OAUTH 2.0 IDENTITYSERVER4

OAuth 2.0 - Guia do Iniciante

BY BRUNO BRITO ON 9 DE JANEIRO DE 2019

READ IN 5 MIN

O OAuth 2.0 é um protocolo padrão para autorização. Permite que aplicativos como Web App, Mobile e Desktop obtenham acesso limitado às informações de usuários através do protocolo HTTP.

Este artigo dará um overview sobre o OAuth 2.0. Vai descrever suas principais funcionalidades. Pode ser utilizado como guia para aqueles que não estão familiarizados com o protocolo.

Se você possui dúvidas sobre as diferenças entre JWT, Bearer e OAuth 2.0. Não entende bem essa sopa de letrinhas, leia este artigo primeiramente: Segurança - JWT x Cookies x OAuth 2.0 x Bearer.

Roles (Papéis)

O OAuth 2.0 define quatro roles.

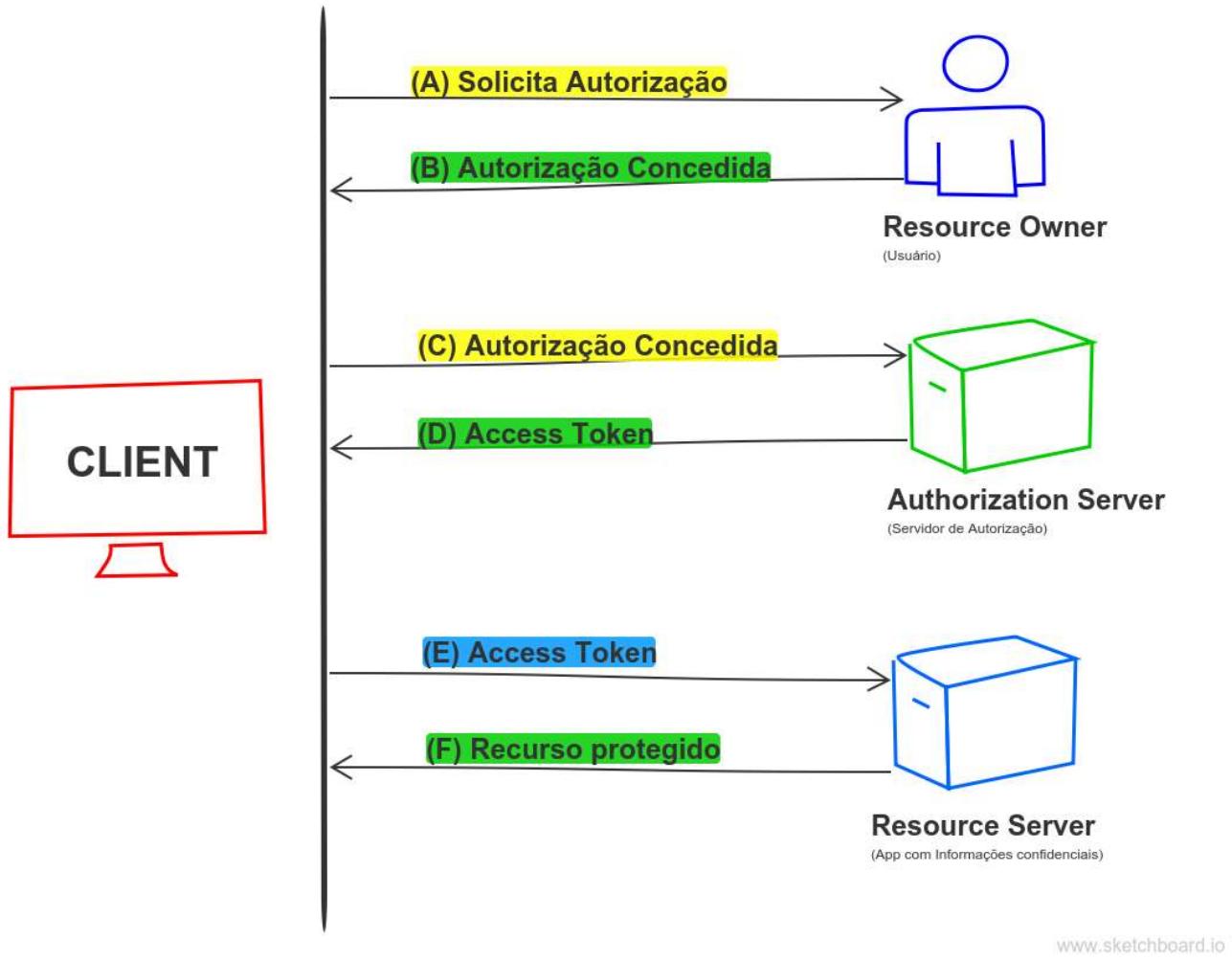
1. Resource Owner - É a pessoa (entidade) que concede o acesso aos seus dados. Literalmente o **Dono do Recurso**. É como o OAuth 2.0 classifica o usuário.
2. Resource Server - É a API. Exposta na internet e contém os dados do Usuário. Para conseguir acesso ao seu conteúdo é necessário um token emitido pelo Authorization Server.
3. Authorization Server - Responsável por autenticação e emitir tokens de acesso (Access Token). Detém informações dos Resource Owner (Usuários) e expõe no formato de

