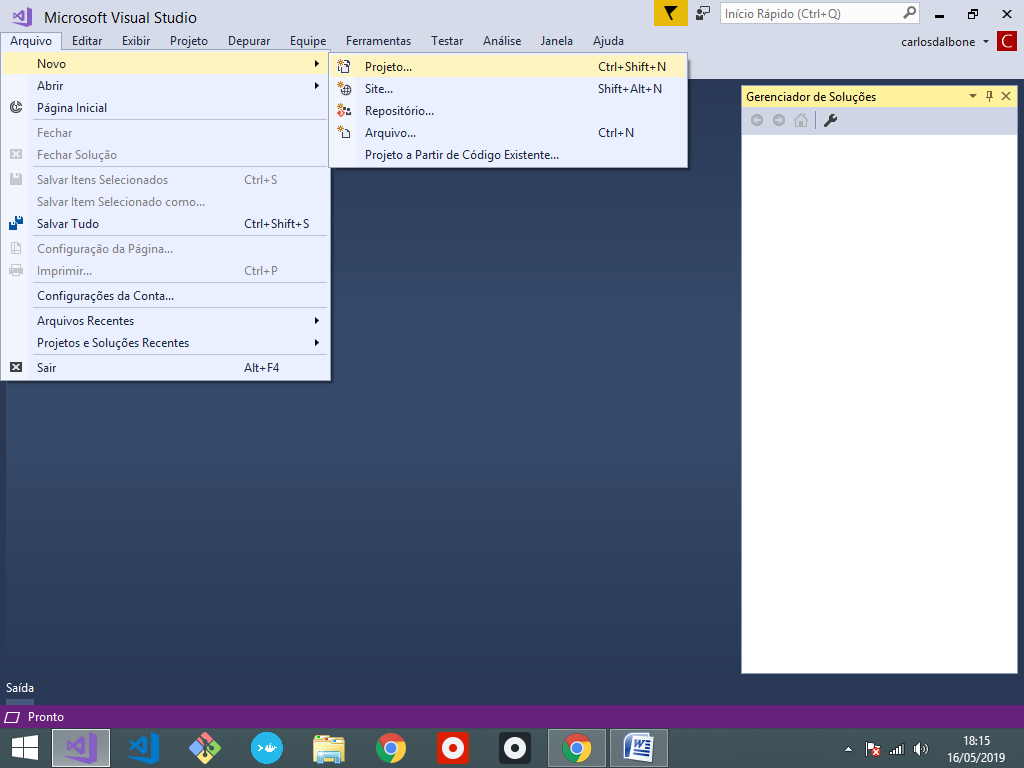
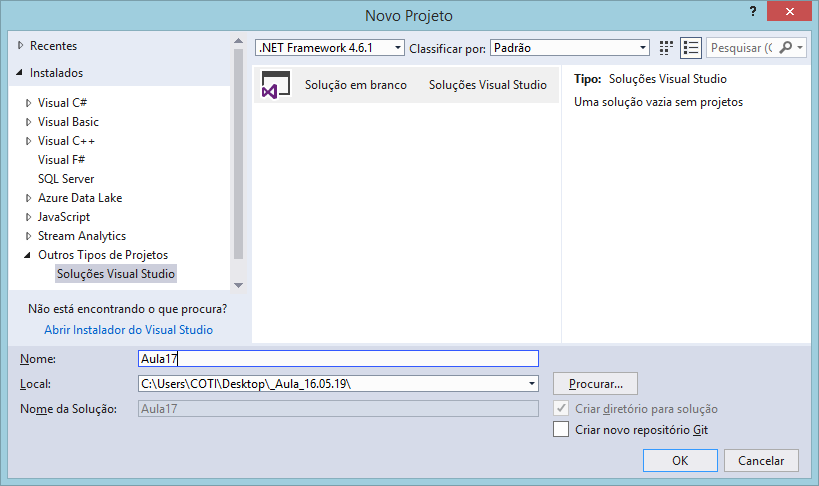
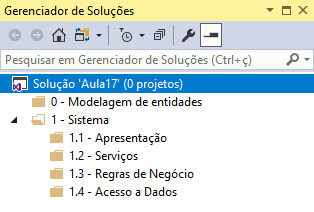
**Criando uma nova solution em branco:**

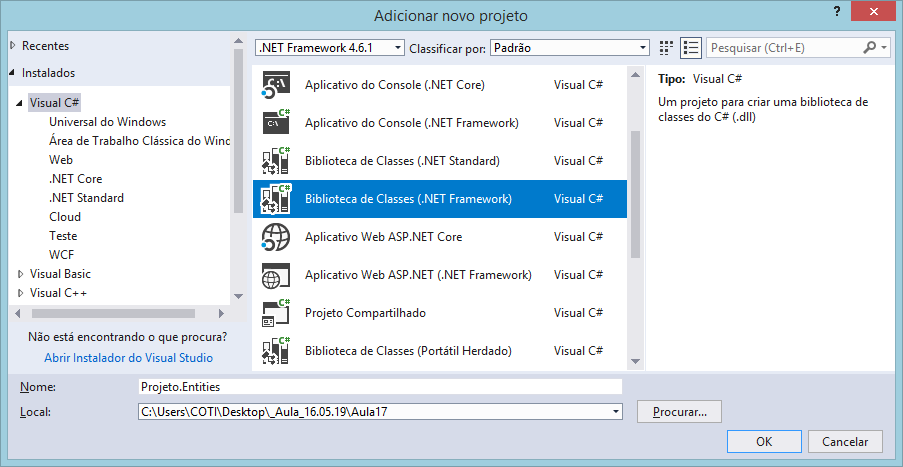






0 - Modelagem de entidades

Biblioteca de Classes .NET Framework



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto.Entities

{

public class Cliente

{

//propriedades -> prop + 2x[tab]

public int IdCliente { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public DateTime DataCriacao { get; set; }

