# OAuth 2.0 &

# OpenID Connect

- História
- Introdução
- Funcionamento

BASE2

# Sobre mim

- Hamon-Rá Taveira Guimarães - 29 anos
- Estudante de Engenharia da Computação - PUC Minas
- Backend no Crowdtest/MeloQA - Ruby on Rails

- GitHub - https://github.com/Pharaoh00
- Linkedin - https://www.linkedin.com/in/hamonra/

Apresentação - https://github.com/Pharaoh00/Google-Oauth2.0-Demo

BASE2

## Problema

História sobre OAuth & OpenID Connect

01

## Os primórdios

Problemas antigos, soluções insegura.

02

## História

Como foi criado. E o futuro do Oauth.

03

## O que é Oauth? OIDC?

Diferença entre Oauth2 e OpenID Connect.

04

## Fluxos básico

Como o protocolo realmente funciona.

05

## Mão na massa!

Testando o protocolo utilizando React e Express.

06

# Disclaimer

- Eu sou gago.
- Todo o conteúdo é “relativo” e interpretado.
- Nosso papo é em cima da implementação do Oauth2.0.
  - Oauth1 é antigo e **perigoso**.
- Empresas diferentes interpretaram a especificações de maneiras diferentes, apesar de implementarem o mesmo protocolo, podem haver mudanças de provedor à provedor.
- É importante ler os guides de cada provedor.
- Alguns slides são da apresentação do Dominick Baier.

# 01

# Problema

Como conectar toda a internet, acessar dados de outros serviços, utilizando somente um único **provedor**.

# Problema

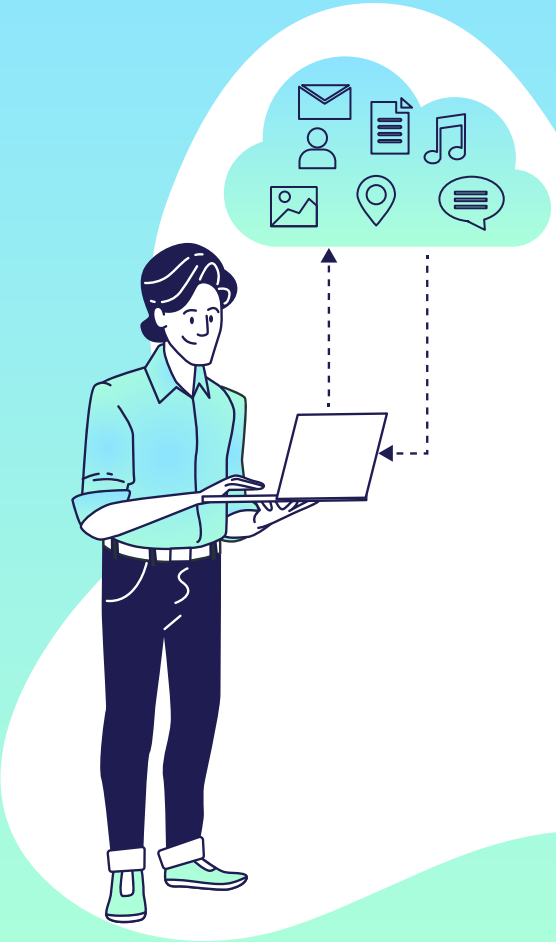
- Como conectar serviços de api's diferente?
- Como autenticar o usuário em diferentes serviços?
- Como assegurar os dados do usuário?

Parecem problemas simples, mas acabam sendo complexos, principalmente quando o foco é a segurança.