Claims através do **Bearer Token**. Autentica e interage com o usuário após identificar e autorizar o client.

4. Client - É a aplicação que interage com o Resource Owner. No caso de uma App Web, seria a aplicação do Browser.

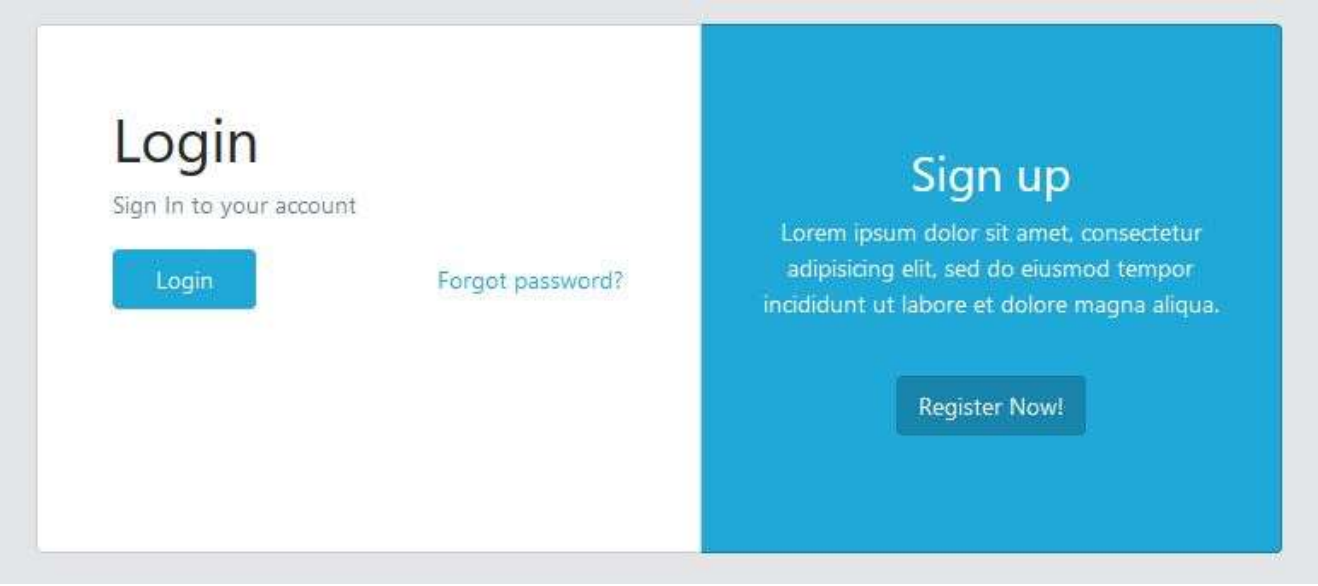
Fluxo de autenticação

Este é fluxo padrão do protocolo. Demonstra de forma simplificada o relacionamento e os papéis dos envolvidos.

