

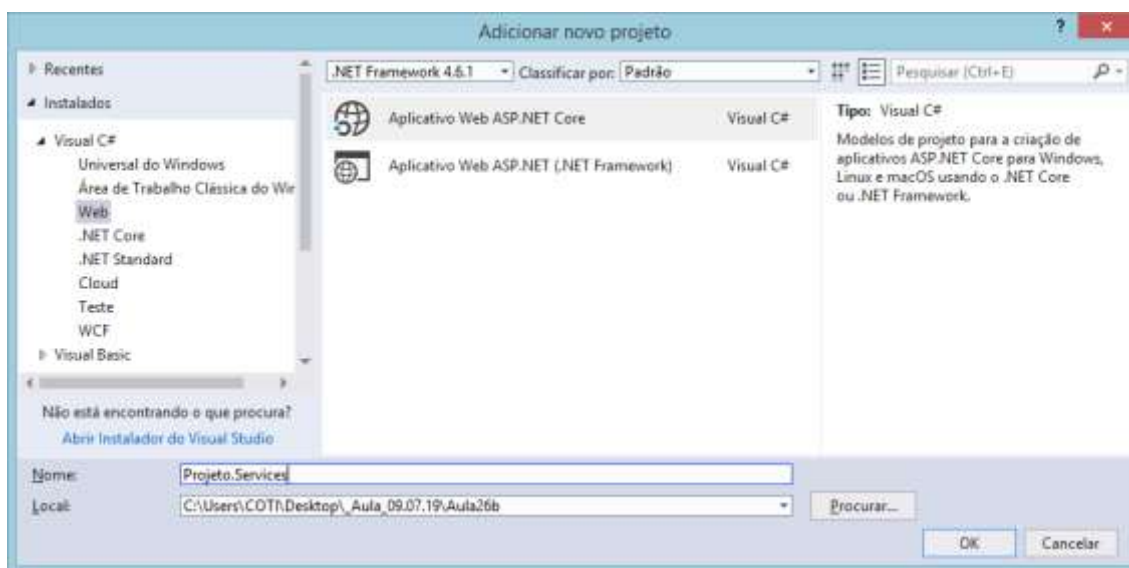
ASP.NET Core

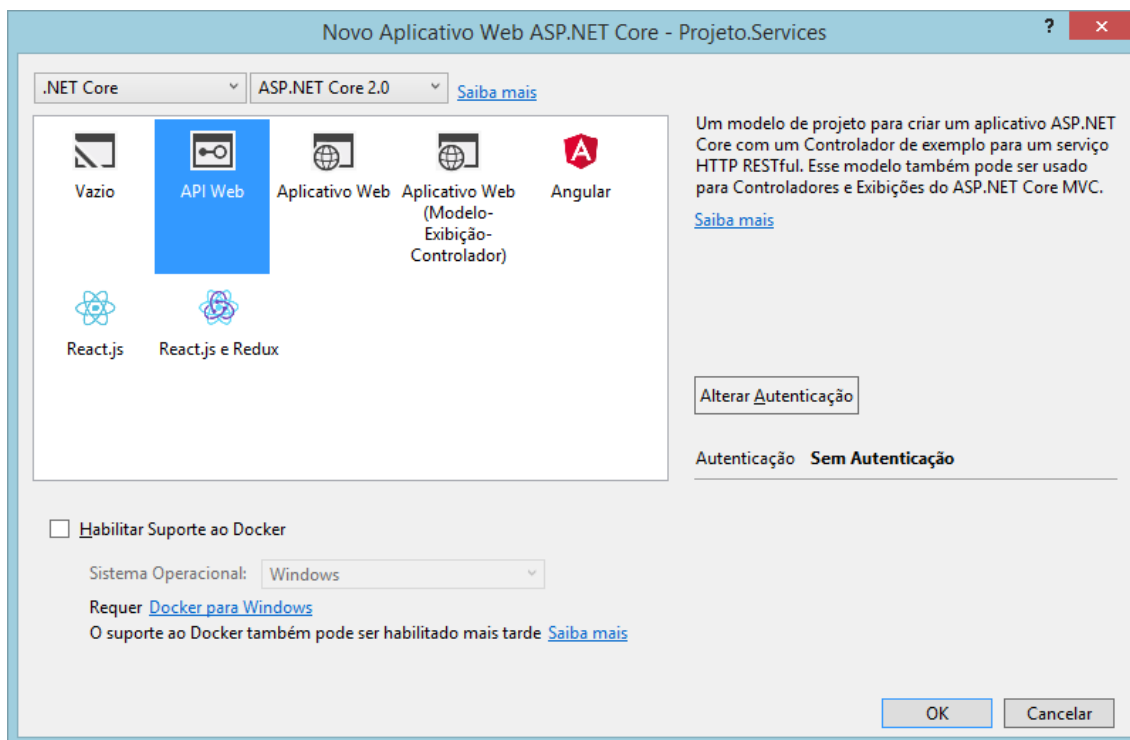
Asp.Net Core

Nova implementação do .NET que veio com a proposta de otimizar as tecnologias do framework através das seguintes características:

- Plataforma mais robusta e performática
- Simplifica estrutura do Asp.Net pois não utiliza mais a biblioteca System.Web
- Unifica o MVC e WebApi em uma única tecnologia
- Multiplataforma (proposta de execução em Linux, IOS, etc)

Criando um projeto em Asp.Net:





Startup.cs

(Classe que irá "substituir" o uso do Global.asax)

Nesta classe é que iremos configurar a maioria das tecnologias que serão instaladas no projeto.

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

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();
        }
    }
}
```

```
// 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();
    }

    app.UseMvc();
}
}
```

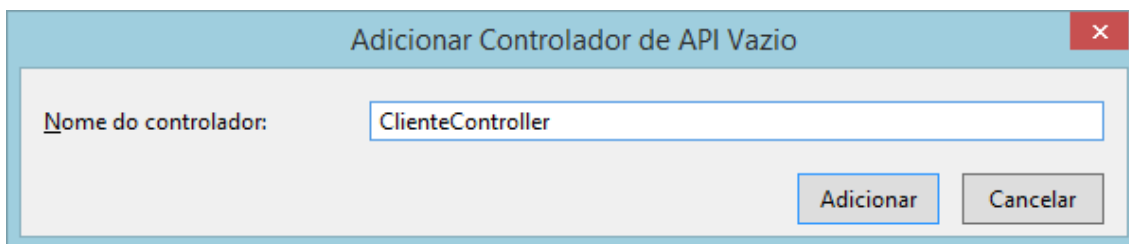
appsettings.json

Utilizado para mapeamento de parametros de forma a substituir o antigo Web.config.xml

```
{
  "Logging": {
    "IncludeScopes": false,
    "Debug": {
      "LogLevel": {
        "Default": "Warning"
      }
    },
    "Console": {
      "LogLevel": {
        "Default": "Warning"
      }
    }
  }
}
```

Criando um controller API:





```
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
{
    [Produces("application/json")]
    [Route("api/Cliente")]
    public class ClienteController : Controller
    {
    }
}
```

Criando classes Model para serviços da API de Cliente:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models
{
    public class ClienteCadastroViewModel
    {
        [Required(ErrorMessage = "Campo obrigatório")]
        public string Nome { get; set; }

        [EmailAddress(ErrorMessage = "Email inválido")]
        [Required(ErrorMessage = "Campo obrigatório")]
        public string Email { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations;

namespace Projeto.Services.Models
{
    public class ClienteEdicaoViewModel
    {
        [Required(ErrorMessage = "Campo obrigatório")]
        public int IdCliente { get; set; }
    }
}
```

```
[Required(ErrorMessage = "Campo obrigatório")]
public string Nome { get; set; }

[EmailAddress(ErrorMessage = "Email inválido")]
[Required(ErrorMessage = "Campo obrigatório")]
public string Email { get; set; }
}

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

namespace Projeto.Services.Models
{
    public class ClienteConsultaViewModel
    {
        public int IdCliente { get; set; }
        public string Nome { get; set; }
        public string Email { get; set; }
    }
}
```