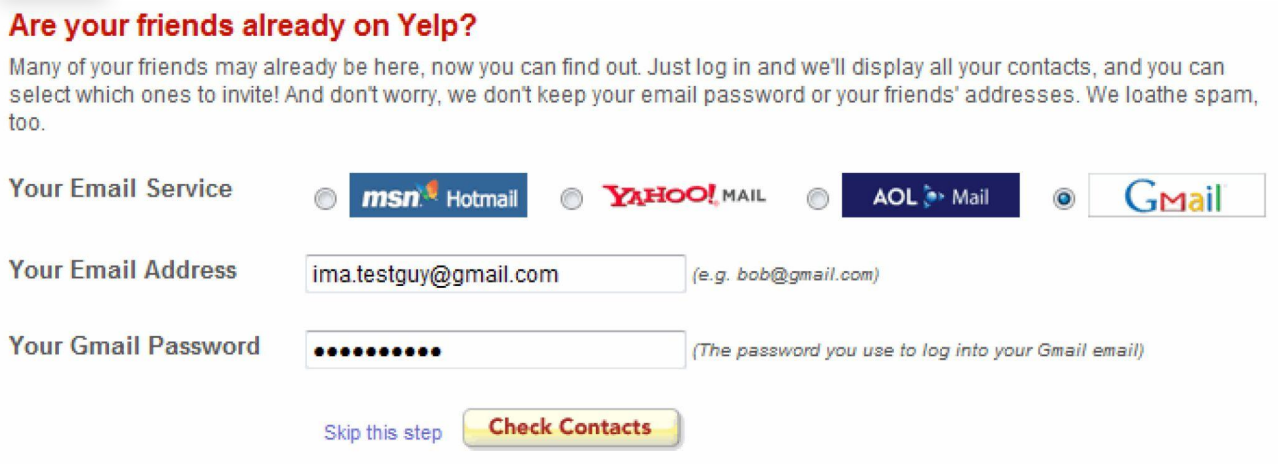
**Na internet nada é seguro!**

# 02
# Os primórdios

Como as empresas, antigamente, tentaram resolver um **problema moderno**.



# Os primórdios




OAuth and OpenID Connect (IN PLAIN ENGLISH) - NATE BARBETTINI

- Essa página era como a empresa Yelp tentou resolver um problema, moderno, e de difícil implementação, há 15 anos ou mais.
- Era uma “prática comum” em pedir o e-mail e senha do usuário para acessar recursos protegidos. Hoje em dia é uma prática **perigosa** e **desatualizada**.

# 03

# História

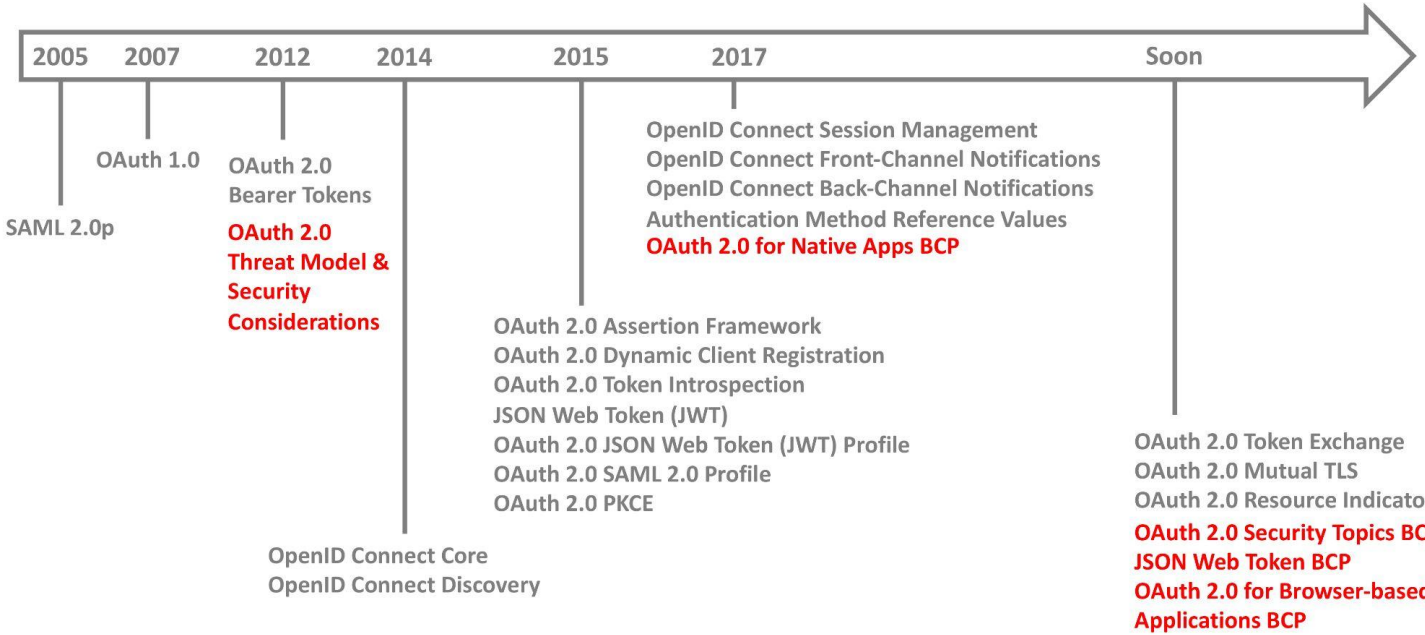
"History Time".
Como veio a ser o que conhecemos hoje em dia com **Oauth**.

