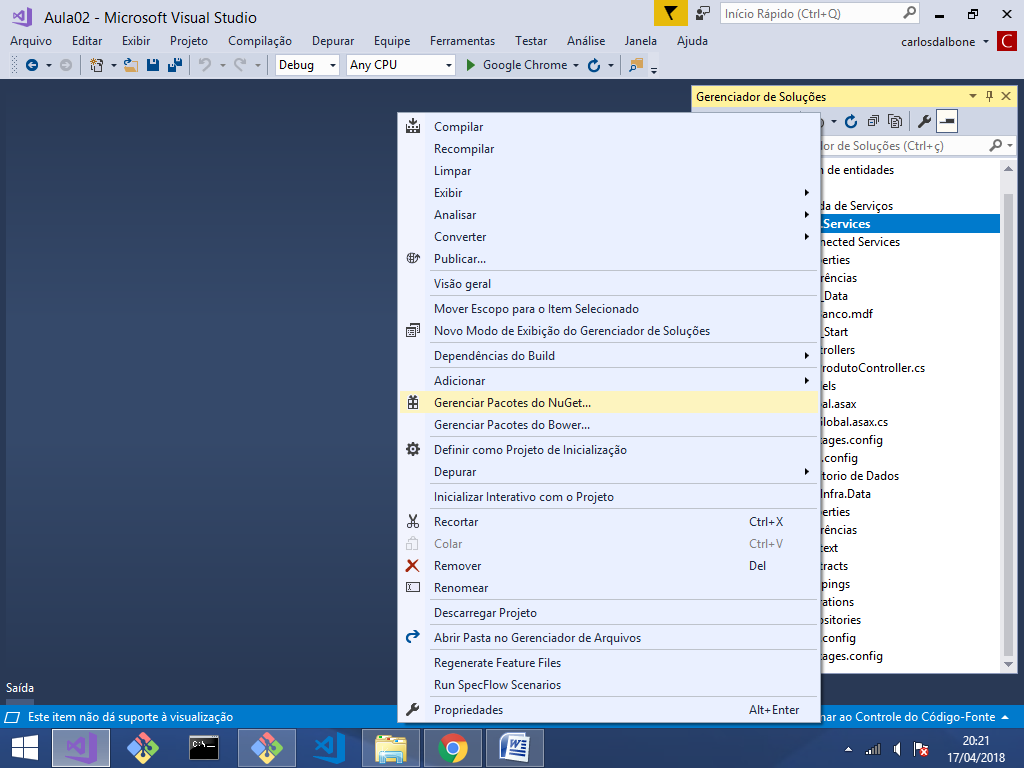
}

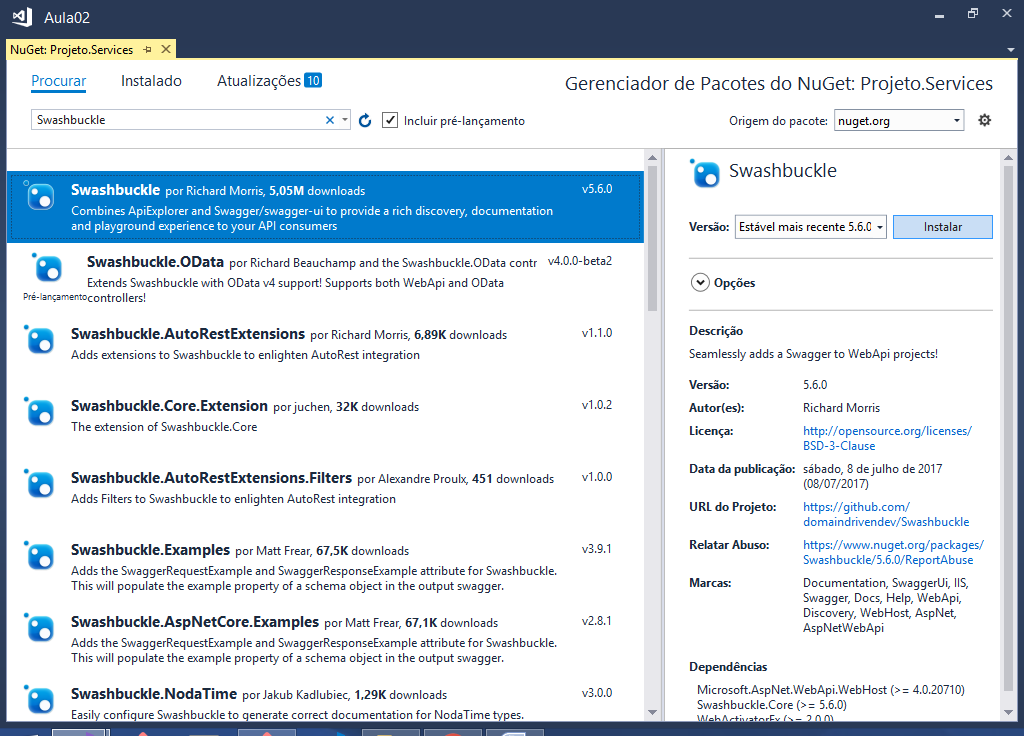
}

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

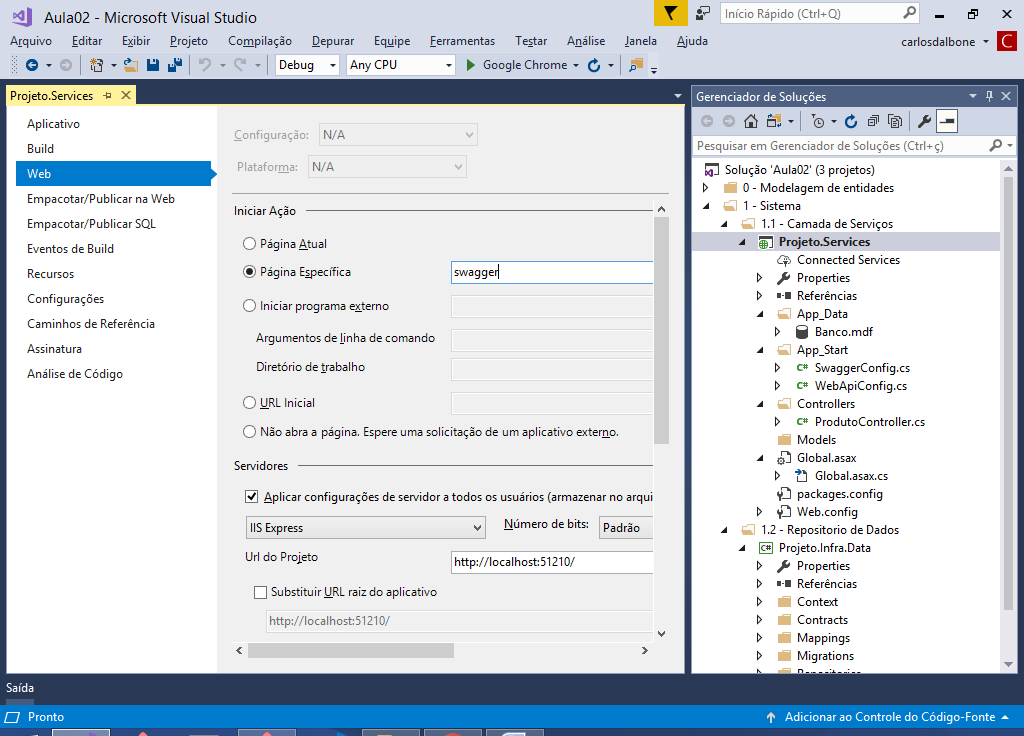
Instalando o Swagger

Documentação da API REST

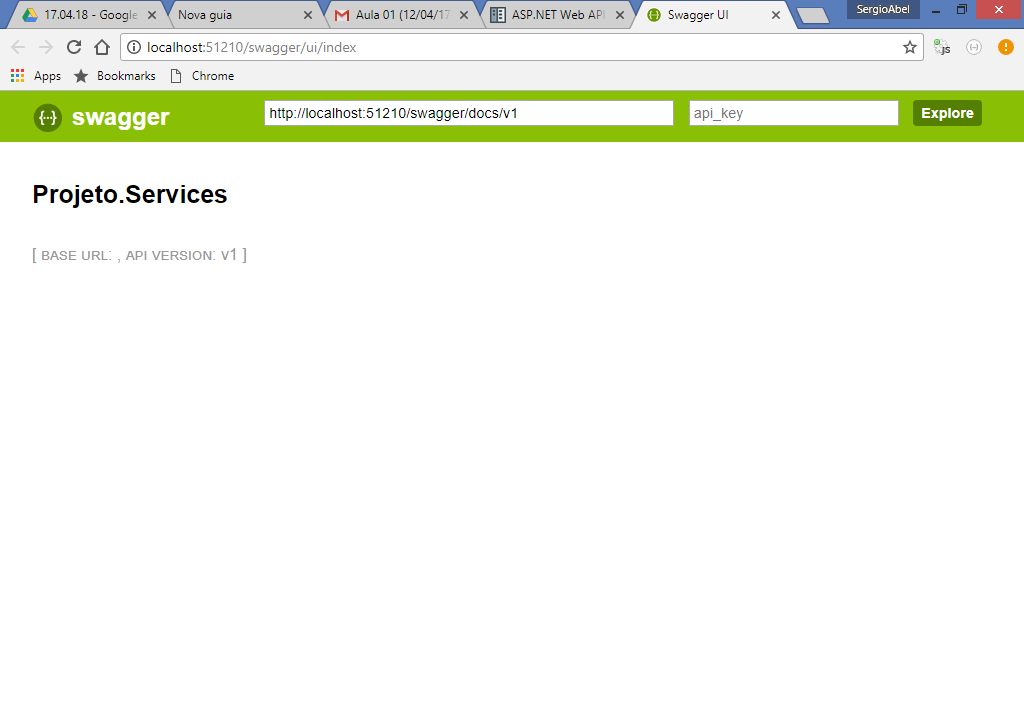




**Alterando a página inicial do projeto:**

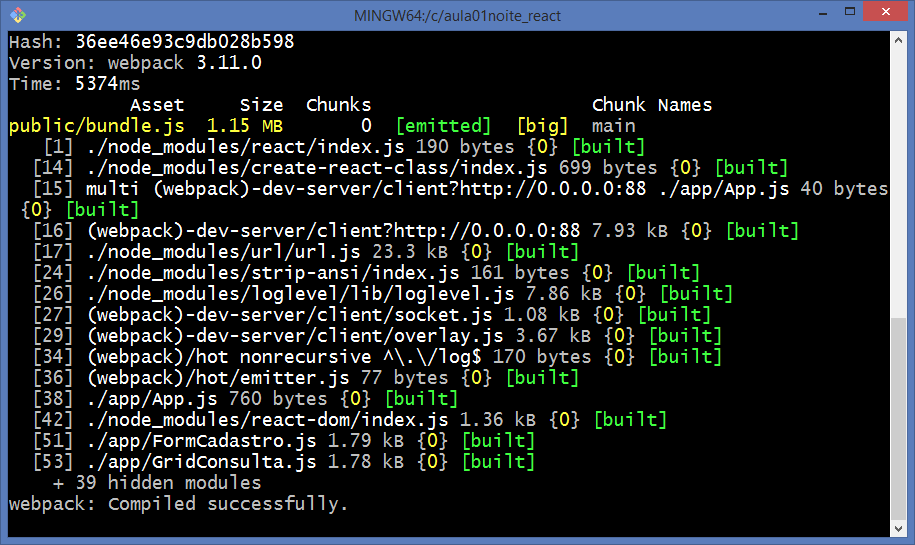