Criando os serviços da API:

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

namespace Projeto.Services.Controllers
{
    [Produces("application/json")]
    [Route("api/Cliente")]
    public class ClienteController : Controller
    {
        [HttpPost]
        public IActionResult Post([FromBody] ClienteCadastroViewModel model)
        {
            if (ModelState.IsValid)
            {
                return Ok("Cliente cadastrado com sucesso.");
            }
            else
            {
                return BadRequest();
            }
        }

        [HttpPut]
        public IActionResult Put([FromBody] ClienteEdicaoViewModel model)
        {
            if (ModelState.IsValid)
            {
                return Ok("Cliente atualizado com sucesso.");
            }
        }
    }
}
```

```

    }
    else
    {
        return BadRequest();
    }
}

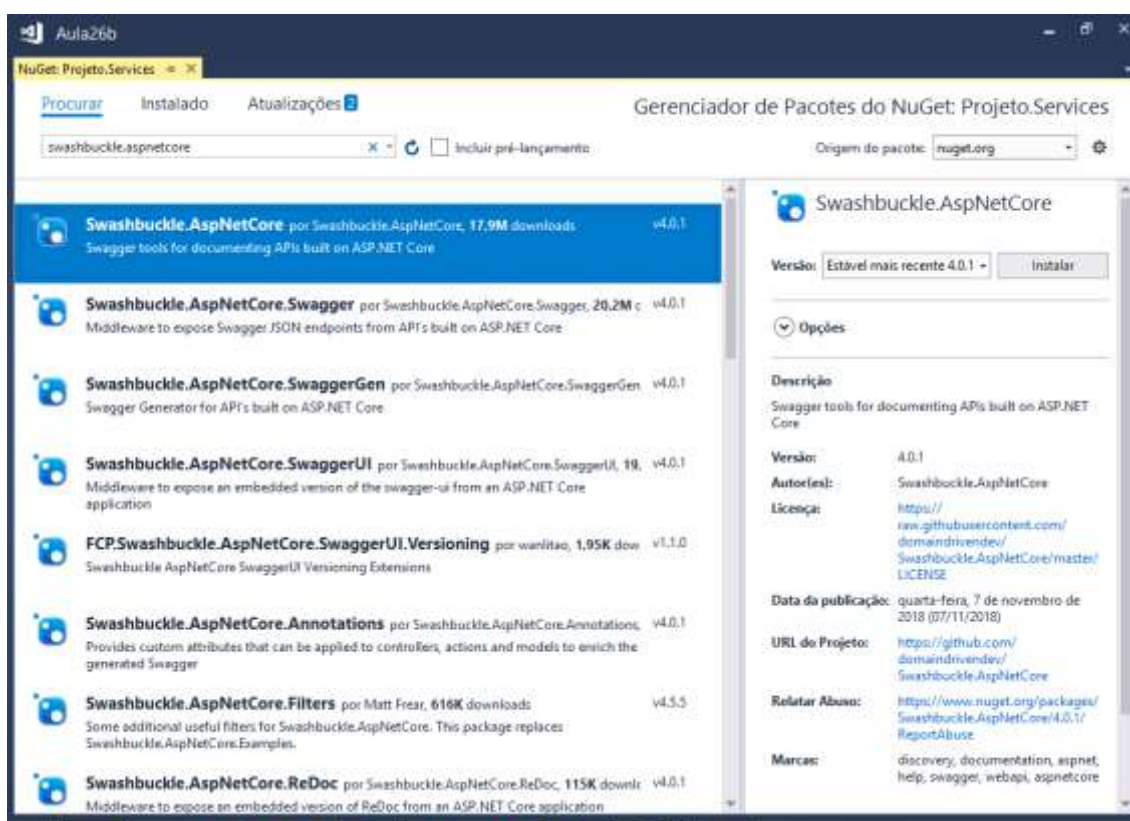
[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    return Ok("Cliente excluído com sucesso.");
}

[HttpGet]
[Produces(typeof(List<ClienteConsultaViewModel>))]
public IActionResult GetAll()
{
    return Ok();
}

