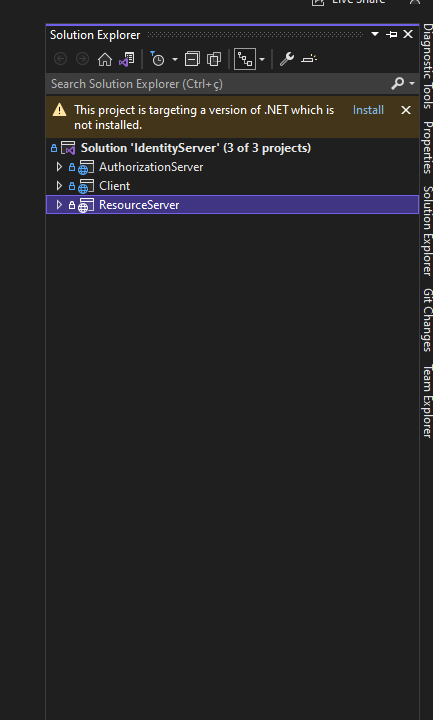
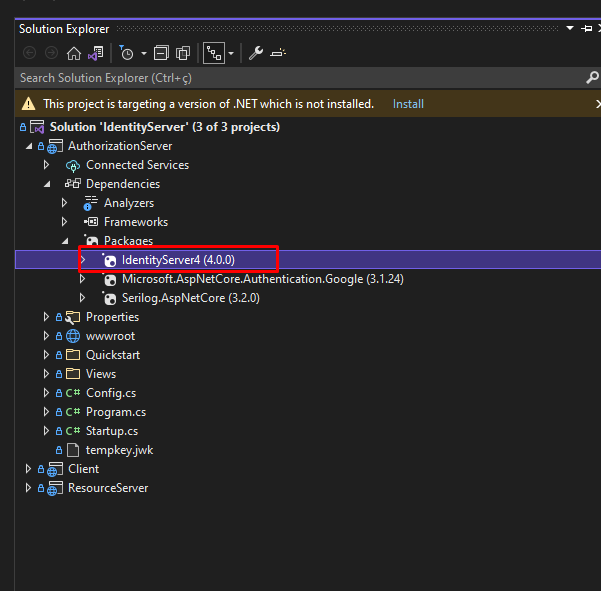
Temos aqui 3 projetos em uma solução. Cada um será personagem do OAuth2 no fluxo authorization code

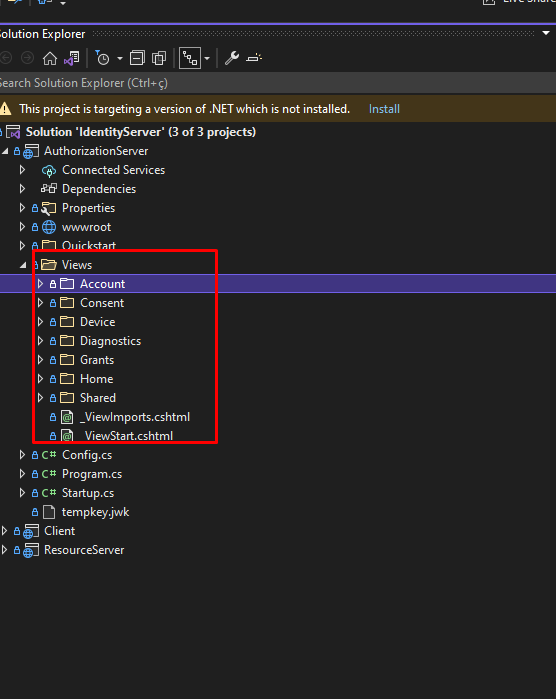


O primeiro que vamos falar é o Authorization Server. É o authorization server **que conhece os resourcers owners** (donos dos recursos) e **libera o acesso ou não aos recursos** conforme o consentimento do owner.

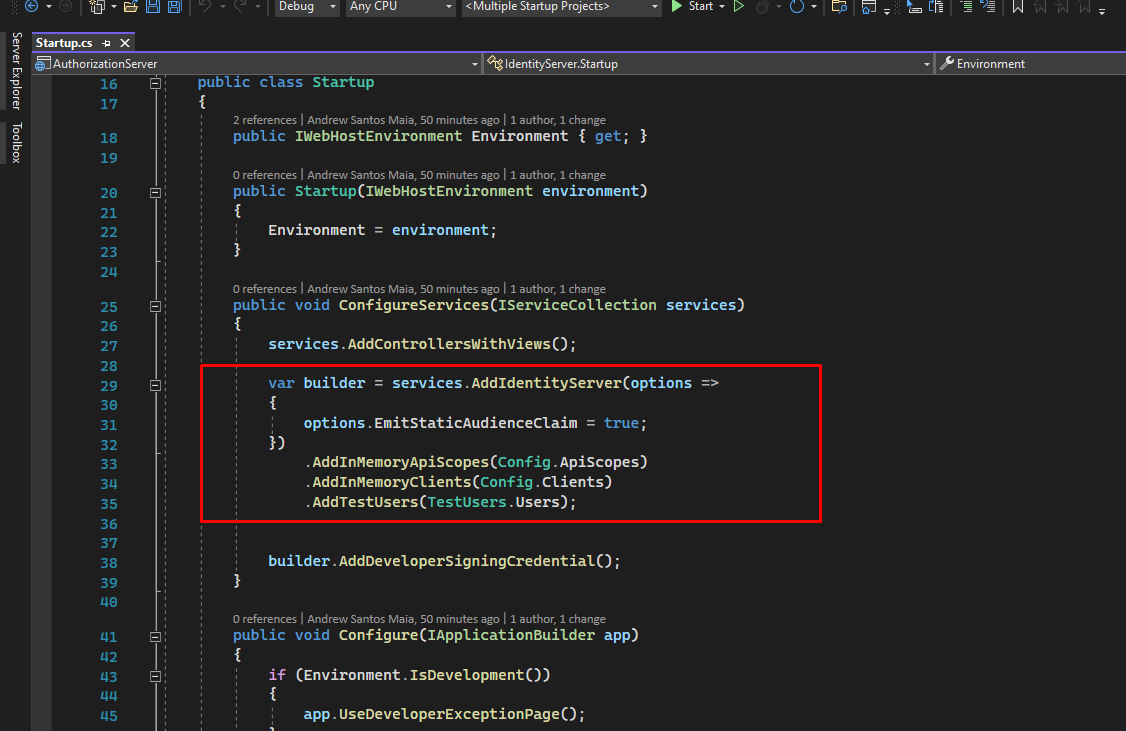
Aqui eu usei uma biblioteca chamada Identity Server 4, que já implementa o protocolo do OAuth2. Como eu havia dito não precisa desenvolver do zero uma aplicação que implemente o Oauth 2.



