	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte



**IES LOS SAUCES
BENAVENTE (ZAMORA)**



**TÍTULO GRADO SUPERIOR DE
DESARROLLO DE APLICACIONES WEB**

DEPARTAMENTO: INFORMÁTICA

TÍTULO PROYECTO: Sistemas de autenticación en servicios apiREST

AUTOR: Luis Mateo Rivera Uriarte

CURSO: 2019/2020


PERIODO: JUNIO

TIPO DE PROYECTO: Proyecto de investigación y desarrollo


CÓDIGO DE PROYECTO: I074

TUTORÍA COLECTIVA: Amor Rodríguez Navarro


TUTORÍA INDIVIDUAL: Heraclio Borbujo Moran

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

Sistemas de autenticación en servicios apiREST

	<p>CÓDIGO DEL PROYECTO: I074</p> <p>TÍTULO: Sistemas de autenticación en servicios apiREST</p> <p>AUTOR: Luis Mateo Rivera Uriarte</p>
---	---



	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

El presente proyecto de investigación y desarrollo cuyo título es “Sistemas de autenticación en servicios apiREST” del módulo PROYECTO ha sido realizado por Luis Mateo Rivera Uriarte, alumno del IES LOS SAUCES del ciclo Ciclo Formativo Grado Superior Desarrollo de aplicaciones web con el fin de la obtención del Título de Técnico Superior en Desarrollo de Aplicaciones Web.

La tutoría individual de dicho proyecto ha sido llevada a cabo por D./D^a. Heraclio Borbujo Moran, profesor de segundo curso de CFGS Desarrollo de Aplicaciones web.

En Benavente, 11 de mayo de 2020

Autor


Vº Bº
Tutoría colectiva

Vº Bº
Tutoría individual

Fdo. Luis Mateo
Rivera Uriarte

Fdo. Amor
Rodríguez Navarro

Fdo.: Heraclio
Borbujo Moran

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

1	Resumen del proyecto.....	5
2	Conceptualización del problema.....	5
3	Objetivos.....	6
4	Implementación.....	7
4.1	IDE y herramientas de desarrollo utilizados	7
4.2	Lenguajes de programación y lenguajes de marcas utilizados.	7
4.3	SGBD o sistema de almacenamiento utilizado	7
4.4	Repositorio de software	8
4.5	Modelo de casos de uso	8
4.6	Modelo físico de datos	8
4.7	Web services – servicios web.....	9
4.7.1	Sistemas de autenticación	10
4.7.2	Sistema de autorización OAuth.....	12
5	Conclusiones y líneas futuras.....	18
6	Referencias Bibliográficas	18
7	Anexos	18


1 Resumen del proyecto

El proyecto se basa en la investigación, aprendizaje, aplicación y presentación de los distintos métodos de autenticación y autorización que existen para controlar el acceso a un servicio apiREST (comunicación servidor-servidor).

Los tres métodos más utilizados son: Autenticación simple, que se resume en autenticar un nombre y una contraseña asociados en una base de datos, similar a una autenticación estándar en una página o aplicación.

Autenticación por token, que consiste en ofrecerle una clave a un cliente cuya función es usarse como sustituto de un nombre y una contraseña. Logrando así dar más seguridad a las llamadas a los servicios web pues una clave no hace ningún tipo de referencia al usuario/servidor que realiza la llamada ni a su clave de acceso original. Así, en caso de que un tercero intercepte las llamadas, puede hacerse con la clave y usarla pero no tendrá acceso a la información privada de la aplicación que la utiliza.

Autenticación por el modelo OAuth 2.0, que es el más complejo de los tres, de hecho, es un sistema de autorización que se puede implementar de diversas formas

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

dependiendo cual sea el uso que le demos al servicio web. Se basa en la oferta a los clientes de tokens de corta duración.

2 Conceptualización del problema.

Campo de estudio.

Los sistemas de seguridad son una de las mayores claves de la digitalización y no es menos en el uso y disfrute de los servicios web apiREST. Son servicios sencillos de usar y, por tanto, fáciles de romper. Por eso es necesario implementar medidas para evitar ataques de denegación de servicio, de suplantación de identidad y de robo de datos sensibles.


Origen.

Mi primer planteamiento del tema surgió mientras estudiaba los servicios web durante la etapa final del segundo año de DAW. Me resultó un tema muy interesante y con mucho potencial, por lo que aprovecho esta oportunidad para aumentar mi conocimiento sobre el tema. Dado que un servicio web no deja de ser un código que devuelve valores, decidí profundizar más en los sistemas de seguridad que los rodean ya que la seguridad informática es algo que apenas he cursado durante mis estudios.

Importancia.

