DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

REDEC UNAM

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS
DE INFORMACIÓN Y COMUNICACIÓN

DIPLOMADO

Desarrollo de sistemas con tecnología Java

Módulo 11

OAuth2.0

Mtro. Alfonso Gregorio Rivero Duarte

1. OAuth2.0

1.1 Introducción

Problema que OAuth2.0 Soluciona

Twitter App

Twitter User

TweetAnalyzer

Sitio web que analiza los datos de los tweets de los usuarios y genera métricas a partir de ellos.

Escenario: el usuario de Twitter desea utilizar un sitio web de terceros llamado TweetAnalyzer para obtener información sobre los datos de sus tweets presentes dentro de la aplicación de Twitter.

Con OAuth2: el usuario de Twitter debe compartir las credenciales de su cuenta de Twitter con el sitio web del analizador de tweets. Utilizando las credenciales de usuario, el sitio web TweetAnalyzer invocará las API de la aplicación Twitter para obtener los detalles del tweet y la publicación que genera un informe para el usuario final.

Pero tiene una desventaja mayor, el TweetAnalyzer puede cometer fraude y realizar otras operaciones en su nombre, como cambiar la contraseña, cambiar el correo electrónico, hacer un tweet rojo, etc.

Con OAuth2: el usuario de Twitter no tiene que compartir las credenciales de su cuenta de Twitter en el sitio web de TweetAnalyzer. En cambio, permitirá que la aplicación Twitter proporcione un token de acceso temporal a TweetAnalyzer con acceso limitado, como si solo pudiera leer los datos de los tweets.

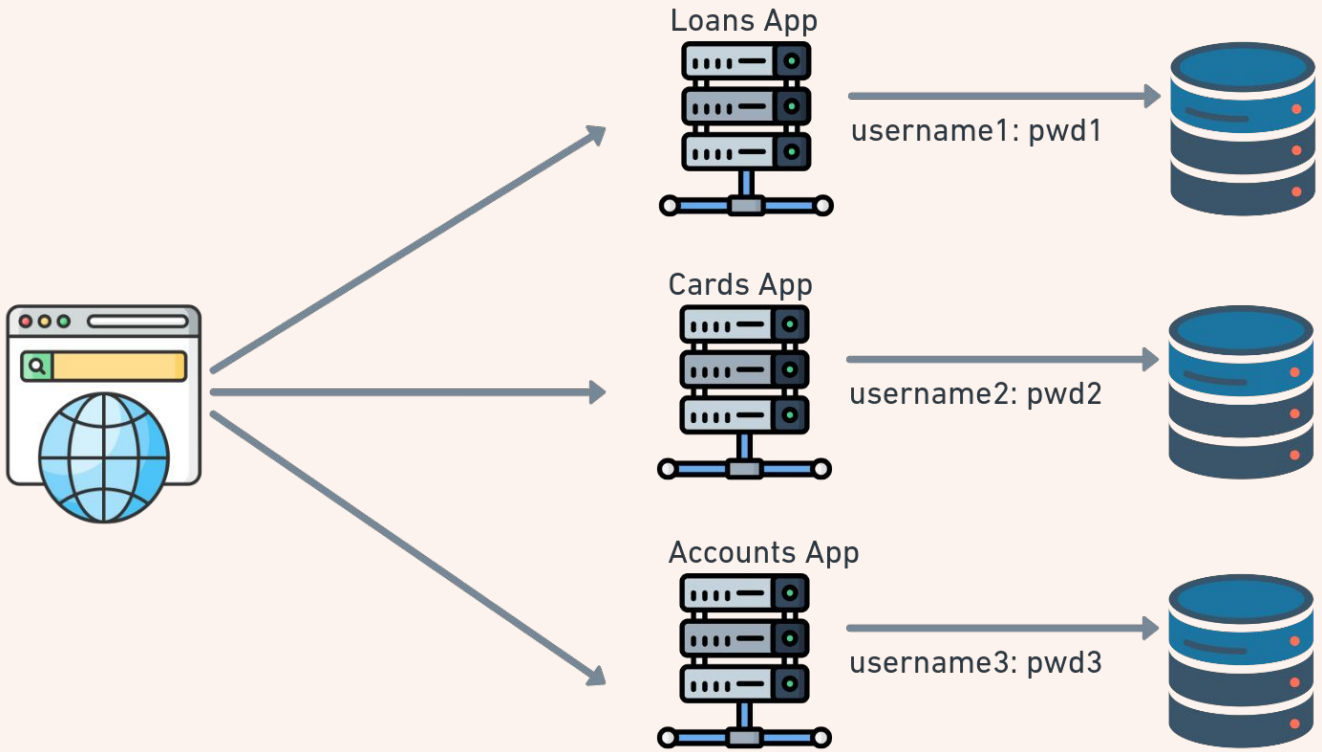
Con este enfoque, TweetAnalyzer sólo puede leer los datos de los tweets y no puede realizar ninguna otra operación.

Problema que OAuth2.0 Soluciona

How come Google let me use the same account in all it's products? Thought they are different websites / apps?

Well the answer is with the help of OAuth2. OAuth2 recommend to use a separate Auth server for Authentication & Authorization

Problema que OAuth2.0 Soluciona



Si un banco tiene varios sitios web que respaldan cuentas, préstamos, tarjetas, etc. Sin OAuth2, los clientes del banco deben registrarse y mantener diferentes perfiles de usuario en todos los sistemas.

Incluso la lógica de AuthN y AuthZ, los estándares de seguridad se duplicarán en todos los sitios web.

Cualquier cambio o mejora futura en materia de seguridad, autenticación, etc. debe realizarse en todos los lugares.

Problema que OAuth2.0 Soluciona

OAuth significa Autorización Abierta. Es un protocolo abierto y gratuito, basado en estándares IETF y licencias de Open Web Foundation.

OAuth2 es un estándar de seguridad en el que le otorga a una aplicación acceso a sus datos en otra aplicación. Los pasos para otorgar permiso o consentimiento a menudo se denominan autorización o incluso autorización delegada. Usted autoriza a una aplicación a acceder a sus datos o a utilizar funciones de otra aplicación en su nombre, sin darles su contraseña.