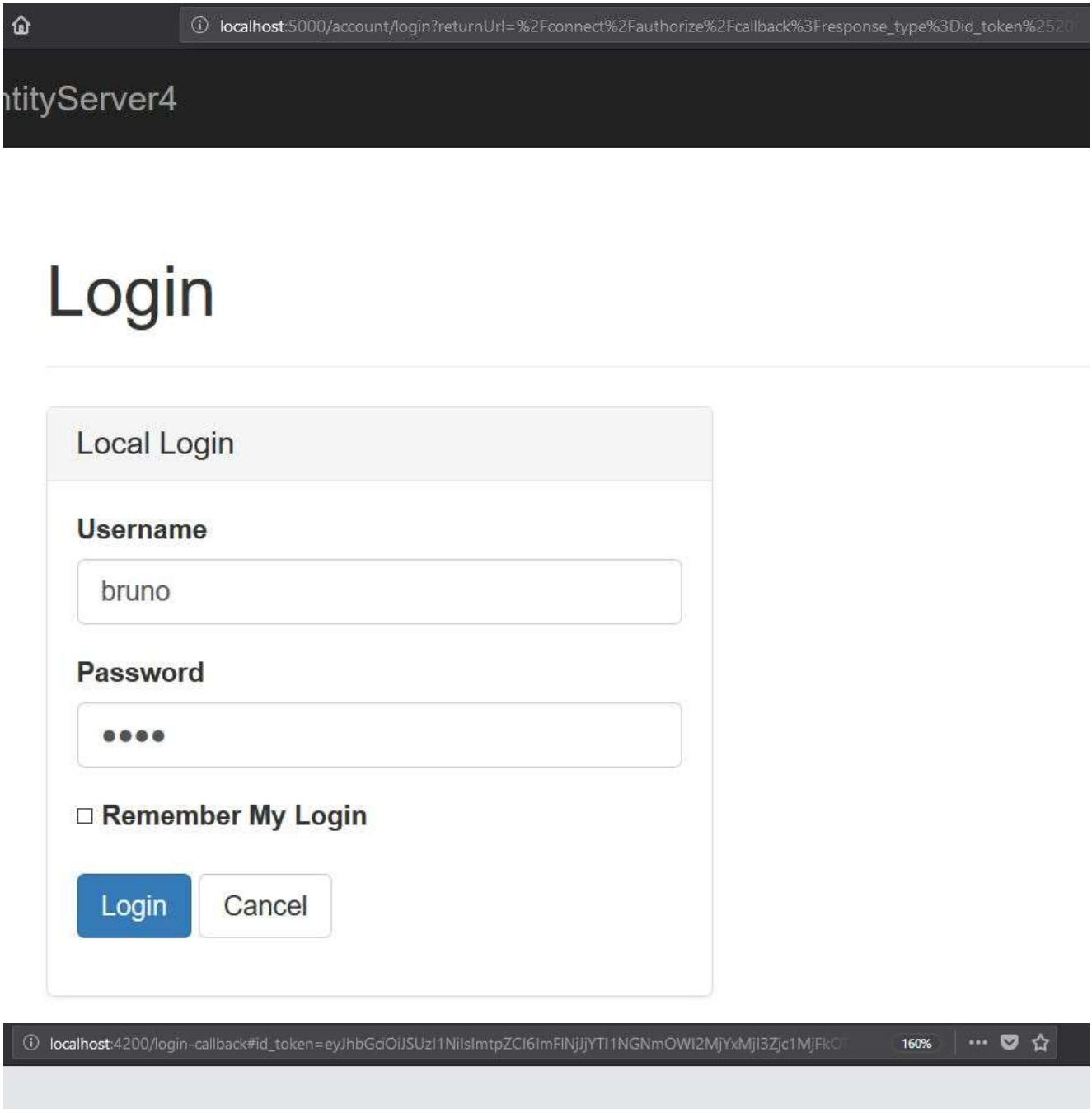


Explicando o fluxo de uma maneira prática.

- (A) O Usuário acessa um client. Para ter acesso ao conteúdo protegido da api (**Resource Server**) o client (A) Solicita Autorização(implicity) ao Resource Owner.
- (B) A **Autorização é Concedida** pelo usuário (**Resource Owner**) ao, por exemplo, clicar no botão `Login`.



- (C) O client solicita um token de acesso ao Authorization Server através da autenticação de sua própria identidade.
- (D) O Usuário (**Resource Owner**) confirma sua identidade através do seu usuário e senha ou através de um terceiro (Facebook, Google). Se tudo ocorrer bem um Access Token será criado e devolvido para o client gerenciar.