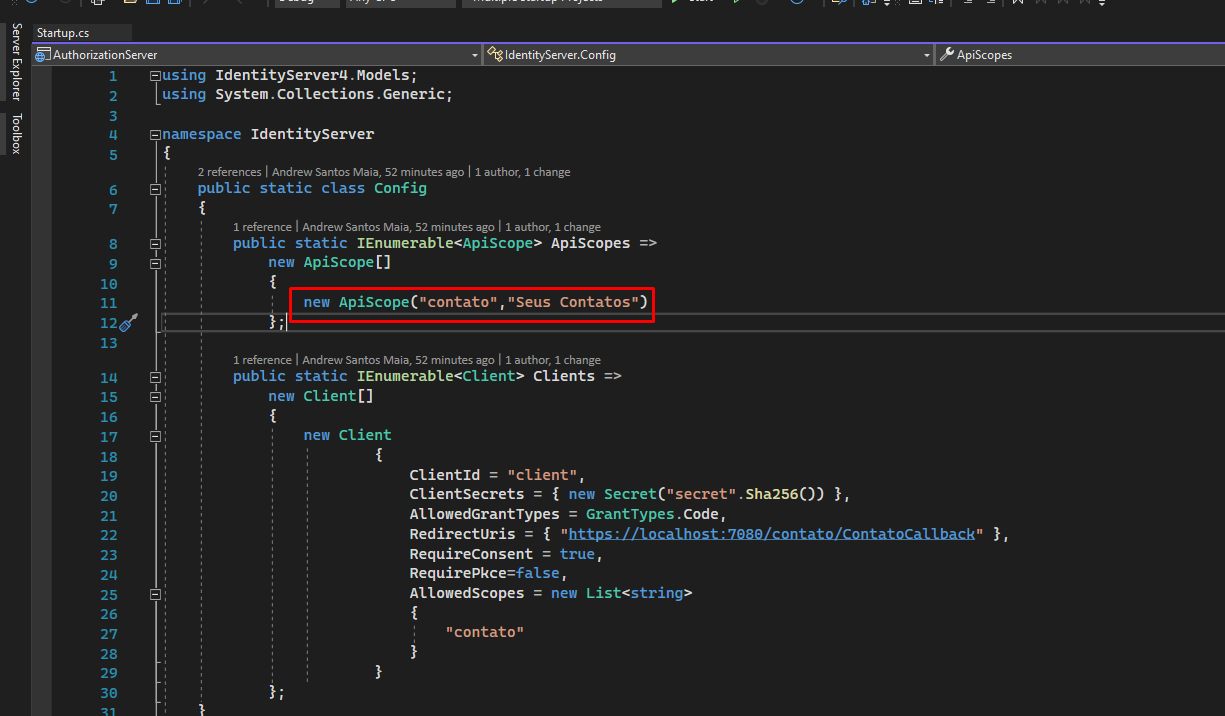
Também usei um template padrão do Identity Server 4 que já vem com as páginas básicas de um Servidor de Autorização: página de login, pagina de consentimento, etc.



Vamos agora para a parte mais importante que é a configuração do serviço do Identity Server 4, que fazemos no startup:

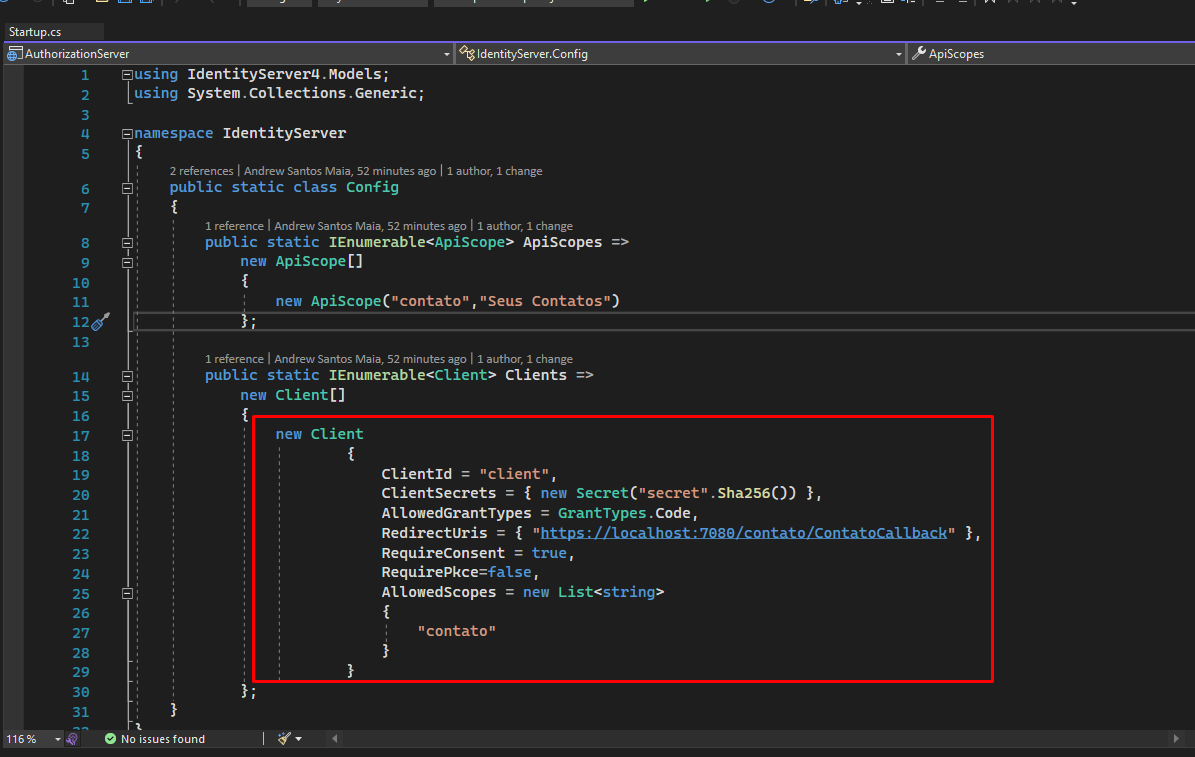


Aqui definimos quais são os escopos que o authorization server vai controlar o acesso:

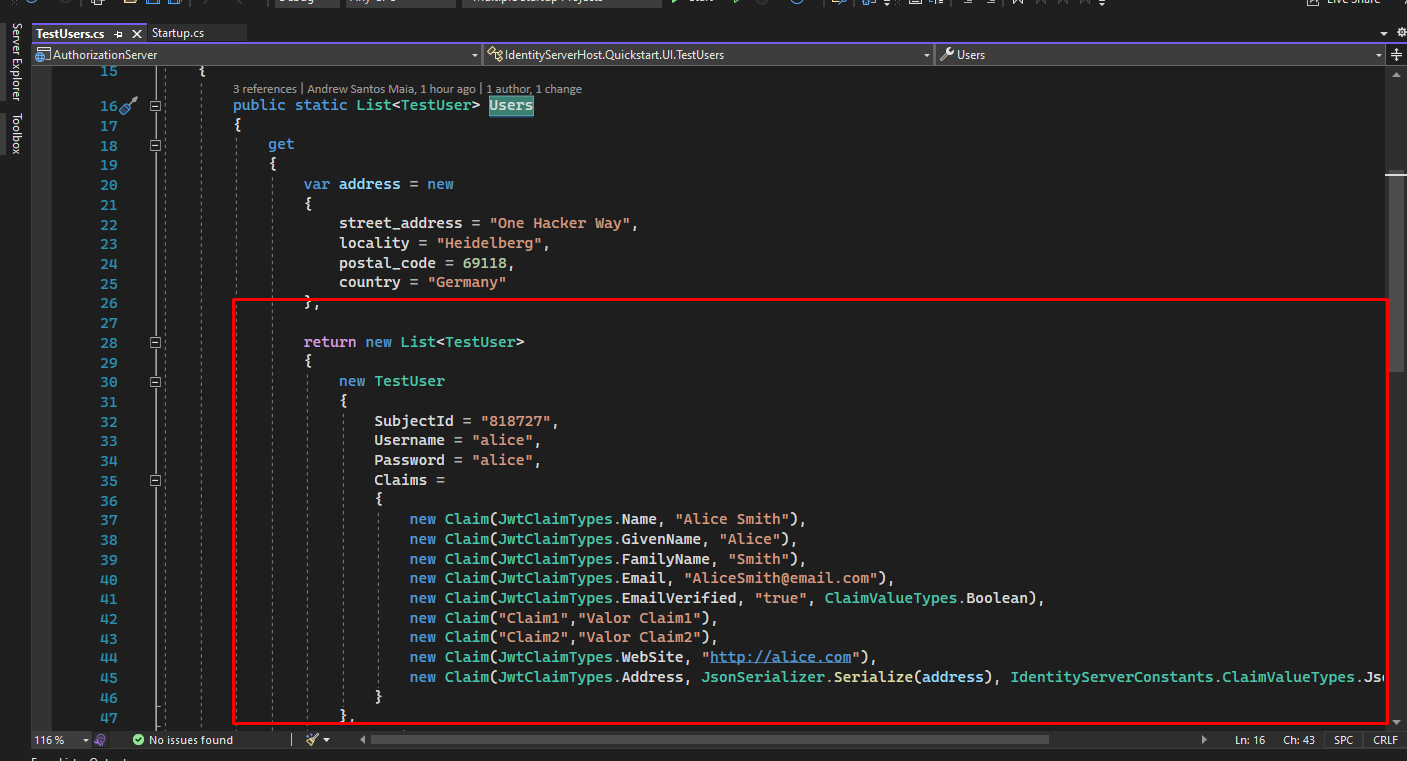


Escopo é a representação de um recurso, é ele que aparece na tela de consentimento para o usuário liberar ou não o acesso.

O segundo cara que precisamos configurar são os clients. Lembra que precisa registar o client no authorization server antes de começar a solicitar acesso a recursos? Na maioria dos casos há sempre uma página de cadastro para os client’s mas aqui no caso eu não tenho página e estou deixando fixo o client. Aqui você precisa atribuir o clientId, o client secret, a página de redirecionamento e os escopos que esse client pode pedir acesso, no caso somente um escopo: contato.