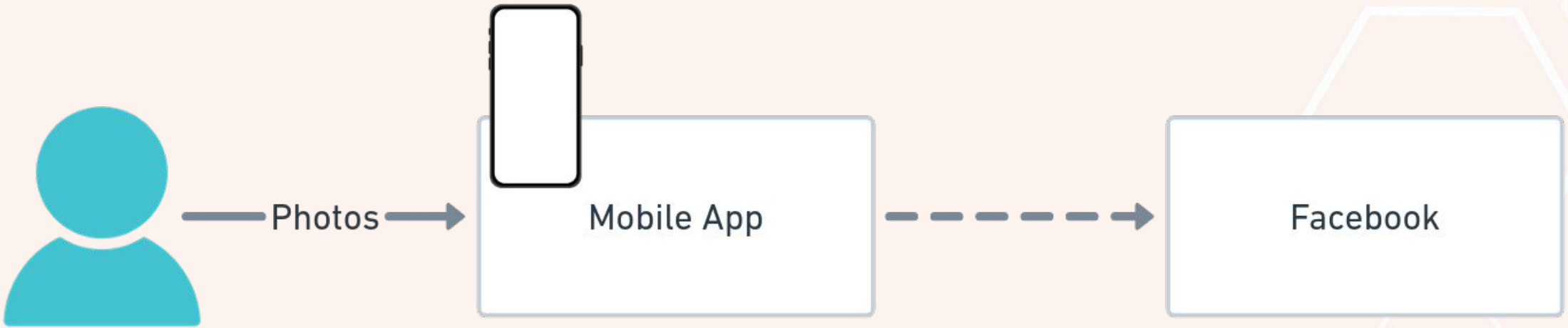
En muchos sentidos, puede considerar el token OAuth como una "tarjeta de acceso" en cualquier oficina/hotel. Estos tokens brindan acceso limitado a alguien, sin tener el control total en forma de clave maestra.

Problema que OAuth2.0 Soluciona

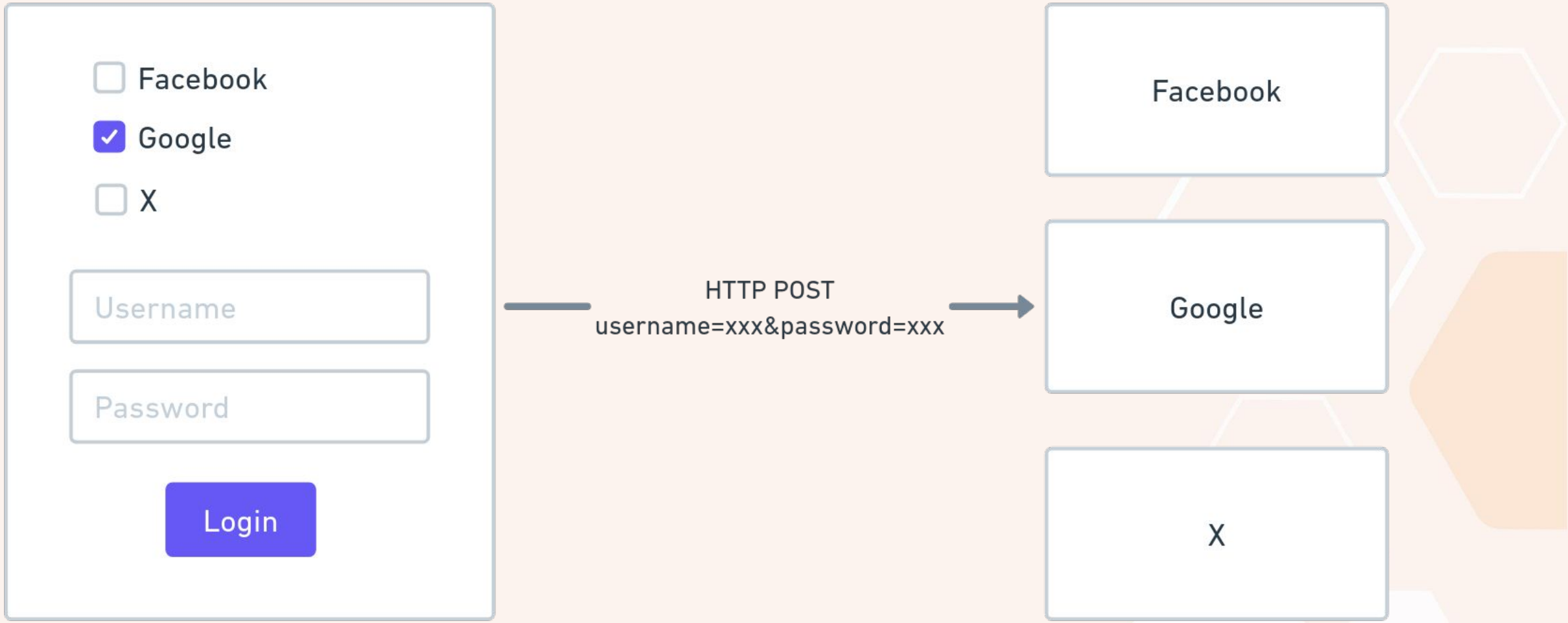
El marco OAuth especifica varios tipos de concesiones para diferentes casos de uso, así como un marco para crear nuevos tipos de concesiones.

