

# Projeto de Banco de Dados e OO .NET

Consumindo SERVIÇOS WEB no C#

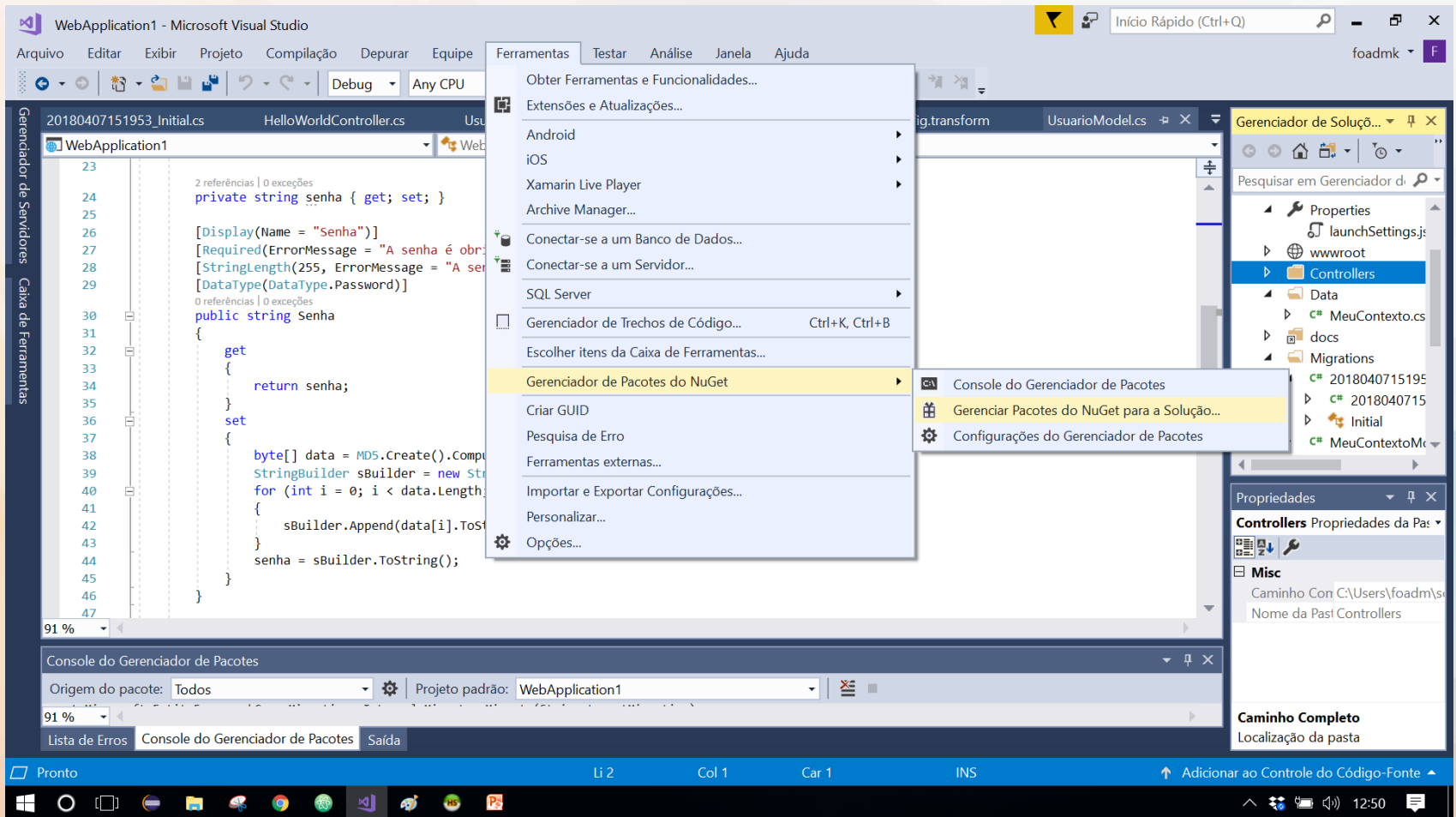
# Consumindo SERVIÇOS WEB no C#

- Crie um novo aplicativo no console. Escolha a opção (.NET Core)