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

public Cliente()

{

//vazio

}

//sobrecarga de métodos (OVERLOADING)

public Cliente(int idCliente, string nome,

string email, DateTime dataCriacao)

{

IdCliente = idCliente;

Nome = nome;

Email = email;

DataCriacao = dataCriacao;

}

//sobrescrita do método ToString()

public override string ToString()

{

return $"Id: {IdCliente}, Nome: {Nome}, Email: {Email},

Data Criação: {DataCriacao}";

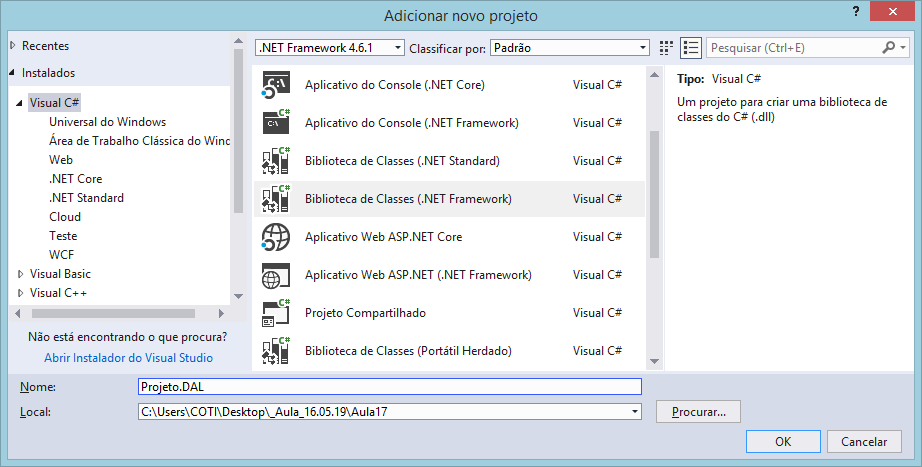
}

}

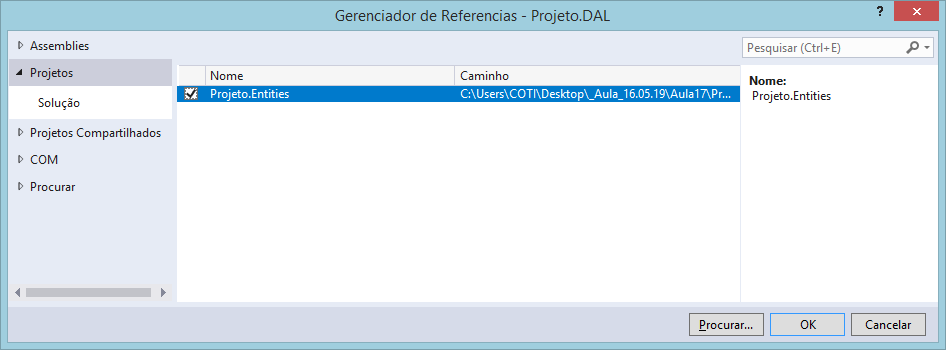
}

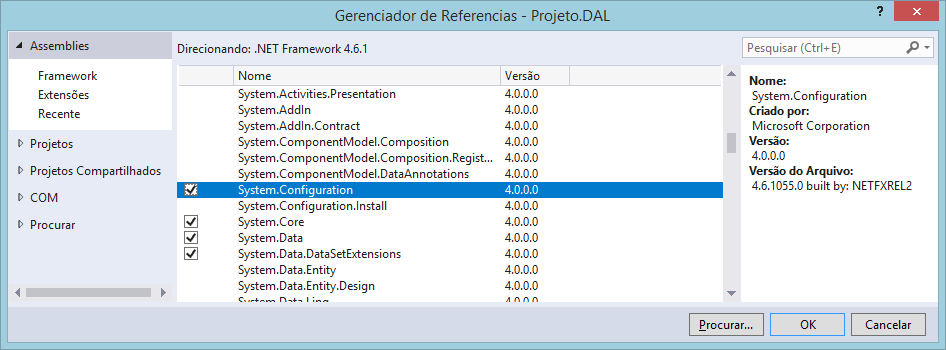
1.4 - Camada de Acesso a dados

DAL - Data Access Layer (Class Library .NET Framework)