- Authorization Code
- PKCE
- Client Credentials
- Device Code
- Refresh Token
- Implicit flow (legacy)
- Password grant (legacy)

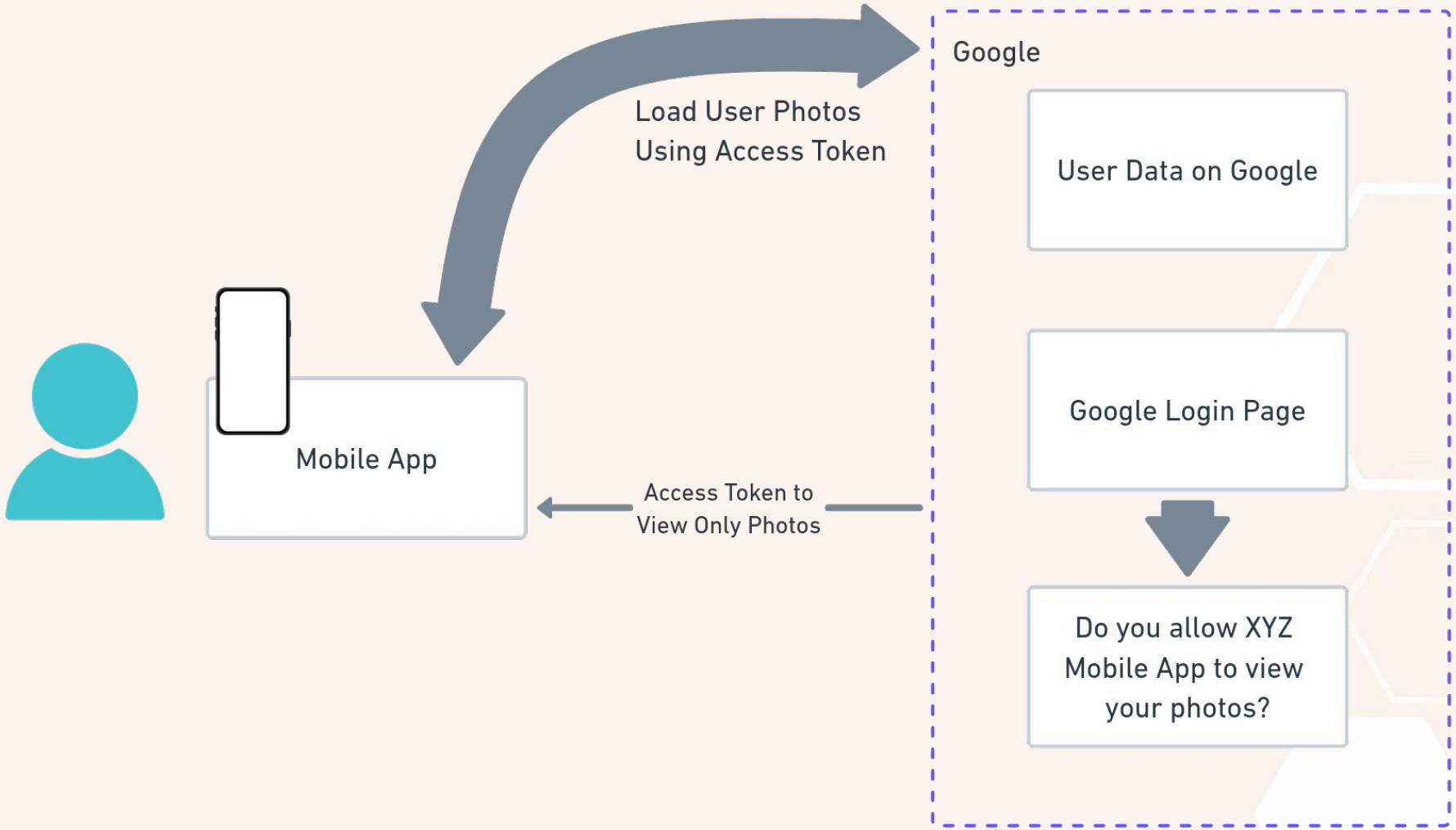
Problema que OAuth2.0 Soluciona



Problema que OAuth2.0 Soluciona



Problema que OAuth2.0 Soluciona



Introducción

OAuth2 es un framework de autorización, que permite a las aplicaciones obtener acceso (limitado) a las cuentas de usuario de determinados servicios, como Facebook, GitHub, Twitter, Steam, BitBucket, LinkedIn y muchos más.

Consiste en delegar la autenticación de usuario al servicio que gestiona las cuentas, de modo que sea éste quien otorgue el acceso para las aplicaciones de terceros.

OAuth2 provee un flujo de autorización para aplicaciones web, aplicaciones móviles e incluso programas de escritorio.

Roles presentes en OAuth2

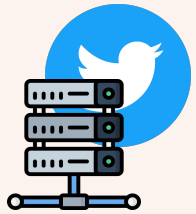
En el protocolo que define OAuth podemos identificar 4 roles.

- Client
- Resource Owner
- Resource Server
- Authorization Server

Resource Owner: eres tú, el usuario final. En el escenario de TweetAnaluzer, el usuario final que desea utilizar el sitio web de TweetAnalyzer para obtener información sobre estos tweets. En otras palabras, el usuario final es dueño de los recursos (Tweets), por eso lo llamamos Propietario del recurso.

Client: el sitio web de Twitter Analizer es el cliente aquí, ya que es el que interactúa con Twitter después de obtener los permisos del propietario del recurso / usuario final.

Authorization Server: Este es el servidor que conoce al propietario del recurso. En otras palabras, el propietario del recurso debe tener una cuenta en este servidor. En el escenario de TweetAnalyzer, el servidor de Twitter que tiene lógica de autorización actúa como servidor de autorización.



Resource Server: Éste es el servidor donde se alojan las API, servicios que el cliente desea consumir. En el escenario de TweetAnalyzer, el servidor de Twitter que tiene implementada la lógica API como /getTweets, etc. En organizaciones más pequeñas, un único servidor puede actuar como servidor de recursos y servidor de autenticación.

Scopes: Estos son los permisos granulares que el Cliente desea, como acceder a datos o realizar determinadas acciones. En el escenario de TweeterAnalyzer, el servidor de autenticación puede emitir un token de acceso al cliente con el alcance de solo LEER TWEETS.

Roles presentes en OAuth2

- **Client.-** Es la aplicación cliente que quiere acceder a la cuenta de un usuario, en un servicio determinado. A fin de conseguir ello, debe contar con una autorización del usuario, y esta autorización se debe validar (a través de la API del servicio).