Cómo dije anteriormente, la autenticación en los servicios apiREST es un tema de seguridad crucial en la red pues permite tener un control sobre los datos que ofrece tu aplicación al mundo. Sin este tipo de protocolos de seguridad. Ejemplos tan cotidianos cómo poder usar una cuenta de Google para autenticarse en una aplicación ajena a esta compañía no sería posible sin la existencia de los servicios web. Así mismo, sin un sistema de seguridad y control, no se podría monitorizar quién accede a los datos ofertados, con qué frecuencia los consulta ni desde donde ocurre esto, por lo que no se podrían evitar ataques de denegación de servicio de manera eficaz y justa.

Mi objetivo principal con este proyecto es entender cómo funcionan estos servicios de autenticación y autorización para comprender mejor el funcionamiento de internet cómo una gran red donde todo está conectado, y más concretamente, las aplicaciones web.

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

3 Objetivos

Busco comprender hasta estar capacitado para implantar distintos tipos de autenticación y autorización de acceso a servicios apiREST ofertados por diversas páginas web, entender más en profundidad el flujo de datos en la red entre distintas aplicaciones, aumentar mi grado de preparación en este ciclo formativo, describir y exponer de forma clara, concisa y amena toda la información de utilidad con respecto al tema a tratar a mis tutores para que se planteen este ámbito como un punto importante a trabajar durante la formación de los compañeros que vengan detrás de mí.

4 Implementación

4.1 IDE y herramientas de desarrollo utilizados


Para la realización de este proyecto utilizaré NetBeans como entorno de desarrollo debido a la práctica que he adquirido con él a lo largo del curso y la comodidad que ello supone.

Además, utilizaré PHPMyAdmin como herramienta para gestionar la base de datos que almacene la información necesaria para el funcionamiento de los servicios web debido a que es rápida, sencilla, fácil de usar y no requiere instalación en el lado del cliente para poder utilizarse, por lo que es perfecto para demostrar y explicar cómo funcionan las bases de datos en cualquier momento que se tenga acceso a la aplicación

4.2 Lenguajes de programación y lenguajes de marcas utilizados.

El lenguaje de programación a usar es PHP debido a que es el lenguaje de programación que tengo más fresco al haberle dedicado centenares de horas a lo largo del curso. Así mismo se trata de un lenguaje orientado a aplicaciones web por lo que es una de las mejores opciones para trabajar con servicios web.

Además, para poder mostrar resultados rápido y sin la necesidad de que la página recargue su información cada vez que se realiza alguna acción, utilizo JavaScript, con la librería de JQuery y con el objetivo de inplantar Ajax en las partes dinámicas de la aplicación.

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

Cómo apoyo usaré HTML+CSS para dar forma a la interfaz de la aplicación buscando un resultado sencillo pero efectivo y claro.

4.3 SGBD o sistema de almacenamiento utilizado

Respecto a la base de datos, se utilizará el sistema gestor de bases de datos MySQL debido a 2 motivos. La enorme popularidad que tiene este sistema y el dominio general que adquirí a lo largo del ciclo sobre el mismo.


Se busca hacer pequeñas tablas con apenas 5 o 10 registros por lo que no es necesario recurrir a gestores más potentes.

4.4 Repositorio de software

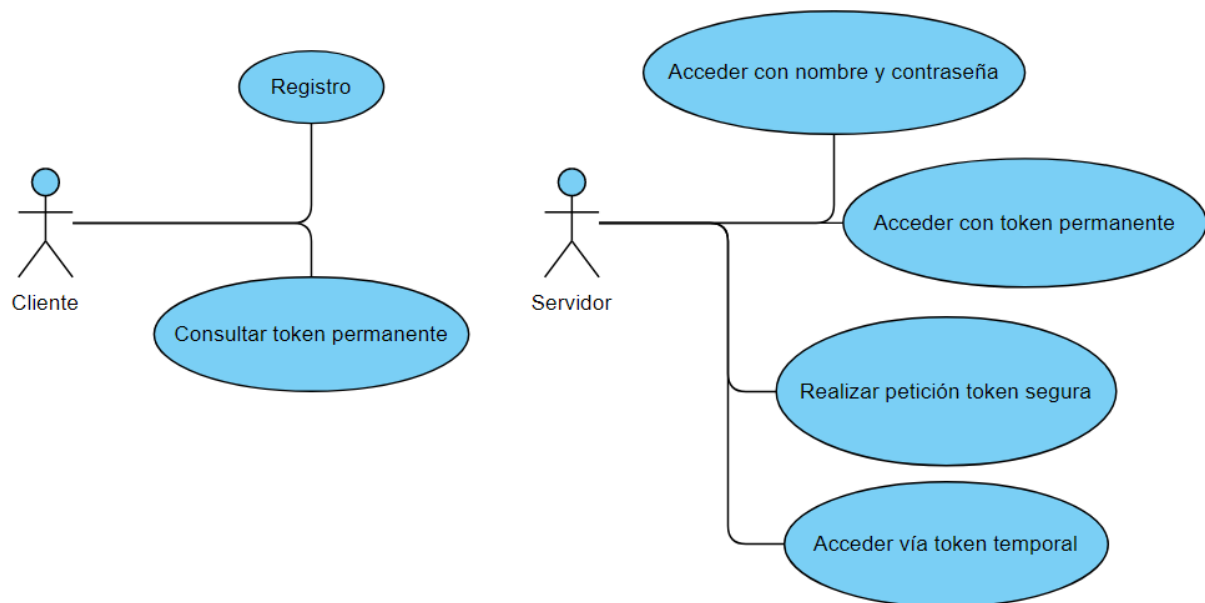
Como repositorio se utilizará GitHub ya que, al funcionar online y no requerir de una infraestructura en el servidor para poder usarse es perfecto para trabajar con él desde distintos entornos sin perder el trabajo.