http://localhost:51210/swagger/ui/index

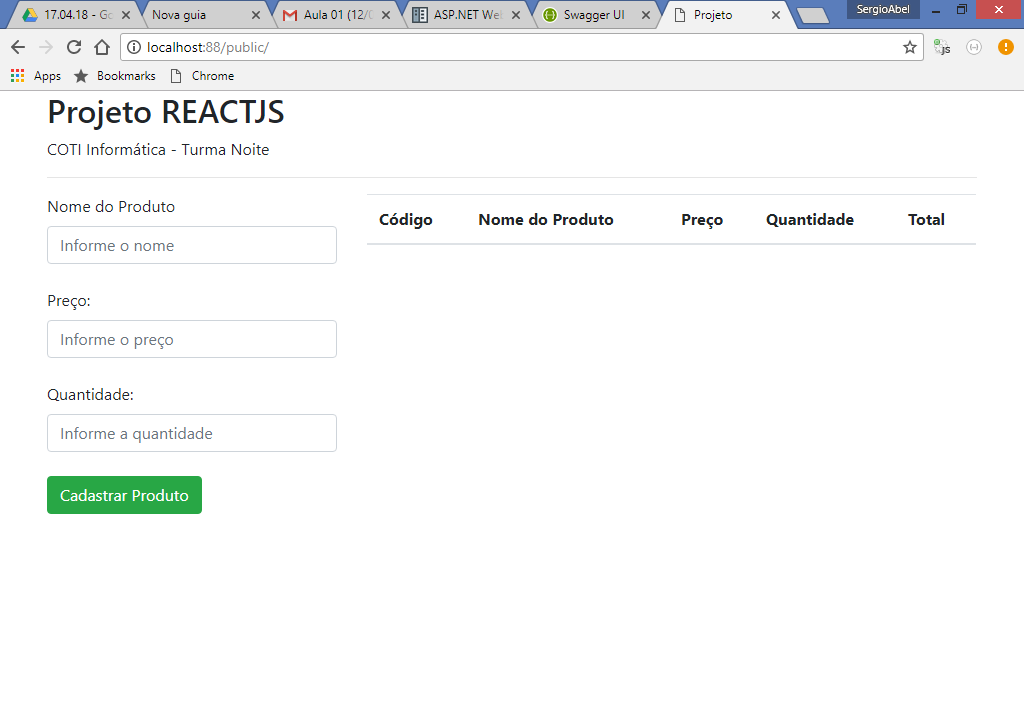


---------------------------------------------------------

**Executando o projeto REACT JS**



http://localhost:88/public/



Global.asax

Access Control Allow Origin (CORS)

Necessário para permitir que a API aceite requisições POST,GET, PUT, DELETE, Corpo e cabeçalho de dados provenientes de projetos externos

using Projeto.Infra.Data.Contracts;

using Projeto.Infra.Data.Repositories;

using SimpleInjector;

using SimpleInjector.Integration.WebApi;

using SimpleInjector.Lifestyles;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Http;

using System.Web.Routing;

namespace Projeto.Services

{

public class WebApiApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

// 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<IProdutoRepository,

ProdutoRepository>(Lifestyle.Scoped);

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

container.RegisterWebApiControllers

(GlobalConfiguration.Configuration);

container.Verify();

GlobalConfiguration.Configuration.DependencyResolver =

new SimpleInjectorWebApiDependencyResolver(container);

