**OAuth2: Open Authorization 2.0 (autorización abierta)**

Protocolo estándar de autorización (no de autenticación) que permite que una aplicación cliente obtenga acceso limitado a los recursos protegidos de un usuario, que se encuentran en un servidor de recursos, sin necesidad de que el usuario comparta sus credenciales (usuario/contraseña) con la aplicación cliente.

**Roles principales en OAuth2**

* **Resource Owner (Propietario de recursos).** El **usuario** final que posee la información o recursos protegidos. Ejemplo: una **cuenta de Gmail o Discord**.
* **Client (Cliente).** La aplicación que **solicita acceso a los recursos** del usuario. Ejemplo: **Spotify**, que quiere acceder a la lista de contactos de Google.
* **Resource Server (Servidor de recursos).** **El sistema que aloja los datos o recursos protegidos**. Ejemplo: los **servidores de Google** que guardan los contactos, correos, entre otros.
* **Authorization Server (Servidor de autorización)**. El que **autentica al usuario y entrega al cliente un token de acceso con los permisos aprobados**. Muchas veces está integrado al Resource Server (ejemplo: **Google o Discord** actúan como ambos).

**Flujo con OAuth2**

El cliente (Spotify) pide al usuario (resource owner) permiso para acceder a cierto recurso (contactos de Google).

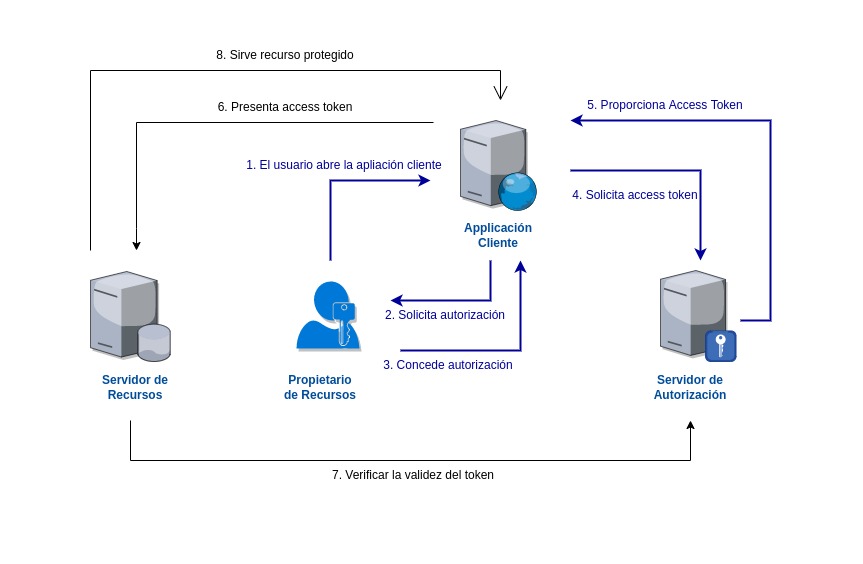
El usuario (resource owner) inicia sesión directamente en el authorization server (Google) y autoriza. Importante: el cliente (Spotify) nunca ve la contraseña del usuario (resource owner).

El authorization server (Google) entrega al cliente (Spotify) un Access Token (token de acceso).

El cliente (Spotify) usa ese token para hacer solicitudes al resource server (Google).

El resource server (Google) valida el token y si es válido, entrega solo los datos autorizados.





**En el caso de Discord y el bot a incluir en un servidor:**

**Resource Owner: la cuenta de Discord.**

**Client: el chatbot.**

**Authorization Server: Discord Authorization Server.**

**Resource Server: la API de Discord, que controla mensajes, roles, canales, etc.**

**Access Token: credencial temporal que le permite al bot (cliente) actuar en el servidor con los permisos autorizados.**

**🔹 Características clave**

**Seguridad: las credenciales del usuario nunca son compartidas con el cliente.**

**Delegación de permisos: el usuario controla qué puede hacer la aplicación cliente.**

**Granularidad: el acceso se limita a los scopes autorizados (ejemplo: “solo leer correo”, “solo enviar mensajes”).**

**Temporalidad: los tokens tienen vencimiento → reducen riesgo en caso de filtración.**

**👉 Resumiendo en una frase académica:**

**OAuth 2.0 es un protocolo de autorización que delega acceso limitado a recursos protegidos a través de tokens, separando la autenticación del usuario de la autorización al cliente.**

Es un estándar de autorización (no de autenticación), se encuentra diseñado para permitir que un sitio web o una aplicación accedan a recursos alojados por otras aplicaciones web en nombre de un usuario de manera segura, sin necesidad de compartir credenciales.

Es decir, le permite a un usuario conceder acceso limitado a sus recursos protegidos.

**Roles**

Dentro de OAuth 2.0 encontramos diferentes roles que van a participar en el proceso:

* Dueño del recurso (Owner).
* Cliente (Client).
* Servidor de recursos protegidos (Resource Server).
* Servidor de autorización (Authorization Server).

**Dueño del recurso**

El propietario del recurso es el usuario que da autorización a una determinada aplicación para acceder a su cuenta y poder hacer algunas cosas en su nombre.

El conjunto de cosas que puede hacer la aplicación en su nombre se denomina alcance, y podría ser, por ejemplo, solamente acceso de lectura y no poder crear ningún tipo de elemento de ningún nuevo recurso en su nombre.

**Cliente**

El cliente sería la aplicación que desea acceder a esa cuenta de usuario.

Antes de que pueda hacerlo debe ser autorizada por el usuario.

El cliente puede ser una aplicación web, una aplicación móvil, una aplicación de escritorio, una aplicación para smartTV, un dispositivo de IoT, otra API que consumiera de esta API, etcétera.

**Servidor de autorización**

El servidor de autorización es el responsable de **gestionar las peticiones de autorizació**n.

Verifica la identidad de los usuarios y emite una serie de**tokens de acceso** a la aplicación cliente.

**Servidor de recursos**

**El servidor de recursos será la API propiamente**, es decir, el servidor que aloja el recurso protegido al cual queremos acceder.

Puede que también forme parte de la misma aplicación que el propio servidor de autenticación.