# História

- 2006
  - Blaine Cook, trabalhava no Twitter, desenvolvia uma versão do OpenID para a mesma.
- 2007
  - Um grupo pequeno foi fundado para a discussão sobre a implementação.
  - DeWitt Clinton, da Google, conheceu o projeto e expressou interesse.
  - Julho as especificações iniciais foram feitas.
  - Eran Hammer se juntou ao grupo como coordenador e contribuir nas especificações.
  - Dezembro OAuth foi lançado.
- 2008
  - Novembro a IETF teve sua 73rd reunião sobre a padronização do OAtuh como protocolo padrão.
- 2010
  - Abril OAuth teve sua publicação como um RFC (Request for Comments) 5849.
  - Agosto todas as aplicações terceira do Twitter precisam utilizar OAuth
- 2012
  - Outubro OAuth 2.0 foi publicado como uma RFC 6749.
  - Outubro Bearer Token foi publicado como uma RFC 6750.

# História

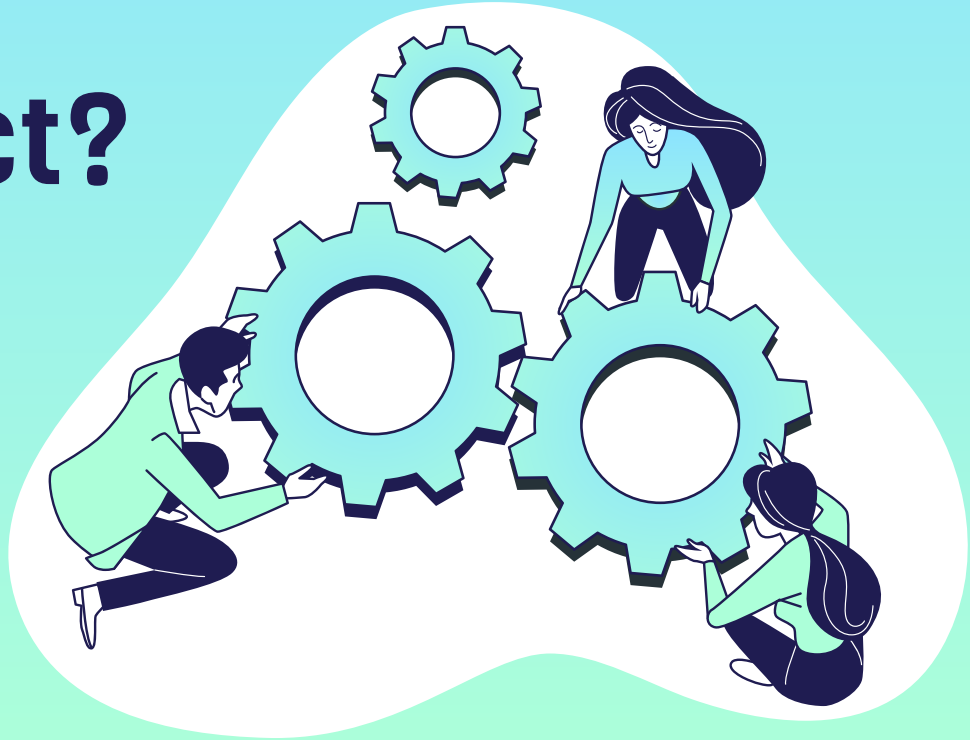
## Some Context…



# 04

# O que é Oauth? OpenID Connect?

O que cada protocolo tenta resolver e principalmente qual a **diferença** entre os dois.



# O que é OAuth 2.0

**The OAuth 2.0 Authorization Framework**

Abstract

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. This specification replaces and obsoletes the OAuth 1.0 protocol described in RFC 5849.

The OAuth 2.0 Authorization Framework - RFC 6749

OAuth 2.0 é um padrão de segurança que delega permissão a um aplicativo para acessar dados de outro.
Basicamente OAuth é um framework de **autorização**, onde o aplicativo irá pedir permissão ao usuário, concebendo assim **permissão,** o mesmo pode acessar conteúdos **protegidos** em nome do usuário.
OAuth 2.0 é utilizado pela Amazon, Google, Facebook, Microsoft, Twitter e vários outros sites.

# O que é OpenID Connect

1. **Introduction**

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 **[RFC6749]** protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

The OpenID Connect Core 1.0 specification defines the core OpenID Connect functionality: authentication built on top of OAuth 2.0 and the use of Claims to communicate information about the End-User. It also describes the security and privacy considerations for using OpenID Connect.

OpenID Connect Core 1.0 incorporating errata set 1

OpenID Connect ou OIDC, no contexto do OAuth, é um protocolo ou camada que tem o seu foco em identificar o usuário final. Sem um framework de autorização OAuth 2.0 é incapaz de saber informações sobre o usuário autorizado. Basicamente OIDC implementa **Autenticação** em cima de um sistema de **Autorização**.

BASE2

# Qual a diferença entre OAuth e OpenID Connect

Podem parecer iguais mas as duas especificações agem de forma diferente. OAuth é sempre para **autorização** e OpenID Connect (OIDC) é para **autenticação**.

OAuth é um framework de delegação, por isso é necessário pedir permissão ao usuário, pois o mesmo está autorizando um aplicativo à utilizar “suas credenciais”. O protocolo foi desenvolvido para justamente não saber as credenciais do usuário.

Então o que OIDC agrega ao sistema?

- Identificação por ID.
- Informações sobre o usuário, logado, aos endpoints.
- ‘Scopes’ padronizados.
- Implementações diferentes padronizadas.

05

# Fluxos básico

Como Oauth2.0 funciona.
E qual a **importância** do usuário em todo o processo.

# Terminologia

- **Access token** - Token utilizado para acessar informações protegidas.
- **Authorization code** - Token intermediário que será trocado pelo Access Token.
- **Authorization server** - Servidor que “conversa” com a aplicação e provê o Oauth.
- **Client** - A aplicação que deseja acessar os recursos protegidos.
- **Resource server** - O servidor que contém as informações protegidas.
- **Resource owner** - O usuário que autoriza o **Client** para acessar os recursos em seu “nome”.
- **Back channel** - Comunicação segura.
- **Front channel** - Comunicação não segura.
- **Scope** - Permissões de acesso a recursos seguros.
- **Consent** - Fluxo para pedir permissão ao usuário.
- **Redirect url** - Endpoint controlado pelo **Client**.