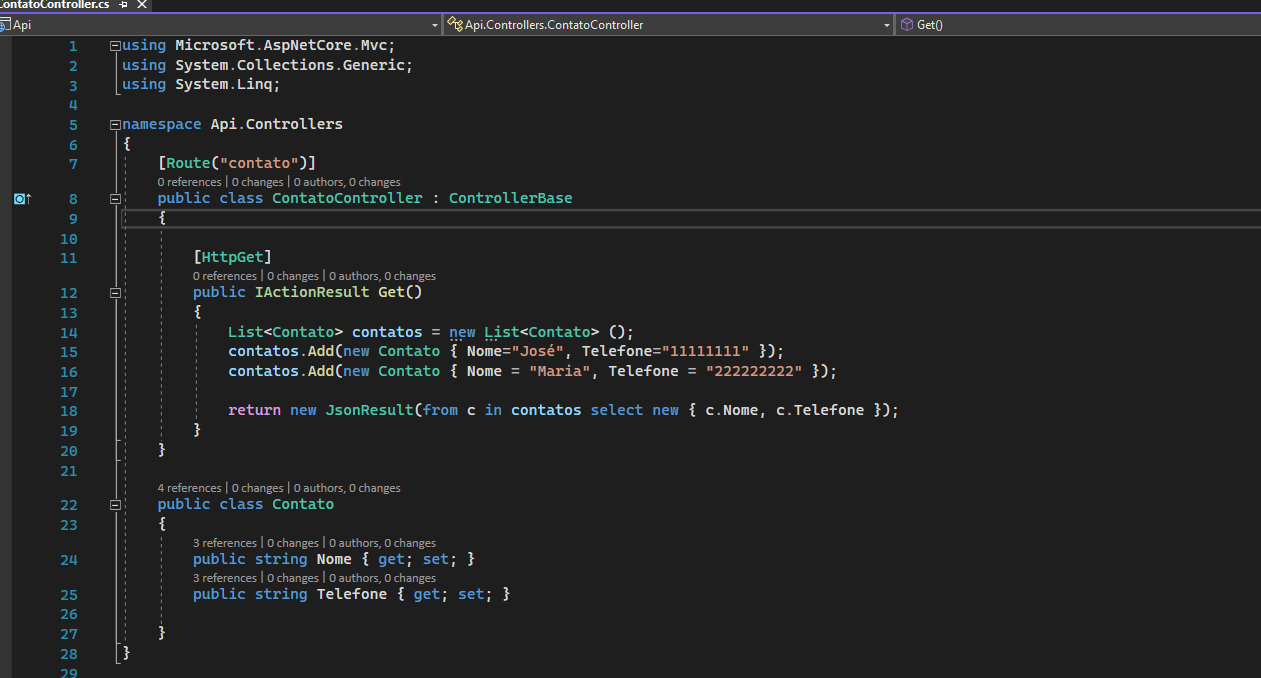


Por fim os usuários que o authorization server vai lidar. Novamente no caso estou criando usuários fixos e fakes.



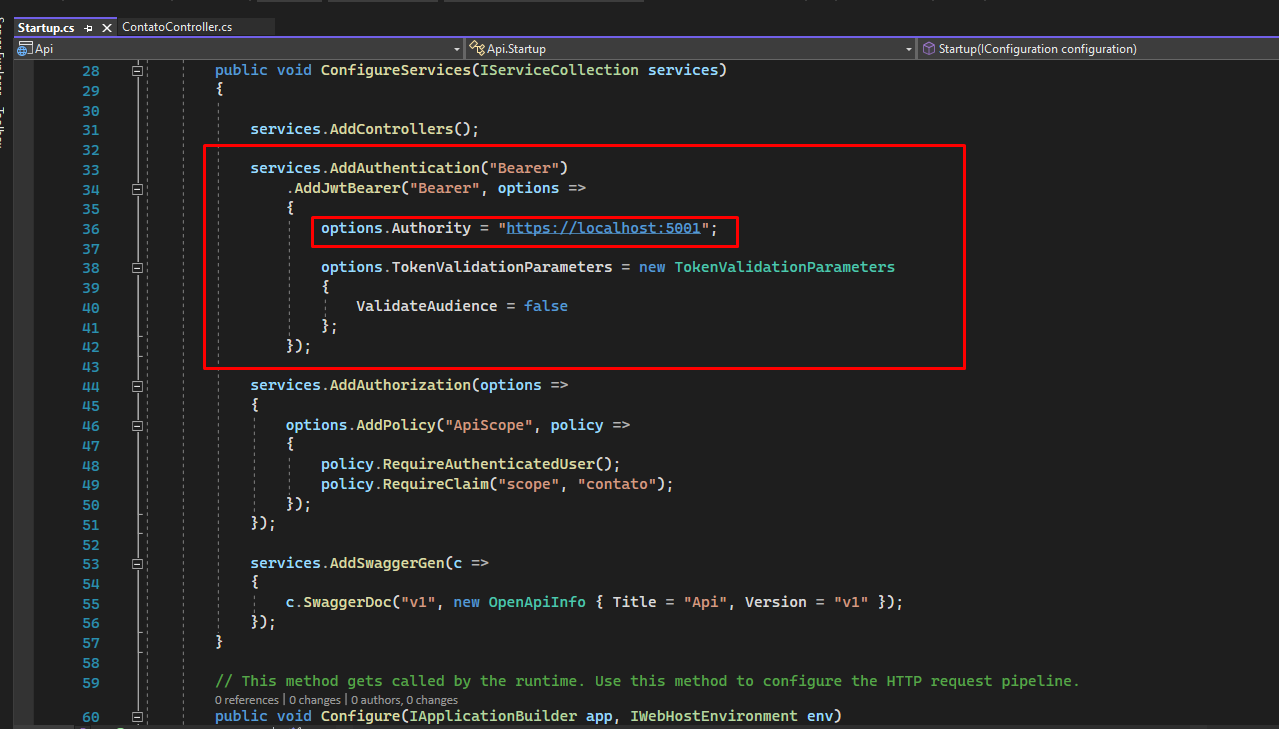
O próximo personagem é o Resource Server. Aqui estamos representando-o por um projeto de WebAPI.

Aqui temos uma API chamada contato que retorna contatos. São dados fakes e fixos que eu coloquei só para a apresentação. Ela retorna dois nomes e seus respectivos telefones



Essa API está protegida, ou seja, para acessar ela precisa de um token de acesso e o claim contato.

Essa proteção é feita por este trecho de código:



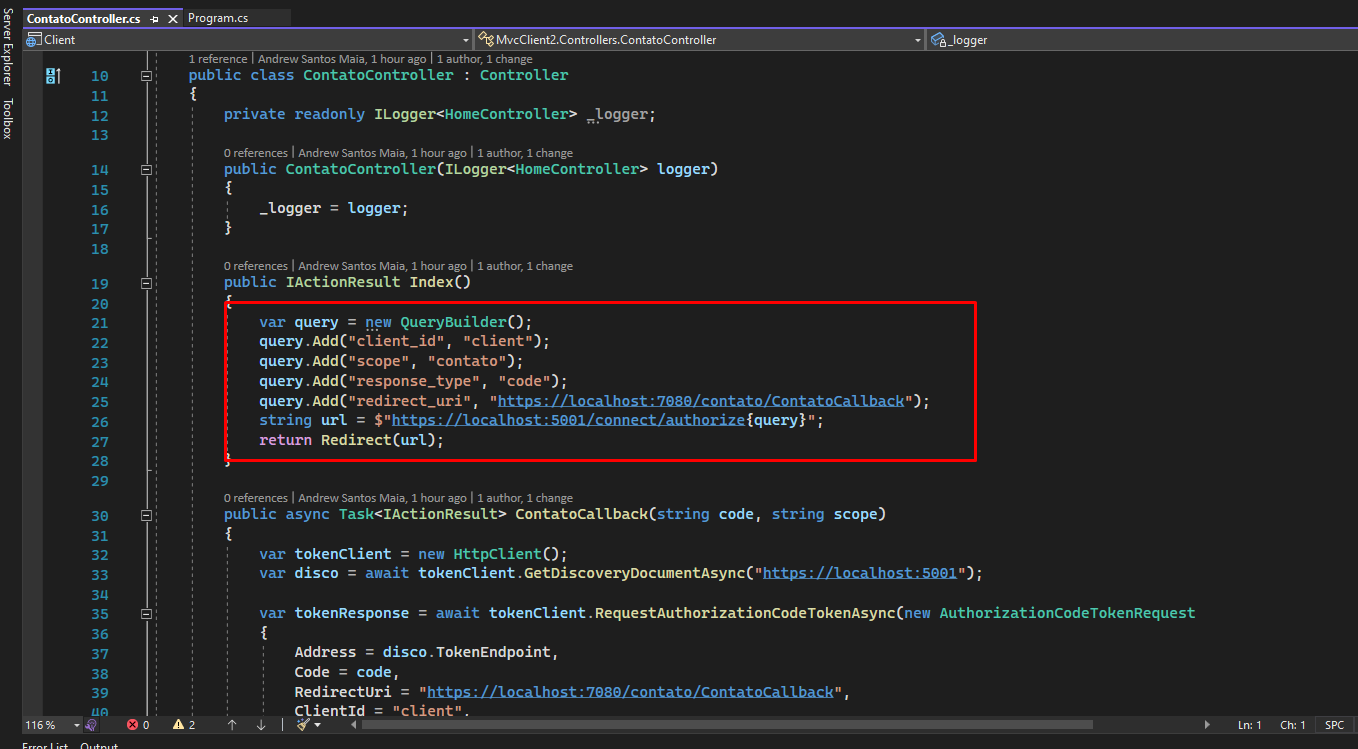
O principal aqui é a propriedade Authority que estamos apontando para o nosso Authorization Server que está rodando na porta 5001. Ou seja, é como se estivéssemos dizendo que quem é responsável por determinar quem pode acessar a API de contatos é o Authorization Server.

Por fim o terceiro projeto é o Client. É um site que quer acessar os meus contatos.

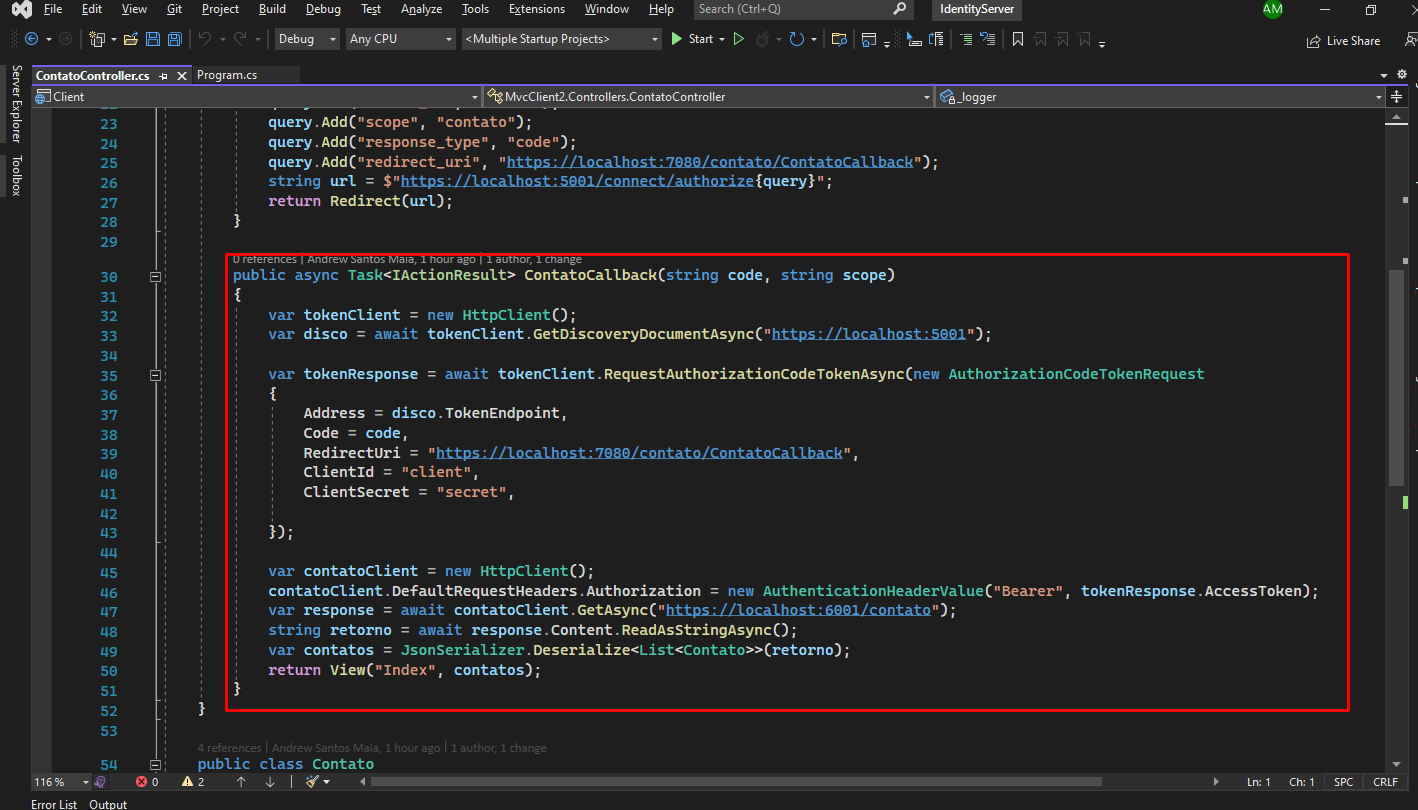
Aqui tem uma controller chamado Contato que quando acesso-la ela vai me redirecionar para o authorization server solicitando o meu consentimento para acessar os meus contatos.

Este é um exemplo usando o fluxo authorization code, que vou explicar com mais detalhes agora:

Aqui nós redirecionamento o usuário para o endpoint de autorização do authorization server. Aqui precisamos passar o clientId do client, o escopo que queremos, no caso contato, response\_type que é o fluxo de autorização authorization code e a URL de redirecionamento.

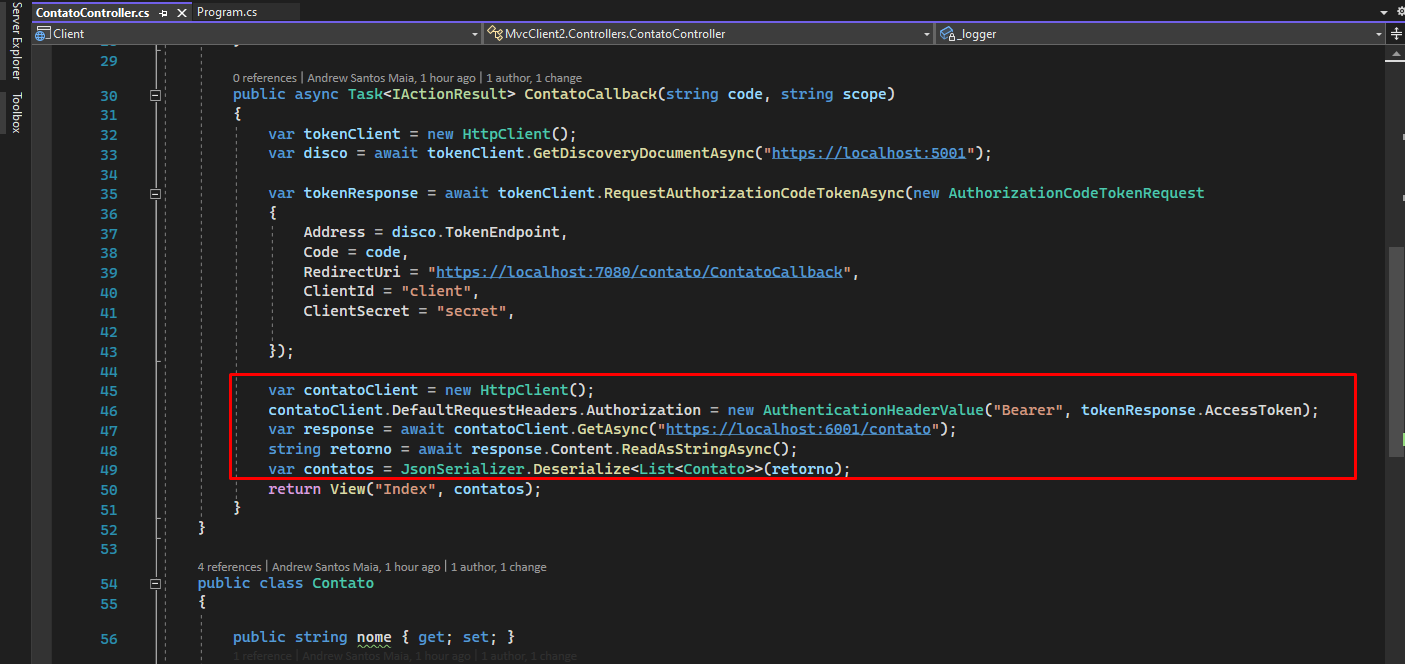
A url de redirecionamento é utilizada pelo o authorization server para retornar para o client depois que o usuário autoriza o acesso ao recurso.

Essa aqui já é o endpoint de retorno depois que o authorization server redireciona o usuário de volta para o client. Repare que o authorization server retorna um code na query string. Esse code faz parte do fluxo authorization code, é através dele que iremos obter o access token para assim conseguir acessar a API de contatos:



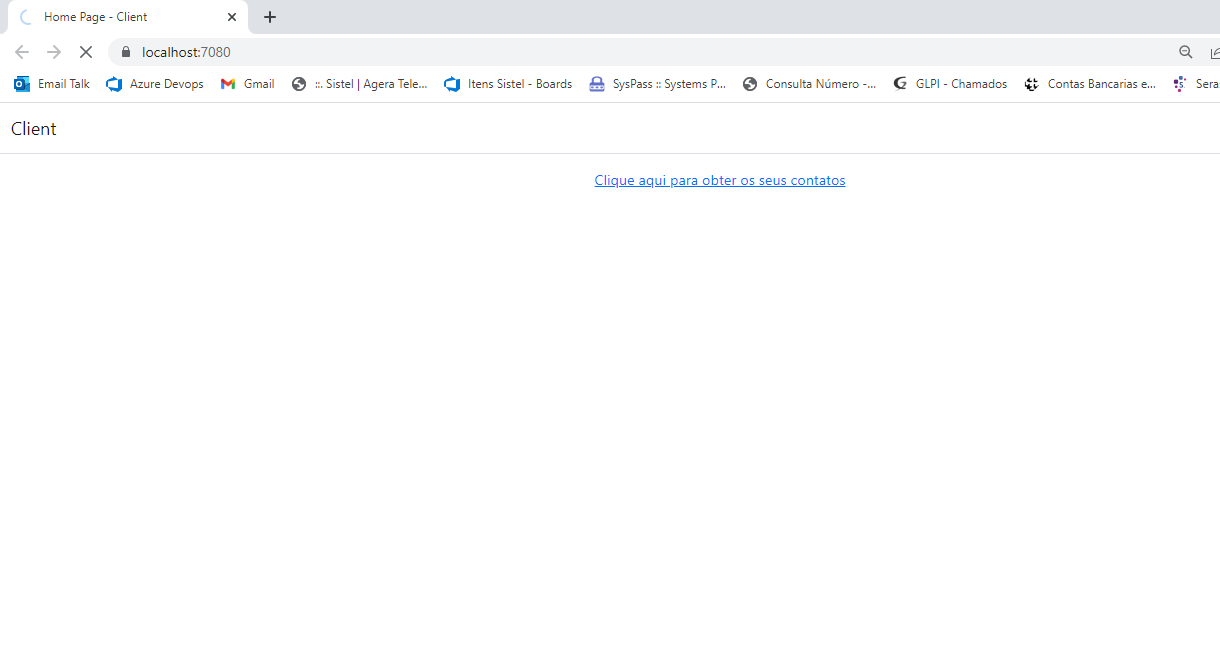
Depois que obtemos o code, vamos de novo no authorization server só que sem a participação do usuário diretamente, de server para server, o que nós chamamos no mundo OAuth2 de **backchannel**.

Aqui fazemos uma requisição para o authorization server passando o code, clientId e client Secret e obtemos o token de acesso

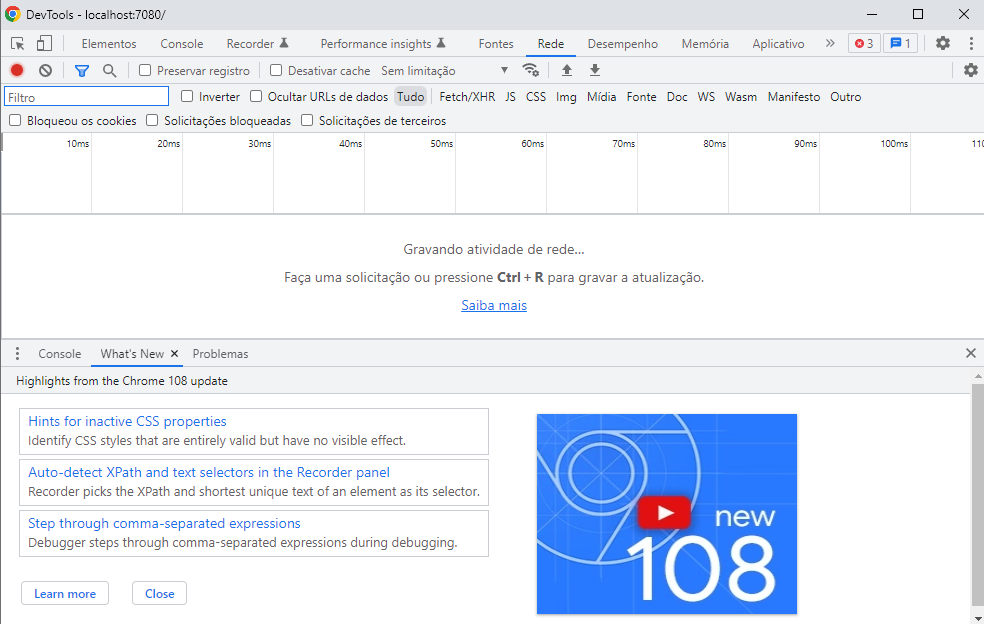
Com esse token podemos utilizar na chamada a API do resource server para obter os contatos do usuário:

Vamos ver na prática.

Acessamos o site do Client que está rodando na porta 7080:



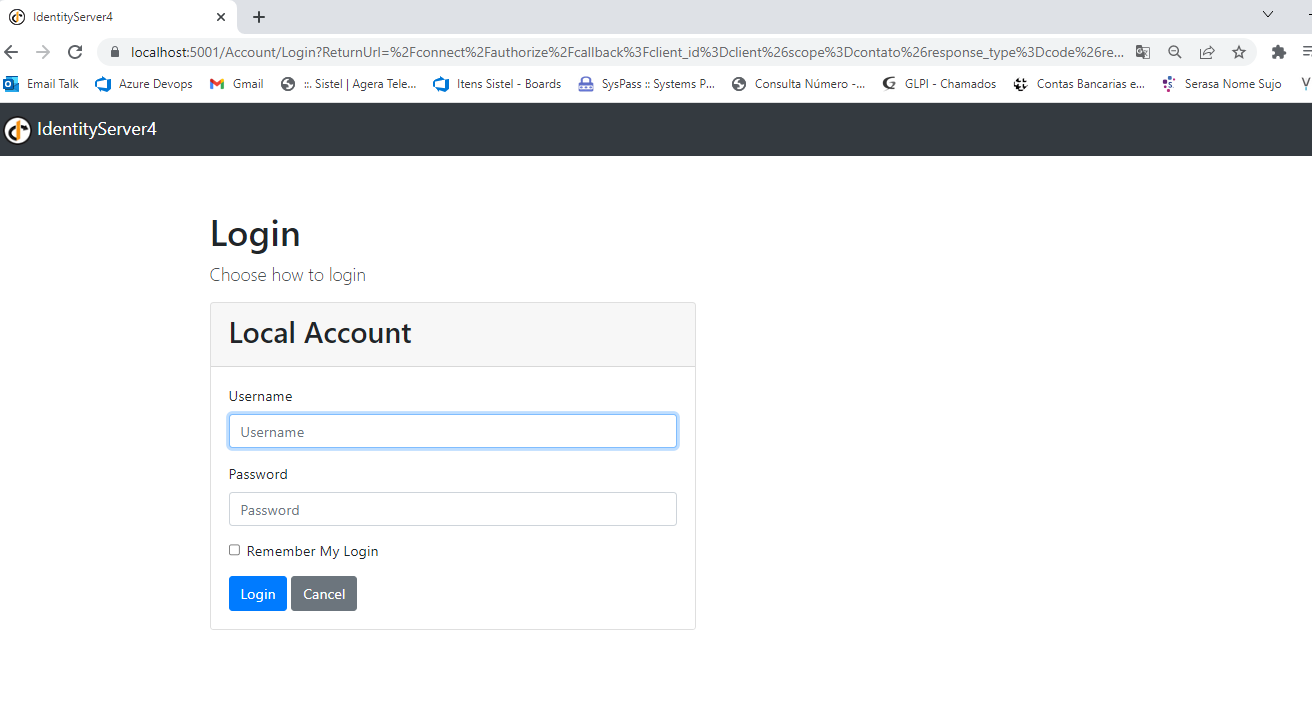
Eu vou deixar o DevTools do chrome aberto para vermos o que ocorre durante a comunicação entre o authorization server e o client:



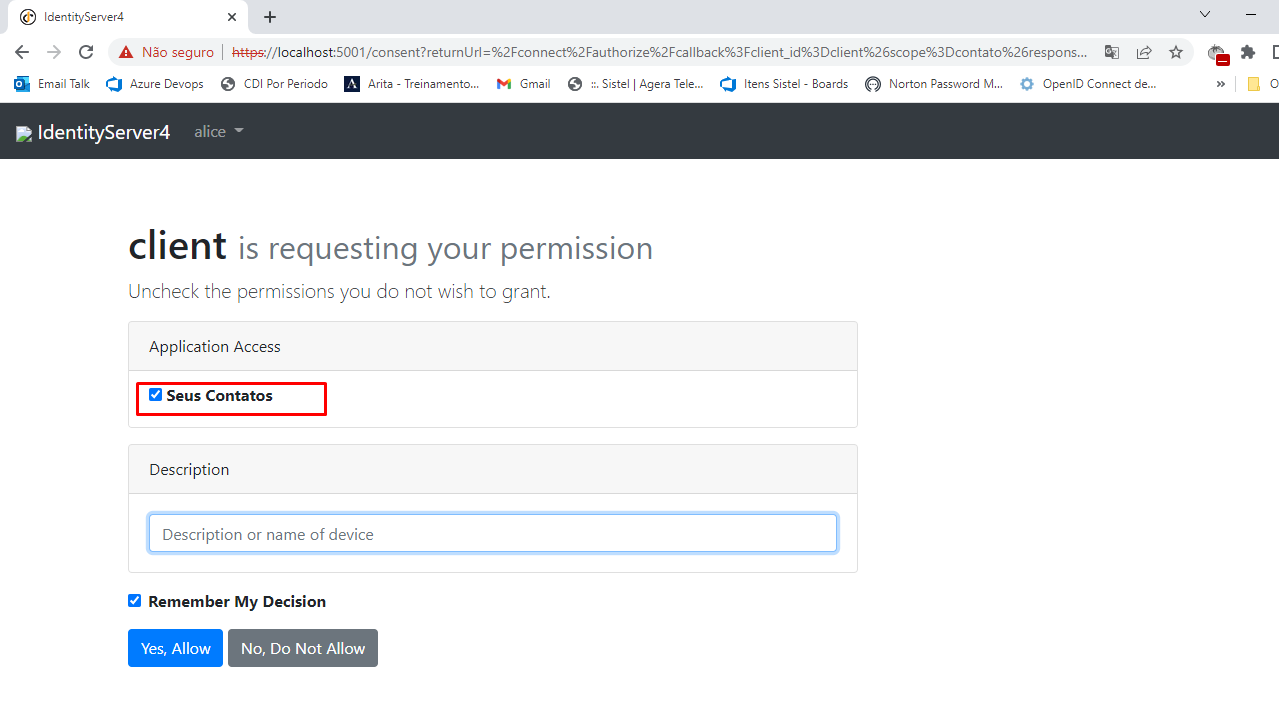
[Ir debugando o código]

Ao clicar no link para obter os contatos, somos redirecionados para o authorization server que está rodando na porta 5001.

Primeiro o usuário é solicitado a fazer o login:



Depois vem a tela de consentimento onde usuário irá autorizar ou não o acesso ao escopo solicitado pelo client. Repare que o que aparece aqui para o resource owner é o nome **amigável** do escopo que definimos no authorization server:

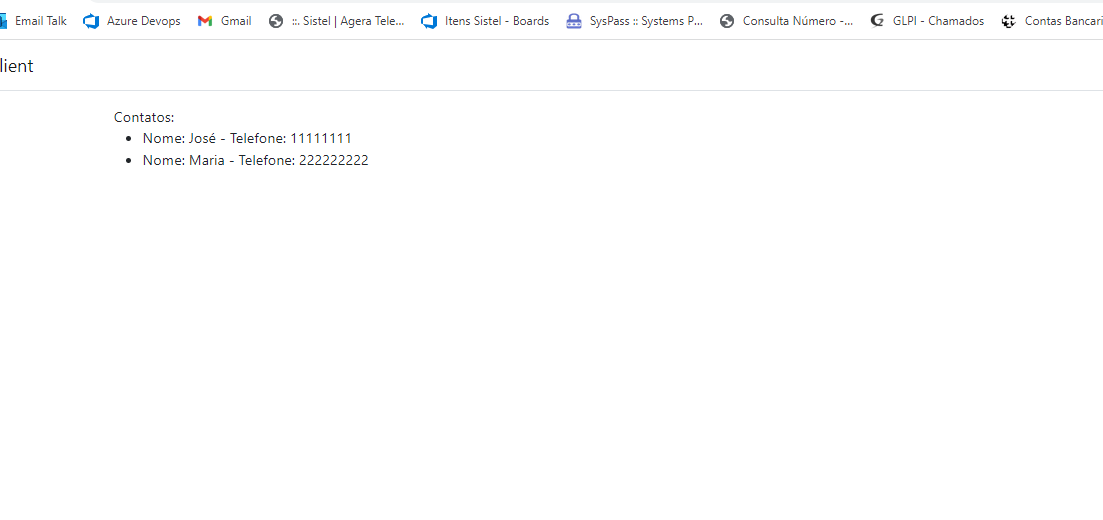


[Ir debugando o código]

Com o consentimento liberado o authorization server retorna o code para o client.

A partir do code ele consegue fazer uma nova requisição para o authorization server para obter o token de acesso via **backchannel** .

Depois ele faz uma requisição para a API de contatos do resource server que está rodando na porta 6001 com o token para obter os contatos do resource owner:



No Devtools podemos observar o que ocorreu:

Primeiro o usuário é redirecionado para o endpoint authorize do authorization server. Depois do consentimento, o authorization server redireciona o usuário para o endpoint redirect uri do client onde o client recebe o code. Depois não podemos observar mais nada aqui no chrome porque todas as requisições ocorrem no backchannel , ou seja, de server para server.

