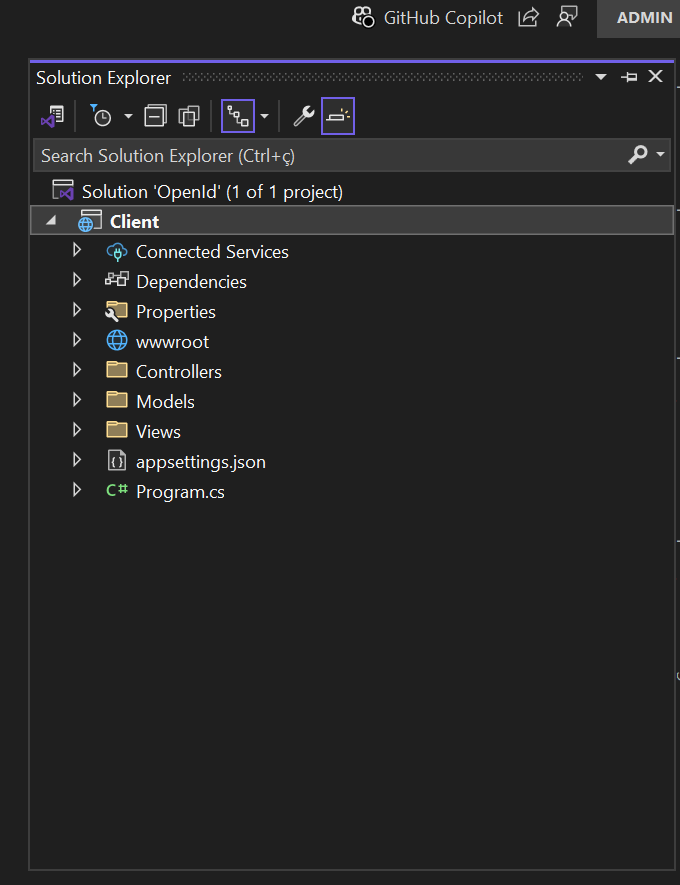
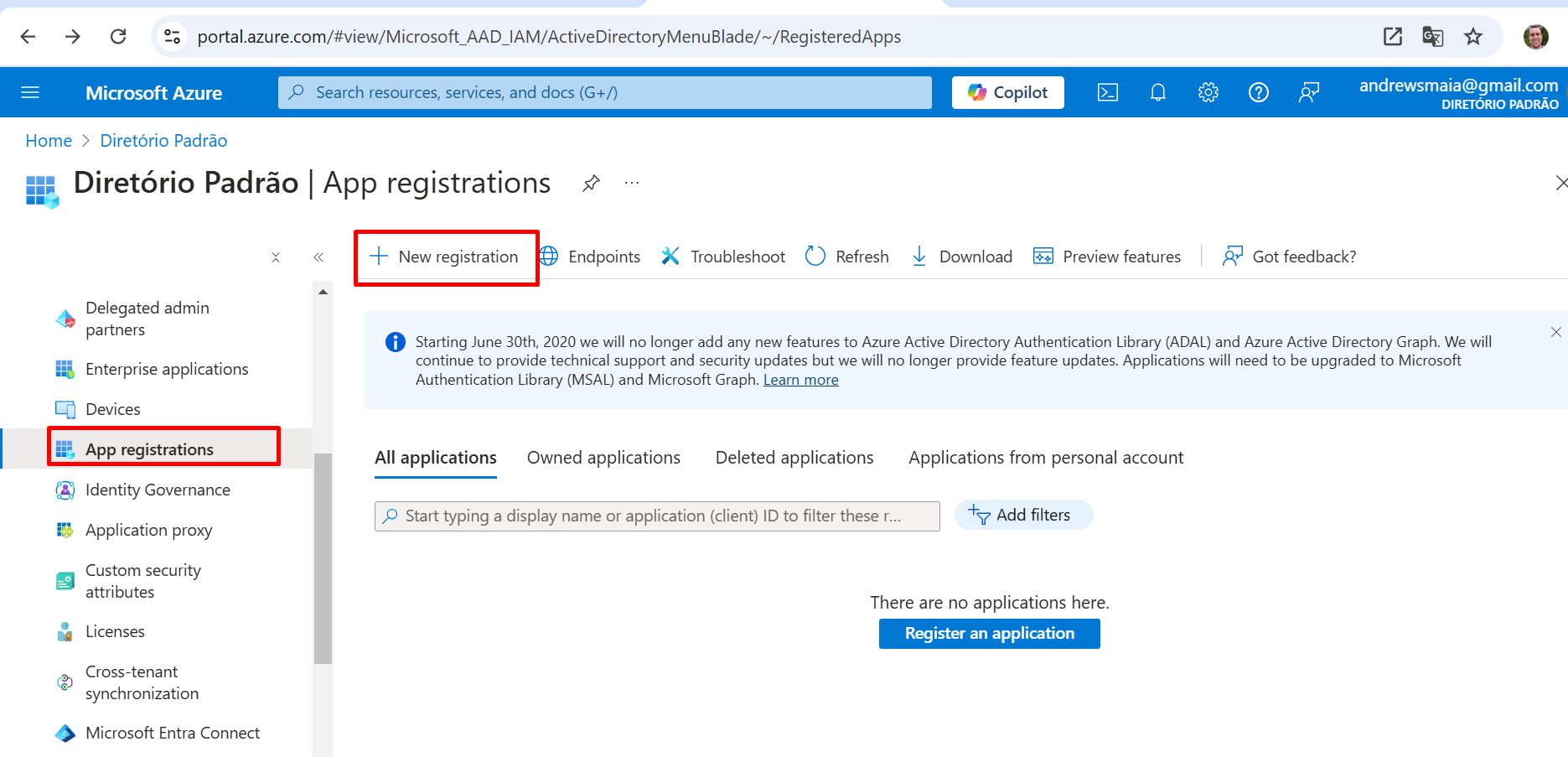
Temos aqui uma solução com um projeto chamado client que representará o personagem client:

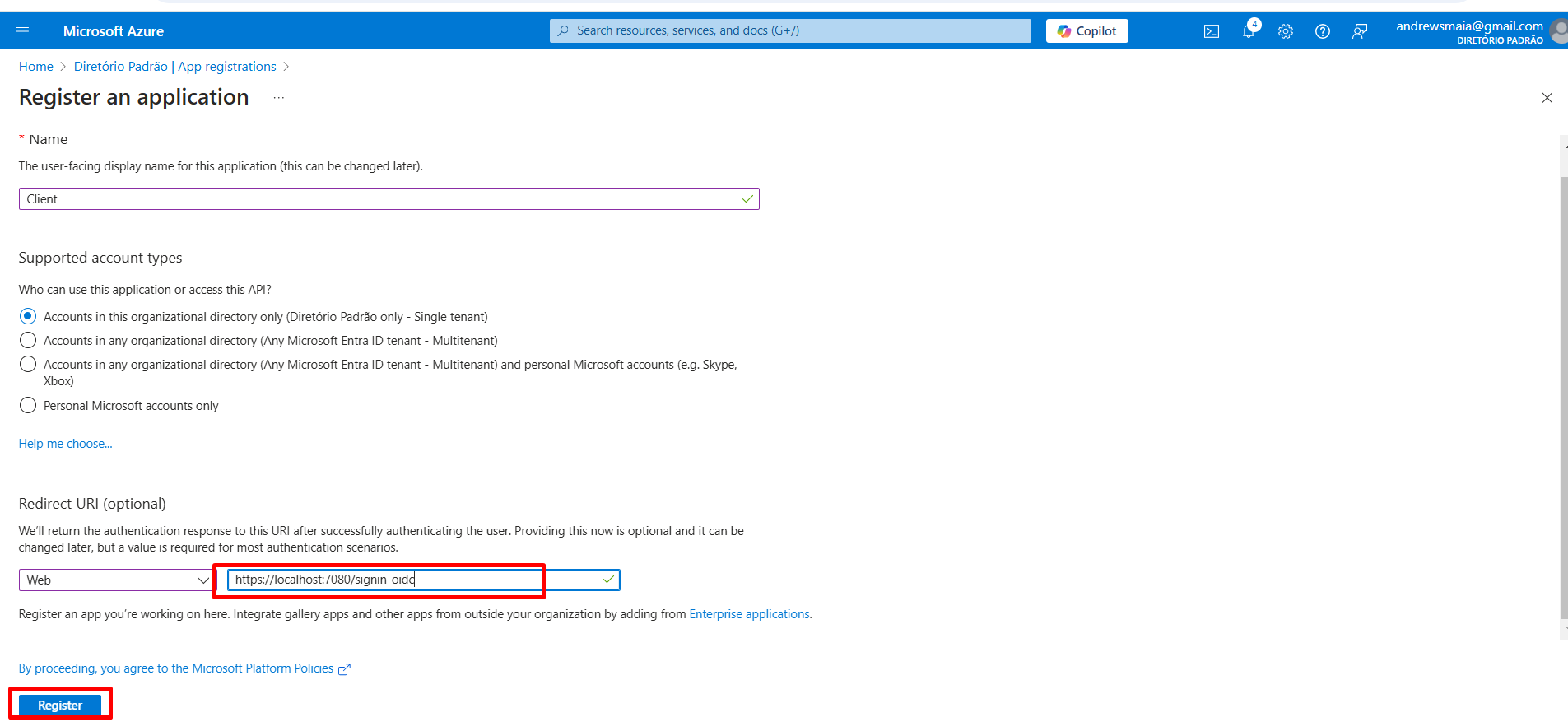


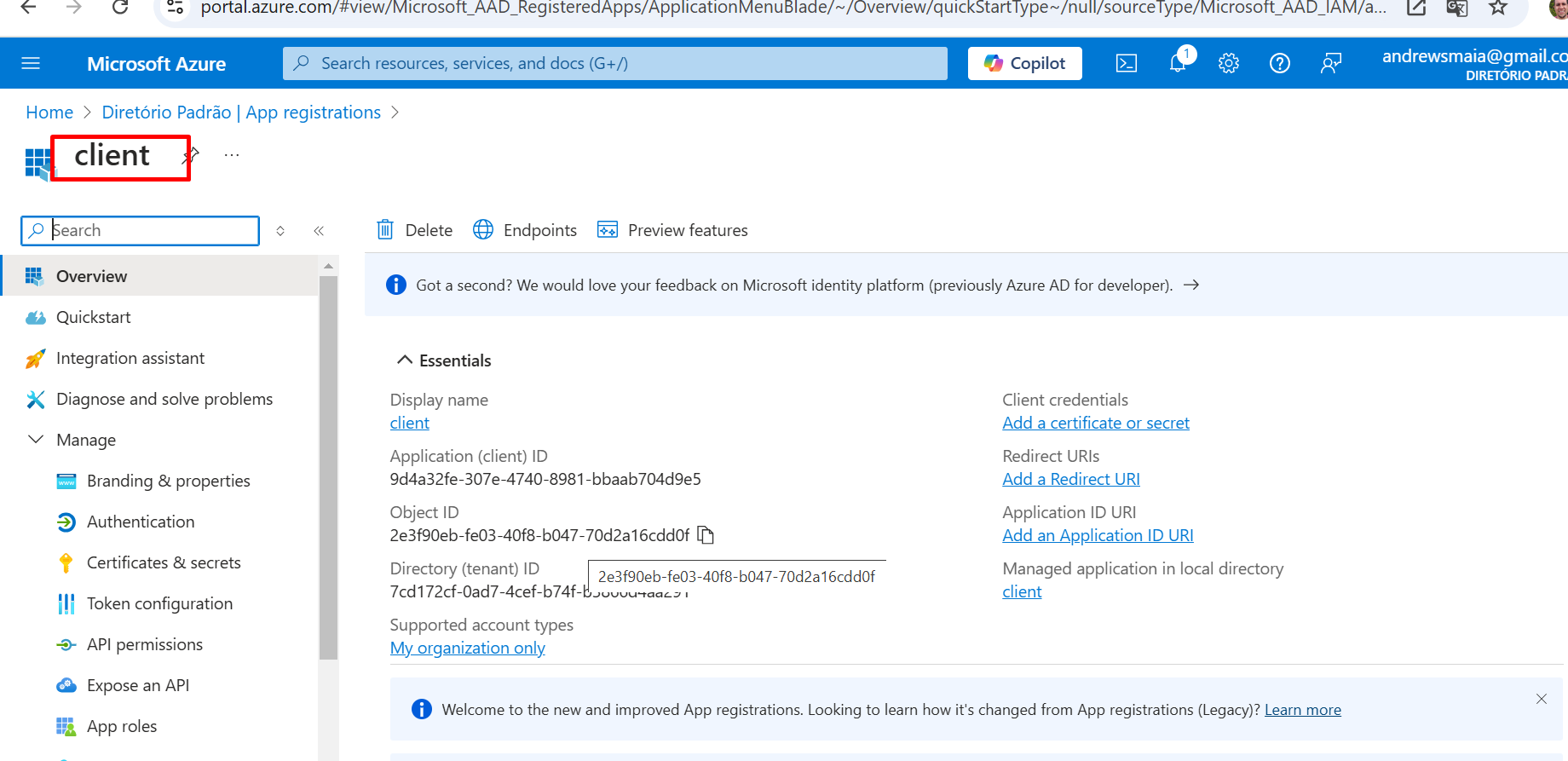
Para representar o Identity Provider teremos novamente o Microsoft Entra.