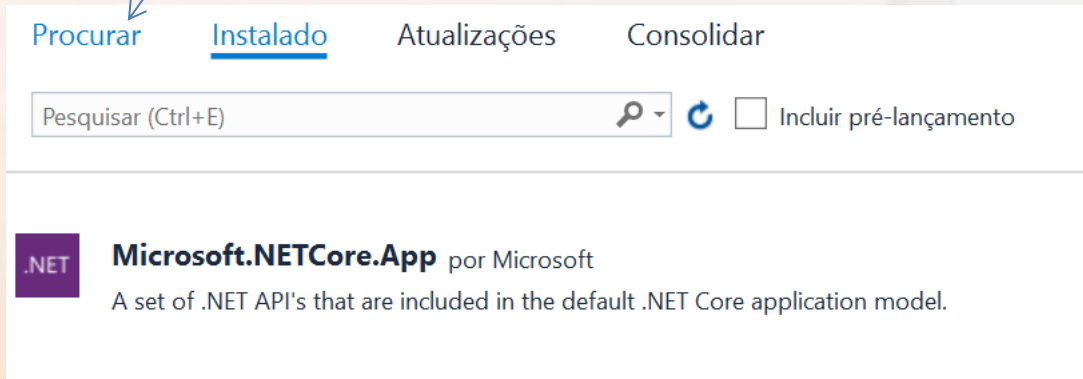
- Somente na opção .NET Core existe a possibilidade de usar o pacote `System.Net.Http`

# Instalando o Pacote com Gerenciador NuGet




# Procurando por pacote no NuGet

Clique em procurar

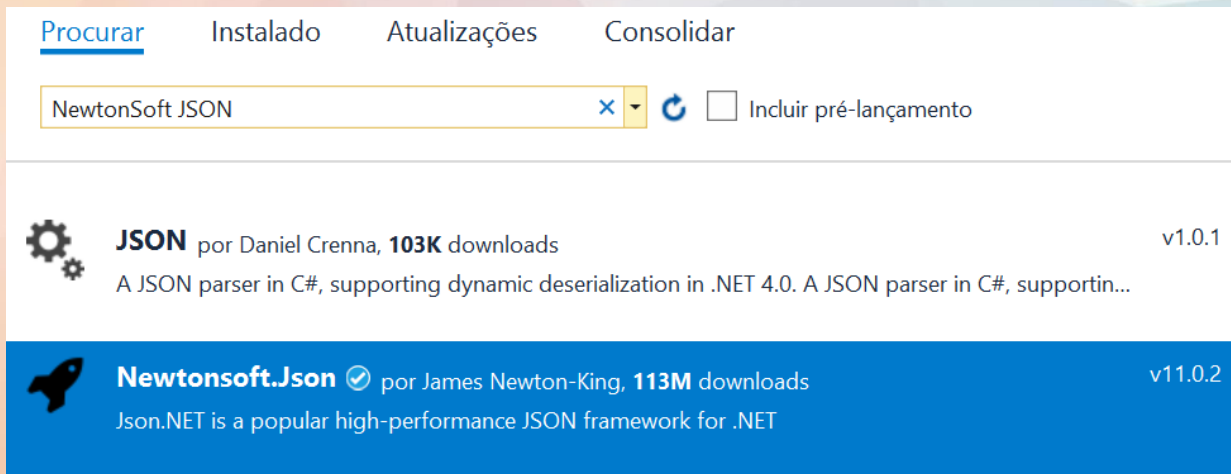


Procurar Instalado Atualizações Consolidar

Pesquisar (Ctrl+E) 🔍 ↺ ☐ Incluir pré-lançamento




 **Microsoft.NETCore.App** por Microsoft  
A set of .NET API's that are included in the default .NET Core application model.