Además, será accesible para todo el mundo se conecte desde donde se conecte. Por lo que sirve como expositor del código además de ser una herramienta de trabajo útil y necesaria.

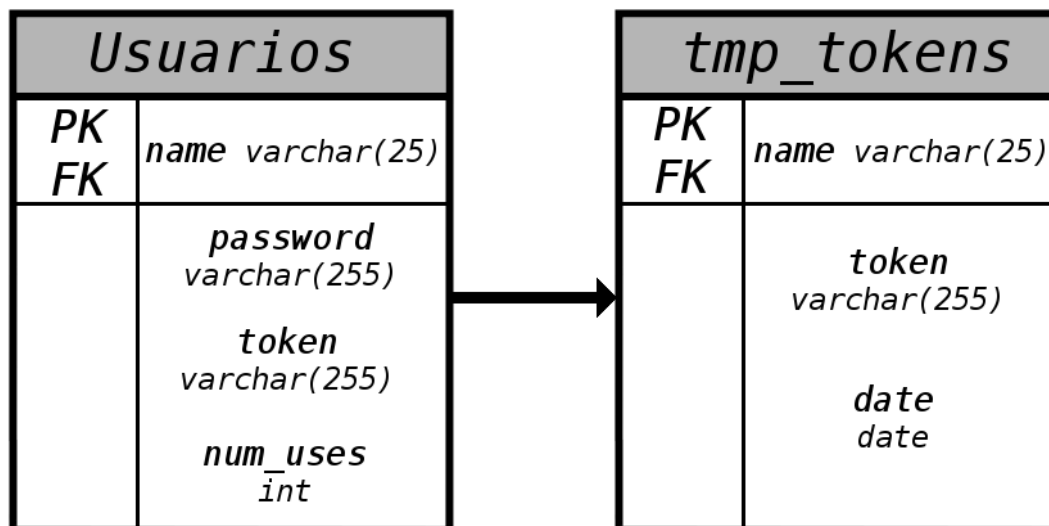
Actualmente el repositorio es accesible desde este [link](#).


	CÓDIGO DEL PROYECTO: 1074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

4.5 Modelo de casos de uso



4.6 Modelo físico de datos



	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

4.7 Web services – servicios web

Un servicio web (web service en inglés) es un programa accesible desde una red, similar a una aplicación web, diseñada para servir a aplicaciones en vez de a usuarios. Se trata de un programa al cual se llama mediante una URL, en caso de ser necesario, se le pasan unos parámetros por los métodos GET o POST, y devuelve una información en función de ello.

Un servicio web de uso común es, por ejemplo, en servicio que usa twitter. Se trata de un servicio que permite extraer información de su base de datos con la que crear cosas cómo mostrar, en una web ajena a twitter, un apartado con los últimos twitts que ha publicado un usuario. Otro ejemplo menos obvio es el servicio web de autenticación que ofrece Google. Un servicio que permite utilizar la información que tiene Google para facilitar el registro o el inicio de sesión en aplicaciones web externas a Google.


Los servicios web son el núcleo del proyecto, son la base sobre la qué implemento los distintos sistemas de autenticación y autorización. No importa lo sencillo o complejo que sea el servicio web ya que la autenticación es una capa extra que no afecta a su desempeño, por lo que se podría decir que “hecho uno, hechos todos”.

