IdentityServer4

Manacés Pereira Follow
Mar 26, 2018 · 6 min read

O que é o IdentityServer4?

E aí, *bixo* solto! Beleza?

Autenticação de APIs REST é um assunto bem comum no desevolvimento de software, pois o que seria dos nossos serviços sem as devidas autenticações e autorizações? Existe um framework específico que eu gostaria de falar um pouco hoje, que é o IdentityServer4. Isso mesmo. Framework!

O IdentityServer4 é um framework que utiliza as tecnologias OpenID e OAuth 2.0 para ASP.NET Core 2. Ele é responsável por criar um serviço de autenticação completo, de única sessão de entrada e de saída para vários tipos de aplicações, sendo elas: mobile, web, nativas ou até mesmo outros serviços. Através disso, conseguimos controlar o acesso entre estas aplicações utilizando o token de acesso disponibilizado pela implementação do IdentityServer4. Por ser um framework OpenID e OAuth 2.0, temos suporte para provedores de identidade como: *Azure Active Directory, Google, Facebook, Twitter* e etc…

Ah, e tem mais! É de código aberto com suporte comercial de GRAÇA! Então te pergunto: por que não usar?

. . .

Breve resumo

Vou tentar mostrar pra vocês, basicamente, como configurar o seu servidor de autenticação e como pode ser feita a autorização de um usuário para sua API. Neste post, não pretendo mostrar todas a *features* do IdentityServer4, mas sim, criar de forma básica, uma API segura utilizando os conceitos mais básicos do IdentityServer4. Caso queiram se aprofundar mais no assunto, deixarei ao final deste post, uma lista de links.

Abaixo listo algumas coisas que você vou mostrar na próxima sessão deste post:

- Criação e configuração de uma API ASP.NET Core 2.0
- Configuração do IdentityServer4
- Autorização de uma API utilizando *login* e *senha* (modo mais comum)
- Criação de um *client* para testarmos nossa API segura

Sem mais delongas, vamos ao que interessa!

. . .

Show me the code

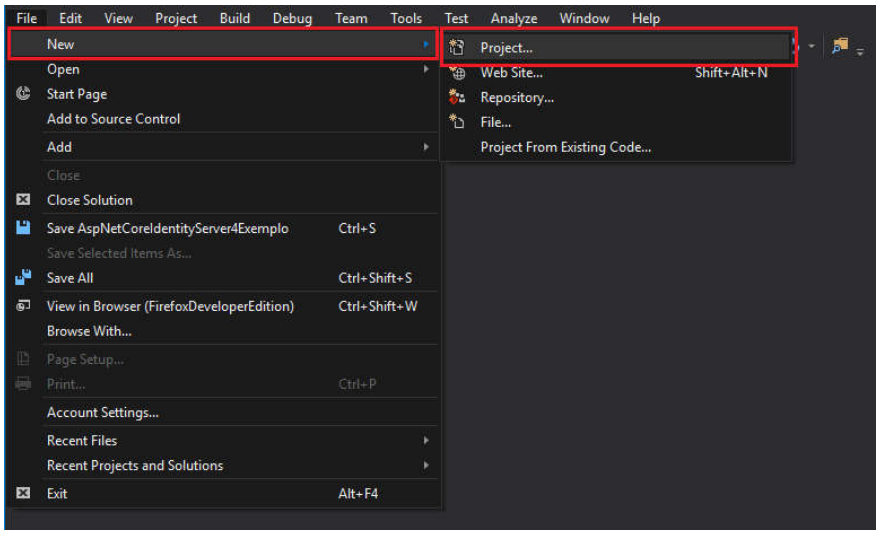
Vou separar esta sessão com base na sessão anterior. Então vamos lá!

1—Criando e configurando a API

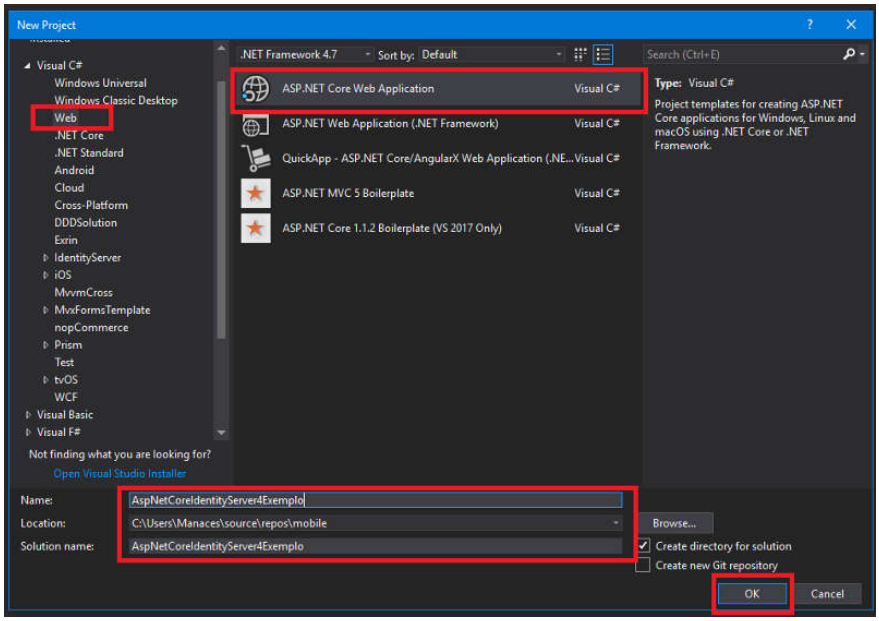
Não é obrigatório o uso do Visual Studio apesar de estar usando esta IDE.

