



ADC
Applaudo
Developers
Conference
2020

Full Stack OAuth2 Flow

- [Must watch](#) -



Arely Viana

Backend

Technologies



/areviana



/areviana



Nicolas Martinez

Frontend

Technologies



/nicokurogane



/nicokurogane

AGENDA

+

1

Concepts
Básicos

2

¿Cómo
funciona?

3

Grant Types

4

Authorization
Code Flow

5

Código
Backend

6

Código
Frontend



Conceptos básicos

- Going back to the basics strengthens your foundation -

Authentication vs Authorization

Autenticación

¿Quién soy?

Certificar, validar que un usuario es quien dice ser.

/signup

/login

/resetpassword

Autorización

¿Qué puedo hacer?

Dar acceso a los recursos de un sistema.

/authorize

/token

OAuth

JWT

OAuth - Open Authorization

OAuth es un [protocolo \(v 1.0\)](#) o [framework \(v 2.0\)](#)

Se utiliza para:

- Autorizaciones entre servicios dando accesos limitados
- Estandarizar el flujo de credenciales entre cliente y servidor
- Dar permisos, no credenciales



JWT - Json Web Token

[JWT](#) son una forma compacta y segura de transmitir información entre 2 partes

Puntos importantes:

- "access_token" ⇒ | Header "Authorization": "Bearer <token>" |
- Codifica JSON en Base64
- Estructura: HEADER.PAYLOAD.SIGNATURE
- PAYLOAD ⇒ [Claims](#)

jwt.io

 Debugger Libraries Introduction Ask Get a T-shirt! Crafted by  Auth0

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

Type of token


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

 Signature Verified

SHARE JWT

Roles

OAuth 2.0 define 4 roles

1. Dueño del recurso (Resource Owner):
Usuario quien autoriza a la aplicación a acceder a sus recursos
2. Servidor de autorización (Authorization Server):
Verifica la identidad del usuario y emite tokens de acceso para la aplicación
3. Servidor de recursos (Resource Server):
Aplicación que aloja los recursos a los que queremos acceder
4. Cliente (Client):
Aplicación que quiere acceder a los recursos del usuario

Tipos de cliente

Cientes privados

- Lado del servidor
- Buen amigo! Guarda tus secretos

Ejemplos: App Nativas, APIs

Cientes públicos

- Lado del cliente
- No es de fiar! No puede guardar secretos

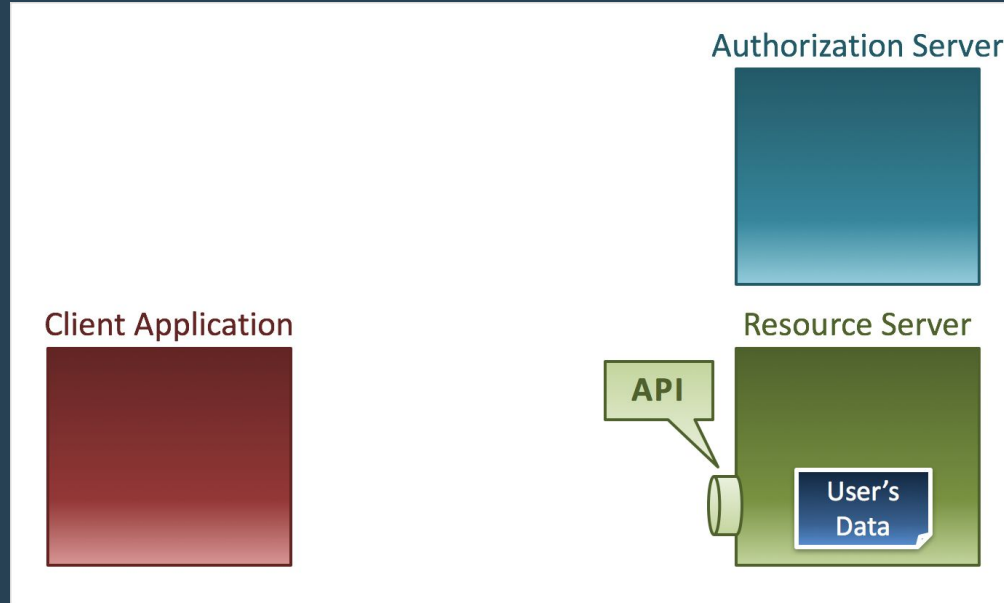
Ejemplos: SPA, Páginas renderizadas en el navegador