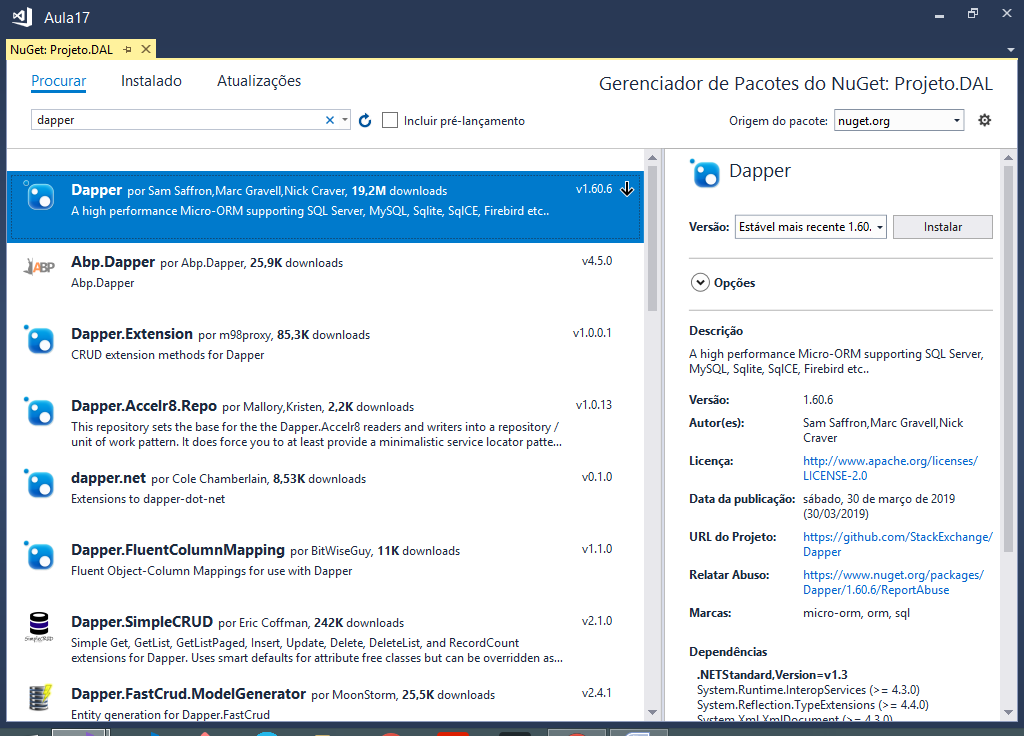
Addicionando referencias no projeto DAL:





Instalando o Dapper:

Gerenciador de pacotes do NuGet



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient; //importando

using System.Configuration; //importando

using Projeto.Entities; //importando

using Dapper; //importando

namespace Projeto.DAL

{

public class ClienteRepository

{

//atributo

private string connectionString;

//construtor

public ClienteRepository()

{

connectionString = ConfigurationManager

.ConnectionStrings["projeto"].ConnectionString;

}

public void Insert(Cliente cliente)

{

using (var conn = new SqlConnection(connectionString))

{

string query = "insert into Cliente(Nome, Email, DataCriacao) "

+ "values(@Nome, @Email, GETDATE())";

conn.Execute(query, cliente);

}

}

public void Update(Cliente cliente)

{

using (var conn = new SqlConnection(connectionString))

{

string query = "update Cliente set Nome = @Nome, Email = @Email "

+ "where IdCliente = @IdCliente";

conn.Execute(query, cliente);

}

}

public void Delete(int id)

{

using (var conn = new SqlConnection(connectionString))

{

string query = "delete from Cliente

where IdCliente = @IdCliente";

conn.Execute(query, new { IdCliente = id });

}

}

public List<Cliente> SelectAll()

{

using (var conn = new SqlConnection(connectionString))

{

string query = "select \* from Cliente";

return conn.Query<Cliente>(query).ToList();

}

}

public Cliente SelectById(int id)

{

using (var conn = new SqlConnection(connectionString))

{

string query = "select \* from Cliente

where IdCliente = @IdCliente";

return conn.QuerySingleOrDefault<Cliente>(query,

new { IdCliente = id });