Abaixo segue o fluxo de criação da nossa API ASP.NET Core 2.0:

Crie um novo projeto



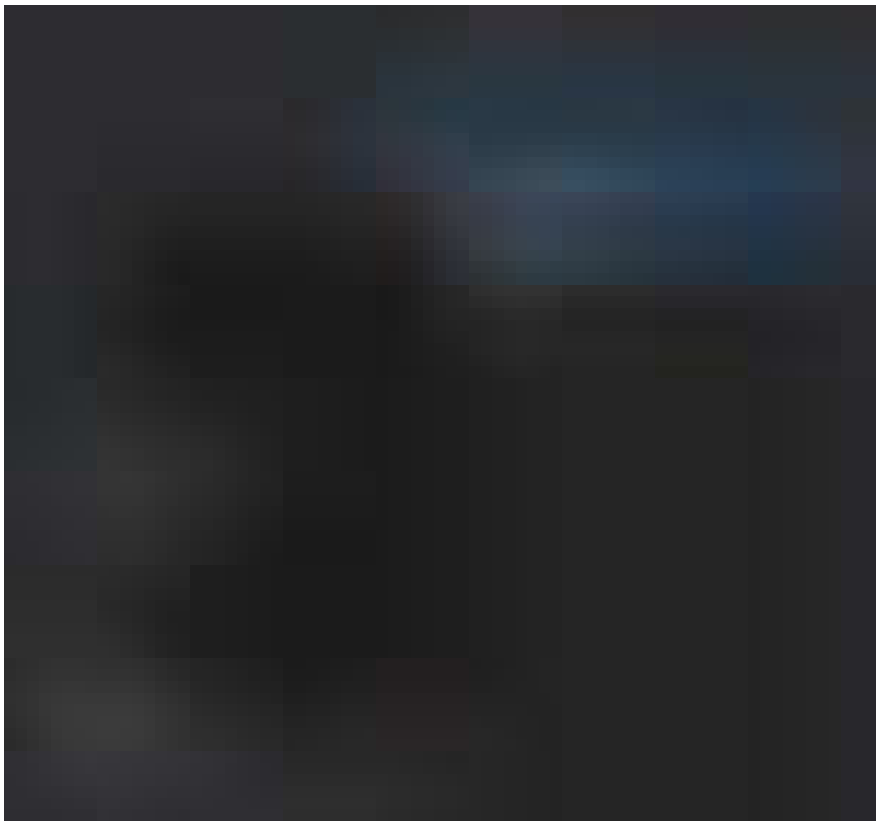
Escolha ASP.NET Core Web Application



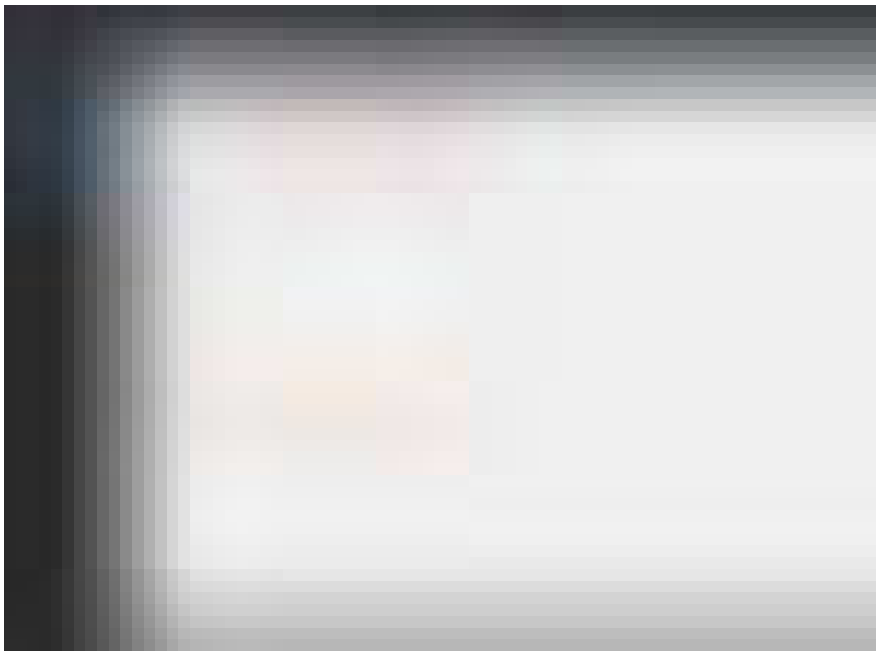
ATENÇÃO: Por padrão, o Visual Studio usa o IIS Express para executar as aplicações web. Porém é recomendado pela documentação oficial do IdentityServer4 rodar a API no console do sistema operacional utilizado, pois isso nos possibilita ver em tempo real o que está acontecendo com a nossa aplicação IdentityServer4 através do rico gerenciamento de log que o IdentityServer4 disponibliza para nós.