# Fluxo básico

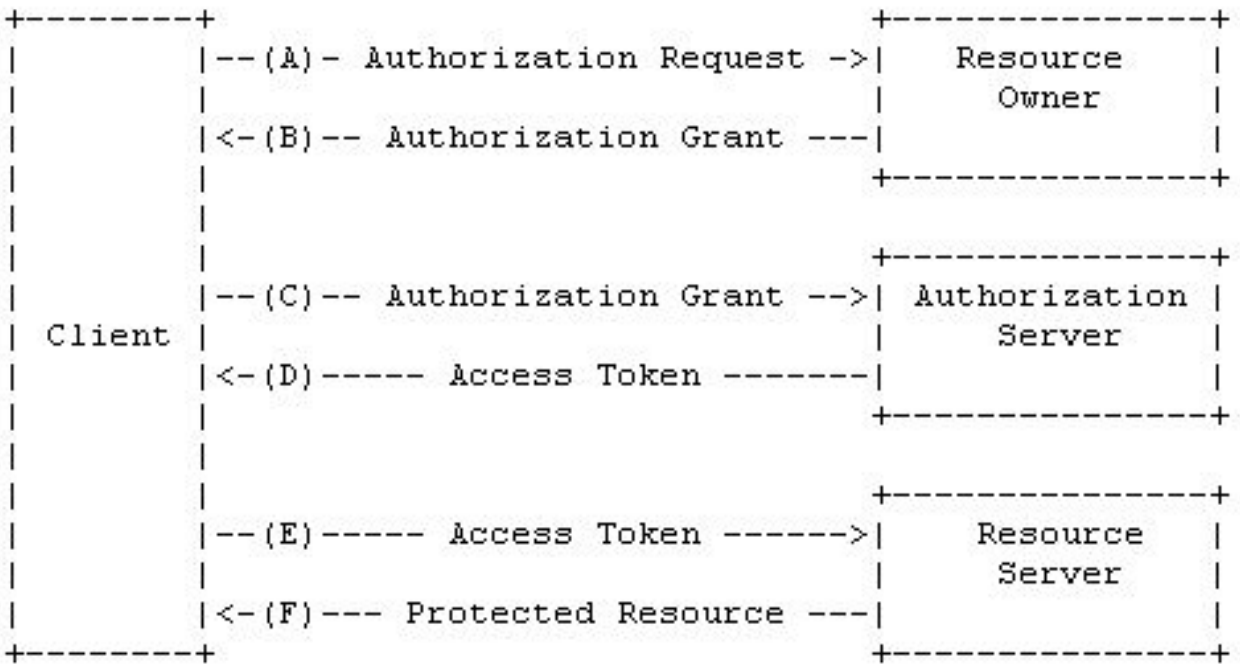



Figure 1: Abstract Protocol Flow

The OAuth 2.0 Authorization Framework - RFC 6749

BASE2

# Fluxos básicos

### 1.3. Authorization Grant

An authorization grant is a credential representing the resource owner's authorization (to access its protected resources) used by the client to obtain an access token. This specification defines four grant types -- authorization code, implicit, resource owner password credentials, and client credentials -- as well as an extensibility mechanism for defining additional types.

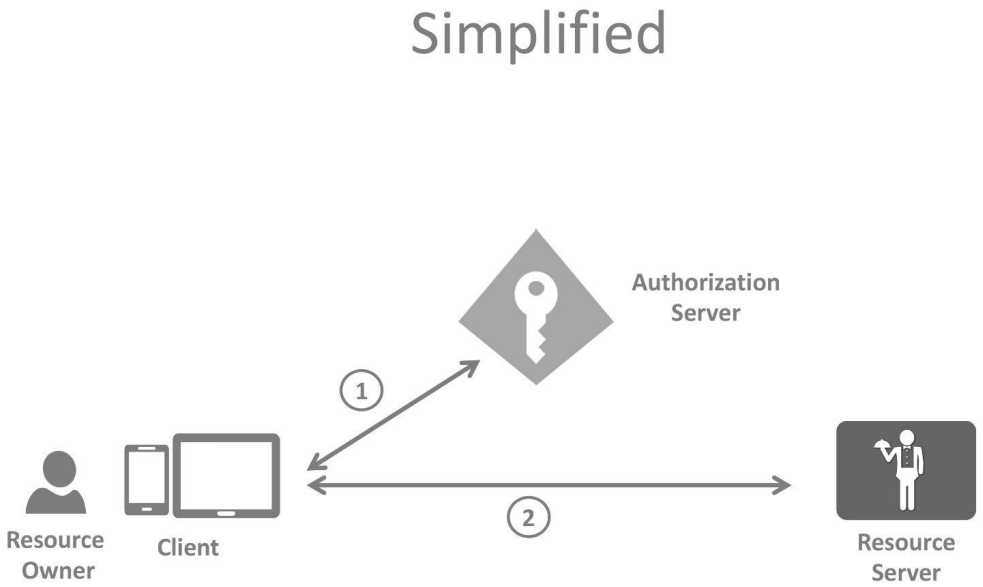
The OAuth 2.0 Authorization Framework - RFC 6749

Existem 4 tipos de fluxos diferentes:

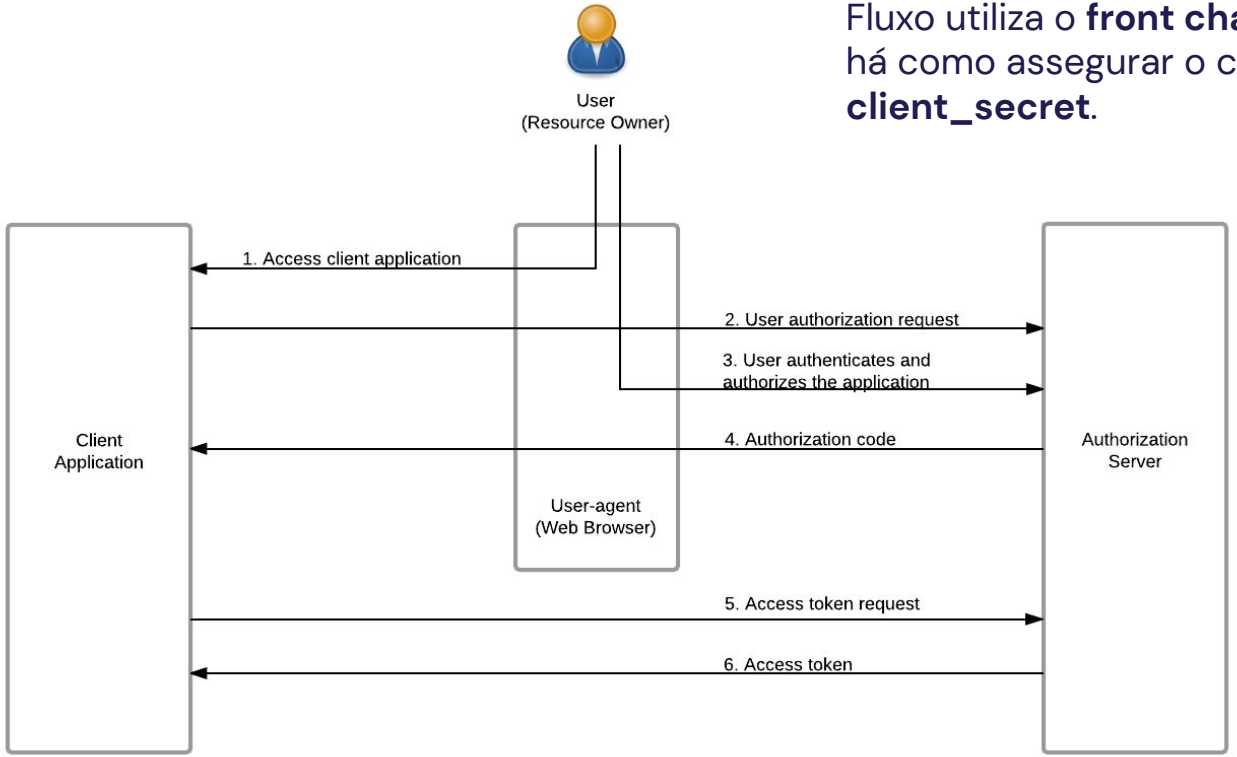
1. Authorization Code
2. Implicit
3. Resource Owner password credentials
4. Client Credentials

No OAuth modern ou BCP são somente utilizados **Authorization Code** e **Client Credentials**, por questões de segurança pesquisadores descobriram falhas e novas especificações e implementações tiveram que se adaptar.

# Fluxos simplificado



# Fluxo 'Authorization Code'

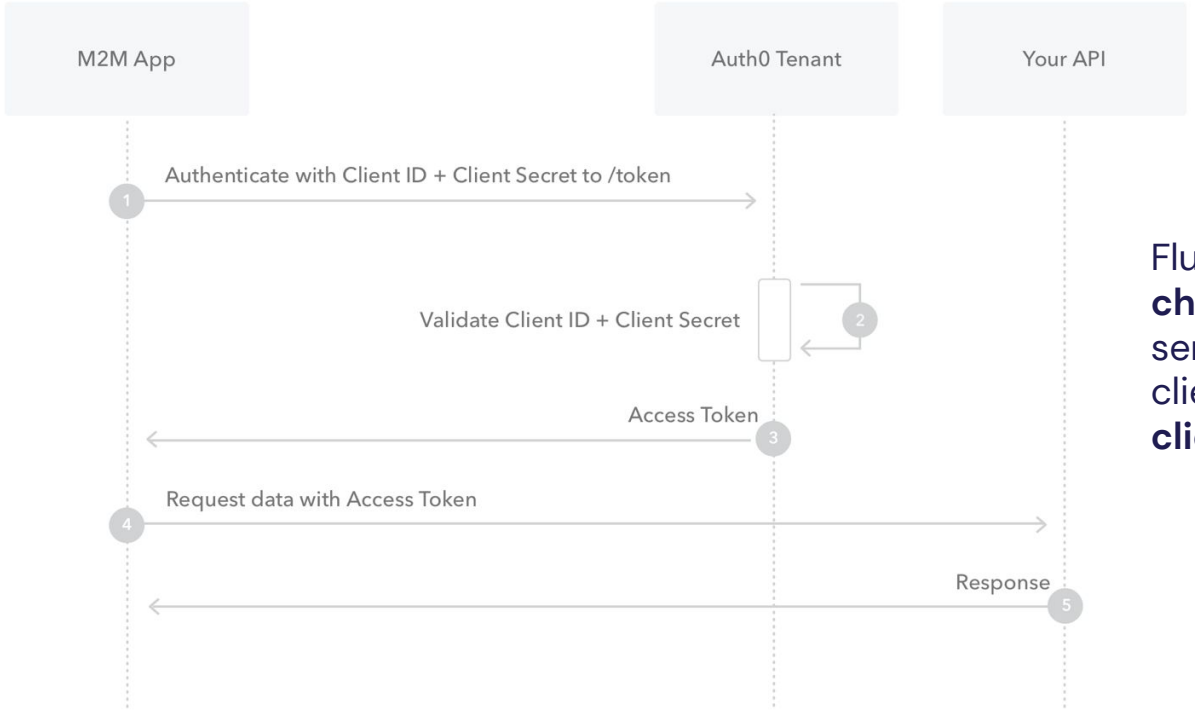



Fluxo utiliza o **front channel**, pois não há como assegurar o client_id e **client_secret**.

https://apim.docs.wso2.com/en/latest/design/api-security/oauth2/grant-types/authorization-code-grant/

BASE2

# Fluxo 'Client Credentials'

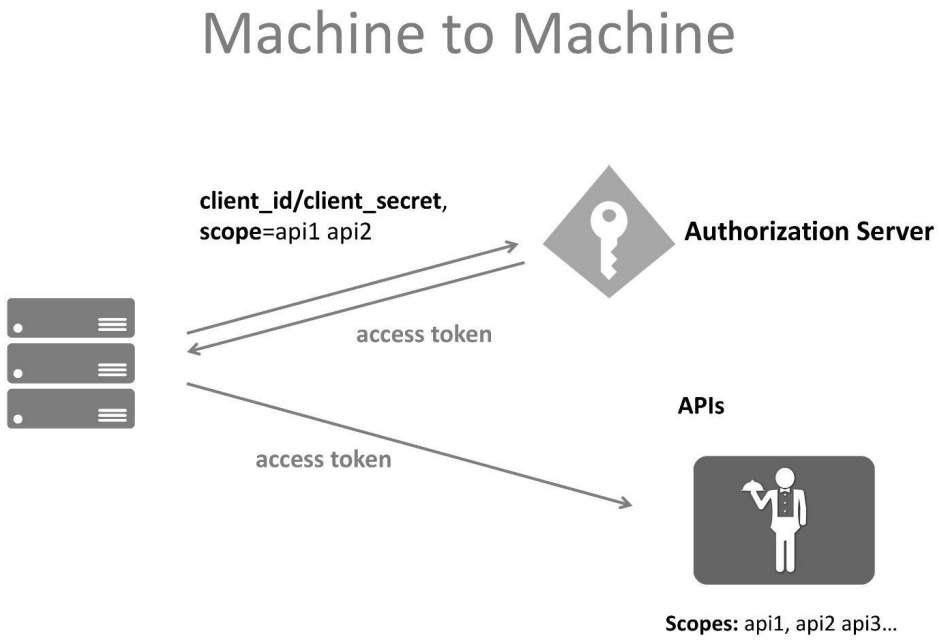



Fluxo utiliza o **back channel** pois o servidor assegura o client_id e **client_secret**.

https://auth0.com/docs/get-started/authentication-and-authorization-flow/client-credentials-flow

BASE2

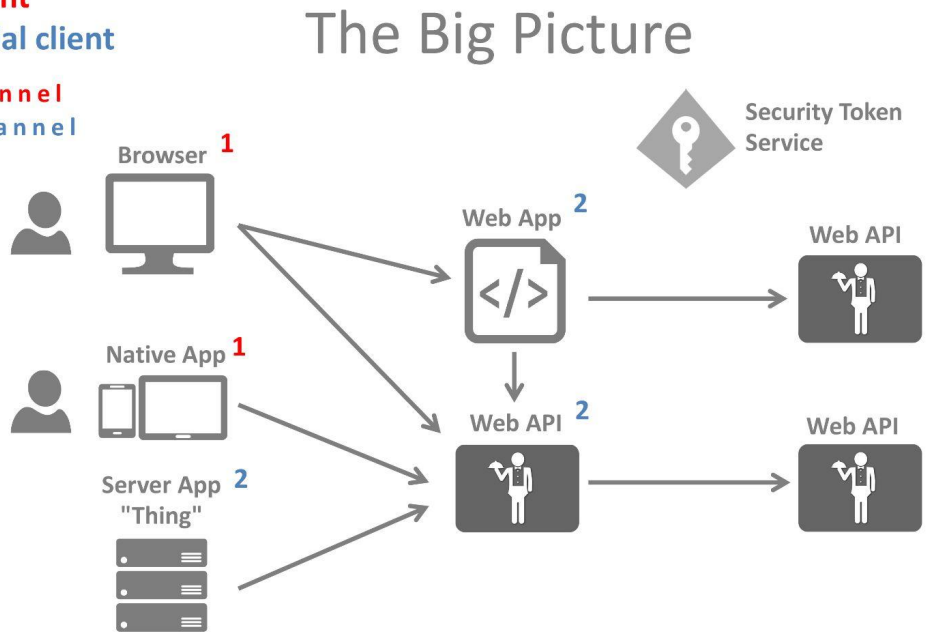
# Fluxos 'Client Credentials' simplificado



# Exemplo ‘mundo real’

1 public client
2 confidential client

1 Back Channel
2 Front Channel



# 06

# Mão na massa!

Vamos programar!
O código está em React e Node/Express.

Iremos passar “manualmente” pelo fluxo e entender a suas etapas.

# O que é OAuth 2.0

JSON Web Token (JWT)

Abstract

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

https://datatracker.ietf.org/doc/html/rfc7519

É um token, compacto, url-safe, seguro para transferência de informações entre duas. É composto de 3 partes, **Header**, **Body/Payload**, **Signature,** dividido entre pontos. A assinatura é parte importante do processo pois a mesma usa algoritmos de criptográfia, como por exemplo HMAC ou RSA, para ter certeza que o jwt não foi alterado entre as partes.

Exemplo de um JWT:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

BASE2

# Começar o fluxo - Authorization request

Request:

GET /authorize?response_type=code&client_id=CLIENT_ID&
redirect_uri=STRING_URL_SAFE HTTP/1.1
Host: server.example.com

Response:

HTTP/1.1 302 Found
Location: https://STRING_URL_SAFE/code=ALGUM_TOKEN

Trocar token:

POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic CLIENT_SECRET
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=ALGUM_TOKEN&
recirect_uri=STRING_URL_SAFE

# Começar o fluxo - Authorization request

Resposta:

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
“access_token”: ALGUM_TOKEN,
“token_type: “Bearer”,
“Expires_in”: TEMPO_EM_S_DO_JWT,
“id_token”: JWT
}

# Bibliografia


- An Illustrated Guide to OAuth and OpenID Connect
  - Vídeo
- Terminology Reference
- OpenID Connect Flows
- OAuth 2.0 and OpenID Connect (in plain English)
  - Slides
- Authentication as a Microservice
- Implementing OpenID Connect and OAuth 2.0 – Tips from the Trenches - Dominick Baier
  - Slides
- OAuth 2.0 Security Reinforced
- OAuth 2.0 for native Applications
- OAuth 2.0 for Browser-Based Applications
- OAuth 2.0 Security Best Current Practice
- JSON Web Token Best Current Practices
- JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens
- OAuth 2.0 Threat Model & Security Considerations