}

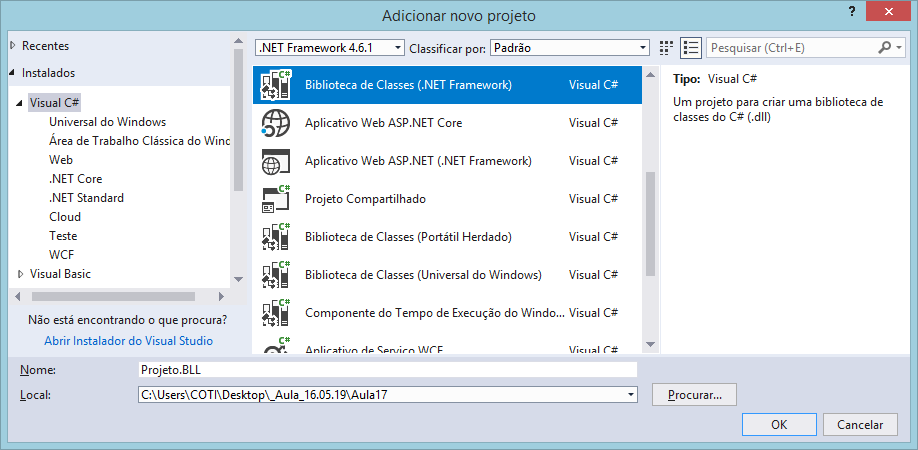
}

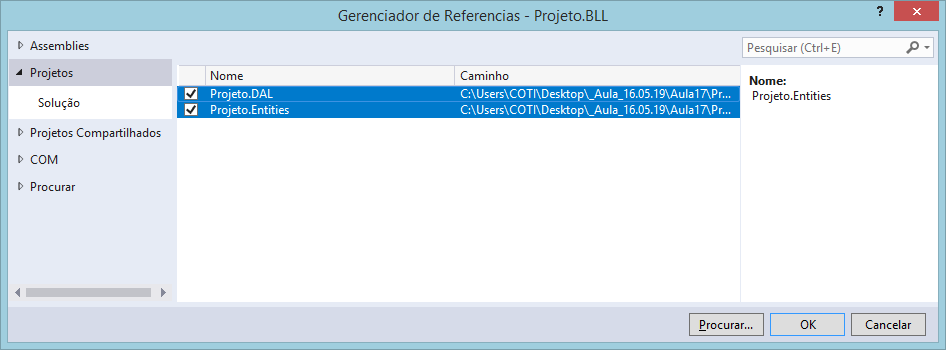
}

}

1.3 - Camada de Regras de Negócio

Biblioteca de Classes .Net Framework





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto.Entities;

using Projeto.DAL;

namespace Projeto.BLL

{

public class ClienteBusiness

{

//atributo..

private ClienteRepository repository;

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

public ClienteBusiness()

{

repository = new ClienteRepository();

}

//método para cadastrar cliente

public void CadastrarCliente(Cliente cliente)

{

repository.Insert(cliente);

}

//método para atualizar cliente

public void AtualizarCliente(Cliente cliente)

{

repository.Update(cliente);

}

//método para excluir cliente

public void ExcluirCliente(int id)

{

repository.Delete(id);

}

//método para listar todos os clientes

public List<Cliente> ConsultarTodos()

{

return repository.SelectAll();

}

//método para consultar cliente por id

public Cliente ConsultarPorId(int id)

{

return repository.SelectById(id);

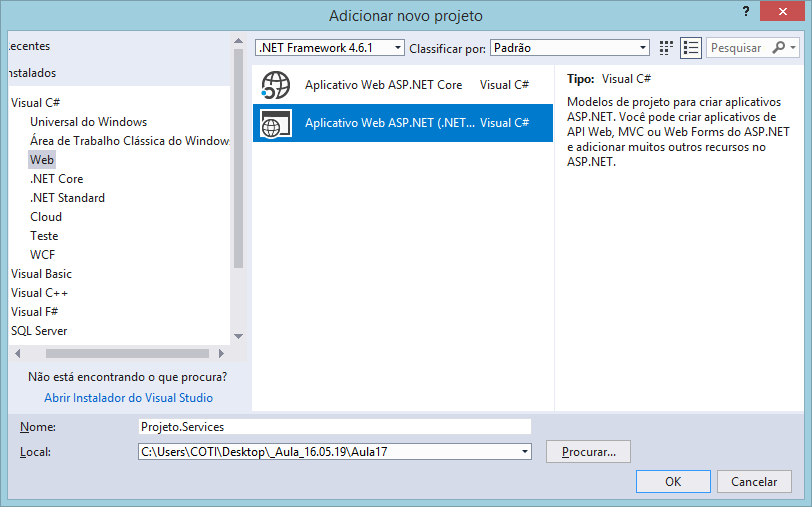
}

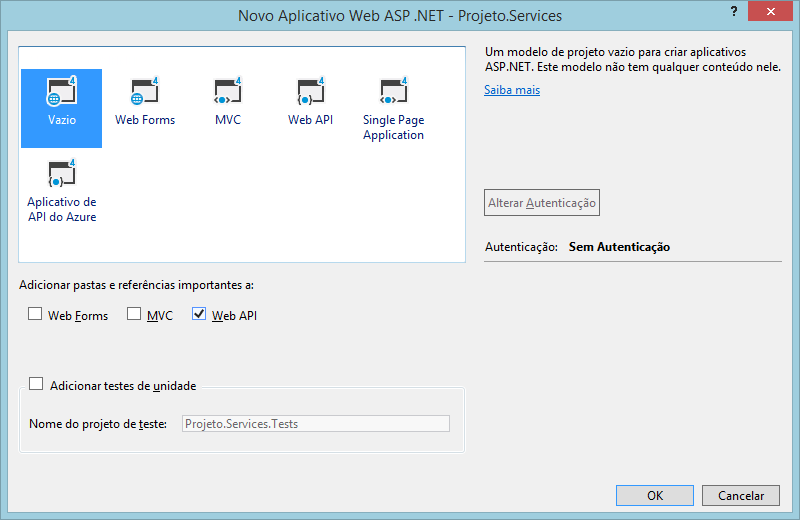
}

}

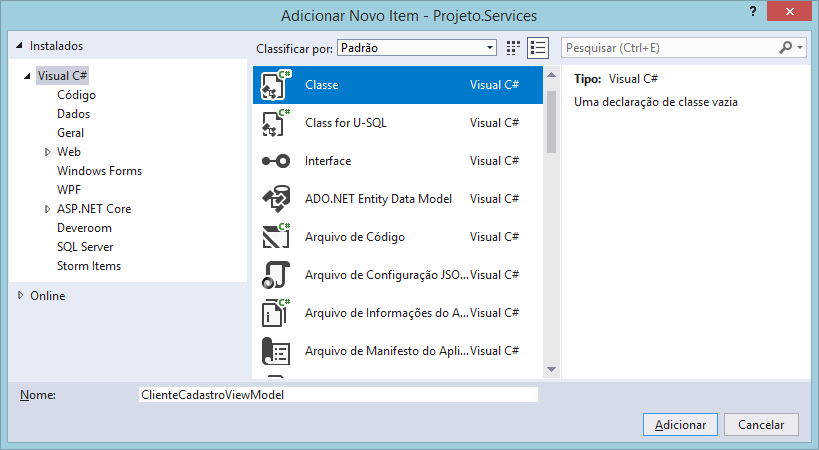
1.2 - Serviços

Projeto Asp.Net .NET Framework (WebApi)