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

```

Instalando o Swagger Gerenciador de pacotes do NuGet



Startup.cs

Configurando o Swagger

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using Swashbuckle.AspNetCore.Swagger;

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();

            services.AddSwaggerGen(
                sw =>
                {
                    sw.SwaggerDoc("v1",
                        new Info
                        {
                            Title = "Aula - C# WebDeveloper Turna Noite",
                            Description = "Projeto Asp.Net Core API",
                            Version = "v1",
                            Contact = new Contact
                            {
                                Name = "COTI Informática",
                                Email = "contato@cotiinformatica.com.br",
                                Url = "http://www.cotiinformatica.com.br"
                            }
                        }
                    );
                }
            );
        }

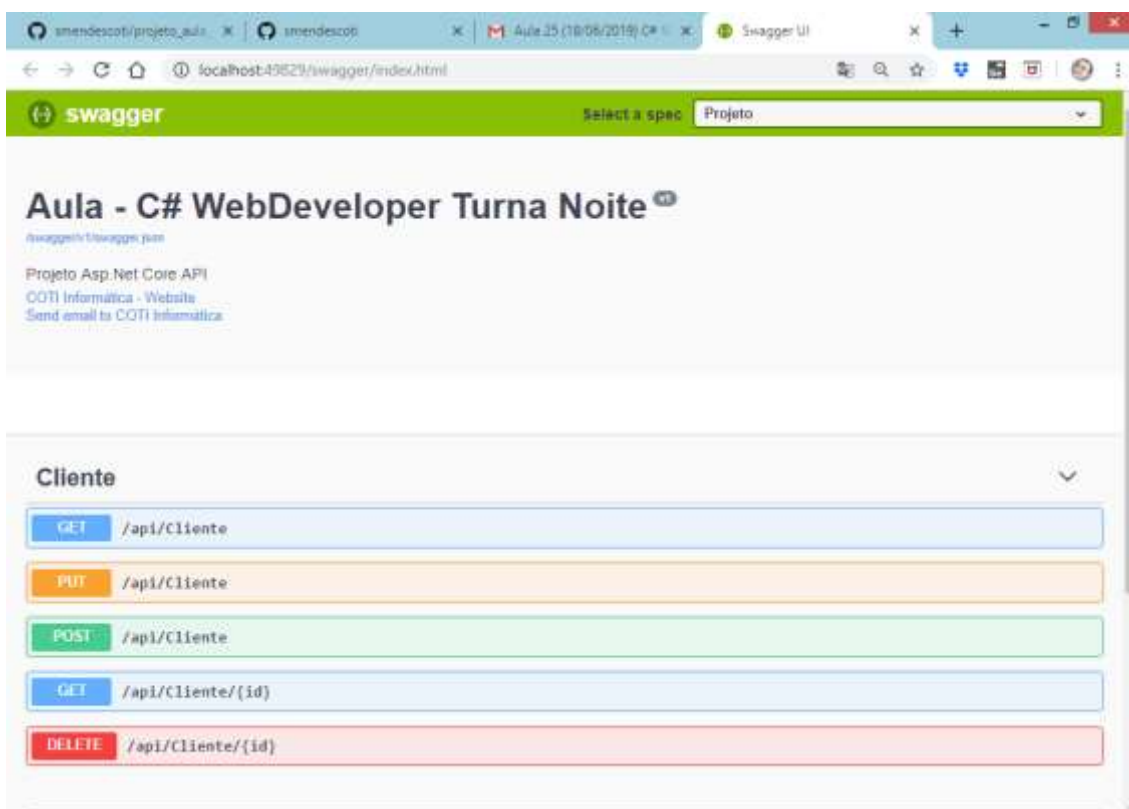
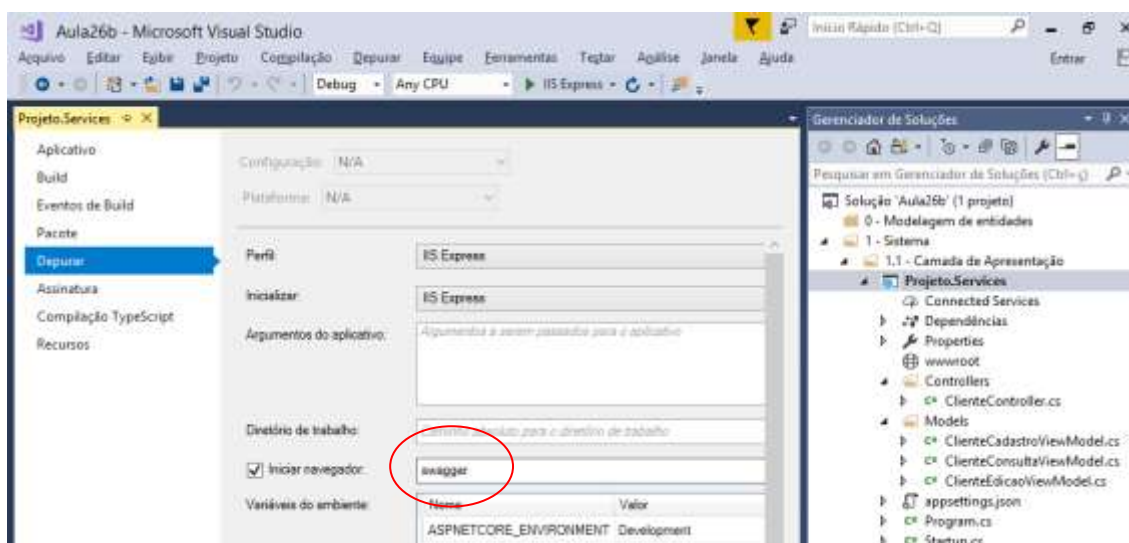
        // This method gets called by the runtime. Use this method to configure
        // the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
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



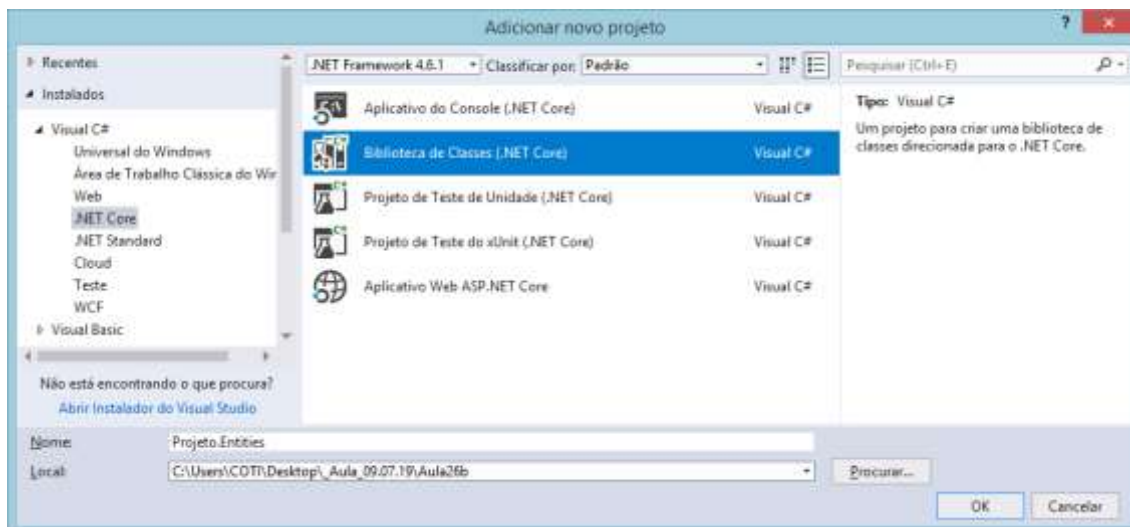
```
app.UseMvc();

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

Configurando a página inicial do projeto para "swagger"



Criando as demais partes do projeto:



using System;

namespace Projeto.Entities

```
{
    public class Cliente
    {
        public int IdCliente { get; set; }
        public string Nome { get; set; }
        public string Email { get; set; }
    }
}
```