Pesquise por Newtonsoft JSON



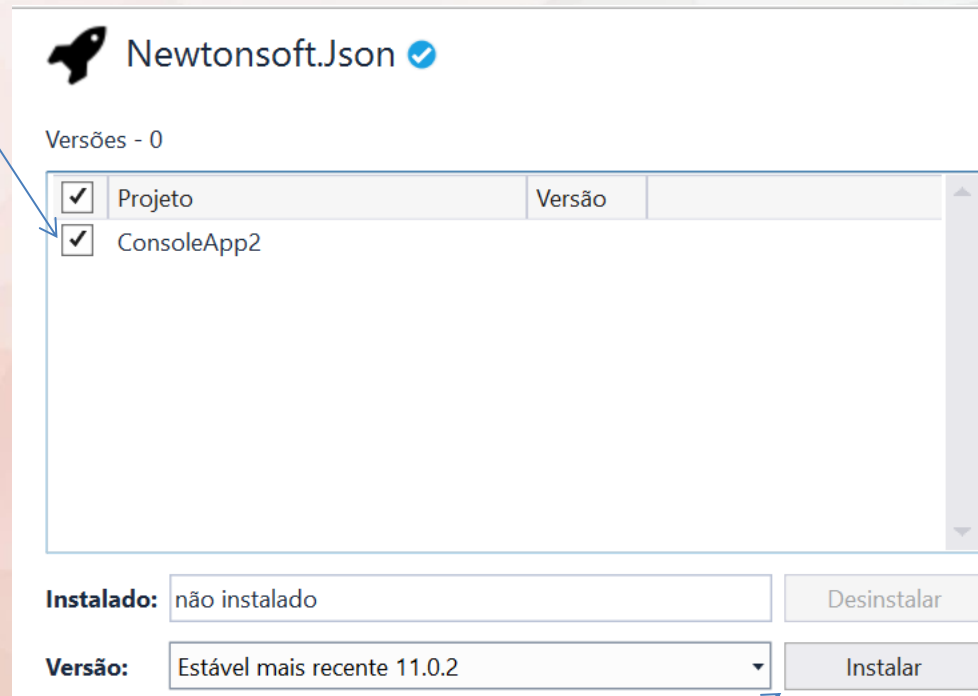
Procurar Instalado Atualizações Consolidar

NewtonSoft JSON ✕ ↺ ☐ Incluir pré-lançamento

	<b>JSON</b> por Daniel Crenna, <b>103K</b> downloads A JSON parser in C#, supporting dynamic deserialization in .NET 4.0. A JSON parser in C#, supportin...	v1.0.1
	<b>Newtonsoft.Json</b>  por James Newton-King, <b>113M</b> downloads Json.NET is a popular high-performance JSON framework for .NET	v11.0.2

# Instalando o Pacote

1. Selecione o seu Projeto



The image shows the package manager interface for Newtonsoft.Json. At the top, there is a rocket icon and the text "Newtonsoft.Json" with a blue checkmark. Below this, it says "Versões - 0". There is a table with two columns: "Projeto" and "Versão". The first row has a checked checkbox and the text "Projeto". The second row has a checked checkbox and the text "ConsoleApp2". Below the table, there are two fields: "Instalado:" with the value "não instalado" and a "Desinstalar" button; and "Versão:" with a dropdown menu showing "Estável mais recente 11.0.2" and an "Instalar" button. A blue arrow points from the text "1. Selecione o seu Projeto" to the checked checkbox in the "Projeto" row. Another blue arrow points from the text "2. Instale o Pacote" to the "Instalar" button.

Projeto	Versão
<input checked="" type="checkbox"/> Projeto	
<input checked="" type="checkbox"/> ConsoleApp2	

**Instalado:** não instalado Desinstalar

**Versão:** Estável mais recente 11.0.2 Instalar

2. Instale o Pacote

# Crie uma Struct Usuário na class Program

```
0 referências
class Program
{


0 referências
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
}
```

```
struct Usuario
{
    0 referências
    public int Id { get; set; }
    0 referências
    public string UserName { get; set; }
    1 referência
    public string Nome { get; set; }
    0 referências
    public string Senha { get; set; }
    0 referências
    public string CPF { get; set; }
    0 referências
    public DateTime Aniversario { get; set; }
}
```

# Crie um método myGET

```
0 referências
class Program
{
    2 referências
    struct Usuario...
    ...

    0 referências
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

A blue arrow points from the 'Main' method to the 'Usuario' struct, indicating a reference.

```
async static void myGET(string RestUrl)
{
    HttpClient client;
    client = new HttpClient();

    var uri = new Uri(string.Format(RestUrl, string.Empty));

    var response = await client.GetAsync(uri);

    List<Usuario> usuarios;

    Console.WriteLine("Ok, vamos lá");

    if (response.IsSuccessStatusCode)
    {
        string str = await response.Content.ReadAsStringAsync();
        usuarios = JsonConvert.DeserializeObject<List<Usuario>>(str);
        Console.WriteLine(usuarios[0].Nome);
    }
    else
    {
        Console.WriteLine(response.StatusCode);
    }
}
```

# Código para CTRL-C CTRL-V