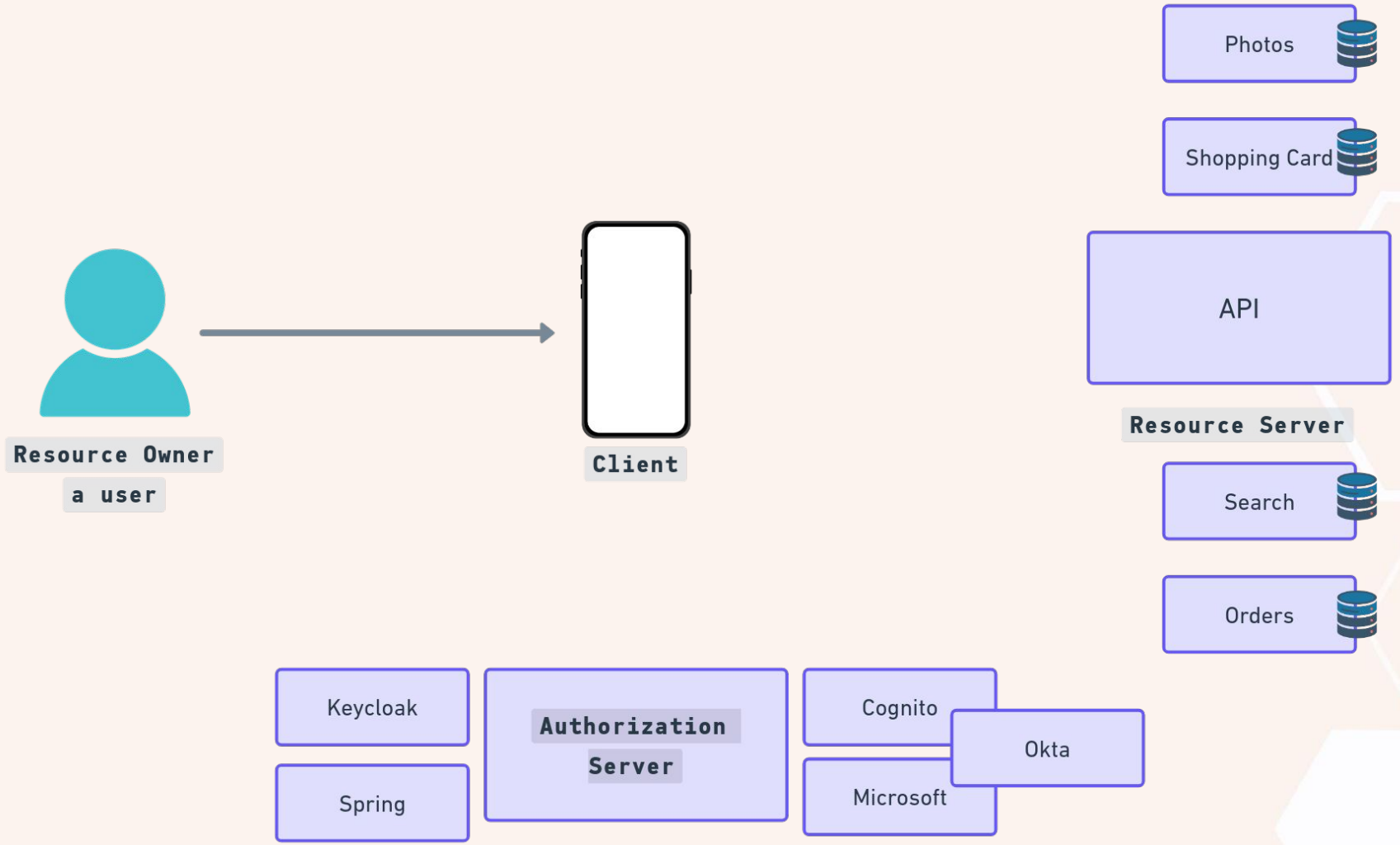
- **Resource Owner.-** El "dueño del recurso" es el usuario que autoriza a una aplicación, para que pueda acceder a su cuenta. El acceso está limitado en función del "scope" que concede el usuario durante la autorización.

Roles presentes en OAuth2

- **Resource & Authorization Server.-** Resource server es el servidor que almacena las cuentas de usuarios, y authorization server es el servidor que verifica la identidad de los usuarios y emite access tokens a la aplicación cliente.

Desde nuestro punto de vista como desarrolladores, la API del servicio representa muy bien a estos 2 últimos roles. Por tanto, podemos referirnos a ellos en conjunto, bajo el nombre de servicio o API.

Roles presentes en OAuth2



Flujo del protocolo OAuth

- El flujo descrito a continuación es un flujo genérico que representa al protocolo OAuth.
- Sin embargo, podemos encontrar diferencias en función al authorization grant type. Es decir, OAuth puede implementarse bajo diversas condiciones, y varía principalmente si se trata de una aplicación web, móvil o de escritorio.

1. La aplicación cliente solicita una autorización para acceder a los recursos de un usuario, en un servicio determinado.

Flujo del protocolo OAuth

2. Si el usuario autoriza esta solicitud, la aplicación recibe una authorization grant (concesión de autorización).

3. La aplicación cliente solicita un access token al authorization server, demostrando que es un cliente válido, y el permiso concedido anteriormente.

4. Si la identidad de la aplicación cliente se valida adecuadamente por el servicio, y la concesión de autorización es válida, el authorization server emite un access token a la aplicación cliente. Con esto la autorización se ha completado.

Flujo del protocolo OAuth

5. La aplicación cliente puede presentar el access token recibido en el paso anterior, y "solicitar un recurso" al resource server.