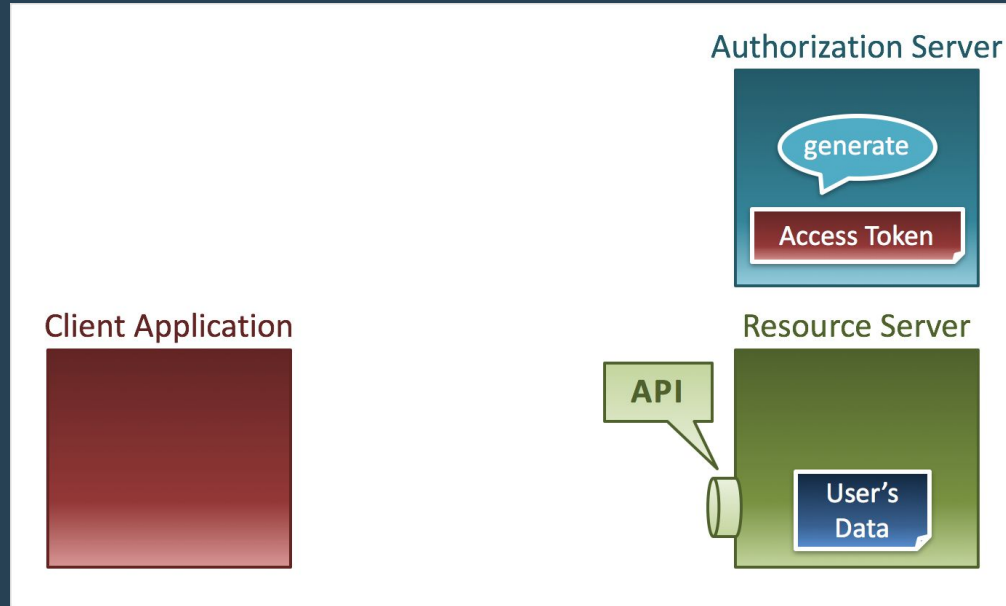
¿Cómo funciona?

- [OAuth2 Introduction](#) -

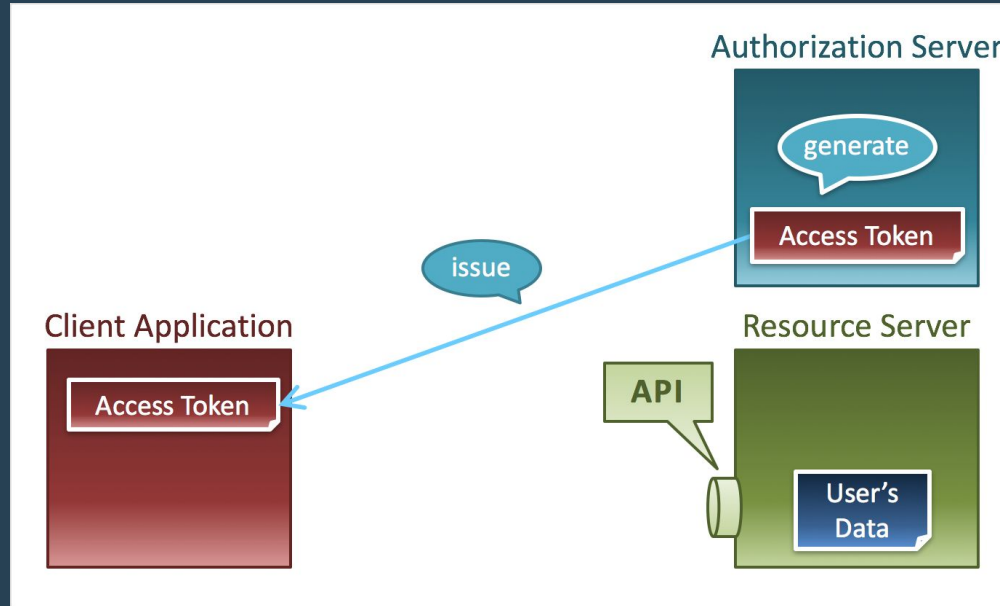
Flujo Abstracto de autorización

- Abstract protocol flow -

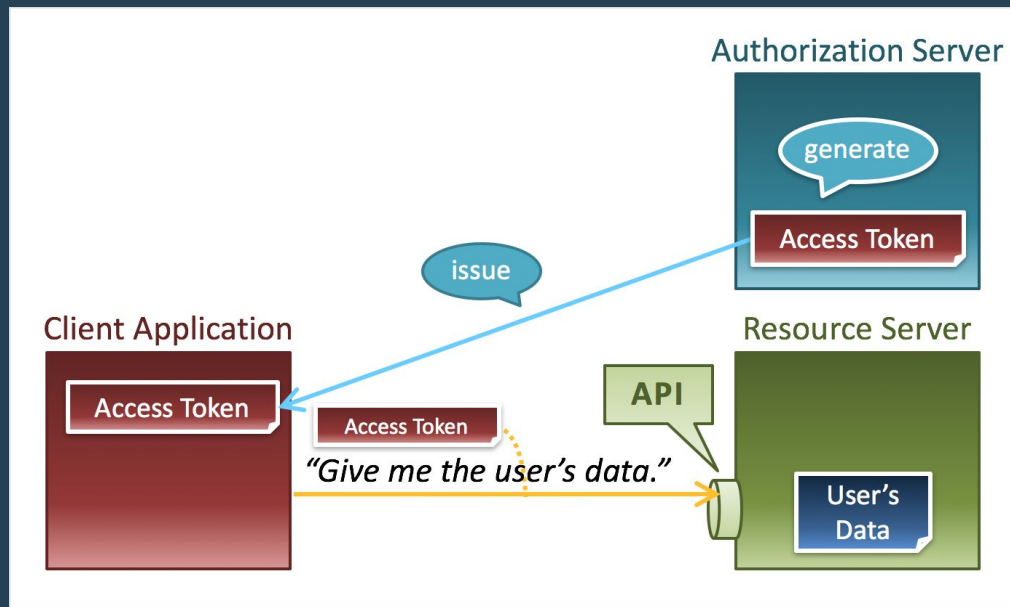




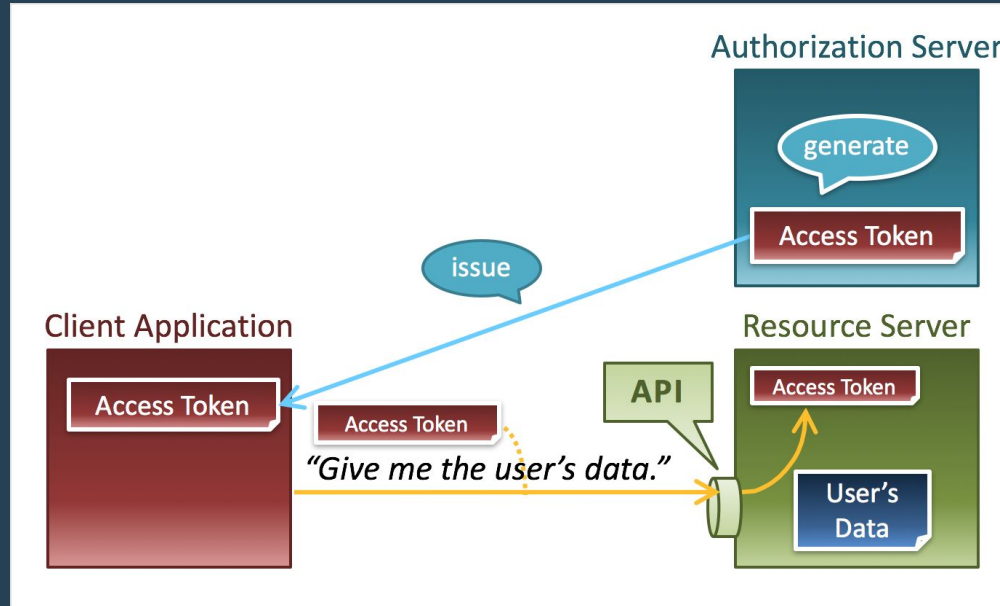
El authorization server genera un token de acceso



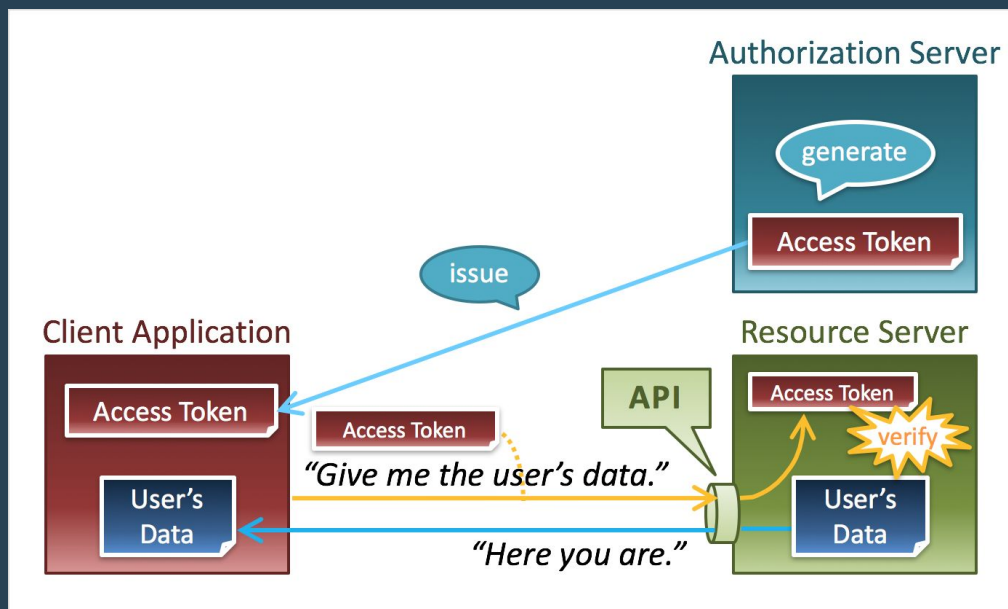
Y entrega el token al cliente que lo solicitó. Para que se entregue el token, requiere la aprobación del usuario.



Luego de tener el token de acceso, el cliente puede pedir la información al resource server con dicho token.



El resource server extrae el token de acceso y hace las verificaciones al token.



Y si todo esta en regla, el resource server entrega la información a la cual se le ha concedido permiso

Grant Types

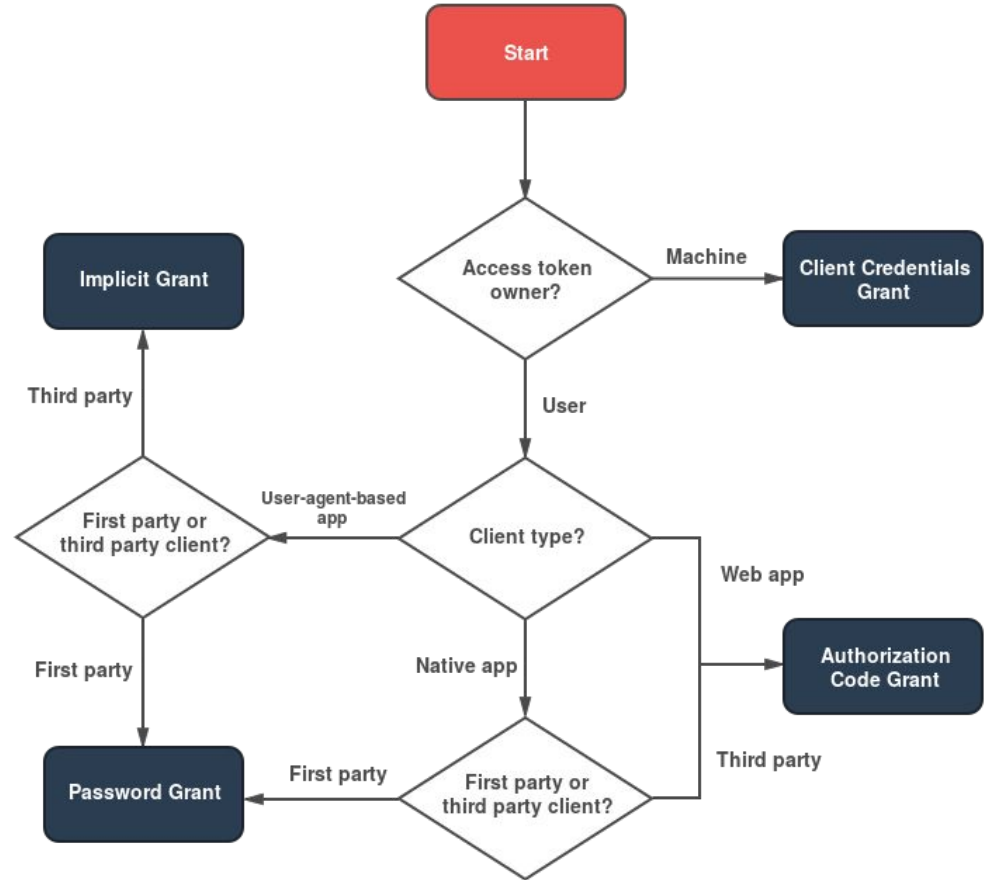
- [Grant Types](#) -

Tipos

Son 4 tipos:

1. Client credentials grant
2. Implicit flow
3. Authorization code flow
4. Password grant

¿Cuál debería escoger?

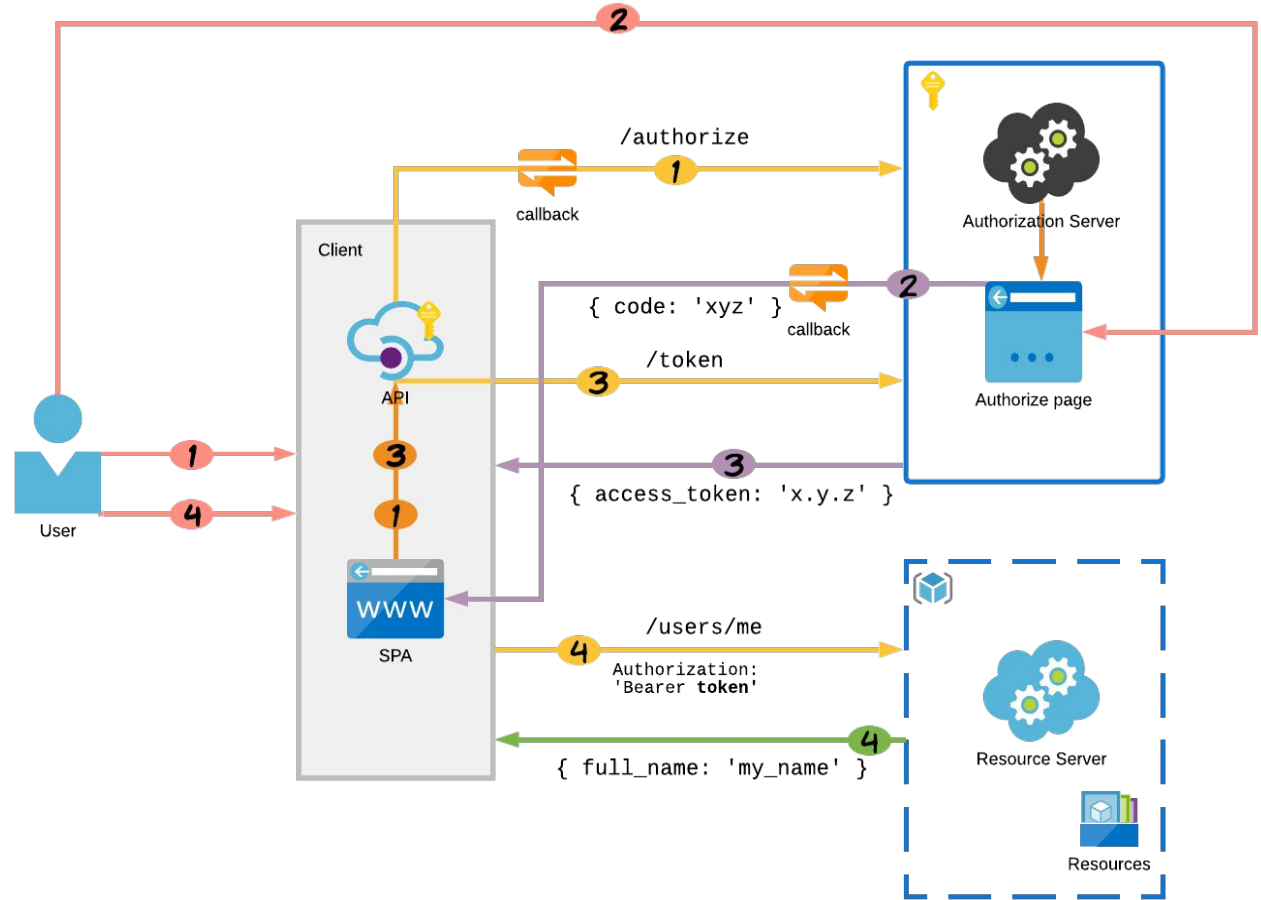


- ¿Cuál uso? -

Flujo de autorización por código

- Authorization Code Flow -

Diagrama de Flujo de Autorización por código



¿Quién usa
este flujo?



Get a Code

Query the `GET oauth/client_code` endpoint. The redirect URI must be the exact value you set in your app dashboard under the **Facebook Login > Settings Client > OAuth Settings** card.

```
curl -i -X GET "https://graph.facebook.com/{graph-api-version}/oauth/client_code?  
  client_id={app-id}&  
  client_secret={app-secret}&  
  redirect_uri={app-redirect-uri}&  
  access_token={long-lived-user-access-token}"
```

Sample Response

```
{  
  "code": "{code-for-your-client}"  
}
```

Nota: Aqui mandamos client_secret como query param.

- [Facebook access tokens](#) -

Código Frontend

- Client side -



<https://github.com/nicokurogane/oauth2-example-frontend>

Código Backend

- Server Side -



<https://github.com/AreViana/oauth2-example>

Ejemplo de Flujo de Autorización por código

Roles involucrados

1. Dueño del recurso: Usuario
2. Servidor de autorización: Microsoft - Azure Active Directory
3. Servidor de recursos: Graph Microsoft
4. Cliente: Backend + Frontend

Endpoints

- /azure/auth/authorization
- /azure/auth/token
- /azure/users/me

Ejemplo de Flujo de Autorización por código

¿Qué utilizamos?

1. Backend Node - [NestJs](#)
2. Npm - [simple-oauth2](#)
3. Servicios de [Microsoft](#) (Azure Active Directory , Graph API)
4. Frontend [ReactJs](#), [React Router](#),
5. Cliente HTTP [Axios](#)



Powered by  **Applaudo**Studios™