Para isso, precisamos configurar o modo como inicializamos a nossa aplicação. É só seguir os passos abaixo:

Propriedades da aplicação

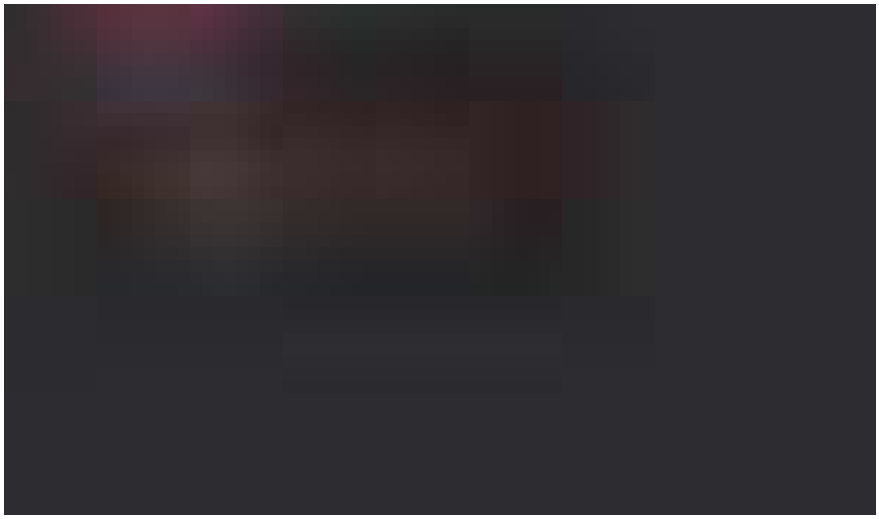


Agora vamos em Debug e configuramos como na imagem abaixo



Não esquecer de desmarcar a opção *Launch browser* como na imagem acima.

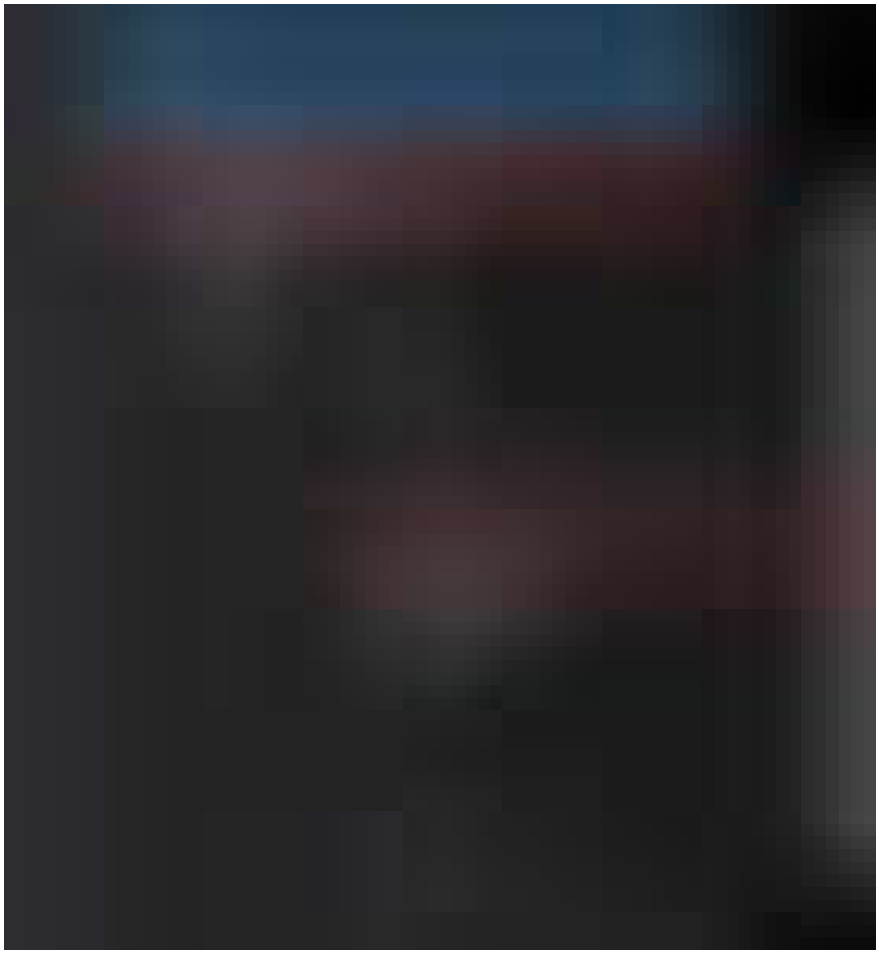
Agora selecione a aplicação e execute. Deve aparecer um console com o projeto iniciado.