E a primeira coisa que precisamos fazer é registrar o client:

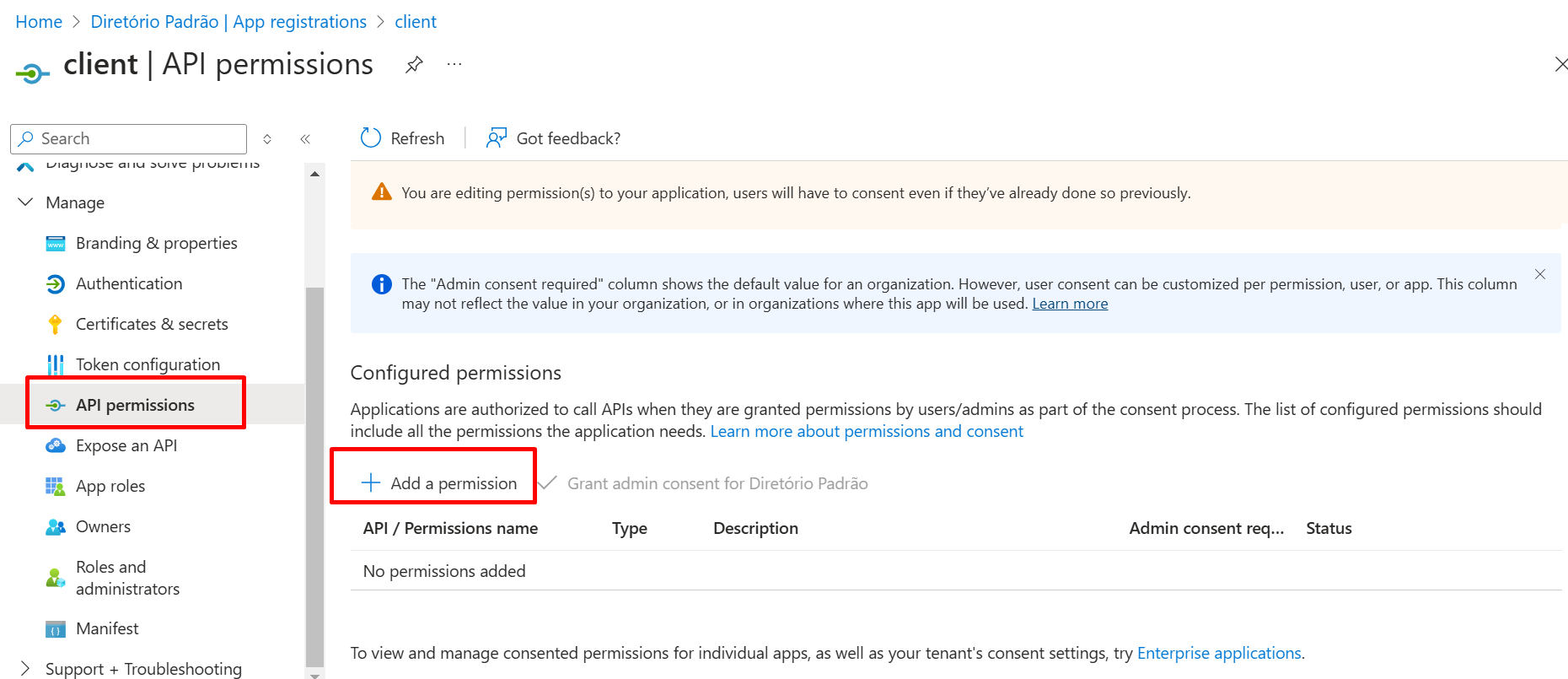


Para adiantar aqui já podemos adicionar a URI de redirecionamento que vamos utilizar. No caso: https://localhost:7080/signin-oidc

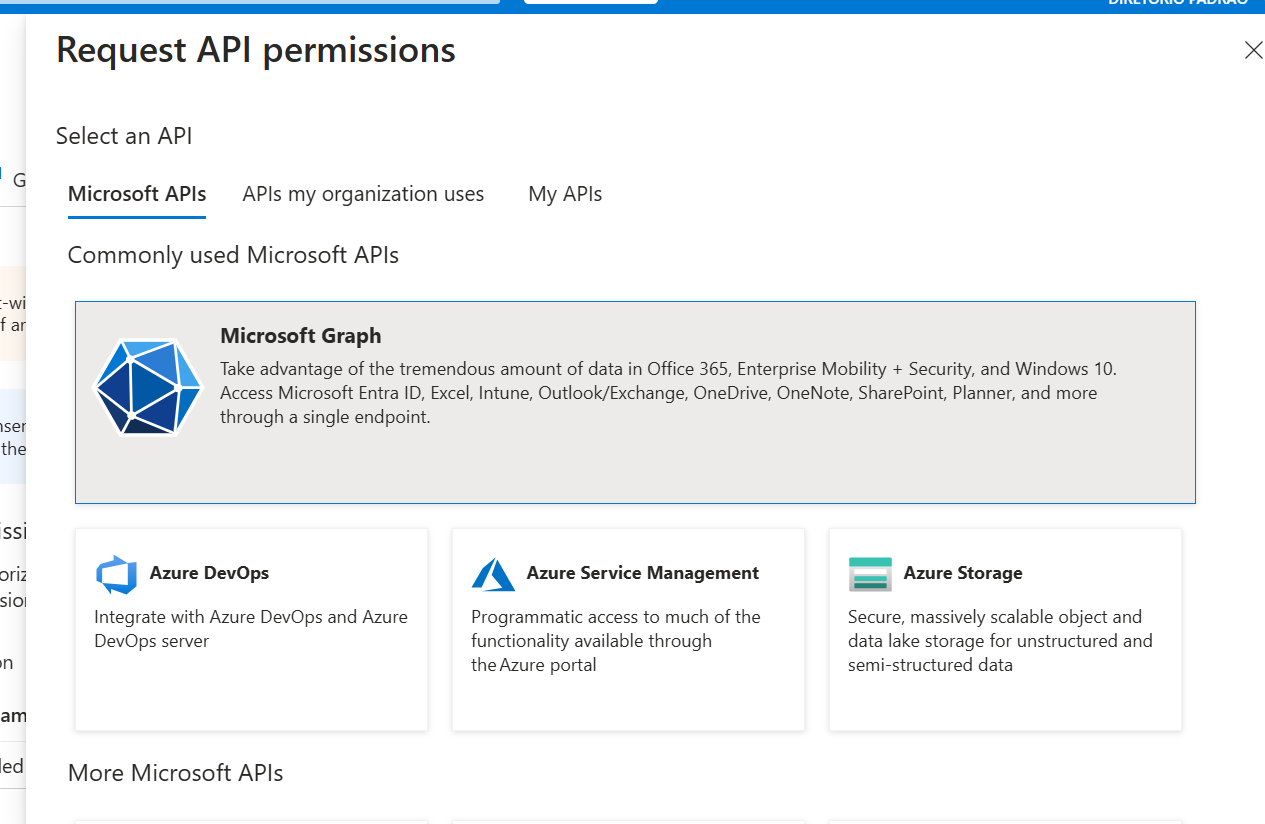




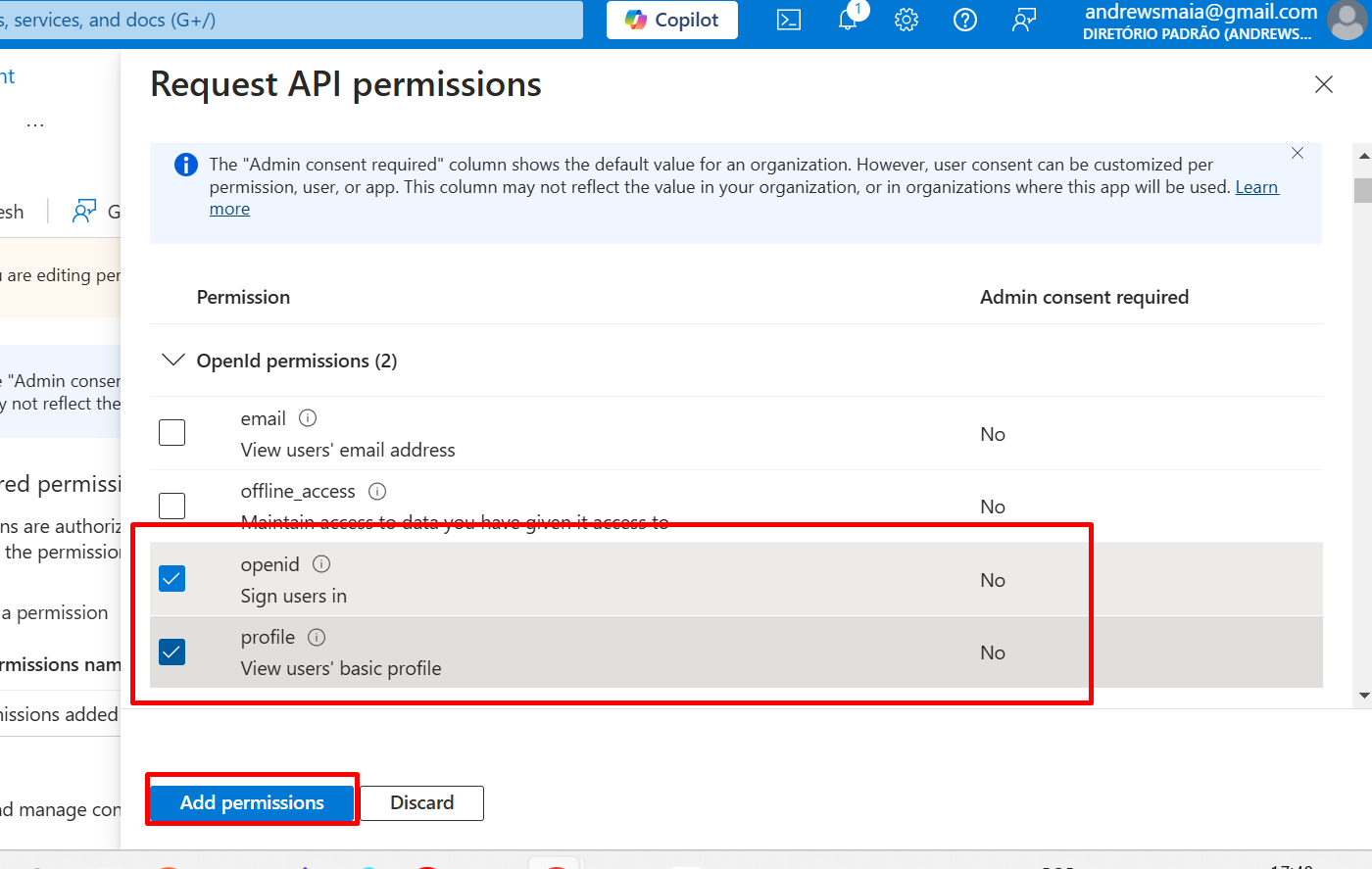
Depois precisamos dar a permissão aos escopos que esse client terá acesso:

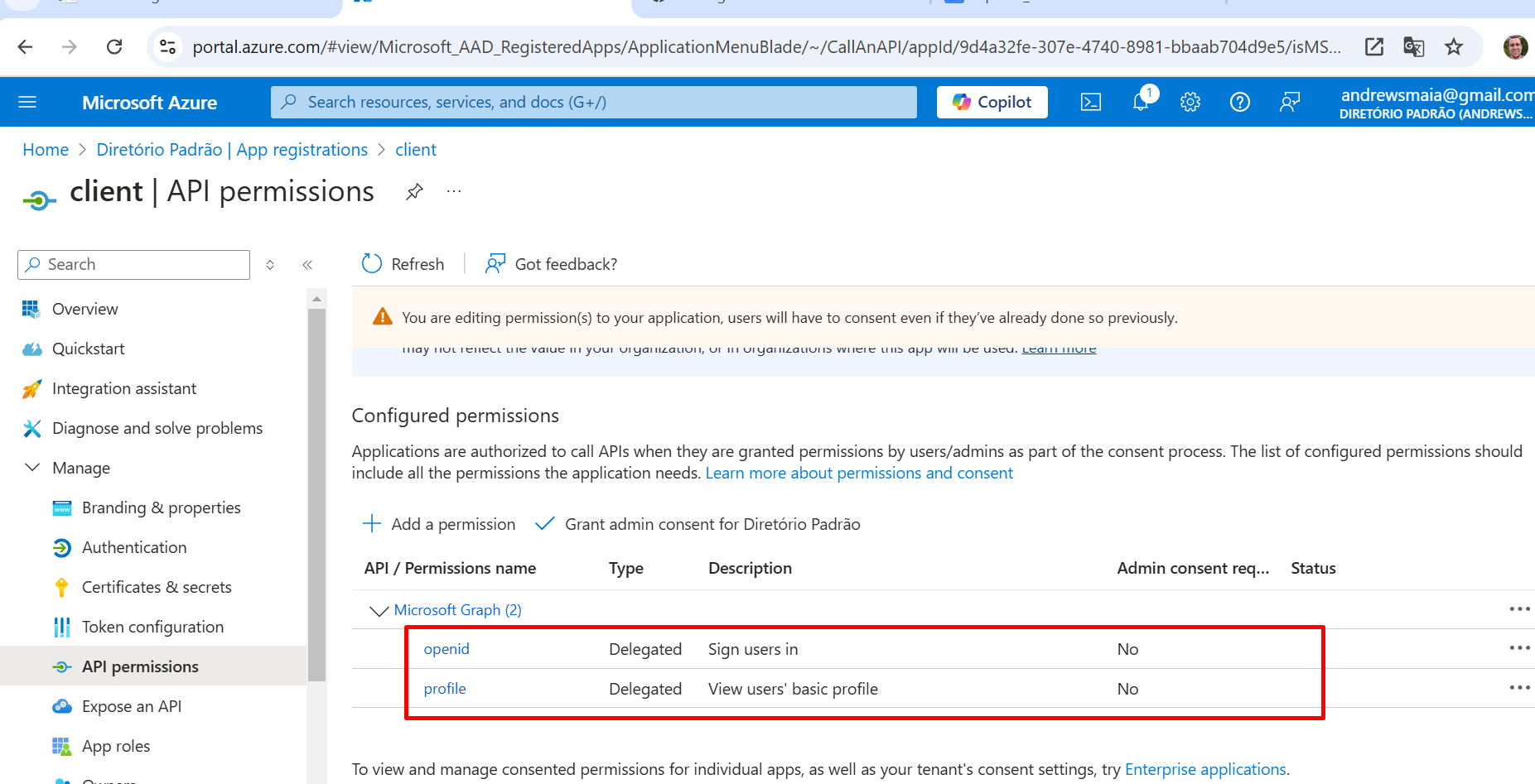


Aqui não precisamos criar um escopo. Eu vou aproveitar alguns escopos que já existem nativamente no Microsoft Entra através do aplicativo Microsoft Graph

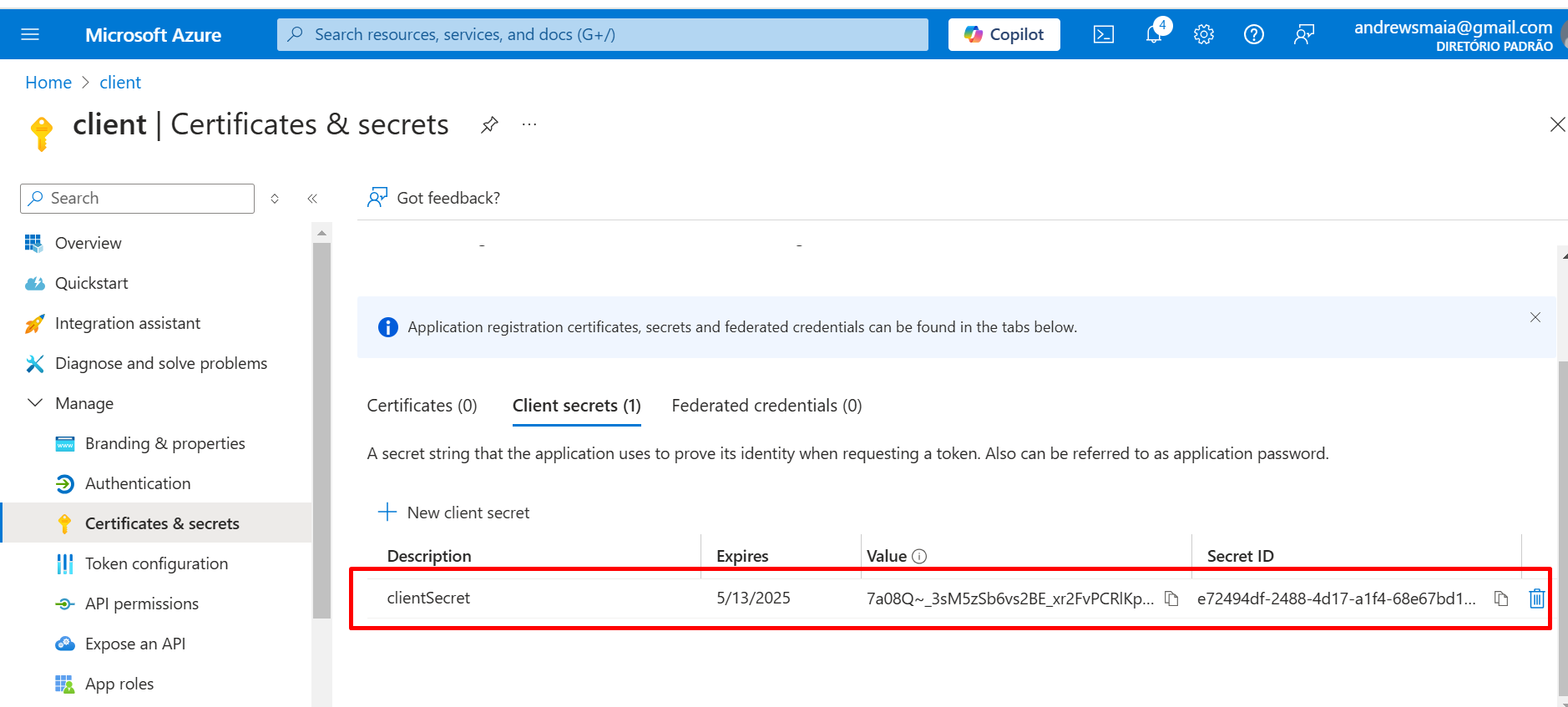


Esse App já possui escopos voltados para o openID:

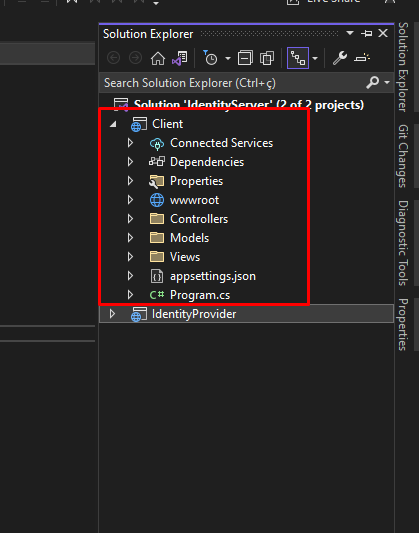




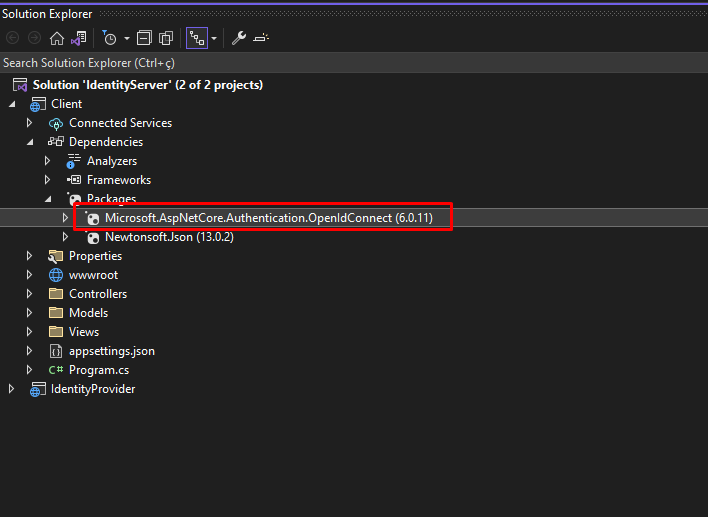
A outra coisa que já vamos deixar pronto aqui já é o secret do client:



Agora vamos para o projeto do client que fará uso da autenticação do Identity Provider:

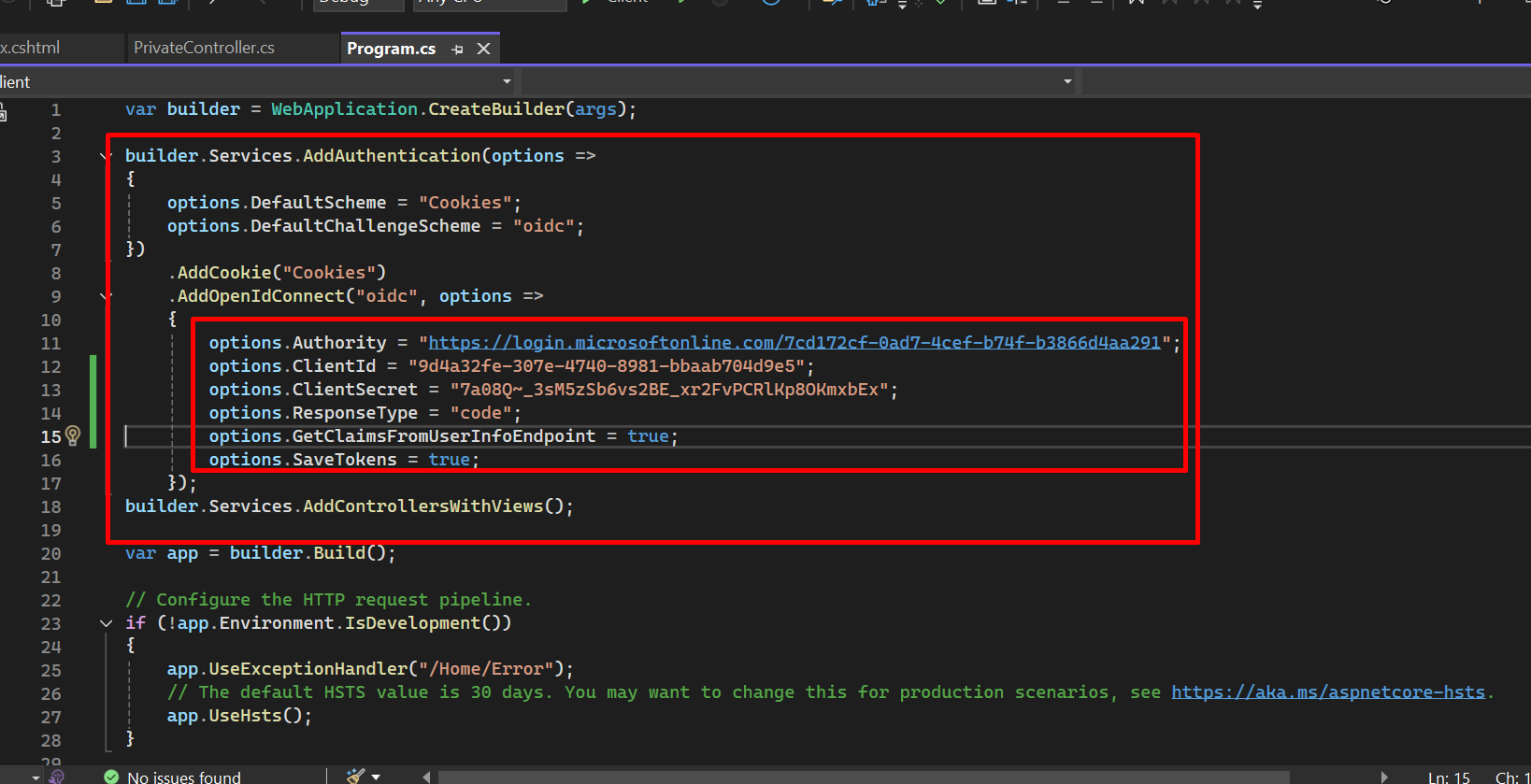


A primeira coisa que devemos fazer é instalar o pacote ‘Microsoft.AspNetCore.Authentication.OpenIdConnect’:



É através desse pacote que faremos a autenticação usando o protocolo OpenID Connect.

Através do método AddOpeIdConnect configuramos a autenticação via nosso Identity Provider.



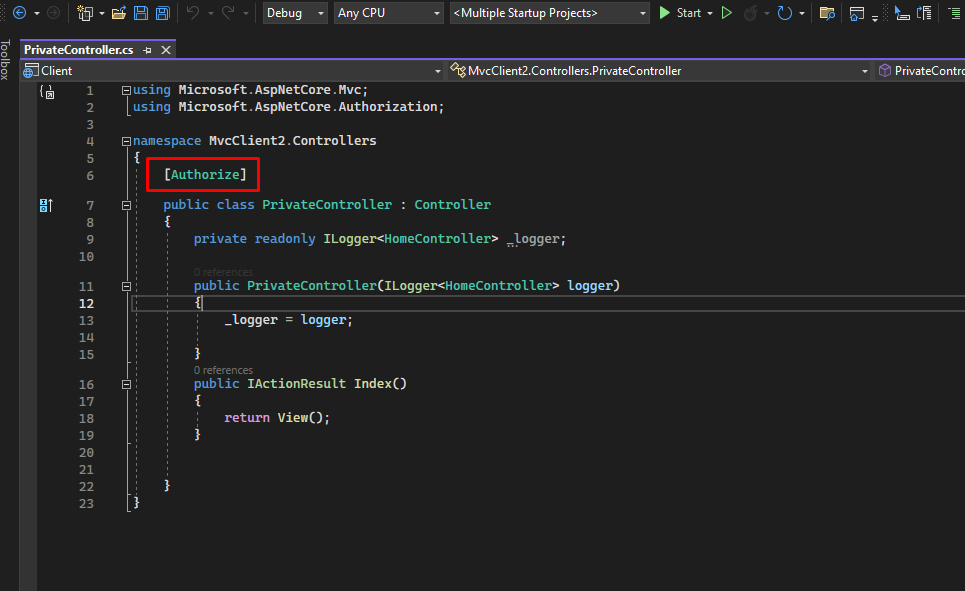
Através do método AddOpeIdConnect configuramos a autenticação via nosso Identity Provider.

Aqui temos que fornecer a url da autoridade que é o endereço da Microsoft Entra com o meu tenantId. O clientId do meu client e o secret gerado. O response type code.

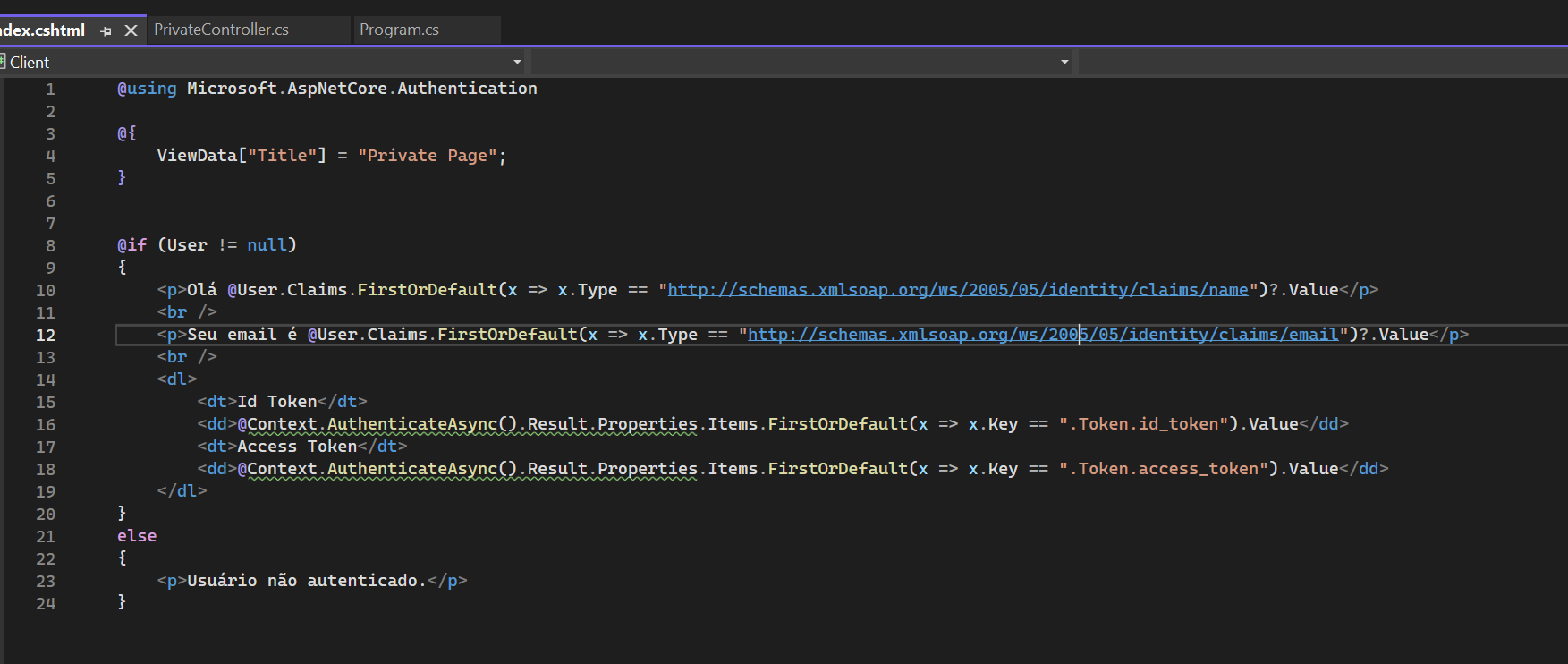
Repare também que não precisamos especificar os escopos OpenId e Profile na configuração. Por padrão a biblioteca ‘Microsoft.AspNetCore.Authentication.OpenIdConnect’ já inclui na requisição esses escopos.

A opção **SaveTokens é importante para salvar o token** retornado pelo Identity Provider no cookie de autenticação do usuário. Assim é possível obter os tokens a partir do cookie a qualquer momento.

Por fim eu decorei o controller ‘Private’ com o atributo Authorize para que somente usuários autenticados possam acessa-lo.

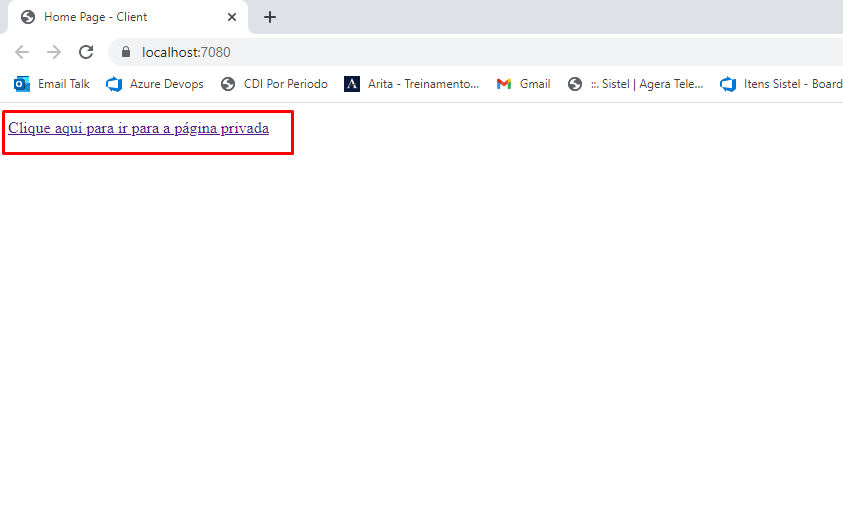


Esse endpoint basicamente exibe informações do usuário autenticado que serão provenientes do Microsoft Entra:



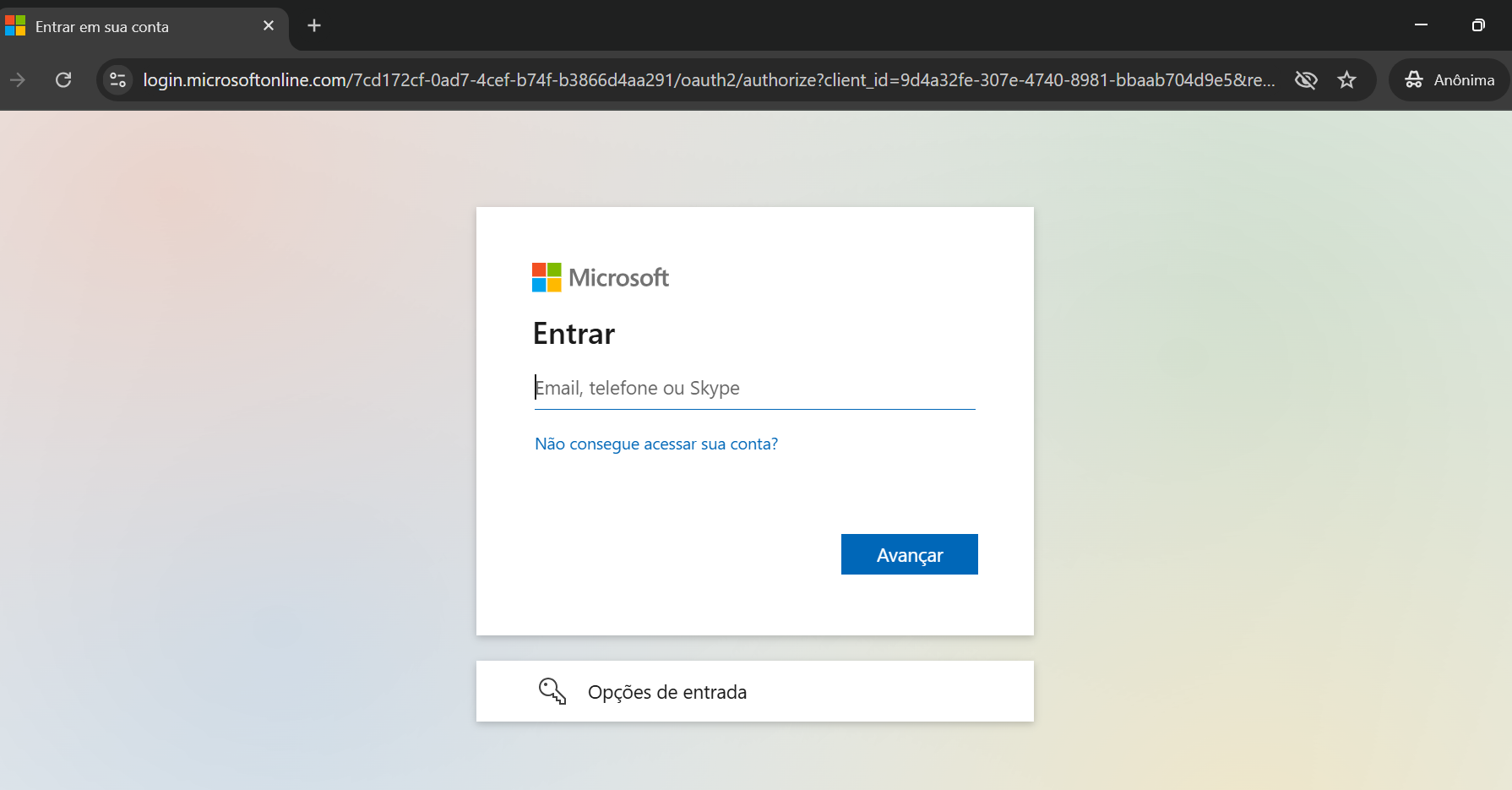
**Rodando o projeto**

Então ao rodar o projeto client, na home temos um link para o endpoint Private que necessita de autenticação [Ativar Devtools]:

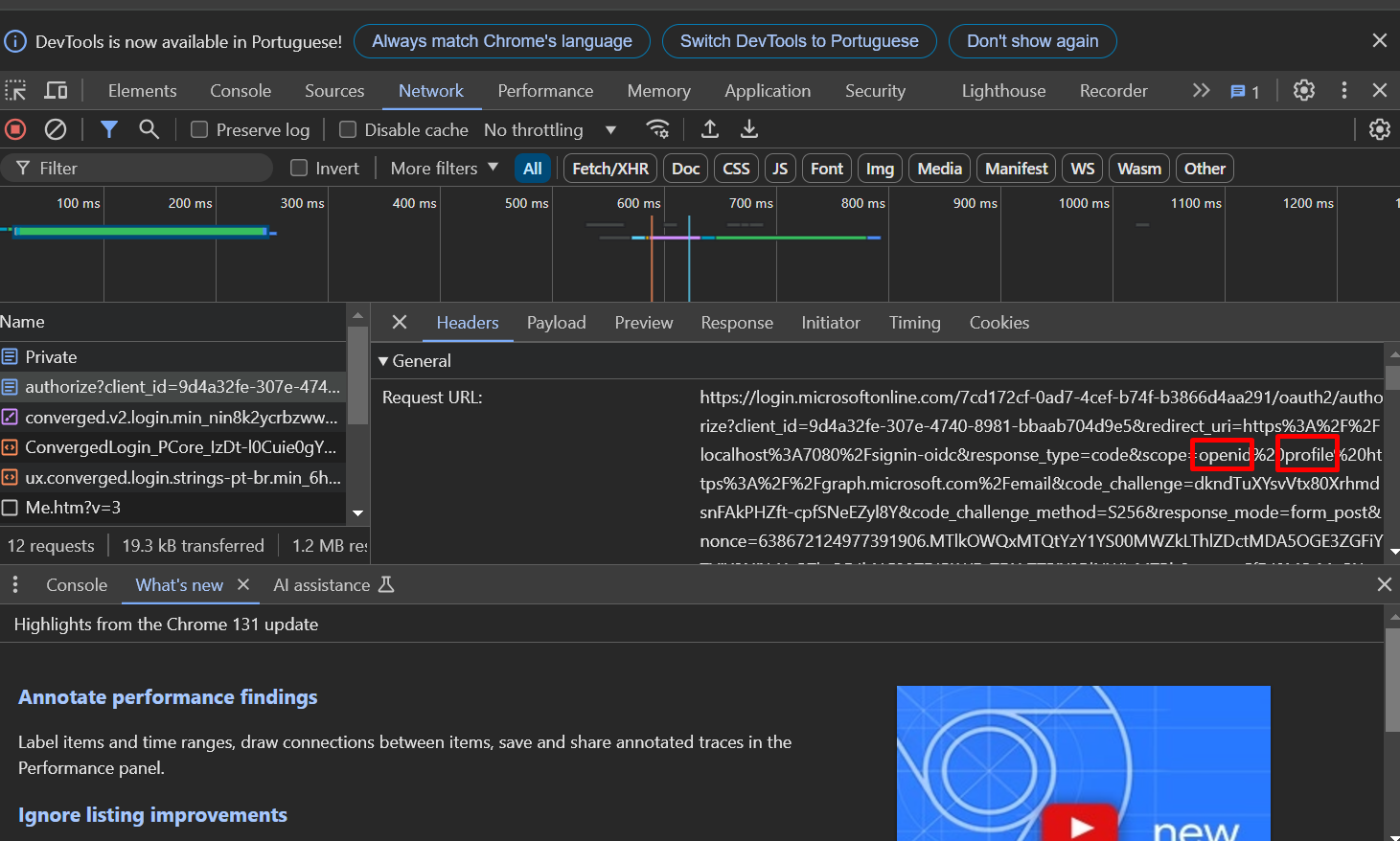


Ao clicar nele eu sou redirecionado automaticamente para o Microsoft Entra.

Diferentemente do exemplo anterior em que eu mesmo via programação redirecionei o usuário, aqui a biblioteca ‘Microsoft.AspNetCore.Authentication.OpenIdConnect’ faz isso automaticamente para mim.



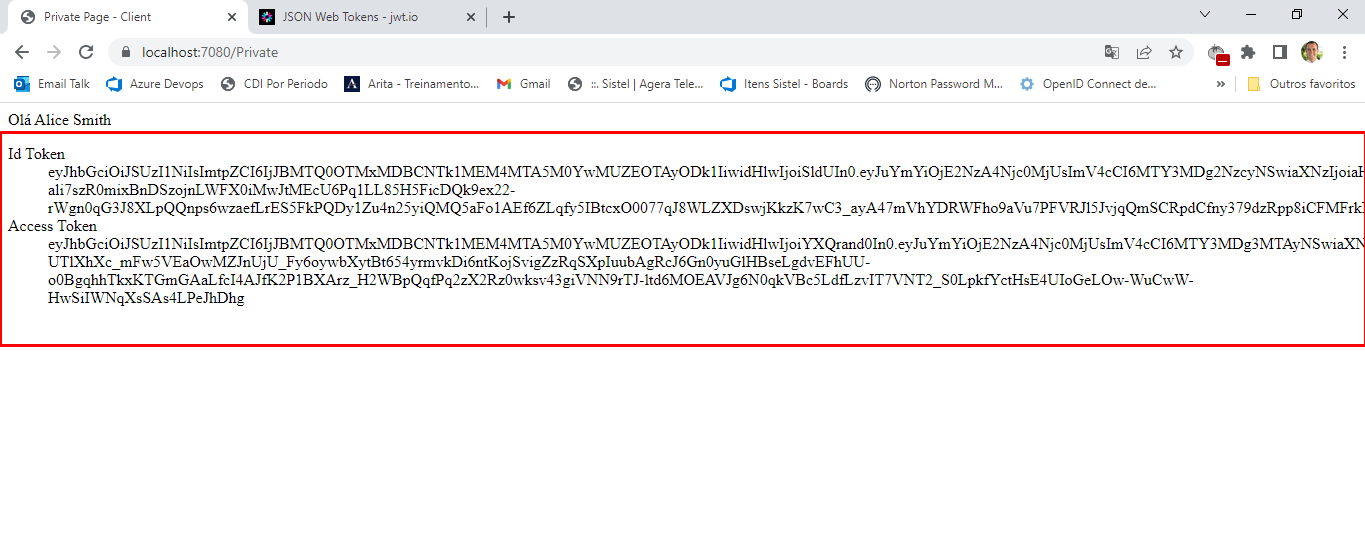
Podemos ver no Devtools na aba Network como foi feito esse redirecionamento. Repare que o pacote do OpenId da Microsoft adicionou automaticamente aos escopos solicitados o openId e o profile.



Depois de logar o usuário é redirecionado de volta para o client e temos acessos as informações básicas de um usuário autenticado como seu nome e email:



Também temos o acesso ao Id Token e o access Token gerado:



**O Id token é um hash no formato JWT que identifica a sessão do usuário logado e o access token é o mesmo token que expliquei no fluxo do Auth2, ele é utilizado para obter recursos do Resource Server**