**Criando as classes ViewModel para os serviços da API:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models

{

public class ClienteCadastroViewModel

{

[Required(ErrorMessage = "Informe o nome do cliente.")]

public string Nome { get; set; }

[EmailAddress(ErrorMessage = "Email inválido.")]

[Required(ErrorMessage = "Informe o email do cliente.")]

public string Email { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models

{

public class ClienteEdicaoViewModel

{

[Required(ErrorMessage = "Informe o id do cliente.")]

public int IdCliente { get; set; }

[Required(ErrorMessage = "Informe o nome do cliente.")]

public string Nome { get; set; }

[EmailAddress(ErrorMessage = "Email inválido.")]

[Required(ErrorMessage = "Informe o email do cliente.")]

public string Email { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Projeto.Services.Models

{

public class ClienteConsultaViewModel

{

public int IdCliente { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

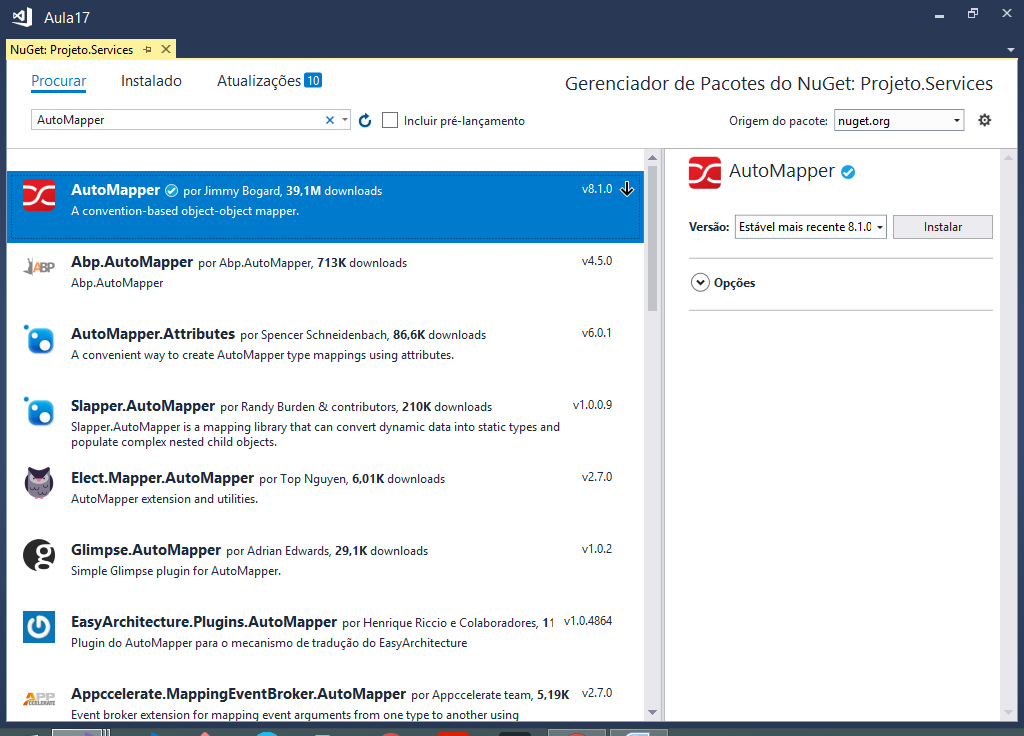
public DateTime DataCriacao { get; set; }

}

}

Instalando o AutoMapper:

Gerenciador de pacotes do NuGet



Mapeando as transferencias de dados entre ViewModels e Entidades e vice-versa

**\Mappings\ViewModelToEntityMap.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using AutoMapper;

using Projeto.Entities;

using Projeto.Services.Models;

namespace Projeto.Services.Mappings

{

//REGRA) Herdar Profile

public class ViewModelToEntityMap : Profile

{

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

public ViewModelToEntityMap()

{

CreateMap<ClienteCadastroViewModel, Cliente>();

CreateMap<ClienteEdicaoViewModel, Cliente>();

}

}

}

**\Mappings\EntityToViewModelMap.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using AutoMapper;

using Projeto.Entities;

using Projeto.Services.Models;

namespace Projeto.Services.Mappings

{

//REGRA) Herdar Profile

public class EntityToViewModelMap : Profile

{

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

public EntityToViewModelMap()

{

CreateMap<Cliente, ClienteConsultaViewModel>();

}

}

}

Configurando o AutoMapper

na classe Global.asax

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Http;

using System.Web.Routing;

using AutoMapper;

using Projeto.Services.Mappings;

namespace Projeto.Services

{

public class WebApiApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

//registrando as classes de mapeamento

//feitas com o AutoMapper..