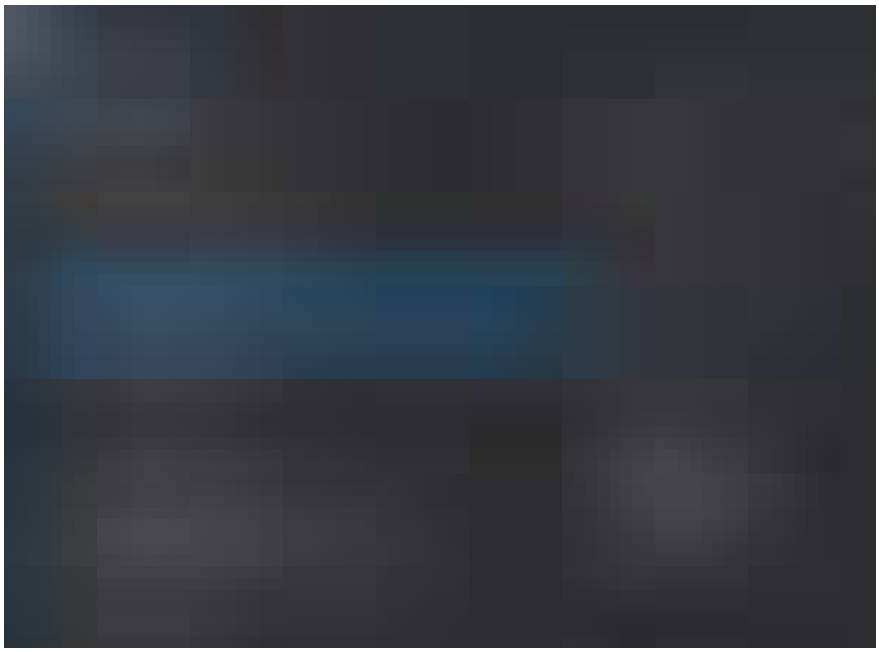
Pronto. É algo desse tipo que deverá aparecer para você.

Agora precisamos instalar os pacotes necessários para rodar o IdentityServer4 em nossa aplicação, e para isso, precisamos seguir os seguintes passos:

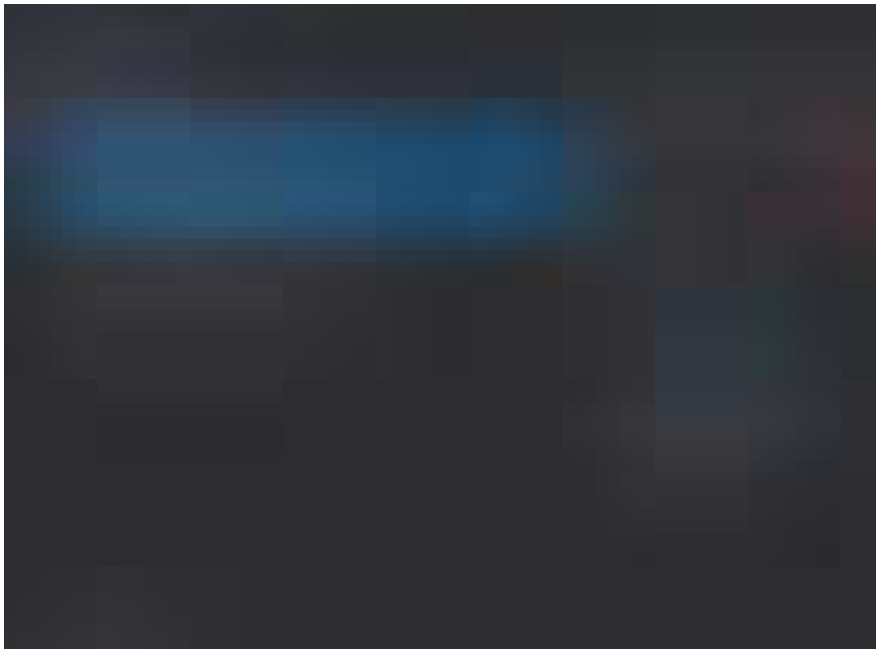
Vamos abrir o gerenciador de pacotes NuGet da nossa API