Un caso de prueba sencillo puede ser un servicio web al que se le pase un nombre y devuelva una cadena similar a “Buenas, [nombre]”. A este servicio se le puede implantar un sistema de autenticación que, en caso de ser incorrecto, en vez de devolver la cadena deseada, devuelva un mensaje de error tal que así: “El sistema no pudo obtener la información del servicio web”

Algunos sistemas de autenticación se pueden considerar en sí mismos servicios web, ya que, por ejemplo, un sistema de autenticación por tokens temporales puede funcionar de la siguiente manera:

- 1- Solicitas la información a un servicio web con una autenticación simple (nombre y contraseña)

- 2-El servicio, si la autenticación es correcta, te devuelve 2 parámetros. El primero es un token que hace las veces de llave y el segundo una url con la información que se desea en un principio con el servicio web.

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

3-Se llama a la nueva url utilizando el token como sistema de autenticación y se obtienen los datos esperados en un principio. Este token se genera cada vez que se llama al servicio web y generalmente es de un solo uso por lo que es realmente complicado atacar el servicio desde fuera.

4.7.1 Sistemas de autenticación

1-Autenticación básica

Este es el sistema de autenticación más básico, consiste en trasladar un nombre de usuario y una contraseña en la llamada de la apiREST, a poder ser a través del método POST para que sea algo más discreto.


El funcionamiento interno es muy sencillo, es prácticamente igual que un sistema de acceso a una aplicación. Se comprueba en una base de datos si existen el usuario con su contraseña asignada, se comparan y, si está bien, se devuelve la información del servicio web, y si no, se devuelve null o algún mensaje que explique la razón por la cual no se ha obtenido la información deseada.

El problema de este sistema de autenticación es que es relativamente sencillo de atacar. Es decir, de que un tercero también escuche la llamada y tenga acceso al nombre de usuario, la contraseña y la url del servicio web. Debido a esto, lo más aconsejable es formatear y encriptar el nombre y la contraseña para aumentar sobremanera la seguridad y no comprometer de forma descarada la información personal de los usuarios y servicios que utilizan el servicio web.

2-Autenticación por token permanente

Este método es más seguro que la autenticación básica ya que, si se produce un ataque, el pirata no podrá conocer la información privada de usuario o servicio al que ha robado.

Consiste en entregarle a un usuario registrado en una aplicación, un token asociado a su cuenta con el que puede autenticarse en el servicio web. El funcionamiento es muy sencillo.

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

Un usuario se registra en la aplicación que oferta el servicio web. Esta genera una clave que asocia al usuario y se la ofrece para que este la use. El usuario programa la llamada al servicio web de la aplicación y utiliza el token para autenticarse. El servicio web recibe la llamada y, en vez de comprobar nombre y usuario en la base de datos cómo en la autenticación básica, comprueba el token y con ello decide si devolver o no la información solicitada.

3-Autenticación por token temporal


En esencia, este sistema de autenticación es idéntico al anterior. Consiste en vincular la cuenta del usuario a un token por el cual autenticarse cada vez que realice una petición al servidor web. Con la única diferencia de que, en vez de ser permanente, este token caduca, ya sea por número de usos o tiempo.

Este sistema es más seguro que el anterior ya que, en caso de que intercepte un tercero un token y lo use a voluntad, este solo durará hasta que caduque. Por lo que los ataques y los robos, aun pudiendo seguir produciéndose, se ven truncados por la caducidad de los tokens.

El mayor problema de este sistema es que, al tener que renovar de forma periódica el token, es más tedioso de automatizar su uso. Por encima de ello, cuanto más se automatice ese proceso, más vulnerable es, haciendo que la opción ideal sea que el administrador del programa que utiliza un servicio web con este sistema de autenticación actualice manualmente el token cada vez que sea necesario.

Un uso común de este sistema de autenticación es usar tokens que caduquen tras un uso, pero eso implica tener que hacer dos peticiones cada vez que se utiliza el servicio web. Una con la autenticación que se utilice para pedir el token y otra utilizándolo. Esto lleva a 2 problemas: el primero y más obvio es que el desempeño de la aplicación se verá altamente afectado y el segundo está relacionado con la seguridad. Al tener que autenticarse de forma estándar cada vez que se hace una llamada, se está exponiendo la información del sistema cada vez que un usuario utiliza el servicio web en la aplicación.

Todos estos métodos se pueden ver desde la [aplicación del proyecto](#).

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

4.7.2 Sistema de autorización OAuth

¿Qué es?

OAuth es un framework que facilita la implantación de diversos métodos para dar seguridad a un servicio web. Este framework es un estándar en la web debido a su sencillez y la fiabilidad que ha demostrado merecer tras años de uso.