Mapper.Initialize(cfg

=>

{

cfg.AddProfile<ViewModelToEntityMap>();

cfg.AddProfile<EntityToViewModelMap>();

});

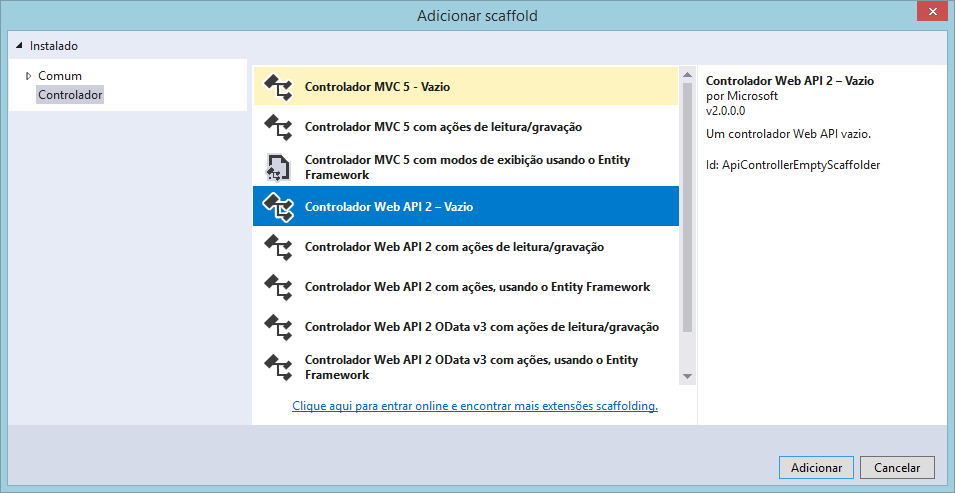
}

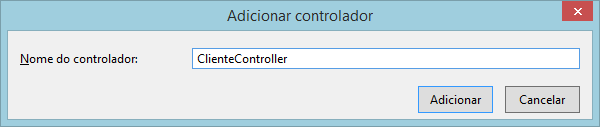
}

}

Criando controller do projeto WebApi

Classe utilizada para disponibilizar os serviços de Cliente no projeto API





using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using AutoMapper; //importando

using Projeto.BLL; //importando

using Projeto.Entities; //importando

using Projeto.Services.Models; //importando

namespace Projeto.Services.Controllers

{

[RoutePrefix("api/Cliente")]

public class ClienteController : ApiController

{

//atributo

private ClienteBusiness business;

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

public ClienteController()

{

business = new ClienteBusiness();