Após isso, vamos na aba Bowse, procuramos por IdentityServer4 e clicamos em Install



Faremos o mesmo para o pacote IdentityServer4.AccessTokenValidation

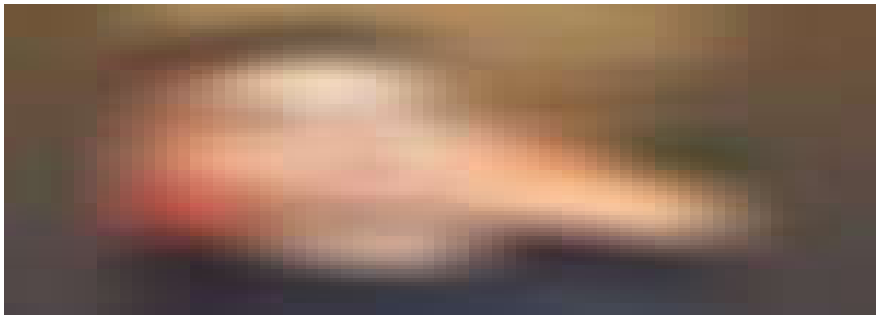


Pronto! Com nossa aplicação pronta para configuração do IdentityServer4, passamos para o passo 2 logo abaixo.

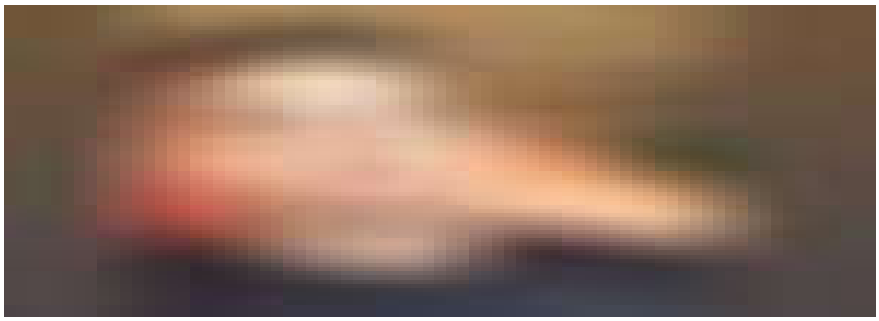
2—Configurando o IdentityServer4

OBS: Neste código temos um ambiente simples, ou seja, sem nenhum tipo de complexidade. Essa é a intenção para que todos possam aprender. Não é recomendado colocar este código em produção e sim desenvolver um código com persistência de dados, utilizando SqlServer. Também não tenho intenção de mostrar um código perfeito, pois é apenas para estudo como falei anteriormente.

Vamos criar uma classe *Config.cs* que será responsável pelas configurações do nosso servidor de identidade.



Nossa classe *Startup.cs* deverá conter o seguinte código



Para ver se deu tudo certo, podemos executar a aplicação e acessar *http://localhost:5000/.well-known/openid-configuration.* Deverá aparecer as configurações do nosso servidor dessa forma

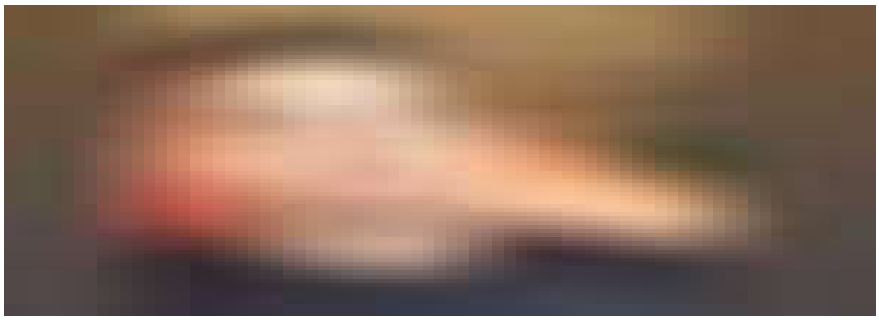
Deu tudo certo, se apareceu algo parecido com isto. Se tiver algum problema, sugiro seguir desde o passo 1 novamente.

Simples assim! Nosso servidor de identidade está configurado. Agora vamos criar um controlador seguro para que possamos testar nossa validação.

3—Criando um controlador seguro

Precisamos agora criar um controlador que será acesso somente através de um token de acesso, que será fornecido pelo nosso servidor de identidade a partir de um login e uma senha.

Vamos criar a classe *IdentityController.cs* que terá o seguinte código

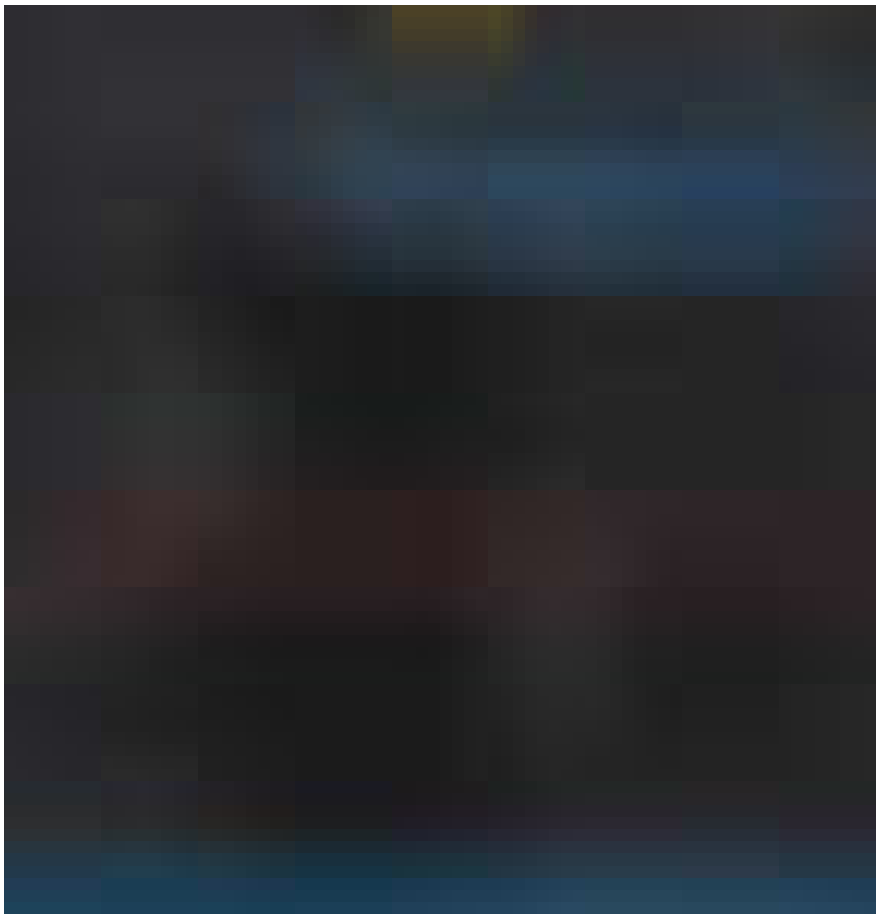


Neste código, estamos retonando um resultado em JSON com todas as Claims do usuário logado em nosso servidor. Vamos ver na próxima etapa, que é quando criamos um client para testar nossa api segura.

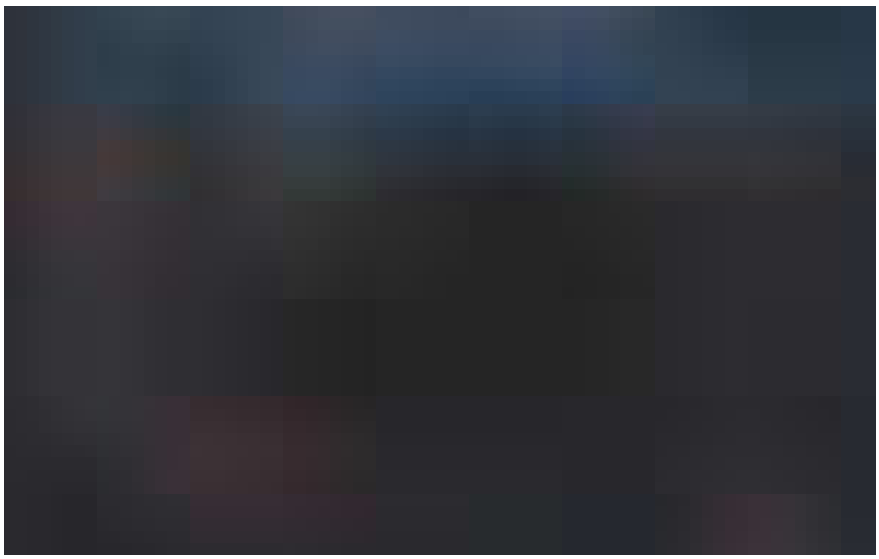
4—Criando o cliente de teste da nossa API segura

Para criar nosso client, vamos precisar criar um projeto do tipo *Console*.