6. Si el access token es válido, el resource server hace entrega del recurso a la aplicación.

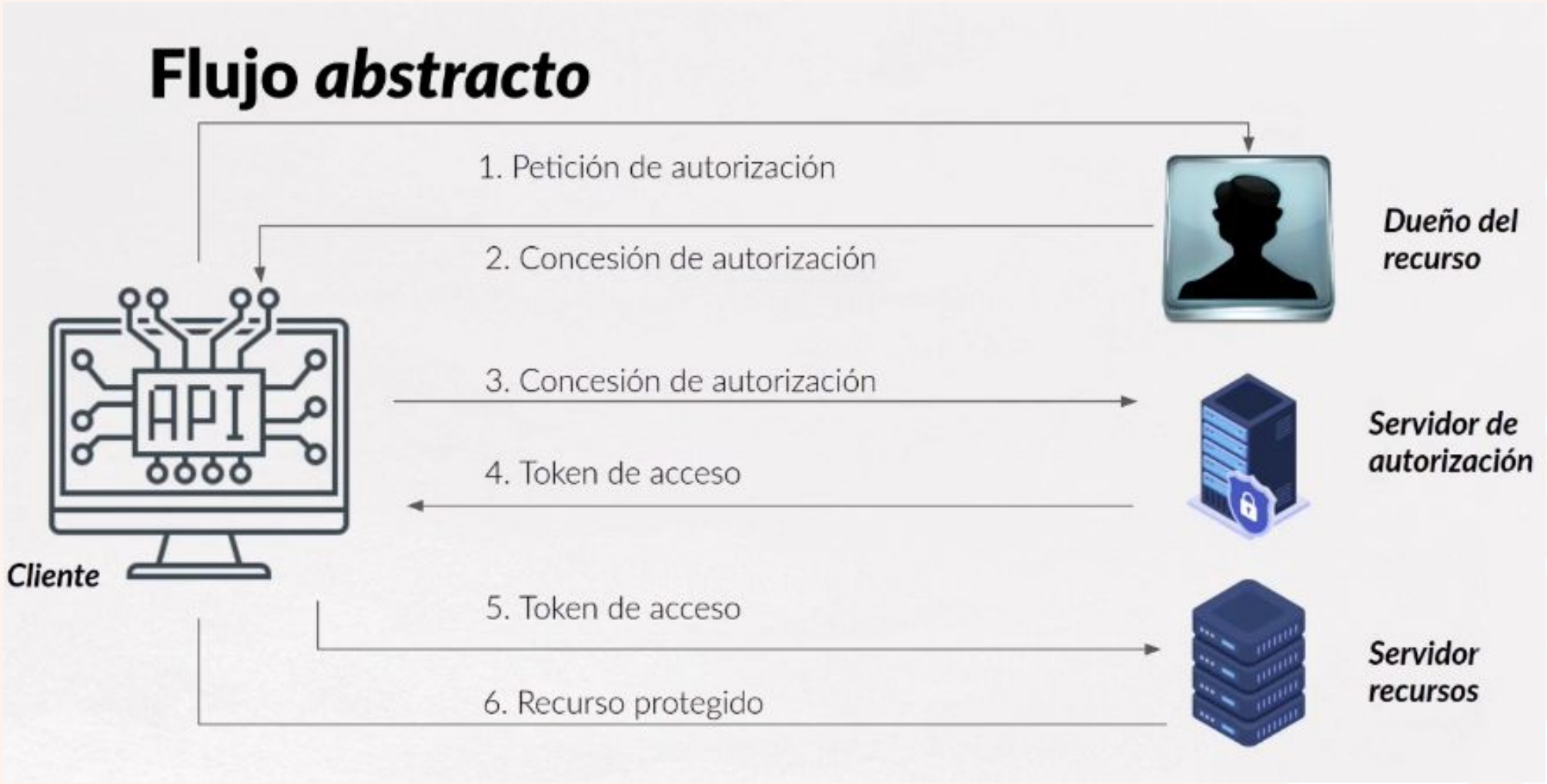
Flujo del protocolo OAuth

Es importante tener en cuenta que con "recurso" nos referimos generalmente a acceder a información del usuario, pero si la API del servicio lo permite (y el usuario nos lo ha autorizado), también es posible ejecutar acciones a nombre del usuario.

Nota:

Puedes volver a leer los pasos anteriores y considerar al authorization server y resource server como la API ofrecida del servicio, ya que generalmente cuando un servicio (como Facebook o Twitter) implementa un servidor de OAuth2, la API que ponen a nuestra disposición desempeña ambos roles.

Flujo del protocolo OAuth

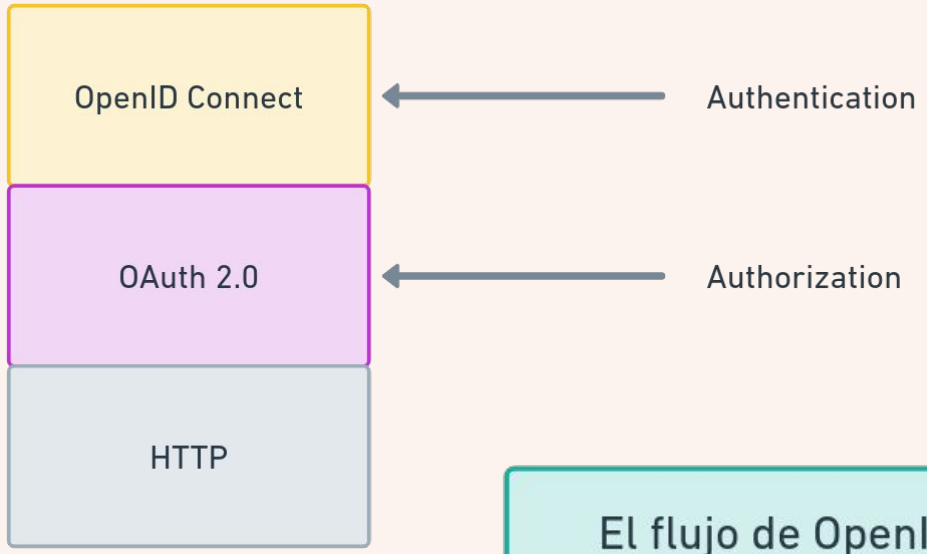


¿Qué es OpenID?

OpenID Connect es un protocolo que se encuentra en la parte superior del marco OAuth 2.0. Mientras que OAuth 2.0 proporciona autorización a través de un token de acceso que contiene ámbitos, OpenID Connect proporciona autenticación mediante la introducción de un nuevo token de identificación que contiene un nuevo conjunto de información y afirmaciones específicas para la identidad.

Con el token de identificación, OpenID Connect ofrece estándares para compartir detalles de identidad entre las aplicaciones.

¿Qué es OpenID?



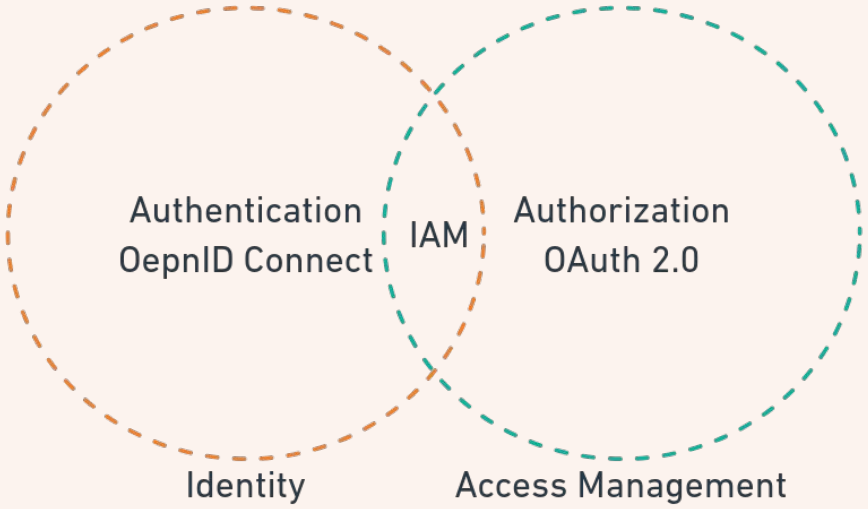
El flujo de OpenID Connect tiene el mismo aspecto que OAuth. Las únicas diferencias son que, en la solicitud inicial, se utiliza un alcance específico de openid y, en el intercambio final, el cliente recibe tanto un token de acceso como un token de identificación.

¿Por qué es importante OpenID Connect?

La identidad es la clave para cualquier aplicación. El núcleo de la autorización moderna es OAuth 2.0, que carece de un componente de autenticación. La implementación de OpenID Connect sobre OAuth 2.0 completa una estrategia IAM (Gestión de identidad y acceso).

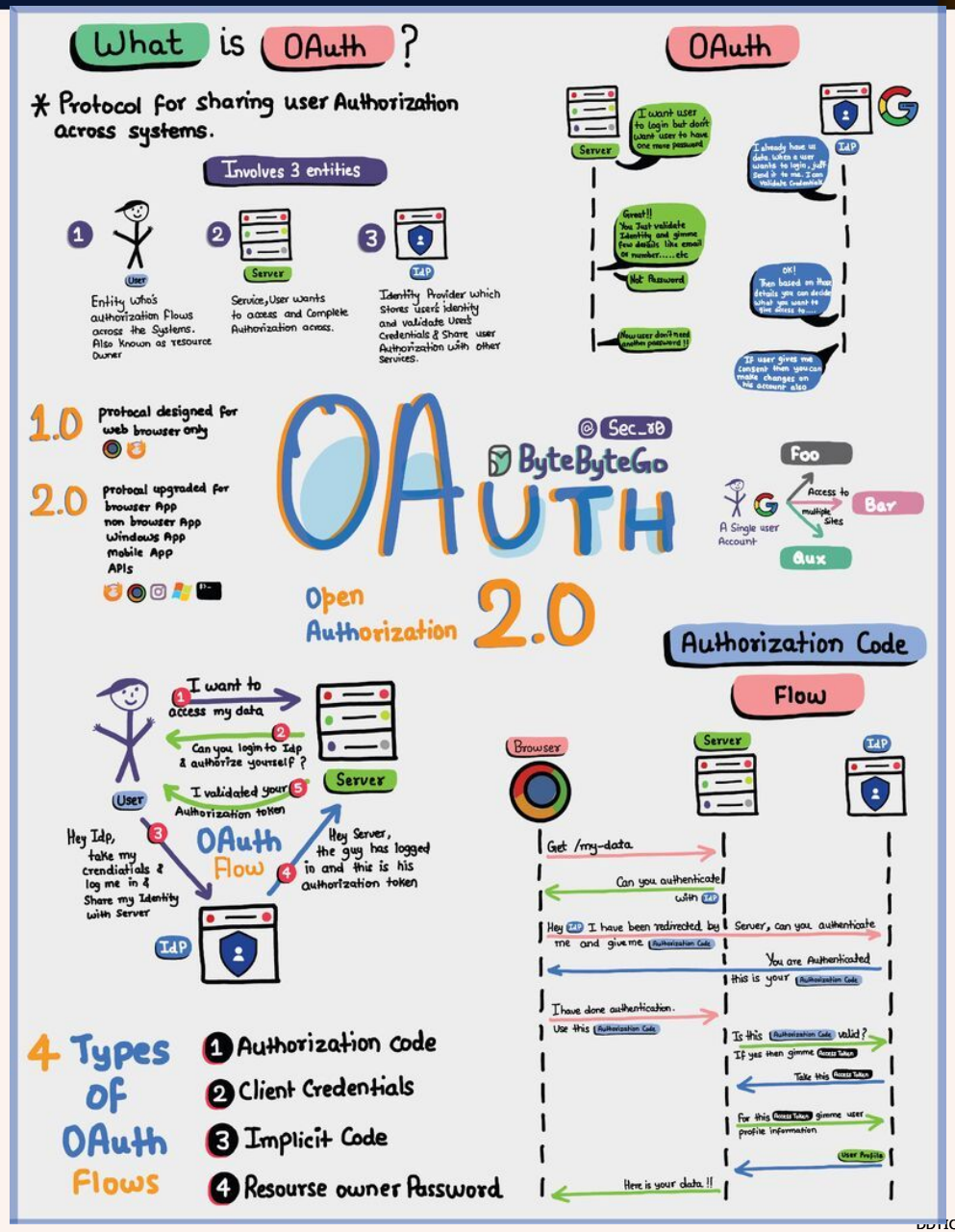
A medida que más y más aplicaciones necesitan conectarse entre sí y más identidades aparecen en Internet, también aumenta la demanda de poder compartir estas identidades. Con OpenID Connect, las aplicaciones pueden compartir las identidades de forma sencilla y estándar.