}

**[HttpPost]**

**public HttpResponseMessage Post(ClienteCadastroViewModel model)**

**{**

**if(ModelState.IsValid)**

**{**

**try**

**{**

**var cliente = Mapper.Map<Cliente>(model);**

**business.CadastrarCliente(cliente);**

**//retornar um status de erro HTTP 200 (OK)**

**return Request.CreateResponse(HttpStatusCode.OK,**

**$"Cliente {cliente.Nome}, cadastrado com sucesso");**

**}**

**catch(Exception e)**

**{**

**//retornar um status de erro HTTP 500 (InternalServerError)**

**return Request.CreateResponse(HttpStatusCode**

**.InternalServerError,**

**"Erro interno de servidor: " + e.Message);**

**}**

**}**

**else**

**{**

**//retornar um status de erro HTTP 400 (BadRequest)**

**return Request.CreateResponse(HttpStatusCode.BadRequest,**

**"Ocorreram erros de validação.");**

**}**

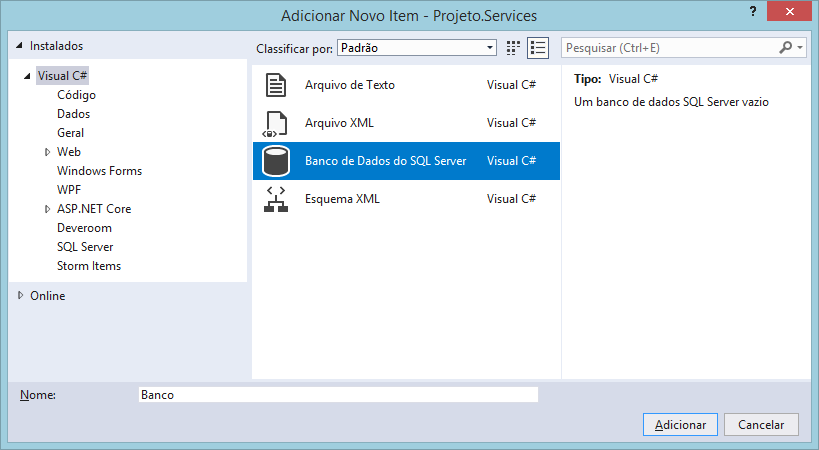
**}**

}

}

**Criando a base de dados:**

MDF - Master Database File



create table Cliente(

IdCliente integer identity(1,1),

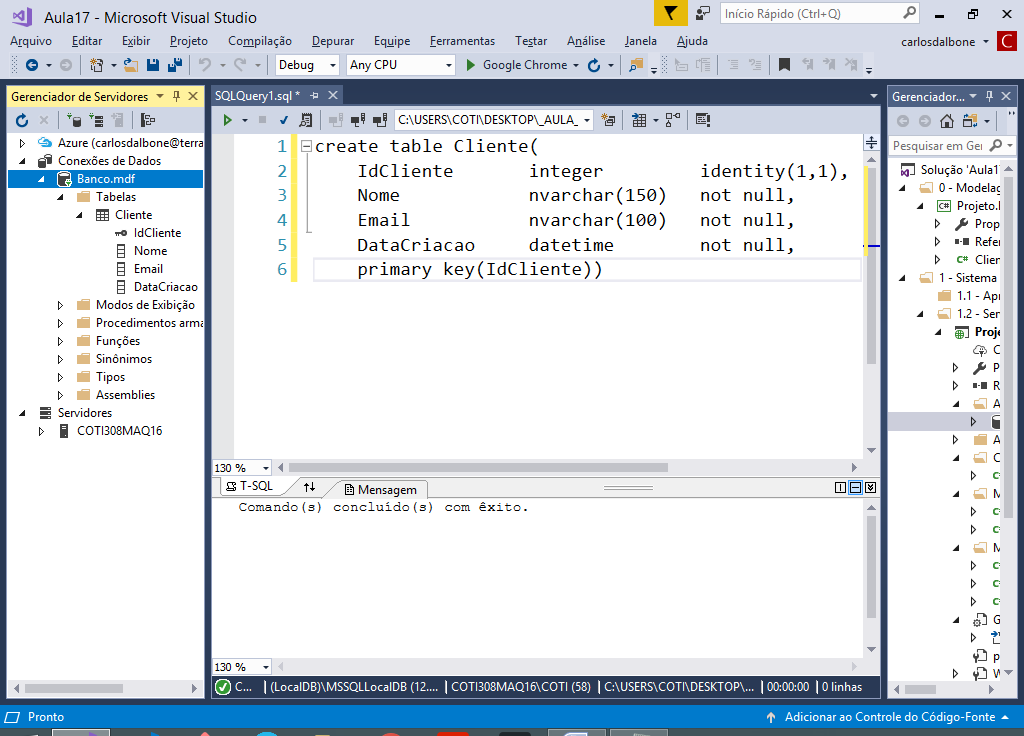
Nome nvarchar(150) not null,

Email nvarchar(100) not null,

DataCriacao datetime not null,

primary key(IdCliente))