Vamos na solução da aplicação e clicamos com o botão direito do mouse e depois clicamos em *Add > New Project*



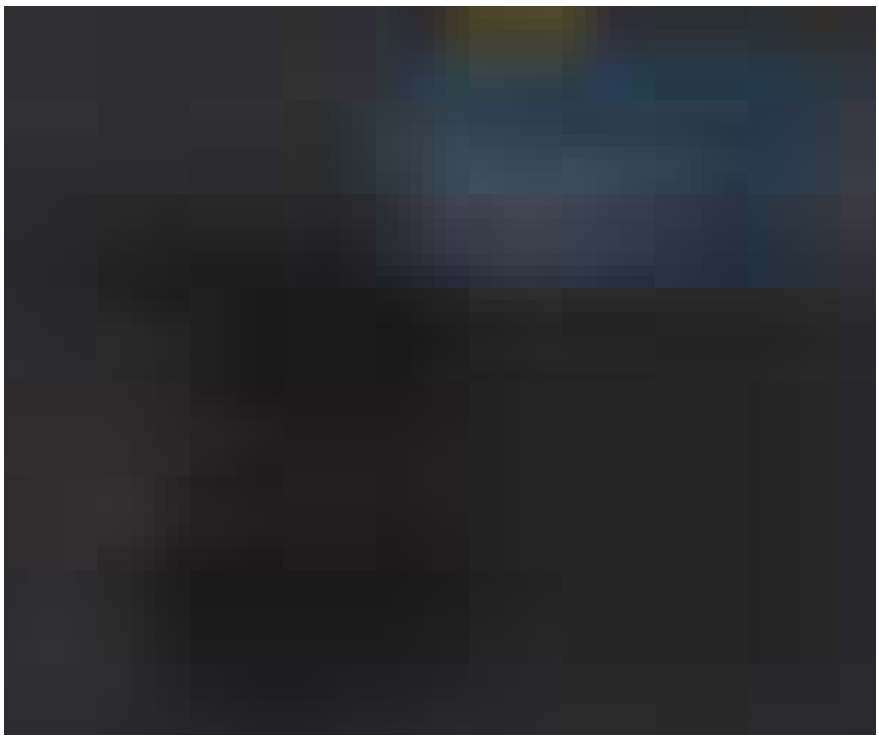
Selecione o projeto do tipo *Console App (.NET Core)*



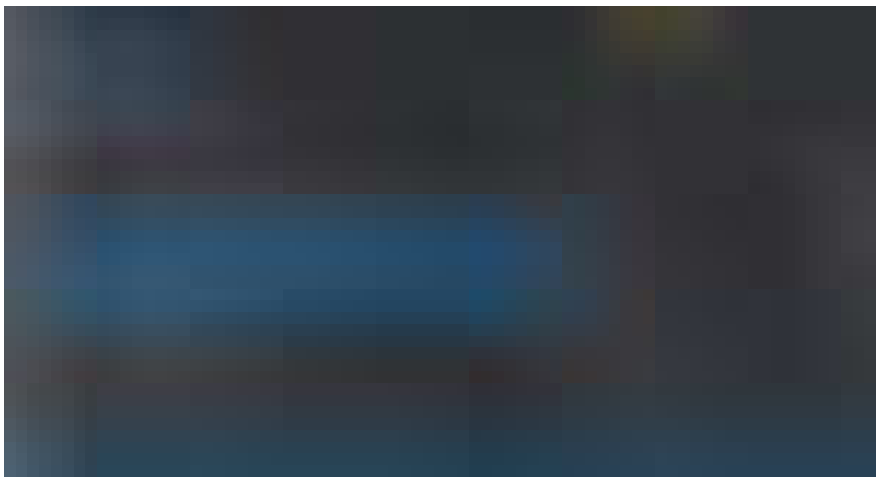
Pronto. Projeto criado. Agora precisamos instalar uma dependência

para criamos um console de testes.

Para isso, vamos no gerenciador de pacotes do NuGet

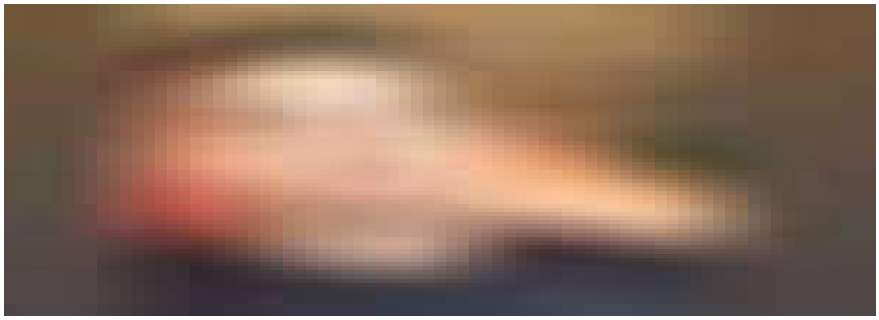


Precisamos pesquisar por *IdentityModel,* selecionar o pacote e em seguida, clicar em *Install*



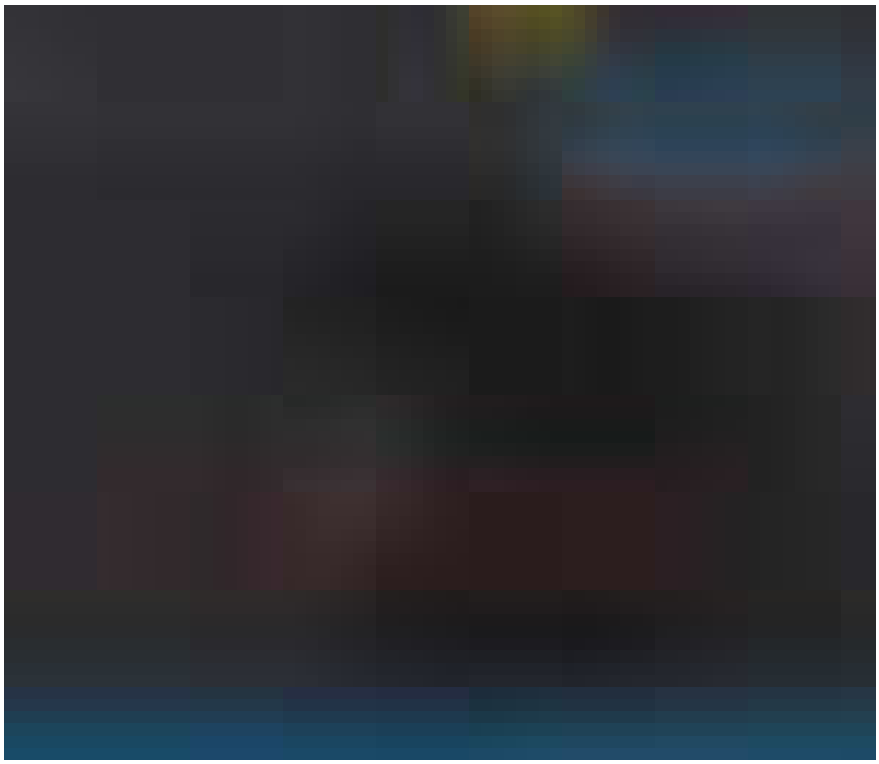
Tudo pronto. Agora vamos ao que interessa que é o código de testes.

Na classe *Program.cs* vamos escrever o código como abaixo. (não irei me alongar sobre o código, pois o mesmo está comentado e explicado)

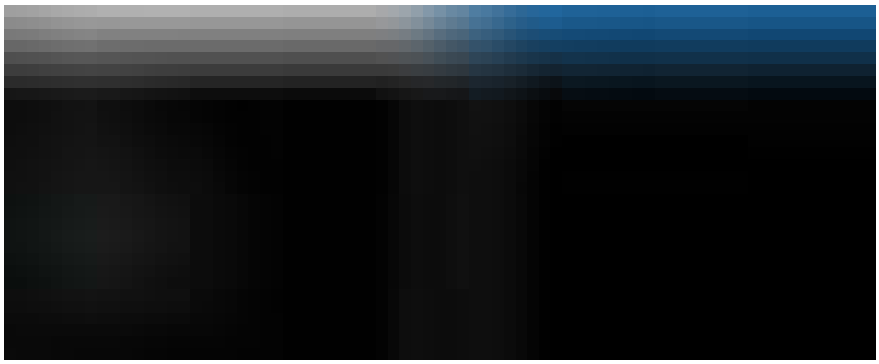


Vamos rodar a nossa API e em seguida, vamos rodar o projeto console.

Para rodar ambos simultaneamente, podemos selecionar com o botão direito o projeto console e clicar em *Debug > Start New Instance* com o projeto API já rodando.



O resultado esperado deverá ser mais ou menos dessa forma



Link do projeto completo:
https://github.com/agenciariseup
/AspNetCoreIdentityServer4Example

Até a próxima, *bixo solto*! Espero que tenham gostado. Estou aberto à dúvidas e críticas construtivas.

Referências:
https://identityserver.io/
https://identityserver4.readthedocs.io/en/release/
https://www.microsoft.com/net