El proyecto OAuth nació en 2007 debido a la necesidad. La idea de crear un estándar abierto para delegar acceso a los servicios web nació de un pequeño grupo de desarrolladores tuvo la necesidad de implementar un sistema de acceso controlado para un servicio web de twitter. Al darse cuenta estos de que, si bien había ya distintos métodos más o menos implementados de forma generalizada, no existía un estándar propiamente dicho. Por lo que tras varios borradores y con la ayuda de “Eran Hammer-Lahav”, un trabajador experimentado de YAHOO!, se publicó el borrador definitivo de OAuth 1.0 el 3 de octubre de 2007.

Esta primera versión estaba basada en FAA (Flickr’s Authorization Api) y Google AuthSub, que eran plantillas para implementar sistemas de autenticación en los servicios web y los simplificó para crear una plantilla más sencilla y fácil de implementar.

La versión 1.0 se mantuvo en pie y se usó durante varios años hasta que en 2012 salió la versión 2.0, la versión actual. Las principales diferencias con su versión anterior son:

- 1-. La simplificación de la implementación referente a la comunicación, pasando del uso de librerías para cifrar distintos elementos en cada llamada a un sistema por tokens mucho más sencillo y claro.

- 2-. Mayor flexibilidad, pasando de un solo esquema de implantación a varios capaces de abarcar distintas situaciones. La forma más sencilla de reflejar esto es con un ejemplo. No es lo mismo crear un sistema de autenticación para un dispositivo como un PC que para un televisor. La versión 2.0 ofrece varias posibilidades según el uso que vaya a tener el servicio web.

	CÓDIGO DEL PROYECTO: 1074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

3-. Mejora considerable en la escalabilidad. La versión 1.0 utilizaba un sistema de tokens de un solo uso, por lo que las grandes empresas sufrían estragos para mantener actualizada toda la sobrecarga de **WIP**

4-. Se simplificó la comunicación pasando de utilizar una clave pública y una privada (encriptada y usada para “firmar” las peticiones en la clave pública) a un solo token que hace las veces de autenticador para el servicio web. Este método es menos seguro que el anterior pero mucho más simple.

5-. Se facilitó el desuso de los tokens. Es decir, en la versión 1.0 los tokens eran válidos por tiempo ilimitado o al menos por un año. En la versión 2.0 se generan tokens con una caducidad mucho más temprana y ofrece la posibilidad de generar nuevos tokens fácilmente para que los servidores puedan automatizar este proceso sin necesidad de involucrar a los clientes.

6-. Se separaron los roles de la autenticación. En la versión anterior, el servidor que pide autorización al usuario para usar sus credenciales y el que ofrece el servicio web eran el mismo. Ahora existe la posibilidad de tener un servidor que pida la autorización a los clientes y otra que ofrezca el servicio web usando esa autorización.


¿Cómo funciona?

El funcionamiento de OAuth se basa en la interacción de tres partes qué, a su vez, se dividen en subcategorías. Estas partes son los **actores**, los **tokens** y los **scopes**.

Los actores intervienen en todos los flujos de OAuth y se dividen en tres tipos.

El **propietario del recurso**, que es el administrador del servicio web y quien decide quién tiene acceso al servicio web, cuanto tiempo y a qué nivel.

El **servidor de recursos** o servicio web que se encarga de recibir las peticiones y responderlas o no en función de lo que le dictamine el propietario del recurso. Así mismo se encarga de garantizar el servicio de los clientes que dan su consentimiento de forma simple y automatizada.

	CÓDIGO DEL PROYECTO: I074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

La **aplicación** o cliente que quiere acceder y utilizar el servicio web. Se encarga de hacer de intermediaria entre el usuario, que da su consentimiento para utilizar un servicio web, y el propio servicio web.

Los scopes son una lista de identificadores separados por comas que sirven para conocer qué información se le solicita al servicio web y a qué tiene permitido el acceso ese cliente. Los scopes son las secciones del token que se utiliza para autenticar al cliente cada vez que se comunica con el servicio web. Actualmente se utilizan 4 tokens diferentes dentro del marco de OAuth:

El **client id** (identificador de cliente) es un identificador que representa a cada aplicación en el sistema de autenticación de un servicio web.

El **client secret** (secreto de cliente) es una clave que pertenece a cada aplicación y que utiliza el sistema de autenticación para comprobar si una aplicación tiene acceso o no a la información de un servicio web.


Los **access tokens** (tokens de acceso) son las claves con fecha de caducidad que ofrece el sistema de autenticación para que la aplicación se identifique al hacer una petición al servicio web.

Los **refresh tokens** (tokens de refresco) son claves que ofrece el sistema de autenticación y sirven para que la aplicación solicite nuevos Access tokens cuando los que utiliza caduquen.