**Executando:**



\Web.config.xml

mapeando a string de conexão do banco de dados

**<connectionStrings>**

**<add**

**name="projeto"**

**connectionString="Data Source=(LocalDB)\**

**MSSQLLocalDB;AttachDbFilename=**

**C:\Users\COTI\Desktop\\_Aula\_16.05.19\**

**Aula17\Projeto.Services\App\_Data\**

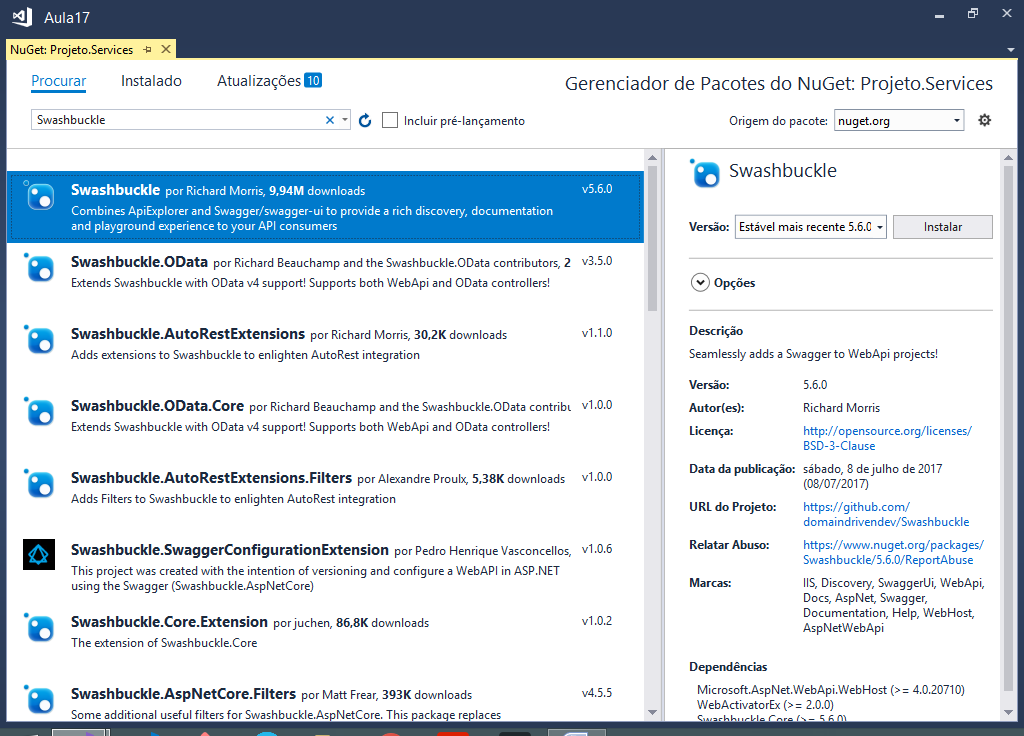
**Banco.mdf;Integrated Security=True"**

**/>**

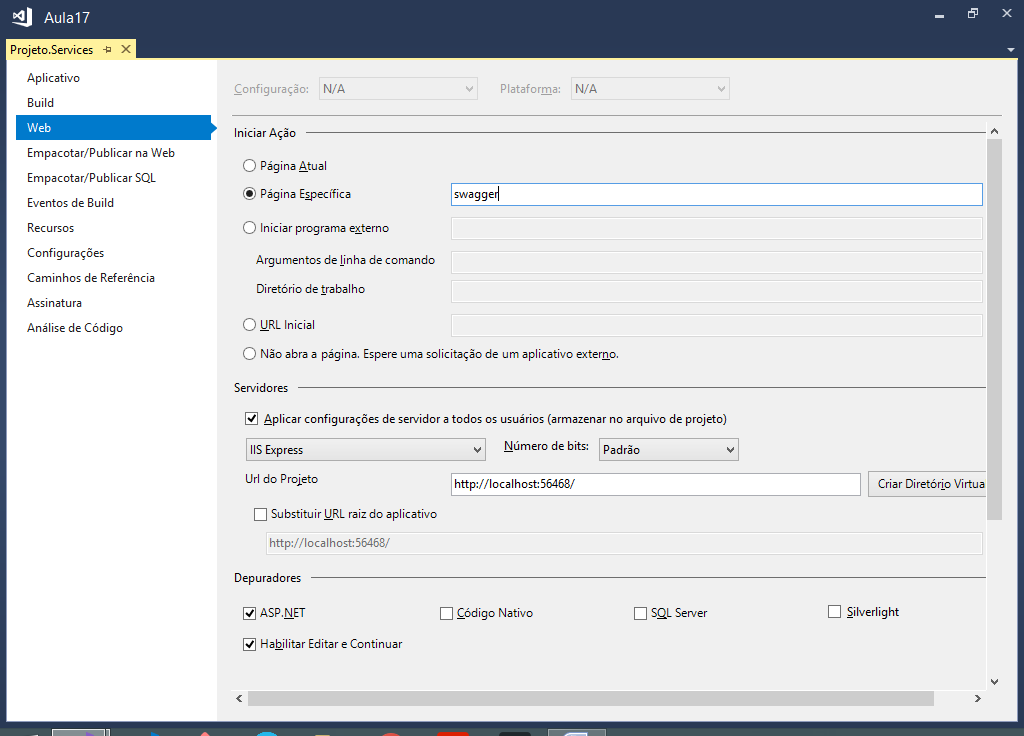
**</connectionStrings>**

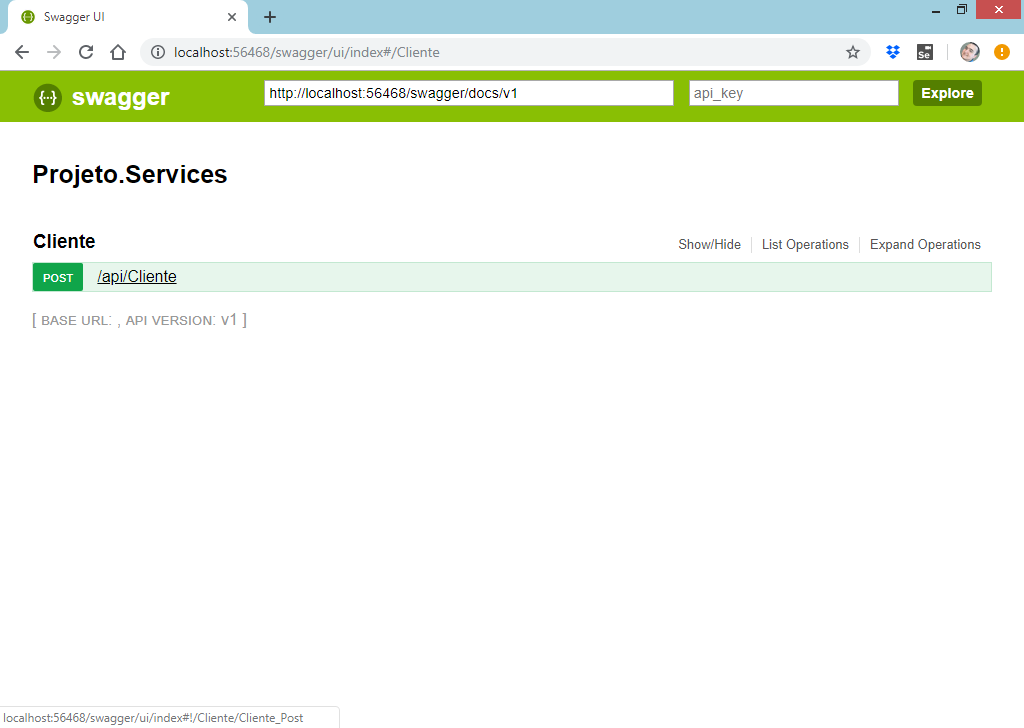
Swagger

Framework utilizado para gerar   
documentação em projetos do tipo API.



Alterando a página inicial do projeto: **swagger**





Testando:

