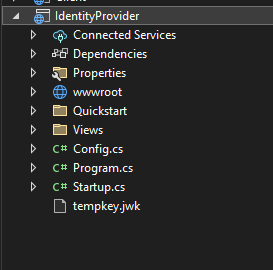
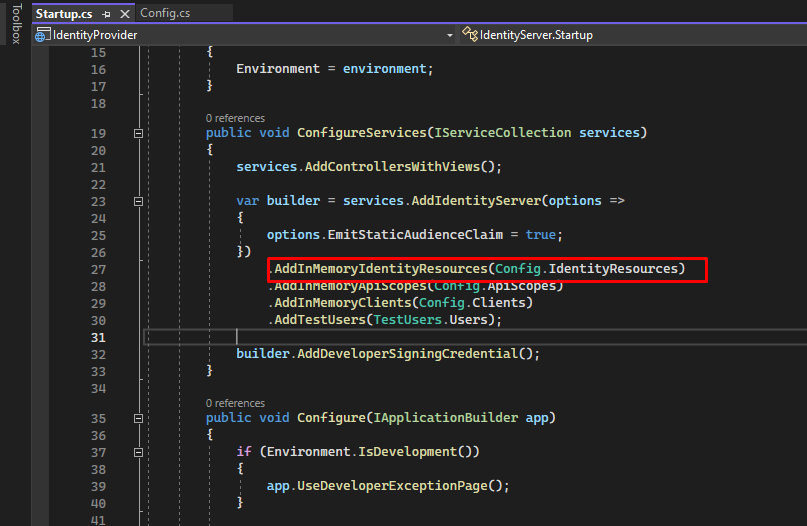
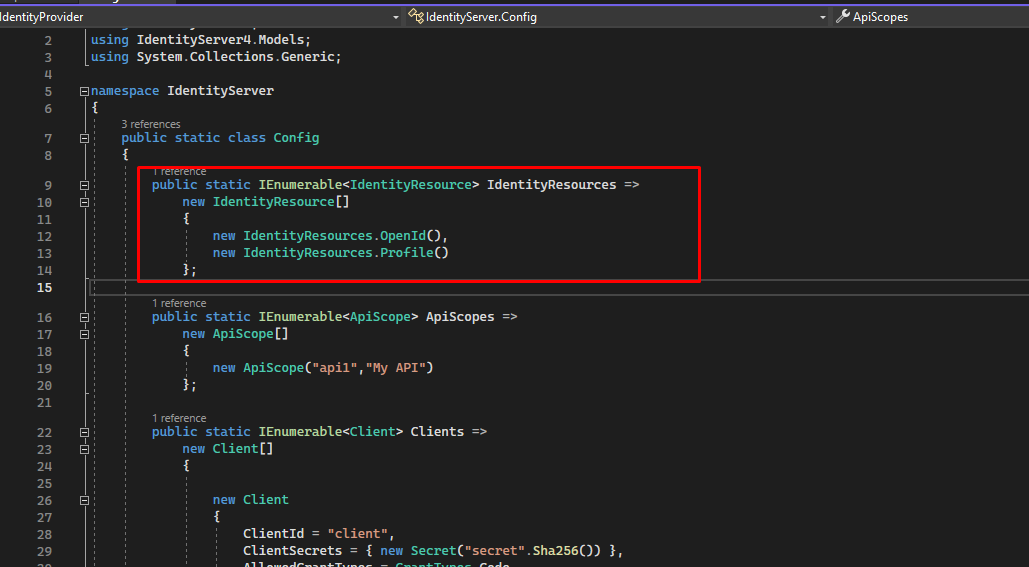
Primeiramente temos o projeto do Identity Provider, que usei exatamente o mesmo template do Identity Server 4 do Authorization Server do último exemplo que mostrei para vocês, só batizei ele de outro nome:



O que o Identity Provider tem de diferente em relação ao Authorization Server?

Primeiramente aqui na configuração do Identity Server 4 temos um novo cara aqui chamado identity Resources:

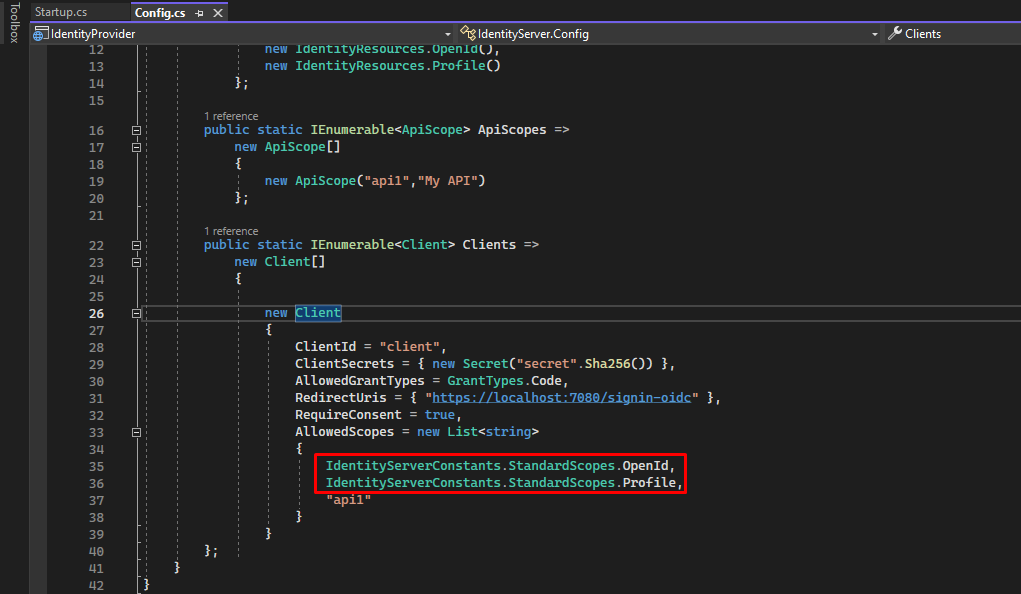




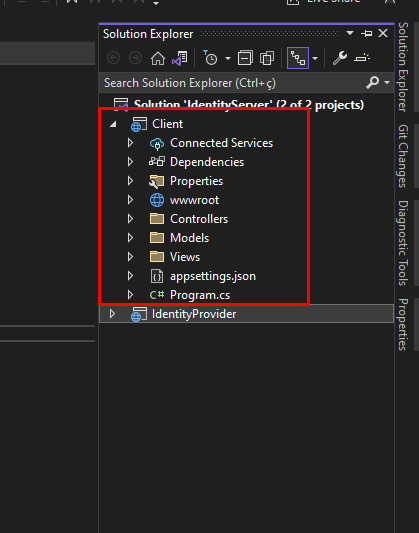
Eles são escopos voltados para Identificação de usuários. No caso adicionamos os recursos OpenID e Profile.

Depois para finalizar temos aqui o registro do client que irá usar nosso Identity Provider para autenticar.

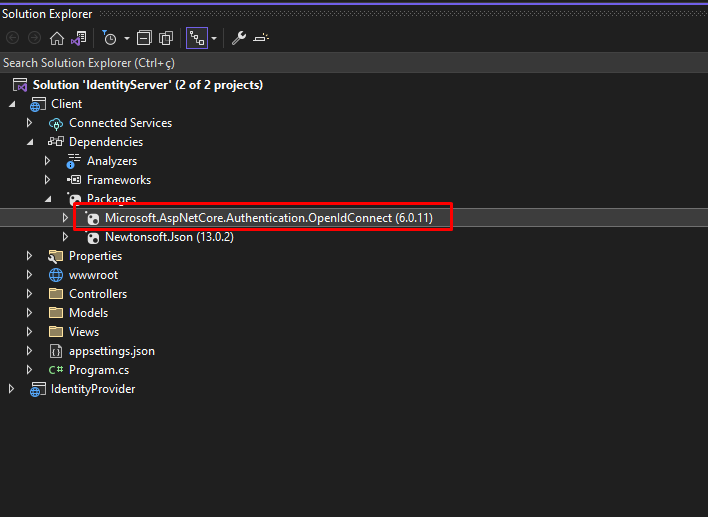
Repare que tem que ser **permitido** a esse client os **escopos do OpenID**:



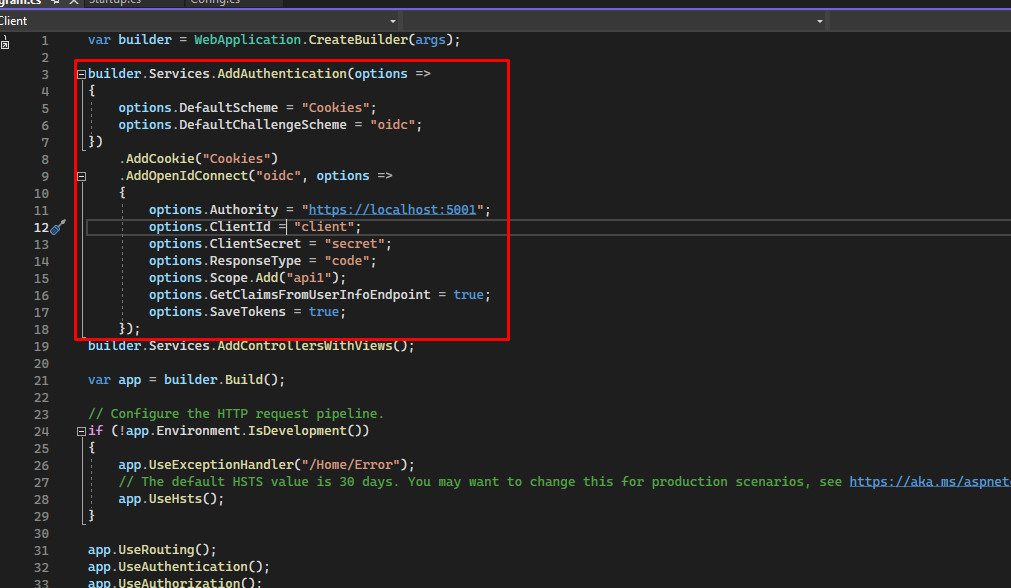
Agora vamos para o projeto do client que fará uso da autenticação do Identity Provider:



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.



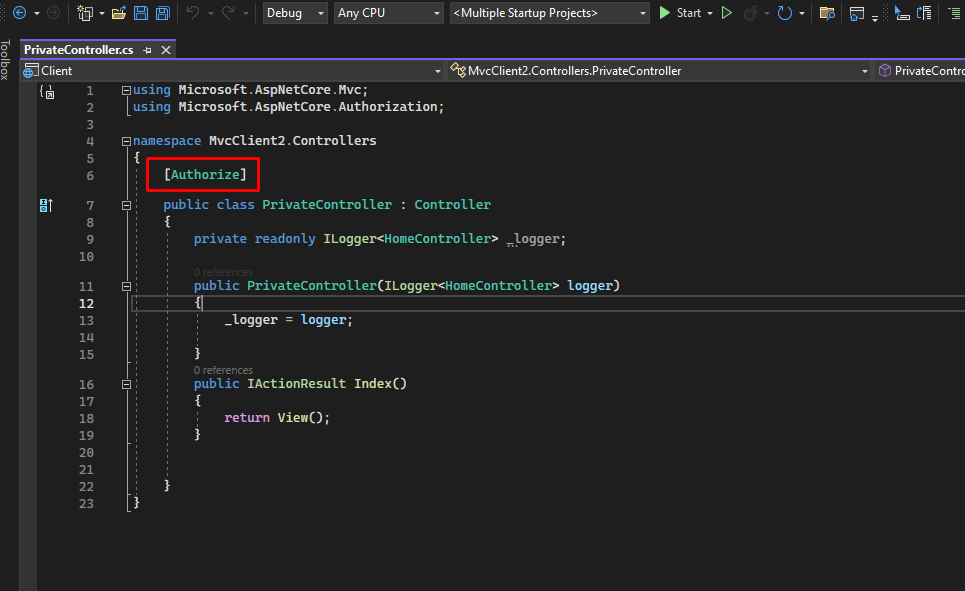
Através do método AddOpeIdConnect configuramos a autenticação via nosso Identity Provider que está rodando na porta 5001.

Repare que basicamente temos que passar as mesmas informações para o Identity Provider que passamos no exemplo anterior: ClientId, ClientSecret, ResponseType, etc.

Repare também que não precisamos especificar os escopos OpenId e Profile no Scope. Por padrão a biblioteca ‘Microsoft.AspNetCore.Authentication.OpenIdConnect’ já inclui na requisição esses escopos.

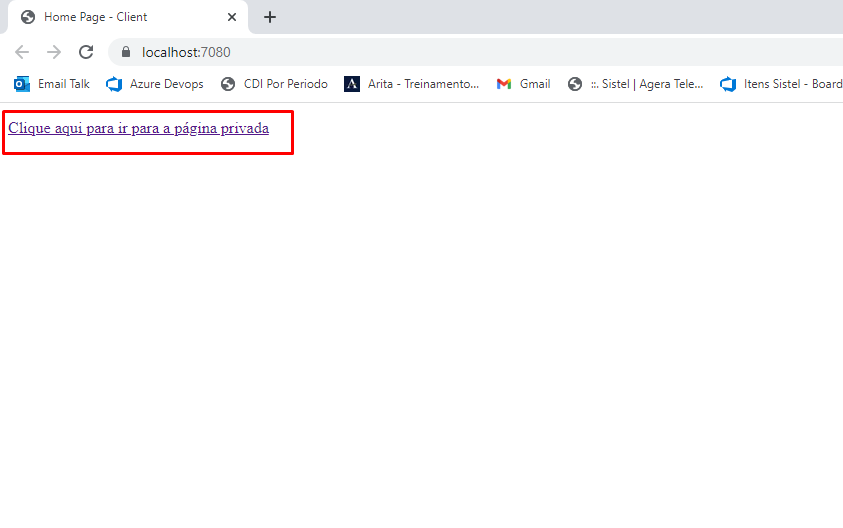
A opção **SaveTokens é importante para salvar o token** retornado pelo Identity Provider no cookie de autenticação do usuário. Assim é possível obter os tokens a partir do cookie a qualquer momento.

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



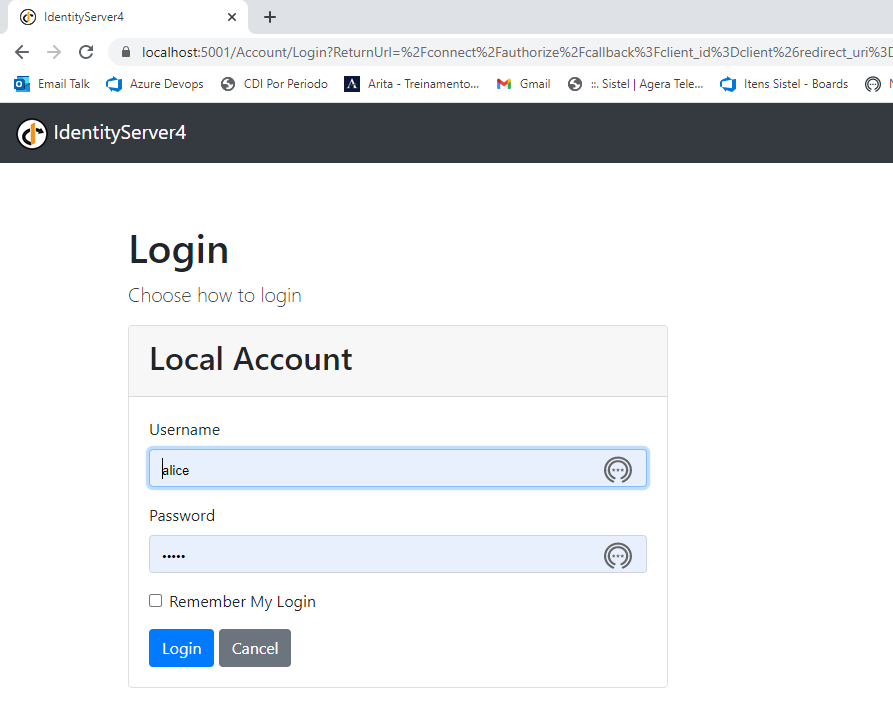
**Rodando o projeto**

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

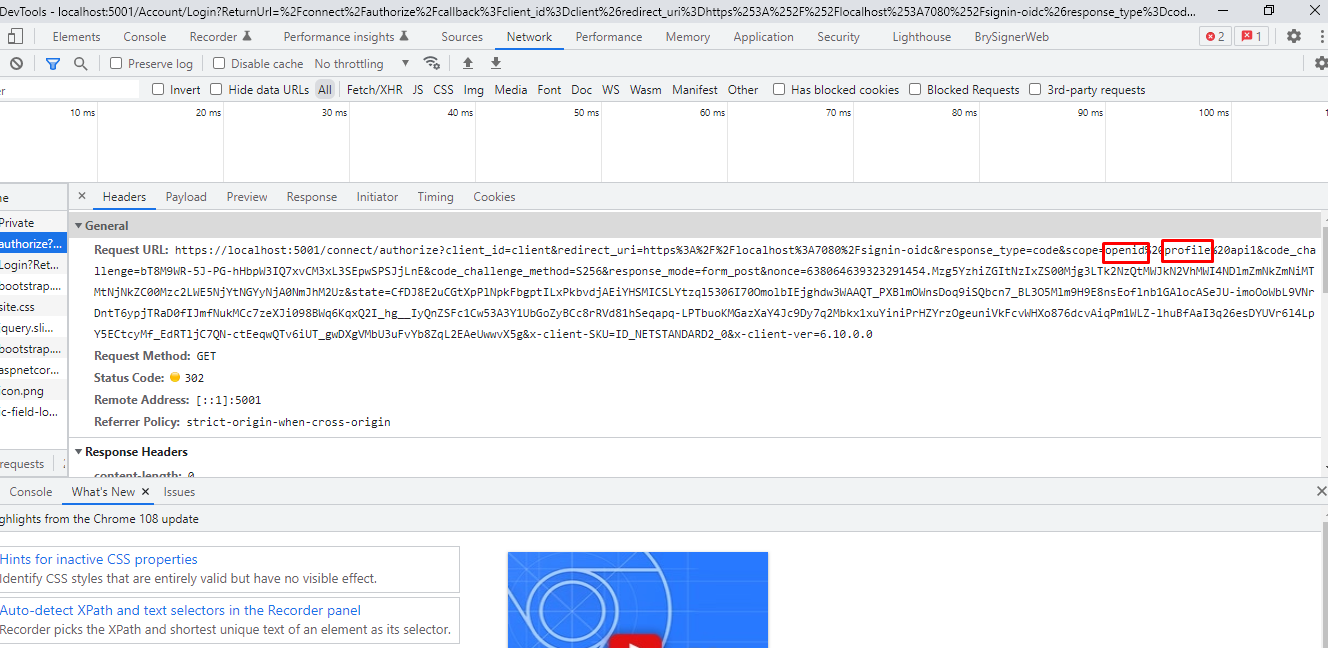


Ao clicar nele eu sou redirecionado automaticamente para o Identity Provider.

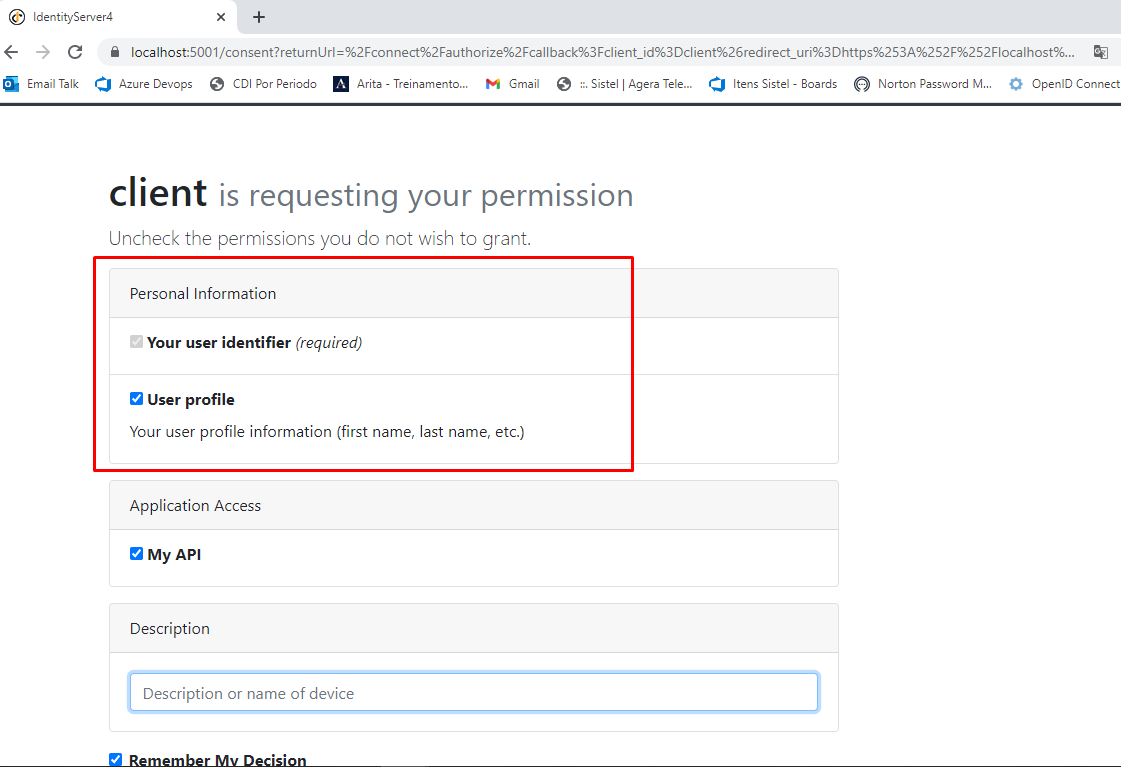
Diferentemente do exemplo anterior em que eu mesmo via programação redirecionei o usuário, aqui a biblioteca ‘Microsoft.AspNetCore.Authentication.OpenIdConnect’ faz isso automaticamente para mim.



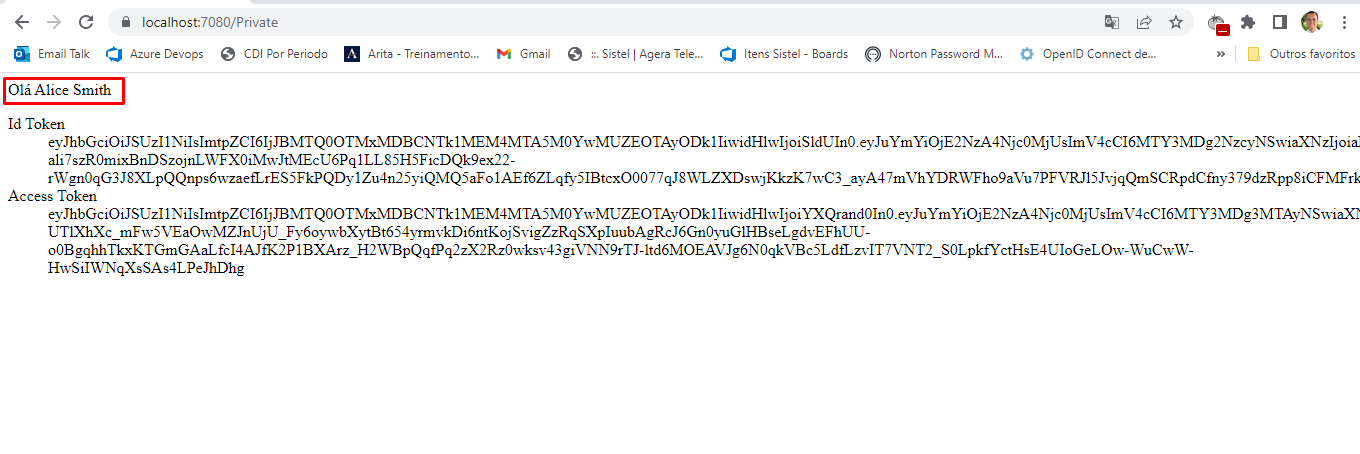
Podemos ver no Devtools na aba Network como foi feito esse redirecionamento. Repare que o pacote do OpenId da Microsoft adicionou automaticamente aos escopos solicitados o openId e o profile.



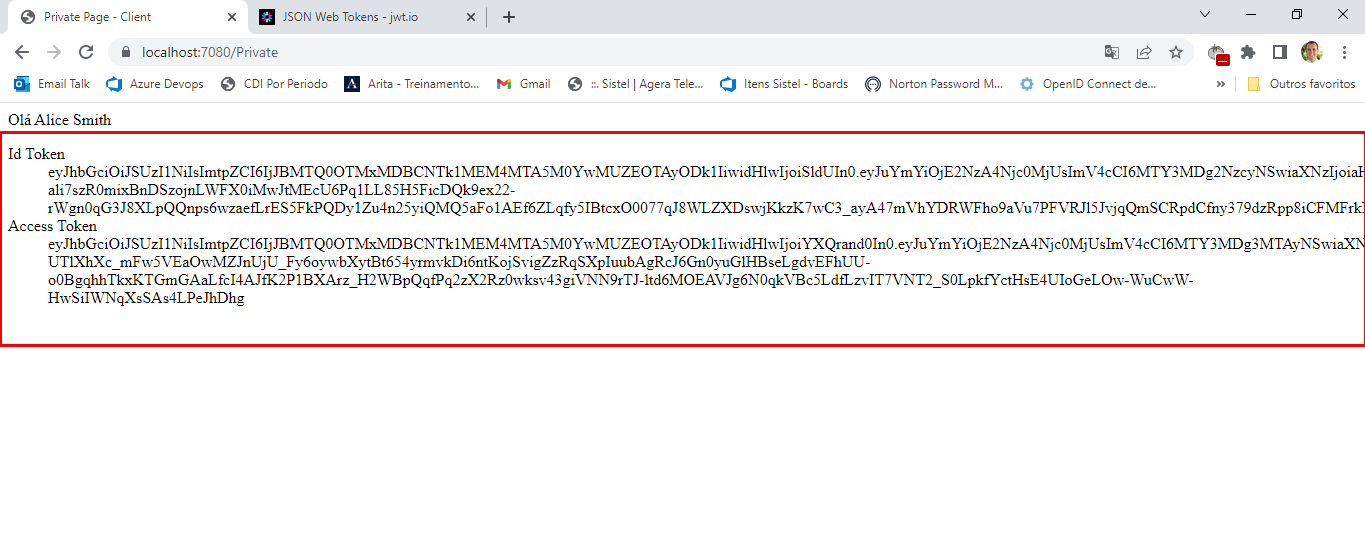
Depois de logar, na tela de consentimento você precisa aceitar ou não que o client acesse os escopos que ele pediu. Repare que não tem como demarcar a primeira opção, porque esse é o recurso do openId em si e é o **escopo mínimo** para realizar o **login terceirizado**:



Depois somos redirecionados de volta para o client e temos acessos as informações básicas de um usuário autenticado como seu nome:



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**