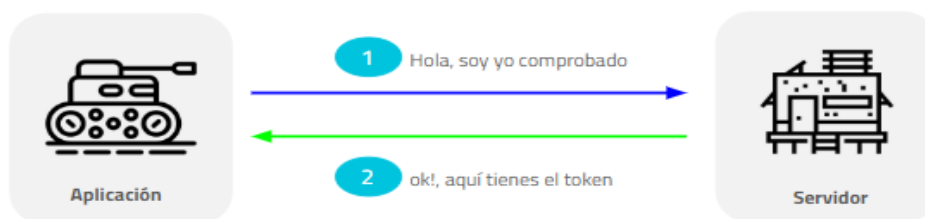
Los flujos o grant types hacen referencia al modo por el cual una aplicación tiene acceso a un access token para identificarse al utilizar un servicio web. La existencia de varios tipos de flujos se debe a la necesidad de tratar los distintos escenarios de negocio que se pueden dar al solicitar el uso de un servicio web.

Existen cuatro tipos de flujo y la implementación de uno u otro se escoge en función del tipo de aplicación consumidora, su grado de confianza y la infraestructura de la que dispone el usuario en el proceso.

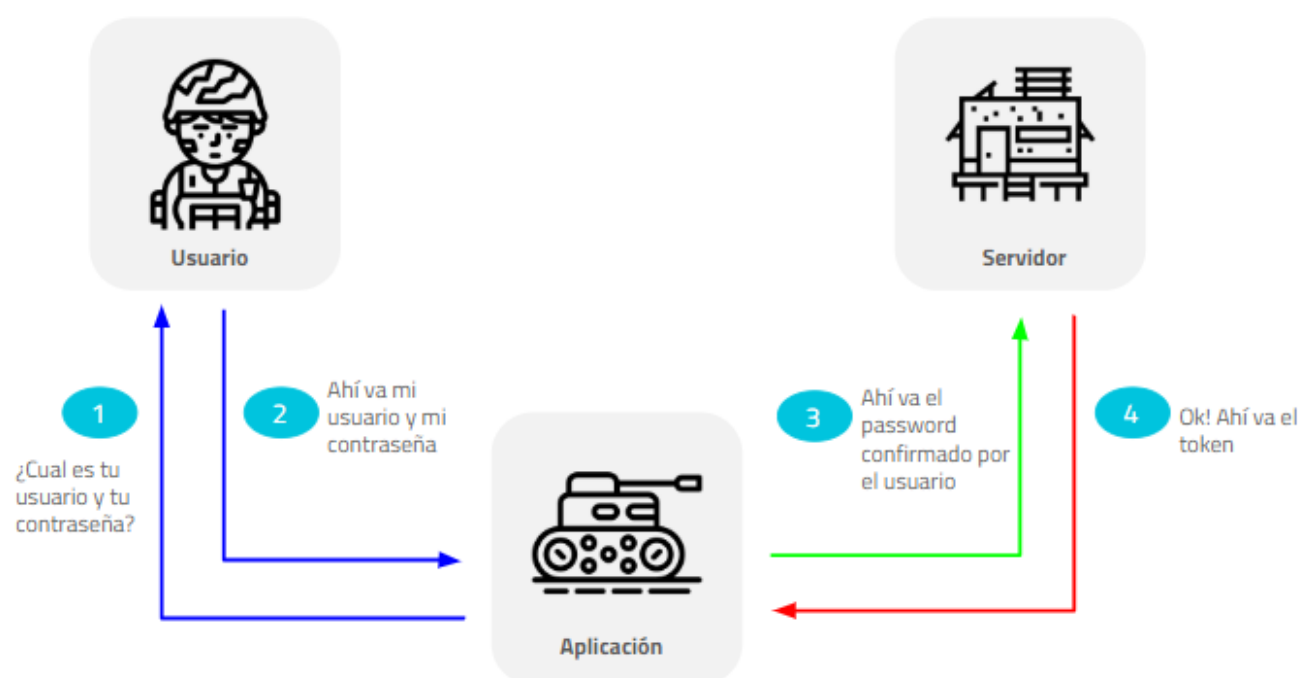
Client credentials grant type (tipo de flujo basado en las credenciales del cliente)
Este flujo se utiliza cuando la aplicación pertenece al usuario, cuando no hay necesidad


	<p style="text-align: right;">CÓDIGO DEL PROYECTO: 1074</p> <p>TÍTULO: Sistemas de autenticación en servicios apiREST</p> <p>AUTOR: Luis Mateo Rivera Uriarte</p>
---	--

de que el cliente se autentique de ninguna forma por lo que este ni siquiera llega a interactuar. Su funcionamiento es muy simple: la aplicación manda un token al servicio web y este comprueba que sea correcto para ver si le devuelve o no la información deseada.



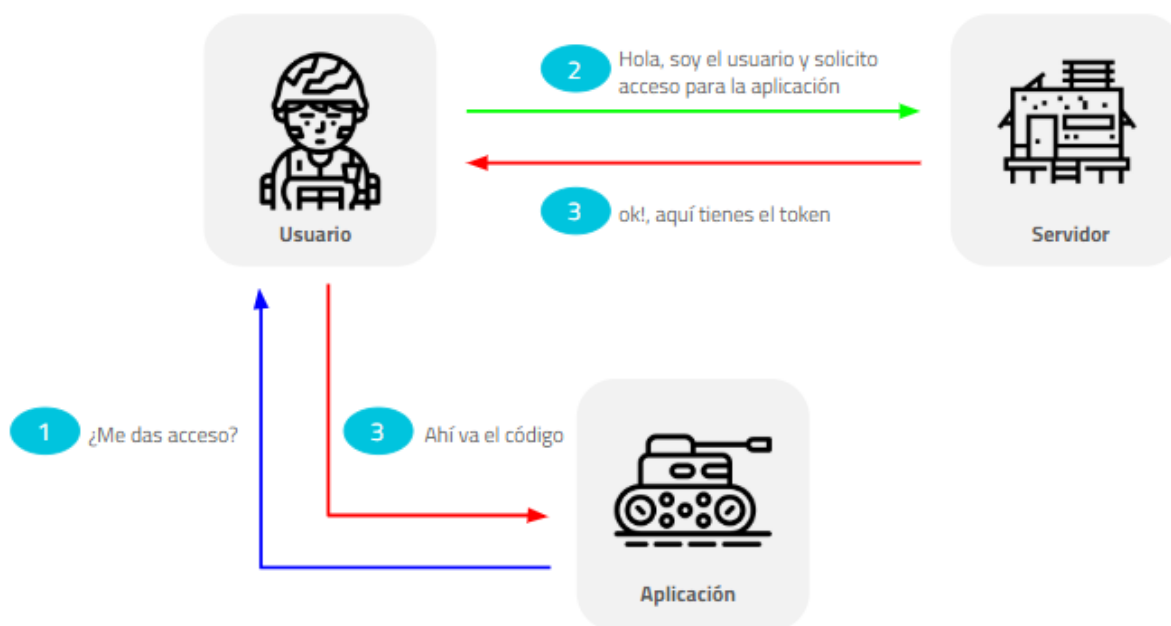
Resource owner password grant type (tipo de flujo basado en que el recurso es el propietario de la contraseña) Este flujo se caracteriza porque el cliente da sus credenciales a la aplicación en vez de al sistema de autenticación. Este flujo solo se debe utilizar en caso de que se tenga una máxima confianza en la aplicación, pues el cliente le brinda sus credenciales y no puede controlar que los utilice para fines ilícitos. Ejemplos de su uso son, por ejemplo, apis de autenticación, es decir, el servicio que ofrece Google a Spotify para que un usuario se registre con su cuenta de Google en la aplicación. Se da por sentado que Spotify es una aplicación de confianza donde el cliente puede poner sus credenciales para que esta le genere el Access token que le



	<p style="text-align: right;">CÓDIGO DEL PROYECTO: I074</p> <p>TÍTULO: Sistemas de autenticación en servicios apiREST</p> <p>AUTOR: Luis Mateo Rivera Uriarte</p>
---	--


permita identificarse en la aplicación. Este flujo es similar al anterior solo que ahora se añade la interacción del cliente: ahora la aplicación le solicita al cliente sus credenciales y esta las usa para identificarse en el sistema de autenticación del servicio web y así obtener un access token.

Implicit grant type (tipo de flujo implícito) Este método se basa en que la aplicación, en vez de pedirle las credenciales al cliente, le pide el Access token directamente, registrándose el cliente directamente en el sistema de autenticación del servicio web para obtener el access token y así dárselo a la aplicación. Este flujo es usado en aplicaciones que no son de confianza ya que no compromete las credenciales del cliente. Este tipo de comunicación se suele ver por parte del cliente cómo una ventana emergente donde se especifica a qué tipo de datos se está concediendo acceso acompañado de un botón de aceptar y otro de cancelar



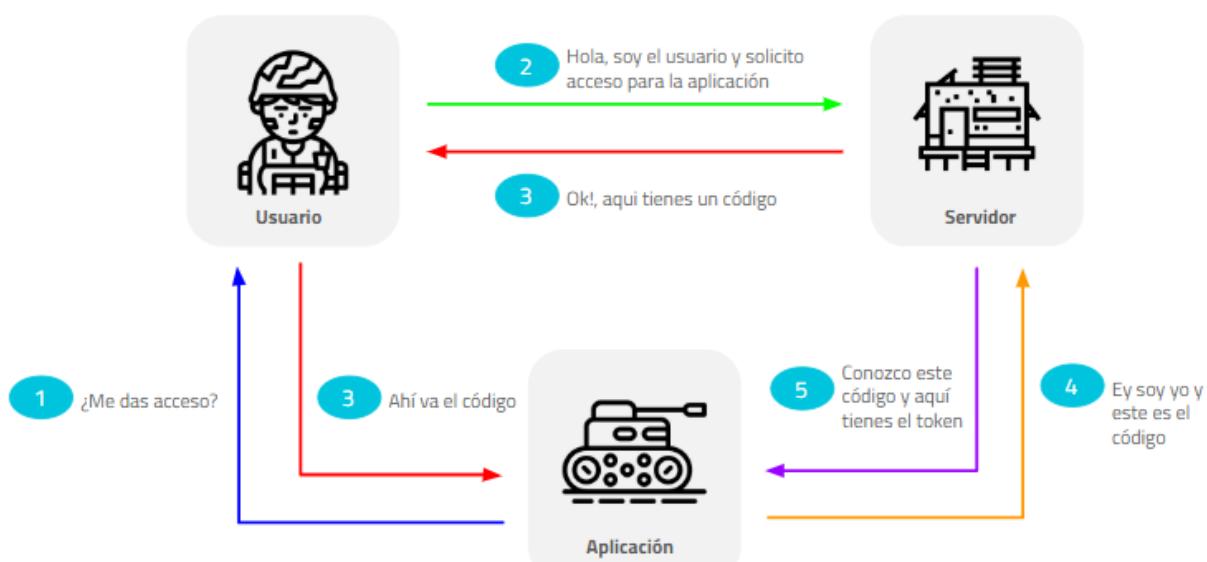
Authorization code grant type (tipo de flujo por autorización de código) Este es el flujo más complejo y seguro de todos. Consta de dos controles de acceso, necesita la intervención del cliente y es, en esencia, una versión más rebuscado que el anterior.


Su funcionamiento es el siguiente: La aplicación le solicita acceso al cliente, este se enfrenta al servidor y decide si quiere permitirse el uso de sus datos en la aplicación. Al aceptar, el servidor le da un código al cliente que llega a la aplicación y, con este

	<p>CÓDIGO DEL PROYECTO: 1074</p> <p>TÍTULO: Sistemas de autenticación en servicios apiREST</p> <p>AUTOR: Luis Mateo Rivera Uriarte</p>
---	---

código, la aplicación le pide un access token al servidor de autenticación. Si todo es correcto, el servidor le devuelve el Access token con el que identificarse de ahí en adelante.

Este es el método más extendido de todos debido a su gran fiabilidad y seguridad. Es sencillo de implementar, amigable con el cliente final y evita al servidor sufrir, entre otras cosas, ataques de denegación de servicio.



	CÓDIGO DEL PROYECTO: 1074
	TÍTULO: Sistemas de autenticación en servicios apiREST
	AUTOR: Luis Mateo Rivera Uriarte

5 Conclusiones y líneas futuras

Los sistemas de autorización de OAuth 2.0 se utilizan sobre todo para utilizar sistemas de registro en aplicaciones, es decir, el típico botón de “Entrar con google” o “Entrar con Facebook” y demás opciones que es común observar en grandes aplicaciones. Me hubiera gustado implementar esta función en mi aplicación pero por falta de tiempo esto me resulta imposible.

De todas formas, si he investigado sobre el tema y dejo aquí dos guías de cómo implementar OAuth 2.0 en aplicaciones de PHP. Una guía es un [blog](#) escrito en inglés donde se explica cómo funciona y cómo se prepara una aplicación para soportarlo y un [vídeo](#) que explica paso a paso en inglés cómo preparar una aplicación en PHP para que permita acceder con una cuenta de Google.

6 Referencias Bibliográficas

OAUTH: <https://www.paradigmigital.com/dev/oauth-2-0-equilibrio-y-usabilidad-en-la-securizacion-de-apis/>

<https://www.oauth.com/oauth2-servers/differences-between-oauth-1-2/>

<https://docs.apigee.com/api-platform/security/oauth/oauth-v2-policy-authorization-code-grant-type>

7 Anexos

Blog explicativo de cómo implementar OAuth 2.0 en una aplicación PHP:

<https://aaronparecki.com/oauth-2-simplified/>

Vídeo explicativo de cómo implementar OAuth 2.0 en una aplicación PHP con el objetivo de permitir el acceso a la misma con una cuenta de Google:

https://www.youtube.com/watch?v=hazMyK_cnz&t=