- (E) Por fim o client informa o Access Token ao Resorce Server.
- (F) O Resource Server faz validação e retorna o **Conteúdo Protegido**.

O fluxo pode diferenciar, dependendo da configuração da aplicação, como por exemplo no Password flow, onde o cliente não é redirecionado para o Authorization Server.

1 - Authorization Server

É o Single Sign On (SSO) em sí. Centraliza as credenciais de acesso dos usuários. Faz a autenticação do usuário. Responsáver também por:

- Identificar e autenticar **Resource Server**, **Clients** e **Usuários**.
- Gerenciar as claims dos usuários
- Emitir Access Token
- Federation Gateway
- Autenticação como serviço

2 - Clients

Entenda como **Client** uma aplicação que roda na máquina do usuario. O OAuth 2.0 define dois tipos de **Clients**.

1. **confidential** - *clients* que são capazes de manter a confidencialidade de suas credenciais. Por exemplo, cenários de integração, onde um serviço externo da organização desenvolve um serviço que irá consumir a WebApi da empresa.
2. **public** - *clients* incapazes de manter a confidencialidade de suas ceredenciais. Uma SPA, por exemplo, ela está exposta e qualquer usuário consegue saber suas credenciais.

O OAuth 2.0 define fluxos de autenticação para o usuário em cada um dos casos. Para isso, no cadastro do **Client** pode especificar:

- O tipo do **client** (public ou confidential)
- Informar os URL's de redirecionamento.
- Salvar informações como Website, Logo e etc.

Clients em Background

O OAuth 2.0 também permite a integração de Clients que rodam em background sem interação do usuário. Por exemplo uma CRON job que consome certas API, para fazer um relatório, por exemplo.

Redirect URI

Para prevenir ataques o server só irá redirecionar o usuário para a URL cadastrada. A URL deve estar protegida por um HTTPS, evitando ataques man-in-the-middle.

ClientId e Secret

Ao registrar um **client**, o server deve disponibilzar um **Client ID** e uma **Secret**. Se o **client** for público, então o Secret não é criado.

Autorização

Os fluxos de autorização são motivos de debate dentro do OAuth 2.0 Group. Atualmente há um consenso geral sobre os fluxos mais seguros e aqueles que devem ser evitados.

O OAuth 2.0 define 4 tipos de fluxos de autorização. O fluxo vai depender do **client**. Como irá solicitar o **Access Token** e o tipo de **client**.

- **Authorization Code** - O código de autorização é obtido usando um **Authorization Server** como intermediário entre o client e o usuário. O **client** redireciona o usuário para um servidor de autorização.

- **Resource Owner Password Credentials** - Mais conhecido como Usuário e senha. Utilizado quando o **Client** solicita o usuário e senha diretamente. Pode ser considerado como impersonate.

- **Implicit** - O implicit é um fluxo de autorização simplificado. Otimizado para **clients** web. Ao emitir um Access Token, o **Authorization Server** não autentica o cliente. Em alguns casos, a identidade do **client** pode ser verificada por meio do URL de redirecionamento.

- **Client Credentials** - Pode ser usado quando a aplicação **client** é protegida. Um serviço que consulta uma api.

Embora a RFC define eses 4 tipos de fluxos de autorização, há um documento OAuth 2.0 Security Best Current Practice onde diz que **Resource Owner Password Credential** está

como Deprecated .

Atualmente o flow recomendado para SPA é **Authorization Code** com PKCE (Proof Key for Code Exchange). A tabela a seguir contém um guia de quando utilizar cada um dos fluxos.

Tipo de client	Fluxo de autorização
SPA's, MVC	Authorization Code
Aplicações não confiaveis (Apps de terceiros)	Authorization Code
Aplicações confiáveis (apps internas da empresa)	Authorization Code ou Implicit
Integrações de sistemas (apps background)	Client credentials

Bearer Token

São credenciais usadas para acessar **recursos protegidos**. Colocando de forma prática, o access_token é enviado para as API's através do Header **Authorization**.

Assim que o usuário faz o login no Auth Server, ele vai entregar um access_token para o **client**.