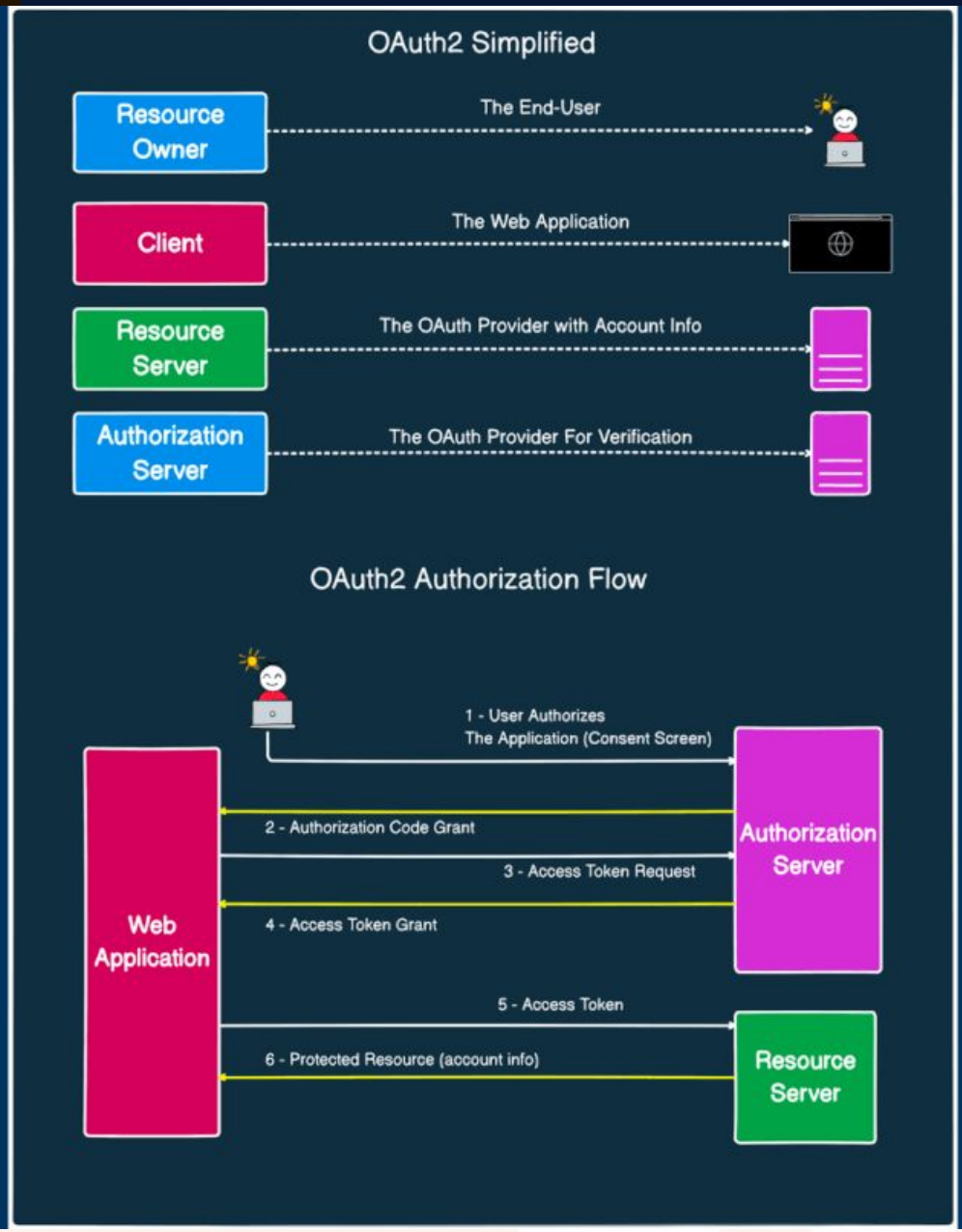
¿Por qué es importante OpenID Connect?

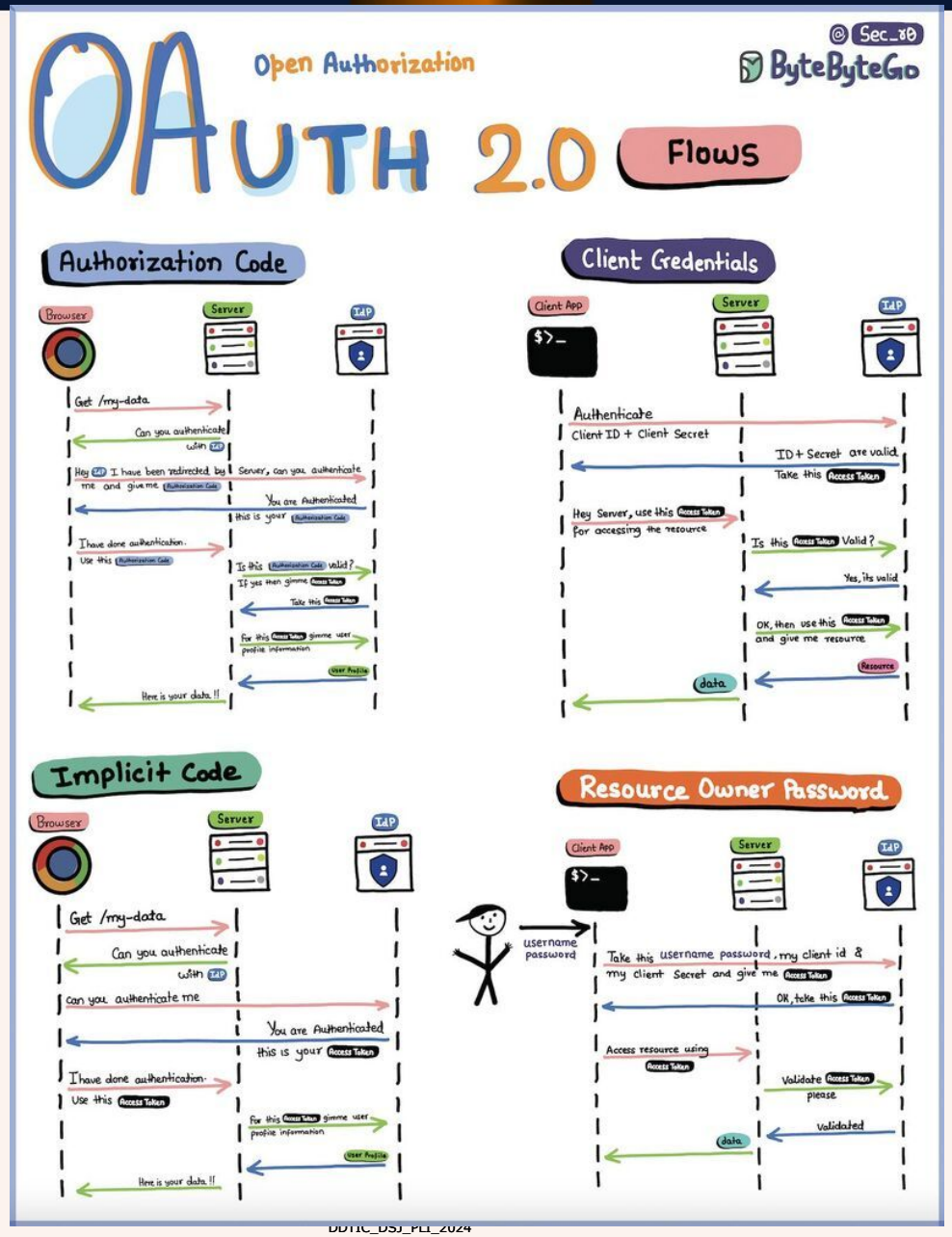


Open ID Connect agrega los siguientes detalles a OAuth 2.0

1. OIDC estandariza los alcances para openid, perfil, correo electrónico y dirección.
2. Token de identificación usando el estándar JWT.
3. OIDC expone el punto final estandarizado "/userinfo".


What is OAuth ?
* Protocol for sharing user Authorization across systems.
Involves 3 entities
1 User
Entity who's authorization flows across the Systems. Also known as resource Owner
2 Server
Service, User wants to access and Complete Authorization across.
3 IdP
Identity Provider which stores users identity and validate Users Credentials & Share user Authorization with other services.
OAuth
Server
I want user to Login but don't want user to have one more password
IdP
Great!! You Just validate Identity and gimme few details like email or number.....etc
Not Password
OK! Then based on those details you can decide what you want to give access to.....
Now user don't need another password !!
If user gives me consent then you can make changes on his account also
1.0 protocal designed for web browser only
2.0 protocal upgraded for
browser App
non browser App
Windows App
Mobile App
APIs
@Sec_t0
ByteByteGo
OAUTH
Open Authorization 2.0
Foo
Bar
Aux
Access to multiple Sites
A Single user Account
Authorization Code Flow
I want to access my data
Can you login to Idp & authorize yourself ?
I validated your Authorization token
Server
User
Hey Idp, take my crendiatials & log me in & Share my Identity with Server
OAuth Flow
Hey Server, the guy has logged in and this is his authorization token
IdP
Browser
Server
IdP
Get /my-data
Can you authenticate with IdP
Hey IdP I have been redirected by me and give me Authorization Code
Server, can you authenticate
You are Authenticated this is your Authorization Code
I have done authentication. Use this Authorization Code
Is this Authorization Code valid ?
If yes then gimme Access Token
Take this Access Token
For this Access Token gimme user profile information
User Profile
Here is your data !!
4 Types of OAuth Flows
1 Authorization code
2 Client Credentials
3 Implicit Code
4 Resourse owner Password


OAuth2 Simplified
Resource Owner
The End-User
Client
The Web Application
Resource Server
The OAuth Provider with Account Info
Authorization Server
The OAuth Provider For Verification
OAuth2 Authorization Flow
1 - User Authorizes The Application (Consent Screen)
2 - Authorization Code Grant
3 - Access Token Request
4 - Access Token Grant
5 - Access Token
6 - Protected Resource (account info)
Web Application
Authorization Server
Resource Server


OAuth 2.0
Open Authorization
Sec_80
ByteByteGo
Flows
Authorization Code
Browser
Server
IdP
Get /my-data
Can you authenticate with IdP
Hey IdP I have been redirected by me and give me Authorization Code
Server, can you authenticate
You are Authenticated this is your Authorization Code
I have done authentication. Use this Authorization Code
Is this Authorization Code valid?
If yes then gimme Access Token
Take this Access Token
For this Access Token gimme user profile information
User Profile
Here is your data !!
Client Credentials
Client App
$>_
Authenticate
Client ID + Client Secret
ID + Secret are valid
Take this Access Token
Hey Server, use this Access Token For accessing the resource
Is this Access Token Valid ?
Yes, its valid
OK, then use this Access Token and give me resource
Resource
data
Implicit Code
Browser
Server
IdP
Get /my-data
Can you authenticate with IdP
can you authenticate me
You are Authenticated this is your Access Token
I have done authentication. Use this Access Token
For this Access Token gimme user profile information
User Profile
Here is your data !!
Resource Owner Password
Client App
$>_
Server
IdP
username password
Take this username password, my client id & my client Secret and give me Access Token
OK, take this Access Token
Access resource using Access Token
Validate Access Token please
Validated
data

Contacto

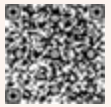
Mtro. Alfonso Gregorio Rivero Duarte

Senior Data Manager - CBRE

devil861109@gmail.com

Tels: (+52) 55 289970 69

Redes sociales:



https://www.linkedin.com/in/alfonso-gregorio-rivero-duarte-139a9225/