```
struct Usuario
{
    public int Id { get; set; }
    public string UserName { get; set; }
    public string Nome { get; set; }
    public string Senha { get; set; }
    public string CPF { get; set; }
    public DateTime Aniversario { get; set; }
}
```

```
static void Main(string[] args)
{
    myGET("http://localhost:51542/api/UsuarioAPI");

    while(true)
    {

    }
}
```

```
async static void myGET(string RestUrl)
{
    HttpClient client;
    client = new HttpClient();

    var uri = new Uri(string.Format(RestUrl, string.Empty));

    var response = await client.GetAsync(uri);

    List<Usuario> usuarios;

    Console.WriteLine("Ok, vamos lá");

    if (response.IsSuccessStatusCode)
    {
        string str = await response.Content.ReadAsStringAsync();
        usuarios = JsonConvert.DeserializeObject<List<Usuario>>(str);
        Console.WriteLine(usuarios[0].Nome);
    }
    else
    {
        Console.WriteLine(response.StatusCode);
    }
}
```

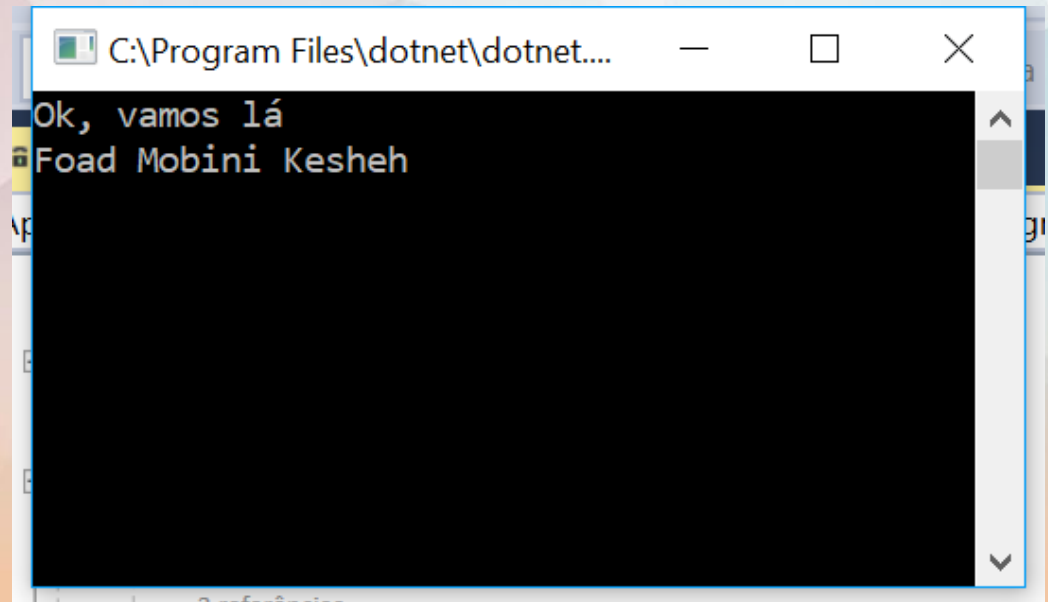


# Altere o método Main

```
0 referências
class Program
{
    2 referências
    struct Usuario...
    ...
    0 referências
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

```
0 referências
static void Main(string[] args)
{
    myGET("http://localhost:51542/api/UsuarioAPI");

    while(true)
    {
        ...
    }
}
```



```
C:\Program Files\dotnet\dotnet...
Ok, vamos lá
Foad Mobini Kesheh
```

# Como funciona?

```
async static void myGET(string RestUrl)
{
    HttpClient client;
    client = new HttpClient();

    var uri = new Uri(string.Format(RestUrl, string.Empty));

    var response = await client.GetAsync(uri);

    List<Usuario> usuarios;

    Console.WriteLine("Ok, vamos lá");

    if (response.IsSuccessStatusCode)
    {
        string str = await response.Content.ReadAsStringAsync();
        usuarios = JsonConvert.DeserializeObject<List<Usuario>>(str);
        Console.WriteLine(usuarios[0].Nome);
    }
    else
    {
        Console.WriteLine(response.StatusCode);
    }
}
```