É uma string que representa uma autorização emitida para o **client**. Os tokens possuem informações que o usuário concedeu ao **Authorization Server**, como nome, e-mail e endereço. Possui os dados da sessão como o tempo de duração do acesso.

Atualmente o formato padrão adotado por quase todas as implementações de OAuth 2.0 é o **JWT**.

No entanto a especificação do OAuth 2.0 não menciona o JWT como o padrão. Apenas especifica os critérios de segurança que o Bearer Token deve ter.

A RFC diz que a natureza do token deve permitir camadas de abstração, permitindo informações adicionais. Por exemplo os dados do usuário, endereço. Ou qualquer outra informação. O conceito de claims é empregado para lidar com esse conjunto de dados.

A RFC é abrangente em relação aos tokens. Permite que tenham formatos e estruturas diferentes. Pode ser utilizados de muitas maneiras. Desde que garantam a consistência e

segurança dos dados.

Devido essa abrangencia, possui uma RFC somente para ele, RFC6750.

Referencias

- RFC6750
- RFC6749
- OAuth 2.0
- Understanding OAuth2 and Building a Basic Authorization Server of Your Own: A Beginner's Guide
- What is the OAuth 2.0 Authorization Code Grant Type
- OAuth 2 Simplified
- Uma introdução ao OAuth 2
- OAuth 2.0 Security Best Current Practice

WRITTEN BY

BRUNO BRITO

Primeiramento sou pai! #dev fullStack e MCSA Web Application.

BRAZIL WEBSITE FACEBOOK

COMMENTS

TAMBÉM NO **BRUNOBRITO**

há 2 anos • 4 COMENTÁRIOS
.NET Core - Console Application

há um ano • 7 COMENTÁRIOS
ASP.NET Core - Como assinar ...

há um
Pac

Comentários **Comunidade** **Política de privacidade**

1 **Iniciar sessão**

Recomendar Tweet Partilhar

Mostrar primeiro os mais votados

Escreva o seu comentário...

INICIE SESSÃO COM O

OU REGISTE-SE NO DISQUS ?

Nome

mcadori • há um mês
também procuro auxilio para implementar OAuth denro do domínio da minha empresa que já é cliente
da plataforma Google
Responder • Partilhar ›

Rodrigo Desenheiro • há um ano
onde encontro documentação para implementar o uso da autenticação do gsuite (tenhum dominio próprio que usa o gmail como base de email) para colocar login e senha em uma "intranet" que estou criando mas somente para logar com o gsuite com dominio próprio ?? deu pra entender? tenho um dominio (@minhaempresa que usamos o gsuite,)
Responder • Partilhar ›

Marcelo Cadori ↗ Rodrigo Desenheiro • há um mês
Rodrigo você encontrou algum material para auxiliar? Eu tbm estou querendo implementar algo dentro de um domínio da minha empresa.
Responder • Partilhar ›

NEXT

Segurança - JWT x Cookies x OAuth 2.0 x Bearer

23 DE JANEIRO DE 2019

PREVIOUS

HTTP Security Headers

13 DE NOVEMBRO DE 2018

LATEST POSTS

2020 - Um ano de altos e baixos

31 DE DEZEMBRO DE 2020

IdentityServer4 Pago - E Agora?

14 DE OUTUBRO DE 2020

Medindo a Maturidade de sua API - Richardson Maturity Model

14 DE OUTUBRO DE 2020

ASP.NET Core - Adicionando Cache em qualquer componente com Injeção de Dependência

18 DE JUNHO DE 2020

TAGS

ASP.NET CORE

ARQUITETURA

OAUTH 2.0

SEGURANÇA

IDENTITYSERVER4

SHOW ME THE CODE

ANGULAR

VISUAL STUDIO

.NET CORE

OPEN SOURCE

OPENID CONNECT

NOVIDADE

MODERNIZAÇÃO

OWASP

DESIGN PATTERNS

BLAZOR

ESTRATÉGIA

DESIGN

DOMAIN-DRIVEN DESIGN

WEBASSEMBLY

ASP.NET IDENTITY

AZURE

ENTITYFRAMEWORK CORE

MICROSSERVICOS

MONGODB

DOCKER .NET 5

RESUMO SOLID

COGNITIVE SERVICES

IIS