}

**protected void Application\_BeginRequest(object sender, EventArgs e)**

**{**

**HttpContext.Current.Response.AddHeader**

**("Access-Control-Allow-Origin", "\*");**

**if (HttpContext.Current.Request.HttpMethod == "OPTIONS")**

**{**

**HttpContext.Current.Response.AddHeader**

**("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");**

**HttpContext.Current.Response.AddHeader**

**("Access-Control-Allow-Headers", "Content-Type,**

**Accept, Authorization");**

**HttpContext.Current.Response.AddHeader**

**("Access-Control-Max-Age", "1728000");**

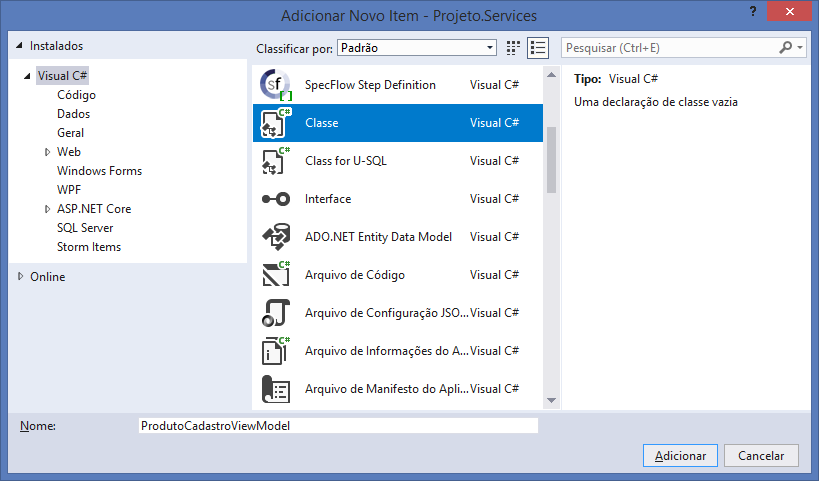
**HttpContext.Current.Response.End();**

**}**

**}**

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models

{

public class ProdutoCadastroViewModel

{

[MinLength(6, ErrorMessage = "Informe no mínimo {1} caracteres.")]

[MaxLength(50, ErrorMessage = "Informe no máximo {1} caracteres.")]

[Required(ErrorMessage = "Por favor, informe o nome do produto.")]

public string Nome { get; set; }

[Required(ErrorMessage = "Por favor, informe o preço do produto.")]

public decimal Preco { get; set; }

[Required(ErrorMessage = "Por favor, informe a quantidade do produto.")]

public int Quantidade { get; set; }

}

}

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

Criando o serviço de cadastro de produto:

using Projeto.Entities;

using Projeto.Infra.Data.Contracts;

using Projeto.Services.Models;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

namespace Projeto.Services.Controllers

{

[RoutePrefix("api/produto")]

public class ProdutoController : ApiController

{

//atributo..

private readonly IProdutoRepository repository;

//construtor com entrada de argumentos

//utilizado pelo framework de injeção de dependência..

public ProdutoController(IProdutoRepository repository)

{

this.repository = repository;

}

[HttpPost] //Requisição HTTP POST

[Route("cadastrar")] //URL: /api/produto/cadastrar

public HttpResponseMessage Post(ProdutoCadastroViewModel model)

{

if(ModelState.IsValid) //passou nas validações?

{

try

{

Produto p = new Produto

{

Nome = model.Nome,

Preco = model.Preco,

Quantidade = model.Quantidade

};

repository.Insert(p); //gravando..

return Request.CreateResponse //HTTP 200 (OK)

(HttpStatusCode.OK, $"Produto {p.Nome},

cadastrado com sucesso.");

}

catch(Exception e)

{

return Request.CreateResponse

//HTTP 500 (Internal Server Error)

(HttpStatusCode.InternalServerError, e.Message);

}

}

else

{

Hashtable mapa = new Hashtable();

//varrer o ModelState..

foreach(var s in ModelState)

{

//verificar se existe erro..

if(s.Value.Errors.Count > 0)

{

mapa[s.Key] = s.Value.Errors

.Select(e => e.ErrorMessage).ToList();

}

}

return Request.CreateResponse(HttpStatusCode.BadRequest, mapa);

}

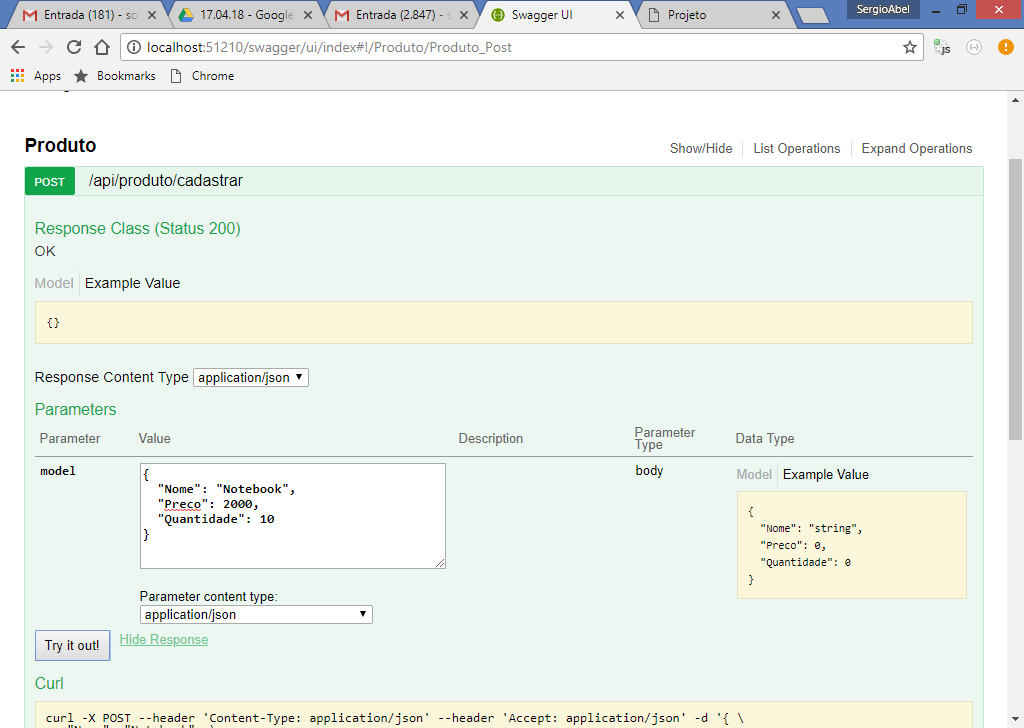
}

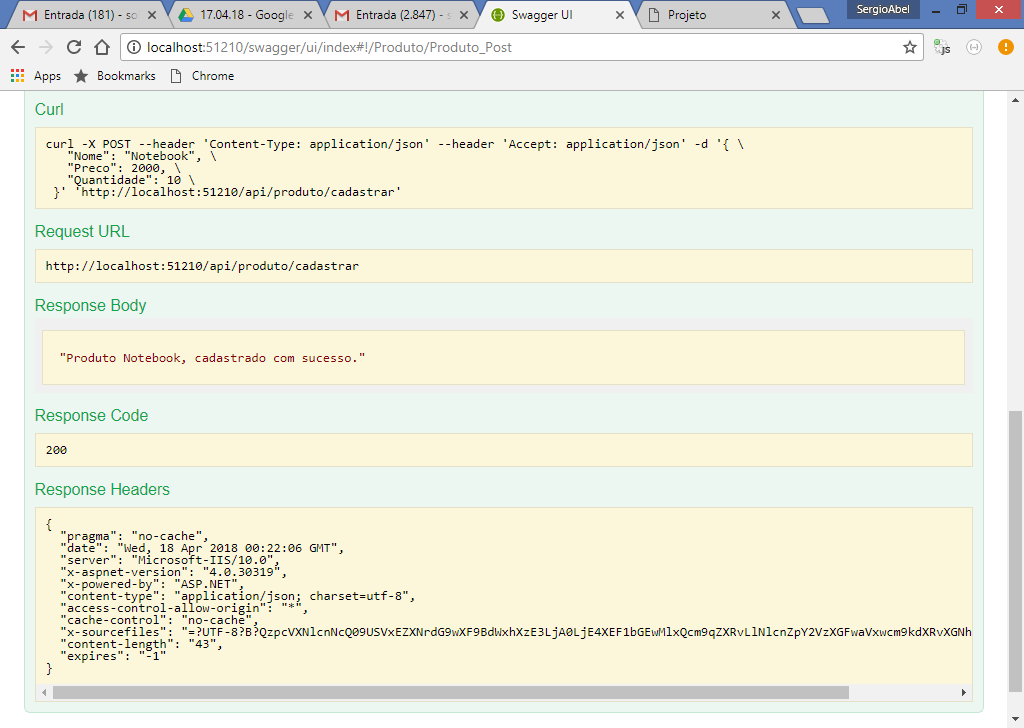
}

}

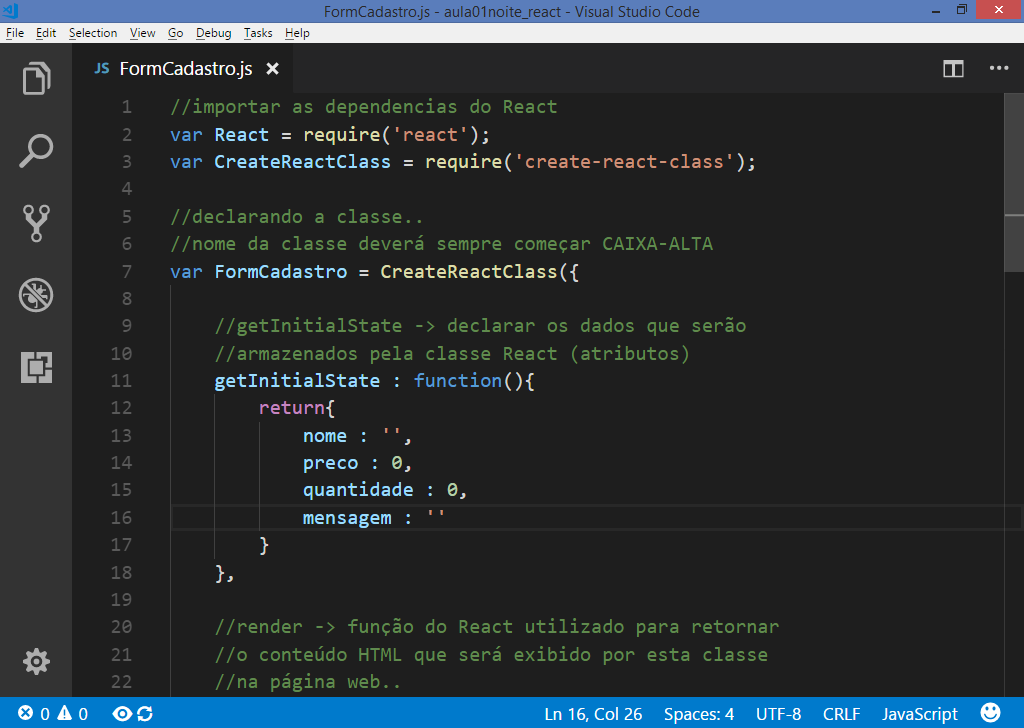
--------------------------------

**Testando:**





Declarando os atributos da classe FormCadastro:



//importar as dependencias do React

var React = require('react');

var CreateReactClass = require('create-react-class');

//declarando a classe..

//nome da classe deverá sempre começar CAIXA-ALTA

var FormCadastro = CreateReactClass({

//getInitialState -> declarar os dados que serão

//armazenados pela classe React (atributos)

getInitialState: function () {

return {

nome: '',

preco: 0,

quantidade: 0,

mensagem: ''

}

},

//função para ler o valor inputado no campo 'Nome'

handleNome: function (e) { //executada em um evento onChange

//e -> variavel que contem o elemento que executou a função..

this.setState({ nome: e.target.value });

},

handlePreco: function (e) {

this.setState({ preco: e.target.value });

},

handleQuantidade: function (e) {

this.setState({ quantidade: e.target.value });

},

//função executada no evento onSubmit..

cadastrarProduto : function(e){ //e -> tag do evento..

e.preventDefault(); //cancelando a ação SUBMIT..

//definir mensagem..

this.setState({ mensagem : "Processando, aguarde..." });

//imprindo..

console.log(this.state);

},

//render -> função do React utilizado para retornar

//o conteúdo HTML que será exibido por esta classe

//na página web..

render: function () {

return (

<div>

<form method="post" onSubmit={this.cadastrarProduto}>

<label>Nome do Produto</label>

<input type="text" className="form-control"

placeholder="Informe o nome"

onChange={this.handleNome}

value={this.state.nome} />

<br />

<label>Preço:</label>

<input type="text" className="form-control"

placeholder="Informe o preço"

onChange={this.handlePreco}

value={this.state.preco} />

<br />

<label>Quantidade:</label>

<input type="text" className="form-control"

placeholder="Informe a quantidade"

onChange={this.handleQuantidade}

value={this.state.quantidade} />

<br />

<input type="submit" value="Cadastrar Produto"

className="btn btn-success" />

<br />

<br />

<div>

{this.state.mensagem}

</div>

</form>

</div>

);

}

});

module.exports = FormCadastro;