→ Cria um cliente HTTP

→ Cria uma URI a partir da URL que passamos

→ Cliente faz uma chamada tipo GET ao webservice

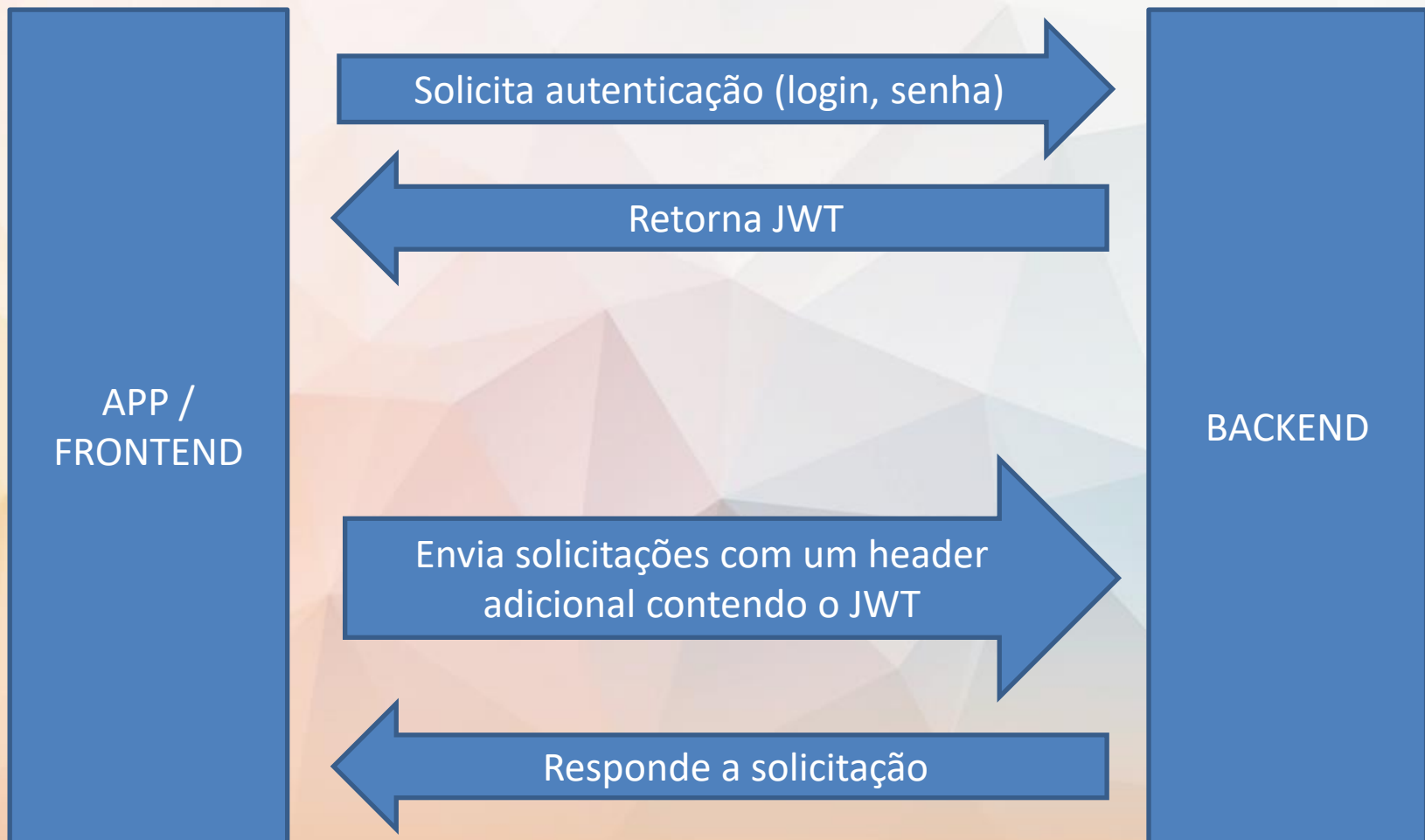
→ Se a resposta da chamada for bem sucedida (Cod. 200)

→ Le a resposta do webservice para uma string

→ Usa o Newtonsoft JSON para transformar a string em uma List<Usuario>

→ Mostramos no console o nome do primeiro usuário da lista.

# Como funciona a autenticação em serviços WEB



# JSON Web Tokens

- JWT - JSON Web Tokens
- Definidos pelo padrão [RFC 7519](#)
- Servem para garantir segurança entre duas parte
- Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

- Payload

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

- Verify Signature

```
HMACSHA256(  
base64UrlEncode(header) + "." +  
base64UrlEncode(payload),  
base64UrlEncode(secretToken))
```

**eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjMONTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ**