Identity Server

1- Finalidade

Com o crescimento de criações de API centralizar a autenticação desses serviços em um apenas um único projeto pode ser vantajoso

2- Breve descrição do funcionamento do projeto

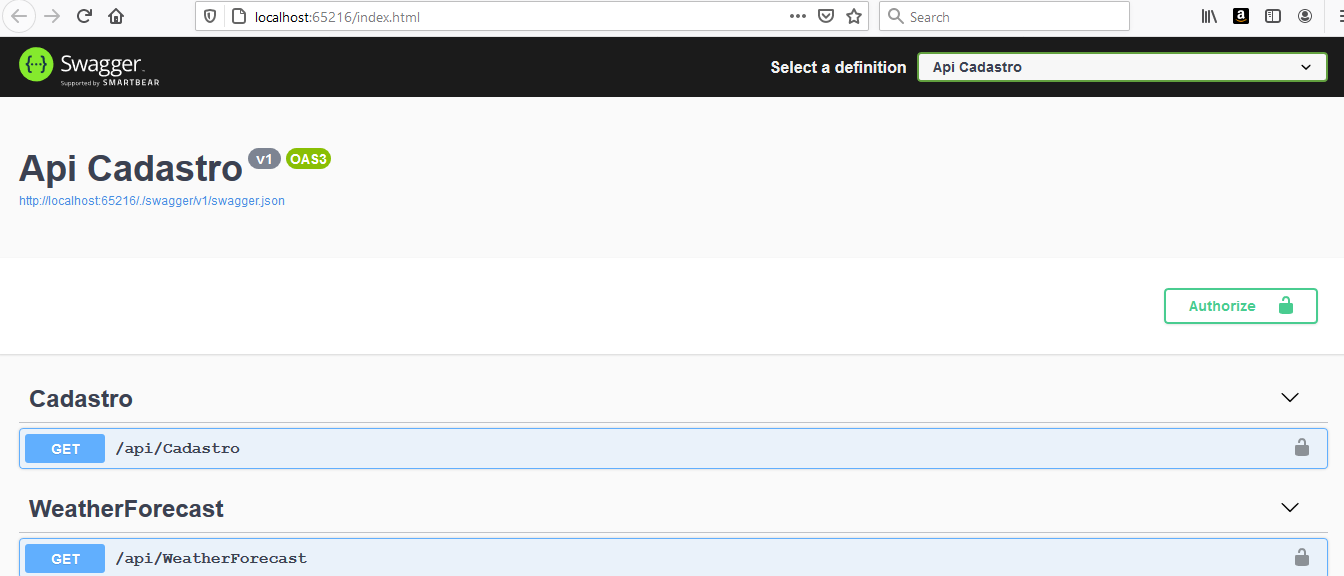
Foi criada uma solução que contem três projetos de API, uma que ficará responsável pela autenticação dos demais e dele próprio, como é um ambiente de teste foi feito dessa, a autenticação poderia ser por usuário e senha definida no banco, assim cada usuária teria o seu token, nesse mesmo documento essa questão será abordada.

3 – ApiCadastro

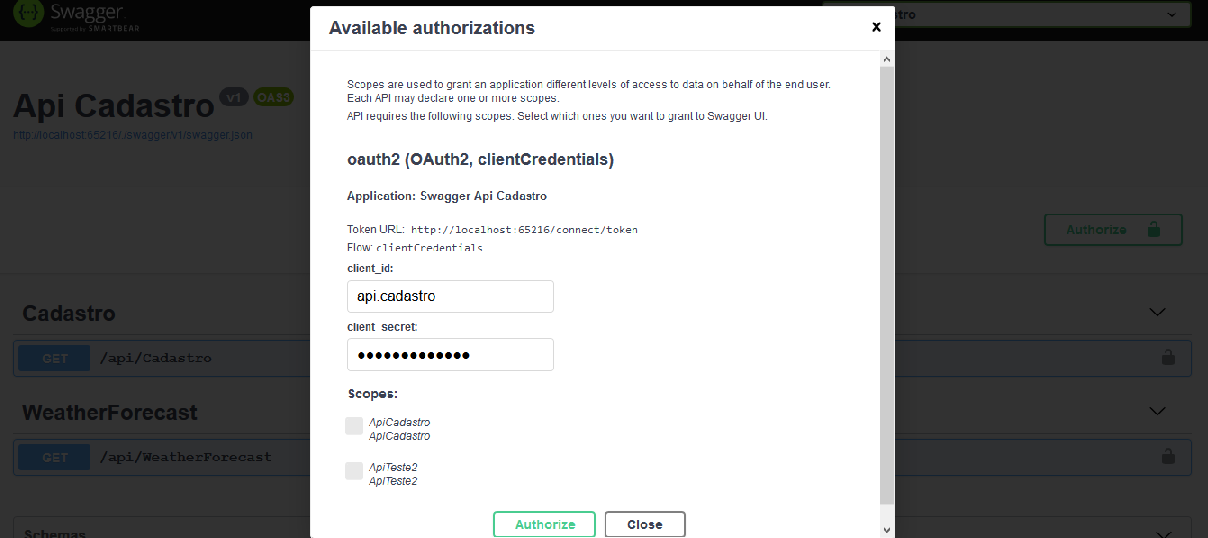
Esse projeto está configurada a autenticação do seu próprio token e os tokens dos projeto ApiTeste1 e ApiTeste2 , logo abaixo serão descritos os pontos mais relevantes que esse projeto possui.

3.1 – Tela

Abaixo a tela da api de cadastro com o swagger e asscociada com o authorize no qual irá autorizar os endpoints

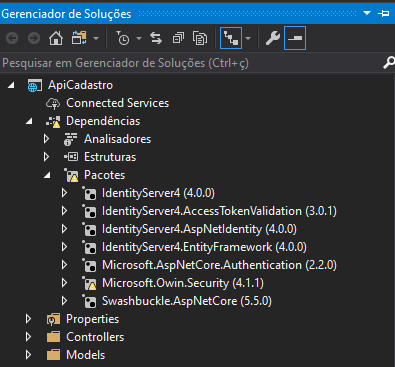


Abaixo a tela que permite autorizar os endpoints via OAUTH2.



3.2 – Packages Instalado

Abaixo seguem os packages instalados



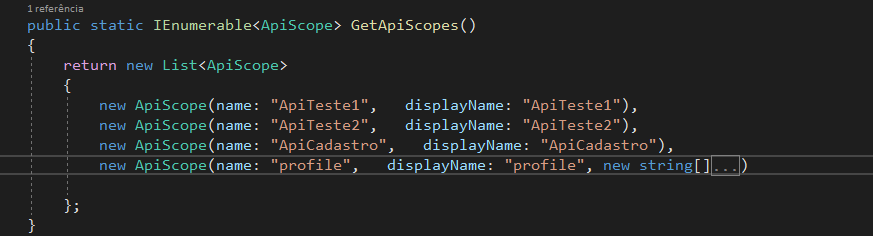
3.3 – Configuração Client IdentityServer

3.3.1 – Classe Config

Para a definição de autenticação do tipo OAUTH será necessário definir escopos, para o OpenID seria necessário apenas os resources, para entender melhor essas duas formas de autenticação a OAUTH ela já possui um clienteid e uma secret definida independente do usuário, ou seja, para um ambiente de desenvolvimento seria bom para testar a aplicação, em ambiente de produção, dependendo das restrições da api, como o caso de uma intranet, também poderia ser utilizado, para Open ID ele pode pedir o usuário e senha cadastrada no banco de dados e assim o token a ser criado será para cada usuário.

Scope (para OAUTH) e ApiResource

Nessas configurações são configurados o recursos da api, como por exemplo os Claims, com finalidade que o projeto que irá consumir a Api, trazer os códigos, permissões diretamente do autenticador que o IdentityServer

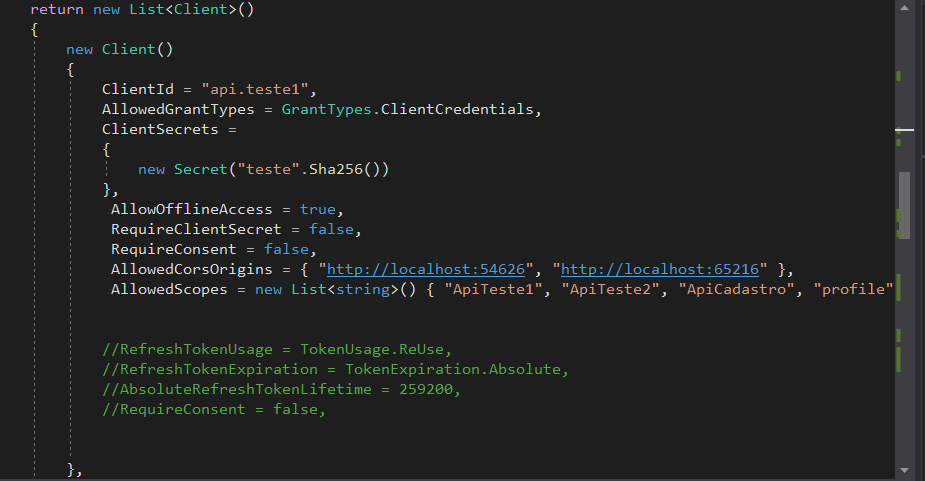


Para o Resource é a mesma ideia



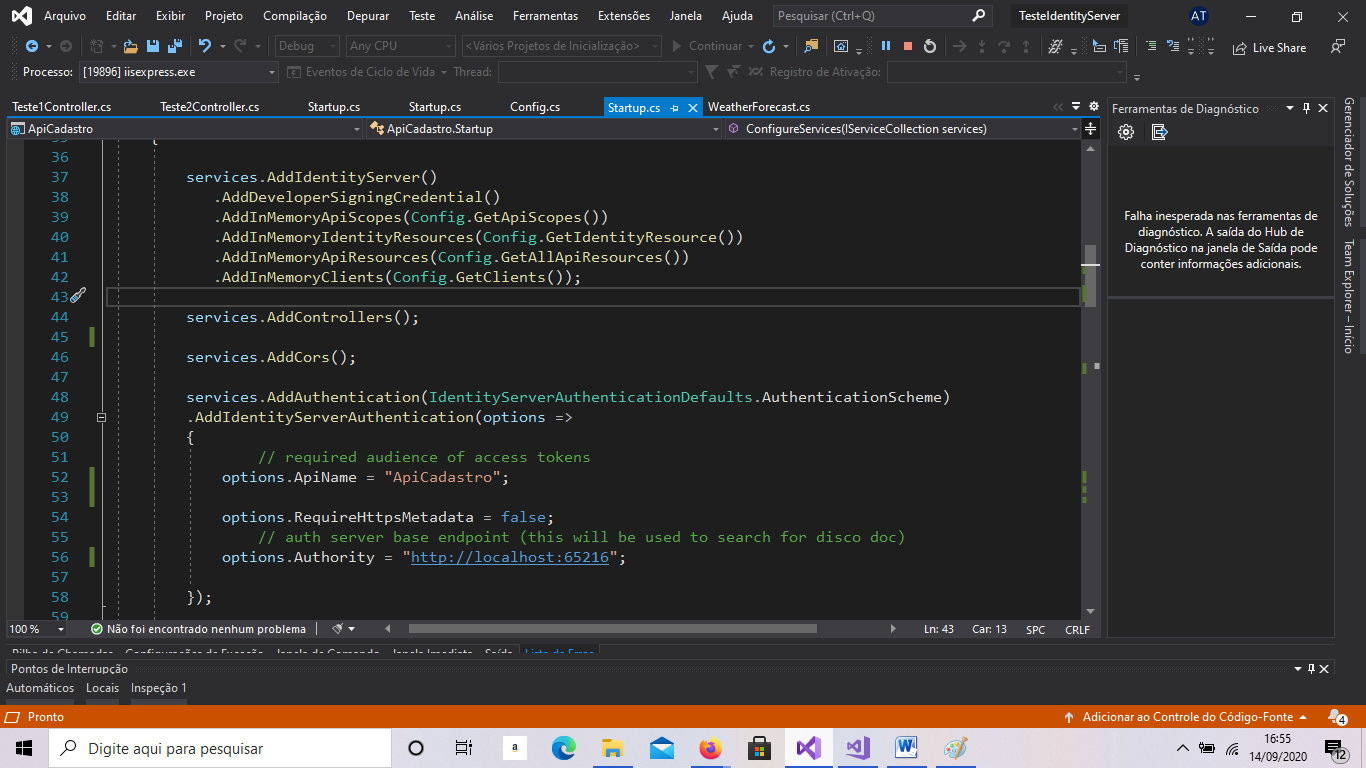
Client

Que podem ser definidos com usuários fixos o por usuários do banco de dados, inclusive definindo o tempo de duração do token, se irá possuir refresh token, re uso do token



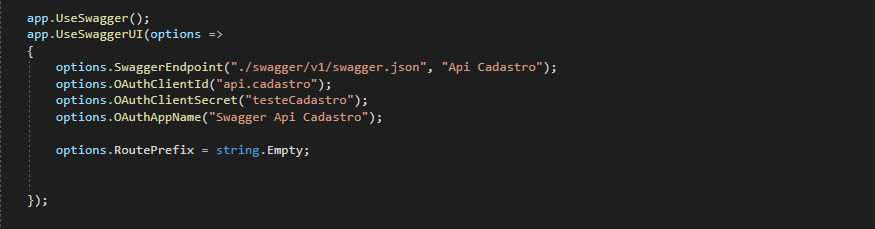
Configuração no Startup

Pode-se notar que a autenticação é feita via token Bearer, o ApiName onde é definido de o compartilhamento de tokens, isto é, se o projeto tiver secrets diferentes ele pode acessar o outro como o mesmo token que foi solicitado o primeiro acesso.



Configuração do Swagger



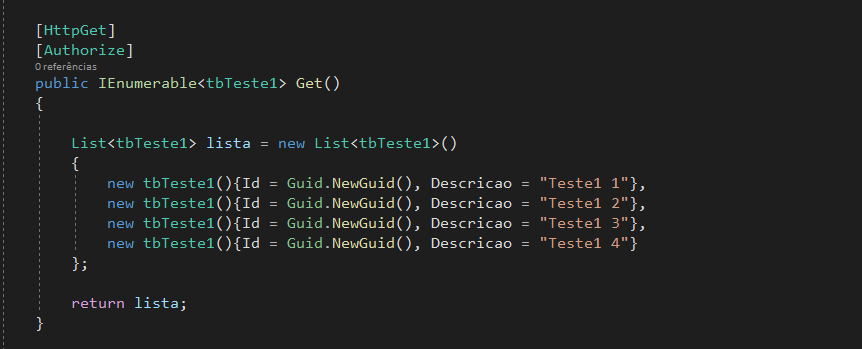


ApiTeste1 e ApiTeste2

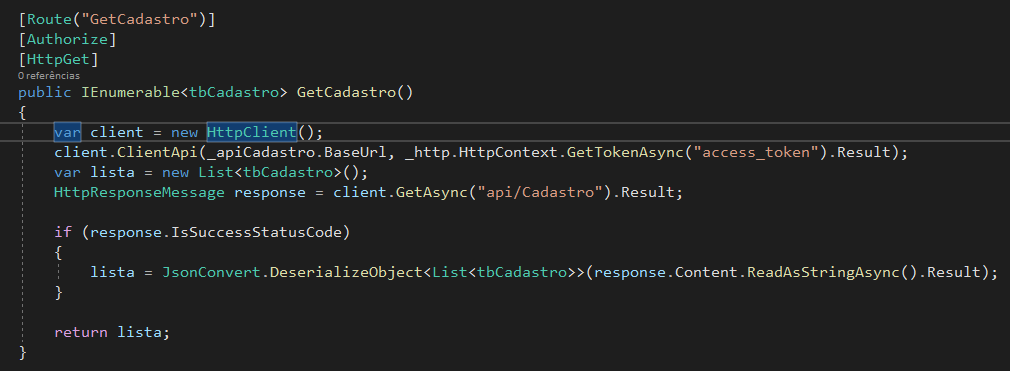
As duas apis possuem a mesma configuração entre elas mudando apenas o clientid e o secret, e apenas um pouco diferente da ApiCadastro, a classe Startup é idêntico

Controller

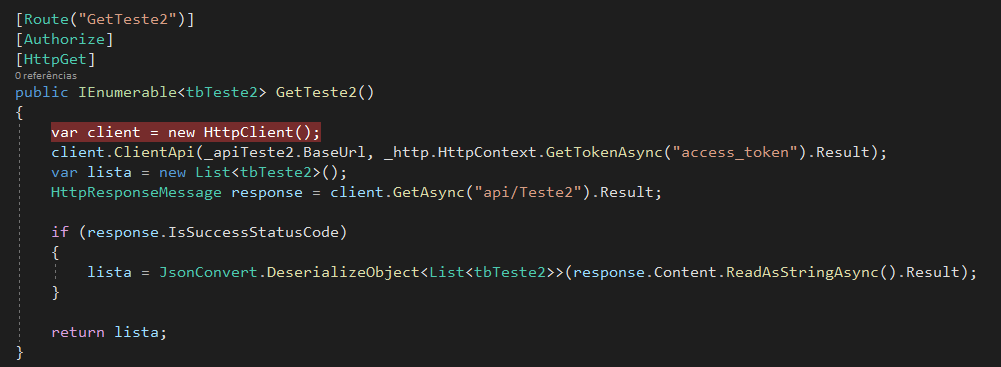
Chamados para própria api no caso ApiTeste1, e para as demais apis de Apicadastro, ApiTeste2



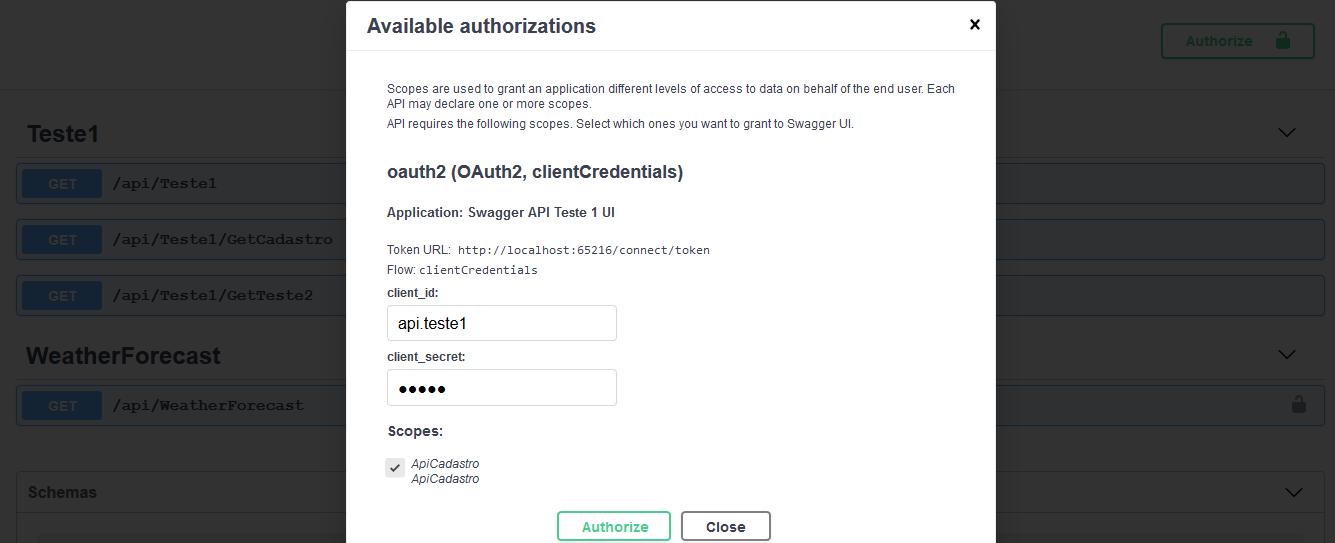
Chamada para ApiCadastro



Chamada para ApiTeste1

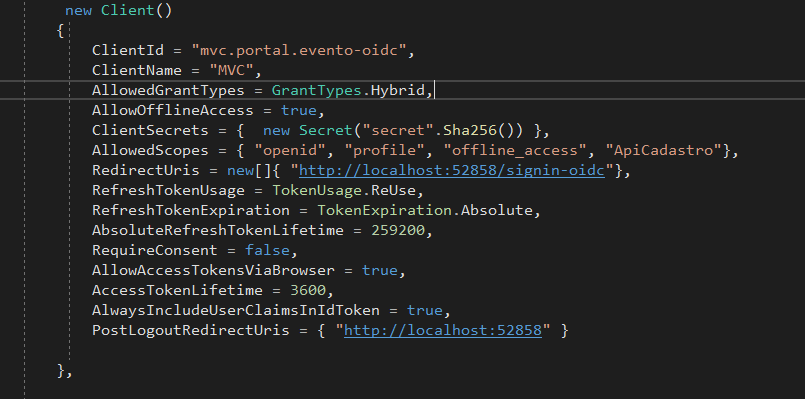


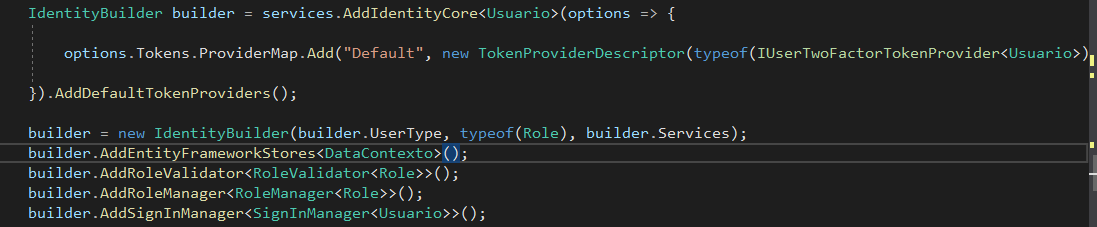
Realizar chamada da ApiTeste1



Ao rodar o projeto os três métodos irão funcionar com o mesmo token

Para configurar para um cliente MVC ou SPA será necessário incluir o identity server a confuguração do Client e o mapeamento das classes do MVC





E no MVC a autenticação se torna diferente, lembrando que os Claims podem ser obtidos através do scope definido na parte de configuração do IdentityServer

