(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 694 423**

(51) Int. CI.:

G06F 21/41 (2013.01)
G06F 21/45 (2013.01)
G06F 21/62 (2013.01)
H04L 29/06 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **04.11.2011 PCT/EP2011/005569**

(87) Fecha y número de publicación internacional: **13.09.2012 WO12119620**

(96) Fecha de presentación y número de la solicitud europea: **04.11.2011 E 11801582 (5)**

(97) Fecha y número de publicación de la concesión europea: **12.09.2018 EP 2684151**

(54) Título: **Un método para proporcionar acceso autorizado a una aplicación de servicio con el fin de usar un recurso protegido de un usuario final**

(30) Prioridad:

08.03.2011 US 201161450376 P
14.03.2011 US 201161452262 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
20.12.2018

(73) Titular/es:

TELEFÓNICA S.A. (100.0%)
Gran Vía 28
28013 Madrid, ES

(72) Inventor/es:

LORENZO, JORGE;
LOZANO, DAVID;
GONZALEZ, DIEGO y
VICENTE, DAVID

(74) Agente/Representante:

ARIZTI ACHA, Monica

DESCRIPCIÓN

Un método para proporcionar acceso autorizado a una aplicación de servicio con el fin de usar un recurso protegido de un usuario final

Campo de la técnica

La presente invención se refiere, en general, a un método para proporcionar acceso autorizado a una aplicación de servicio con el fin de usar un recurso protegido de un usuario final, siendo dicho recurso protegido normalmente una API y estando expuesto por puntos terminales de una pluralidad de dominios administrativos, y realizándose dicho acceso autorizado por medio de un procedimiento OAuth, y más particularmente a un método que comprende usar una entidad intermedia para encaminar dicho procedimiento OAuth al dominio administrativo correspondiente, siendo dicho dominio administrativo el emisor final y controlador de autorizaciones de acceso, y proporcionar, dicho dominio administrativo a dicho usuario final, una ruta directa o de intermediario para acceder a dicha API.

Estado de la técnica anterior

En los últimos años, el mundo de Internet ha experimentado una explosión de API web/servicios web que abren las funcionalidades de los proveedores de servicios a otros sitios y, en muchos casos, a desarrolladores individuales, permitiéndoles así construir rápidamente nuevos servicios o enriquecer los ya existentes a través de la inclusión y combinación de funcionalidades remotamente expuestas. En la actualidad, ésta es una de las tendencias principales en Internet y se espera que siga creciendo y evolucionando, tratando progresivamente escenarios más sofisticados y dinámicos.

Abrir las API implica diferentes problemas que deben resolverse apropiadamente desde un punto de vista técnico para lograr resultados realmente útiles y adecuados. Uno de los problemas principales que deben resolverse es la seguridad y, especialmente, la privacidad del usuario final. Básicamente, las aplicaciones de Internet solo deben acceder a funcionalidades de proveedores de servicios después de haberse autenticado correctamente y después de haber recibido el consentimiento o autorización explícita del propietario de los recursos a los que va a accederse, sin necesidad de que el propietario de los recursos (normalmente un usuario final) comparta su identidad o credenciales con la aplicación. Como solución para este problema, ha emergido la nueva norma OAuth en los últimos años y ahora la comunidad de Internet y el IETF están trabajando en la segunda versión de esta norma.

OAuth proporciona todos los detalles para tratar satisfactoriamente escenarios estáticos en los que las aplicaciones conocen los puntos terminales de API antes de acceder realmente a las API y en los que todas las API pertenecen al mismo proveedor de servicios/dominio administrativo y, por tanto, el acceso puede autorizarse y controlarse desde un único punto central. Sin embargo, a medida que el número de API crece en Internet, hay nuevos escenarios en los que la misma API se expone por diferentes proveedores de servicios o la misma API se expone simplemente desde varias ubicaciones por el mismo proveedor de servicios, con el fin de mejorar la escalabilidad y el rendimiento de la estrategia de exposición de API.

Las principales tecnologías actuales existentes son el protocolo OAuth 1.0 [1] que está normalizado por el IETF y el protocolo OAuth 2.0 [2] que está en proceso de normalización por el IETF.

Tanto OAuth 1.0 como 2.0 definen un protocolo abierto para permitir una autorización de API segura en un método sencillo y convencional desde aplicaciones de escritorio y web, disponibles tanto para consumidores (clientes) confiables como no confiables. OAuth, tal como se especificó, puede aplicarse directamente para conceder acceso a recursos en servicios REST, pero también puede usarse por ejemplo en servicios web basados en SOAP.

El cliente, para acceder a recursos, en primer lugar debe obtener permiso del propietario de los recursos por medio del protocolo OAuth. Este permiso se expresa en forma de un testigo y, opcionalmente, un secreto compartido de coincidencia. El propósito del testigo es, tal como ya se explicó, hacer innecesario que el propietario de los recursos comparta sus credenciales con el cliente. A diferencia de los credenciales del propietario de los recursos, los testigos pueden emitirse con un alcance restringido y tiempo de vida limitado, y pueden revocarse de manera independiente.

En resumen, el propósito principal del protocolo OAuth es proporcionar los medios para que el consumidor obtenga un testigo de acceso válido.

Tal como se mostrará en la figura 1, hay dos testigos con papeles cruciales:

- En primer lugar, el testigo de petición se usa como referencia dentro de los procedimientos de autorización delegados. Más concretamente, según [1], los testigos de petición se usan por el consumidor para pedir al usuario que autorice el acceso a los recursos protegidos. A continuación, el testigo de petición de usuario autorizado, que se recomienda que tenga un tiempo de vida limitado, se intercambia por un testigo de acceso.

- Finalmente, este testigo de acceso se usa por el consumidor para acceder a las API en nombre del usuario, en lugar de usar las credenciales del usuario (usuario y contraseña). Los testigos de acceso pueden limitar el acceso a determinadas API o incluso a recursos dentro de una API dada.

En OAuth 2.0 desaparece el concepto de testigo de petición. Por tanto, el flujo comienza con la petición de autorización para el servicio. En esta petición, en lugar de incluir el testigo de petición, se envía la identificación del consumidor, tal como se mostrará en la figura 2. Hay varios escenarios de OAuth 2.0:

- Código de autorización, tal como se mostrará en la figura 2.
- Concesión implícita: el tipo de concesión implícita es adecuado para clientes que no pueden mantener sus credenciales de cliente confidenciales (para autenticarse con el servidor de autorización) tal como las aplicaciones de cliente que residen en un agente de usuario (por ejemplo, navegador)
- Credenciales de contraseña de propietario de recursos: el tipo de concesión de credenciales de contraseña de propietario de recursos es adecuado en los casos en que el propietario de recursos tiene una relación de confianza con el cliente, tal como su sistema operativo informático o una aplicación altamente privilegiada.
- Credenciales de cliente: el cliente puede solicitar un testigo de acceso usando solo sus credenciales de cliente cuando el cliente está solicitando acceso a los recursos protegidos bajo su control, o los de otro propietario de recursos que se ha dispuesto previamente con el servidor de autorización (cuyo método está más allá del alcance de la especificación OAuth 2.0).

OAuth básico es una norma ampliamente usada y hay muchas implementaciones, tales como las ofrecidas por BlueVia [4], Twitter [5], Google [6] o Yahoo [7]. OAuth 2.0 es aún un borrador, pero muchas implementaciones ya están disponibles o está previsto que ofrezcan OAuth 2.0; ejemplos son Facebook [8] y Google [9]. Las implementaciones existentes advierten acerca del hecho de que OAuth 2.0 es aún un borrador y por tanto puede tener variaciones hasta su normalización definitiva.

También hay varias soluciones que cubren el escenario en el que un servicio OAuth central proporciona OAuth a un proveedor de servicios que ofrece acceso a recursos protegidos. Éste es el caso de Google o Yahoo, que están proporcionando OAuth para que lo use cualquier servicio en Internet.

Adicionalmente y cubriendo el mismo escenario, Google también ha proporcionado una solución que consiste en la unión de OAuth y OpenID [10], [11]. Con esta solución, Google permite a aplicaciones y sitios web de terceros que dejen que sus visitantes se registren usando sus cuentas de usuario de Google. Tal como explican, esta extensión es útil para desarrolladores web que usan tanto OpenID como OAuth, particularmente porque simplifica el proceso para los usuarios solicitando su aprobación una vez en lugar de dos veces.

A diferencia de OAuth convencional, la extensión de OpenID y OAuth no prevé testigos de petición en una solicitud de servidor a servidor desde el consumidor combinado hasta el punto terminal de testigo de petición en el proveedor combinado. En lugar de ello, el proveedor combinado devuelve un testigo de petición ya aprobado al consumidor combinado como parte de la respuesta de autenticación OpenID.

El consumidor combinado entonces intercambia el testigo de petición por un testigo de acceso en el punto terminal de testigo de acceso del proveedor combinado, siguiendo la práctica de OAuth convencional.

El punto común de todas estas soluciones es que cubren el escenario en que hay un servicio OAuth central compartido por todos los proveedores de servicios. En otras palabras, la solución centralizada es responsable de emitir los testigos de acceso.

En otra observación, existen varias soluciones en las que el concepto de ‘testigo’ se usa con propósitos de encaminamiento. Por ejemplo, CISCO y Microsoft [12] tienen una solución que usa información de testigo de encaminamiento para redirigir las sesiones de cliente a servidores de terminales de Microsoft. Otros ejemplos son el uso de Microsoft de redirección de testigos para un agente de conexión [13] o la solución de redirección TransNexus para un intermediario de señalización NexTransitTM [14].

Sin embargo, ninguna de estas soluciones está relacionada con el protocolo OAuth, es decir: el testigo usado para el encaminamiento no es un testigo de acceso OAuth.

Con respecto a los mecanismos existentes para OAuth, tal como se comentó previamente, hay soluciones que cubren dos posibles escenarios:

- Un proveedor de servicios que ofrece un recurso protegido implementa OAuth para proporcionar acceso al recurso protegido. El propio proveedor de servicios ofrece el servicio OAuth. Este escenario coincide con un OAuth normal.
- Un servicio OAuth que proporciona OAuth para N proveedores de servicios que ofrecen sus recursos protegidos

de su propiedad. La autenticación y emisión de testigos se realiza mediante el servicio OAuth central. Por tanto, las credenciales de usuario son credenciales centrales proporcionadas por el servicio central de OAuth. Éste es el servicio ofrecido por Yahoo y Google, por ejemplo.

Además, Google ofrece una combinación de OAuth con OpenID, para un uso más avanzado y más flexible del segundo escenario.

En resumen, en soluciones existentes o bien el proveedor de servicios ofrece el punto terminal de autorización o bien hay un punto terminal de autorización central que autentica y emite los testigos de acceso para varios proveedores de servicios.

Descripción de la invención

Existe la necesidad de contar con escenarios distribuidos en los que los puntos terminales de API no se conocen de antemano por las aplicaciones y en los que no hay ni una sola entidad que pueda usarse como controlador centralizado para emitir autorizaciones y controlar su uso posterior. Con estos requisitos en mente, el objetivo de la invención propuesta en este documento es proporcionar una solución detallada para la aplicación de OAuth para escenarios de exposición de API altamente distribuidos y potencialmente en evolución, sin tener que modificar la norma OAuth en modo alguno. Para realizar esto, en lugar de modificar las interfaces, la invención se centra en cómo resolver el problema dentro de la propia infraestructura de exposición de API.

El siguiente escenario, cubierto por esta invención, no se trata correctamente por las soluciones existentes:

- Varios proveedores de servicios que ofrecen el mismo recurso protegido (es decir, API). Esto normalmente se produce en entornos de compañías de telecomunicaciones, en los que varios operadores ofrecen un recurso protegido, tal como las operaciones de SMS.
- Como el recurso protegido es el mismo, una aplicación que desea acceder al recurso protegido implementa OAuth y espera que esa implementación funcione con todos los proveedores de servicios. La aplicación puede esperar esto incluso sin saber si el usuario final pertenece a un proveedor de servicios o al otro.
- El punto clave es que todos los proveedores de servicios puedan ofrecer un servicio OAuth compartido centralizado, pero cada proveedor de servicios tiene que ser responsable de la autenticación de usuario y de la emisión de testigos de acceso. Un usuario pertenece a un determinado proveedor de servicios, y las credenciales de usuario no pueden conocerse por un servicio OAuth centralizado.

La presente invención proporciona un método para proporcionar acceso autorizado a una aplicación de servicio con el fin de usar un recurso protegido de un usuario final, estando expuesto dicho recurso protegido por puntos terminales de una pluralidad de dominios administrativos, siendo responsable cada dominio administrativo dentro de dicha pluralidad de dominios administrativos de emitir y controlar el uso posterior de autorizaciones para acceder a múltiples recursos protegidos según hayan sido concedidos por usuarios finales que pertenecen a dicho dominio administrativo. A diferencia de las propuestas conocidas, dichos puntos terminales no se conocen previamente por dicha aplicación de servicio y no hay un controlador/emisor de autorizaciones centralizado y el método de la invención, de una manera característica, comprende además

- usar una entidad intermedia o entidad global para:

  a) seleccionar un dominio administrativo de dicha pluralidad de dominios administrativos basándose en criterios flexibles que incluyen al menos la identidad de dicho usuario final; y
  b) realizar, dicho dominio administrativo seleccionado, una autorización segura para conceder acceso a dicho usuario final por medio de un protocolo abierto;

- proporcionar, dicho dominio administrativo seleccionado a dicho usuario final, una vez realizada dicha autorización segura, acceso directo o de intermediario a un punto terminal incluido en el mismo y a dicho recurso protegido, contenido en dicho punto terminal, a través de dicha aplicación de servicio.

Otras realizaciones del método del primer aspecto de la invención se describen según las reivindicaciones adjuntas 2 a 19, y en una sección posterior relativa a la descripción detallada de varias realizaciones.

Breve descripción de los dibujos

Las anteriores y otras ventajas y características se entenderán más completamente a partir de la siguiente descripción detallada de realizaciones, con referencia al dibujo adjunto, que debe considerarse de una manera ilustrativa y no limitativa, en las que:

La figura 1 muestra interacciones actuales seguidas por OAuth 1.0.

La figura 2 muestra interacciones actuales seguidas por OAuth 2.0.
La figura 3 muestra una representación gráfica de escenarios de exposición de API altamente distribuidos y potencialmente en evolución.
La figura 4 muestra el encaminamiento del procedimiento OAuth según una realización de la invención.
La figura 5 muestra el encaminamiento de las peticiones de API según una realización de la invención.
La figura 6 muestra la secuencia para conseguir un testigo de acceso con OAuth 1.0 cuando los testigos de petición pueden configurarse directamente mediante la infraestructura de servicio global y todos los dominios administrativos comparten un portal web común, según una realización de la invención.
La figura 7 muestra la secuencia para conseguir un testigo de acceso con OAuth 1.0 cuando los testigos de petición se generan por la infraestructura local bajo demanda y todos los dominios administrativos comparten un portal web común, según una realización de la invención.
La figura 8 muestra la secuencia para conseguir un testigo de acceso con OAuth 2.0, que no requiere un testigo de petición, y todos los dominios administrativos comparten un portal web común, según una realización de la invención.
La figura 9 muestra la secuencia para obtener el testigo de acceso con OAuth 1.0 cuando los testigos de petición pueden configurarse directamente mediante la infraestructura de servicio global y cada dominio administrativo tiene su propio portal web local, según una realización de la invención.
La figura 10 muestra la secuencia para conseguir el testigo de acceso con OAuth 1.0 cuando los testigos de petición se generan mediante la infraestructura local bajo demanda y cada dominio administrativo su propio portal web local, según una realización de la invención.
La figura 11 muestra la secuencia para conseguir un testigo de acceso con OAuth 2.0, que no requiere un testigo de petición, y cada dominio administrativo tiene su propio portal web local, según una realización de la invención.
La figura 12 muestra el diagrama de secuencia cuando se consume un recurso protegido tanto en modo de intermediario como en modo de redirección, según una realización de la invención.
La figura 13 muestra el diagrama de secuencia cuando se revoca un testigo de acceso, según una realización de la invención.

Descripción detallada de varias realizaciones

La presente invención proporciona una solución para aplicar OAuth 1.0 y/o 2.0 a escenarios de exposición de API altamente distribuidos y potencialmente en evolución. Según las representaciones gráficas de tales escenarios, tal como se muestra en la figura 3, deben tomarse en consideración los siguientes requisitos:

- Los puntos terminales de API (es decir, los servicios o recursos que se exponen) son compatibles con OAuth 1.0 o OAuth 2.0 y no tienen ningún conocimiento unos de otros ni ninguna infraestructura global.
- Cada punto terminal de API pertenece a un dominio administrativo que es responsable de controlar localmente el acceso al punto terminal de API. Un dominio administrativo normalmente corresponde a un proveedor de servicios, pero un proveedor de servicios puede tener múltiples dominios administrativos.
- Las aplicaciones externas no tienen ningún conocimiento previo de los puntos terminales de API o el dominio administrativo que tiene que usar para cada usuario final.
- Los nodos de exposición de API y los dominios administrativos pueden cambiar con el tiempo.

En primer lugar, puesto que las aplicaciones no tienen ningún conocimiento previo de los puntos terminales ni de los dominios administrativos con los que tienen que ponerse en contacto, es necesario que haya URL estáticos en los que las aplicaciones puedan inicialmente activar el proceso OAuth, por tanto, los puntos terminales de protocolo OAuth (URL) se mantienen únicos y válidos para conseguir autorizaciones para acceder a cualquiera de los puntos terminales de API subyacentes. Adicionalmente, puesto que las aplicaciones deben autenticarse durante el proceso, debe haber una “infraestructura global” que soporte el registro de aplicaciones y la gestión de credenciales. Esto proporciona un mecanismo entendible para que los consumidores externos se registren y activen el acceso a API, sin tener que afrontar los detalles de la arquitectura distribuida subyacente. La infraestructura global conoce estos detalles y los oculta a aplicaciones externas. Está fuera del alcance de la presente invención describir cómo la infraestructura global conoce los dominios administrativos locales y/o puntos terminales de API, pero en principio hay diferentes alternativas (configuración estática, procedimientos de registro dinámico, etc.). En cualquier caso, es muy importante destacar que la infraestructura global no se usa como intermediario centralizado para controlar más tarde el uso de API; solo se usa para soportar los dos conceptos principales de la solución tal como se explica a continuación:

- Encaminar el procedimiento OAuth: no hay necesidad de controlar y hacer aplicar centralmente la autorización de acceso, en cambio, debe manejarse y hacerse aplicar directamente la autorización mediante el dominio administrativo local al que pertenece la API. Esto permite que la infraestructura local cambie sin afectar al sistema global y proporciona flexibilidad para que cada dominio aplique sus propias políticas de acceso sin tener que seguir obligatoriamente políticas de seguridad globales. Para hacer esto posible, debe encaminarse el proceso OAuth, siguiendo criterios flexibles, al dominio administrativo apropiado basándose en la información y preferencias del propietario de recursos proporcionadas durante la fase de autorización de OAuth.

Los criterios de encaminamiento pueden variar de un escenario a otro. Por ejemplo, una decisión de encaminamiento evidente puede ser encaminar el proceso OAuth al proveedor de servicios al que el usuario pertenece, pero la presente solución no descarta el uso de otros criterios como los que pueden aplicarse mejor para el escenario objetivo. El punto clave es encaminar de manera flexible el proceso OAuth al dominio administrativo local apropiado basándose en la información y preferencias del propietario de recursos.

- Encaminar peticiones de API: una vez que el dominio administrativo correcto (más específicamente, el servidor de autorizaciones de ese dominio) ha emitido la autorización y la aplicación ha obtenido el testigo de acceso asociado, las peticiones de API enviadas por la aplicación pueden encaminarse basándose en ese testigo de acceso. De hecho, además de ser un identificador o un puntero para conseguir el alcance y políticas de acceso establecidos durante la fase de autorización, el propio testigo de acceso también se usa como puntero para la decisión de encaminamiento tomada durante la fase de autorización. De esta manera, la infraestructura de servicio global, que conoce el testigo de acceso puesto que actúa como intermediario para los procedimientos OAuth, proporciona una base de datos que asocia cada testigo de acceso con el dominio administrativo apropiado y, opcionalmente, los puntos terminales de API específicos a los que puede accederse con ese testigo dentro del dominio administrativo.

Basándose en este concepto, tal como se muestra en la figura 5, el acceso a los puntos terminales de API puede distribuirse sin basarse en ningún elemento centralizado, en cualquiera de las siguientes dos maneras:

- Modo de redirección: antes de enviar una petición de API, las aplicaciones preguntan al punto terminal que debe usarse para un testigo de acceso dado. La respuesta puede almacenarse en caché en el lado de aplicación de modo que no tenga que repetirse de nuevo la consulta para peticiones posteriores.
- Modo de intermediario: un grupo de “intermediarios de API” puede distribuirse a través de Internet. Las aplicaciones envían peticiones de API al intermediario más cercano sin tener que preocuparse del punto terminal de API real que debe usarse. Cuando una petición llega al intermediario, el intermediario extrae el testigo de acceso y obtiene los puntos terminales de API asociados al testigo de acceso preguntando a la infraestructura global. Una vez que tiene el punto terminal, simplemente reenvía la petición a ese punto terminal. El punto terminal asociado al testigo de acceso puede almacenarse en caché por el intermediario para evitar tener que preguntar de nuevo peticiones posteriores que usen el mismo testigo. La manera en que la aplicación averigua cuál es el intermediario más cercano está fuera del alcance de la presente invención.

OAuth distribuido proporciona un enfoque sencillo y eficaz para autorizar y consumir un servicio en nombre de un usuario, en el que la novedad se encuentra en centrarse en la autorización así como en el servidor de recursos (es decir el punto terminal de API) según se decide dinámicamente por una decisión de encaminamiento, por ejemplo, centrarse en el proveedor de servicios del usuario (más exactamente, el dominio administrativo adecuado dentro del proveedor de servicios). Por motivos de simplicidad, en las siguientes descripciones detalladas se asumirá que ésta es siempre la decisión de encaminamiento aplicada, pero tal como se mencionó previamente, hay libertad para aplicar otros criterios de encaminamiento cuando se consideren adecuados para el escenario específico.

Para describir los detalles de la invención propuesta de una manera entendible, se sigue un orden secuencial:

- En primer lugar, se proporciona una descripción de los procedimientos que han de tener lugar primero en situaciones reales, es decir: conseguir la autorización. Los procedimientos para conseguir la autorización se dividen además en dos alternativas diferentes:

    1. Casos en los que todos los proveedores de servicios/dominios administrativos usan un portal de autorización y autenticación global común para interactuar con usuarios finales (obsérvese que la emisión de autorizaciones y el control posterior aún se controlan por cada dominio administrativo independientemente).
    2. Casos en los que cada proveedor de servicios/dominio administrativo tiene su propio portal de autorización y autenticación local para interactuar con usuarios finales.

- En segundo lugar, se describen los procedimientos que permiten encaminar las peticiones de API basándose en el testigo de acceso obtenido después de completar la fase de autorización.

Adicionalmente, para OAuth 1.0 se consideran dos soluciones paralelas para cada escenario: una que supone que el testigo de petición puede configurarse directamente hacia la infraestructura de servicio local, y otra que considera que lo anterior no es posible y los testigos de petición se generan dinámicamente por la infraestructura de servicio local bajo demanda, siguiendo la interfaz de OAuth 1.0 convencional, que está totalmente expuesta de manera local.

• Un portal web global

Antes de consumir un recurso protegido, la aplicación necesita un testigo de acceso que le autorice acceder al

recurso en nombre del usuario. Esta sección considera un portal web global para interactuar con el usuario final con el fin de realizar la autenticación y autorización independientemente del servidor de autorización subyacente o del servidor de recursos asociado al usuario. Este portal web puede ayudar a ofrecer una experiencia de usuario unificada para cada servidor de autorización y recursos. Sin embargo, obsérvese que la autenticación y autorización se realizan en realidad por la infraestructura de servicio local.

Tal como se muestra en la figura 6, este proceso se divide en las siguientes etapas:

1. El usuario interactúa con la aplicación. La aplicación necesita la autorización del usuario para acceder a algunos recursos protegidos.
2. La aplicación consigue un testigo de petición, desde el servidor de autorización global, porque la aplicación no conoce la infraestructura de servicio local asociada al usuario.
3. El usuario se redirige, con el testigo de petición, al portal web global para autenticar y autorizar la aplicación. El testigo de petición se valida por la infraestructura global y el portal obtiene el identificador de aplicación y el alcance (qué recursos se solicitan por la aplicación).
4. El usuario se autentica en el portal web global. El nombre de usuario permite que la infraestructura global averigüe la infraestructura de servicio local asociada al usuario (por medio de una base de datos que correlaciona el nombre de usuario con la infraestructura de servicio local asociada. Tal como se comentó previamente, está fuera del alcance de la presente invención describir cómo esta información local se pone a disposición de la infraestructura global, pero hay diferentes alternativas tales como usar enfoques de suscripción, soluciones de replicación de datos, etc.). La petición de autenticación se propaga a la infraestructura de servicio local asociada al usuario.
5. Pueden producirse un conjunto de interacciones entre la infraestructura global y la infraestructura de servicio local para establecer los términos y condiciones de la autorización. Estas interacciones están fuera del alcance de la presente invención.
6. El usuario autoriza a la aplicación a acceder a algunos recursos protegidos en su nombre conforme a términos y condiciones específicos. El testigo de petición se almacena en la infraestructura de servicio local para adherirse a la operación al proceso OAuth convencional. La instancia local genera un código de autorización, código de verificación en la terminología de OAuth 1.0, que se almacena en caché en la infraestructura global para correlacionar un código de autorización con la infraestructura de servicio local. Obsérvese que esta etapa es en realidad un conjunto de requisitos para la infraestructura global y local, pero no tiene impacto en la API pública de OAuth, puesto que getRequestToken se aplica de hecho globalmente tal como se describe en la etapa 2 del presente escenario, es decir la aplicación externa siempre observa un procedimiento de acceso OAuth convencional.
7. La aplicación recibe el código de autorización (o código de verificación) a través de una redirección HTTP (también serían factibles otros mecanismos). La aplicación solicita un testigo de acceso a la infraestructura de servicio global que actúa de intermediario con el servidor de autorización en la infraestructura de servicio local que en realidad genera el testigo de acceso (y el secreto del testigo). La infraestructura global guarda la correlación entre el testigo de acceso y la infraestructura de servicio local que permite que el directorio de testigos de acceso global encamine las peticiones a los recursos. Finalmente, la aplicación consigue que el testigo de acceso (y el secreto del testigo) consuma los recursos protegidos en nombre del usuario.

En la figura 7 se representó el proceso anterior para el caso en el que los testigos de petición se generan por la infraestructura local bajo demanda. Dentro del flujo, CallBack2 se forma de la siguiente manera: AAPortalURL?rt1=RequestToken1&cb1=callBack1. Esto permite que el portal AA recupere automáticamente el RequestToken1 y el callBack1, que se usarán cuando se regrese a la aplicación una vez que la autorización se haya emitido.

OAuth 2.0 simplifica el proceso global porque la aplicación ya no necesita un testigo de petición. La aplicación también incluye un alcance, en la redirección HTTP para la etapa de autenticación, que especifica los recursos que el usuario debe autorizar, tal como se muestra en la figura 8.

• Portales web locales distribuidos

En este caso no hay ningún portal web global para llevar a cabo las interacciones con el usuario final con el fin de realizar la autenticación y autorización. Cada infraestructura de servicio local proporciona un portal web para realizar ambas acciones. La infraestructura de servicio global no actúa de intermediario ni para la autenticación ni para la autorización. Como resultado, se requiere un enfoque alternativo para centrarse en la infraestructura de servicio local apropiada. Podrían usarse varios mecanismos (por ejemplo mediante el análisis de la dirección IP del usuario) para inferir la infraestructura de servicio local. Este mecanismo podría simplificarse solicitando que el usuario lo seleccione explícitamente, tal como se muestra en la figura 9, que describe la secuencia para obtener el testigo de acceso, asumiendo que los testigos de petición pueden configurarse directamente de manera local. Este proceso se divide en las siguientes etapas:

1. El usuario interactúa con la aplicación. La aplicación necesita la autorización del usuario para acceder a algunos recursos protegidos.
2. La aplicación consigue un testigo de petición, desde el servidor de autorización global, porque la aplicación no conoce la infraestructura de servicio local asociada al usuario.
3. El usuario se redirige, con el testigo de petición, al portal web global. El testigo de petición se valida por la infraestructura global y el portal obtiene el identificador de aplicación y el alcance (qué recursos se solicitan por la aplicación).
4. El usuario selecciona la infraestructura de servicio local (haciendo clic en una página web o mediante mecanismos alternativos fuera del alcance de la presente invención). El usuario se redirige, mediante HTTP, al portal web local (según la selección) y la aplicación de devolución de llamada (callback) se sustituye por un URL de infraestructura de servicio global para correlacionar el código de autorización con la infraestructura de servicio local.
5. El usuario se autentica en el portal web local.
6. Pueden producirse un conjunto de interacciones para establecer los términos y condiciones. Estas interacciones están fuera del alcance de este documento.
7. El usuario autoriza a la aplicación para que acceda a algunos recursos protegidos en su nombre conforme a términos y condiciones específicos. El testigo de petición se almacena en la infraestructura de servicio local para adherirse a la operación al proceso OAuth convencional, pero el secreto del testigo será un valor preestablecido porque no puede adjuntarse en la petición del usuario. Finalmente, la instancia local genera el código de autorización, código de verificación en la terminología de OAuth 1.0. Obsérvese que esta etapa es en realidad un conjunto de requisitos para la infraestructura local, pero no tiene impacto en la API pública de OAuth, puesto que el getRequestToken se aplica de hecho globalmente tal como se describe en la etapa 2 del presente escenario, es decir la aplicación externa siempre observa un procedimiento de acceso OAuth convencional.
8. El usuario se redirige a la infraestructura de servicio global, con el testigo de petición y el código de autorización, de modo que un mapa entre el código de autorización y la infraestructura de servicio local se almacena en caché en la infraestructura global. La aplicación de devolución de llamada se obtiene de los detalles de aplicación. El usuario se redirige a la aplicación de devolución de llamada.
9. La aplicación recibe el código de autorización (o código de verificación en la terminología de OAuth 1.0) mediante la redirección HTTP (otros mecanismos también serían factibles). La aplicación solicita un testigo de acceso a la infraestructura de servicio global que actúa de intermediario con el servidor de autorización en la infraestructura de servicio local para generar el testigo de acceso (y el secreto del testigo). La infraestructura global guarda la correlación entre el testigo de acceso y la infraestructura de servicio local que permite que el directorio de testigos de acceso global encamine las peticiones a los recursos. Finalmente, la aplicación consigue que el testigo de acceso (y el secreto del testigo) consuma los recursos protegidos en nombre del usuario.

En la figura 10 se representó el flujo anterior para el caso en el que los testigos de petición se generan por la infraestructura local bajo demanda. Dentro del flujo, CallBack2 se forma de la siguiente manera: GlobalPortalURL?rt1=RequestToken1&cb1=callBack1. Esto permite que el portal global recupere automáticamente el RequestToken1 y el callBack1, que se usarán cuando se regrese a la aplicación una vez que la autorización se haya emitido.

En OAuth 2.0, tal como se mostró en la figura 11, el estado se incluye por el portal global si es que no se incluyó ya por la aplicación. Esto se usa para correlacionar la segunda redirección de vuelta desde el portal local, después de la autorización de usuario.

• Consumir un recurso protegido

Una vez que la aplicación ha obtenido un testigo de acceso - que se emite por el proveedor de servicios del usuario tal como se explicó anteriormente - para acceder a un recurso protegido, la petición de servicio de la aplicación necesita centrarse en el servidor de recursos del usuario. Aunque la aplicación no conozca qué servidor de recursos se asocia con el usuario, el testigo de acceso ayudará a averiguarlo mediante las siguientes alternativas:

- Autocontenido. El servidor de recursos del usuario está autocontenido en el testigo de acceso. En este caso, hay dos posibilidades adicionales:

    1. La información del servidor de recursos se cifra para impedir que la aplicación (y terceros) infieran la información a partir del testigo de acceso. Este caso es útil si el servidor de recursos del usuario se considera información delicada.
    2. La información del servidor de recursos está disponible en texto no cifrado, lo que simplifica el procesamiento del testigo de acceso.

- Basado en directorio. Un directorio de testigos de acceso asocia cada testigo de acceso con un servidor de recursos.

Como resultado, el encaminamiento al servidor de recursos apropiado se basa en el testigo de acceso. Se consideran dos enfoques diferentes:

- Modo de intermediario. Un intermediario actuará de intermediario entre la aplicación y el servidor de recursos. El intermediario asume la responsabilidad de encaminar las peticiones de servicios de la aplicación al servidor de recursos del usuario basándose en el testigo de acceso.
- Modo de redirección. La aplicación recupera el punto terminal del servidor de recursos a partir de un directorio de testigos de acceso. Entonces la aplicación tiene que componer el URL para acceder al recurso protegido en el servidor de recursos del usuario.

El modo de intermediario tiene dos ventajas: a) el URL es único; el mismo URL es válido para acceder al mismo recurso protegido en todos los servidores de recursos, y b) la aplicación se simplifica porque no conoce los aspectos de redirección. Por otro lado, la desventaja principal de este modo es que un intermediario aumenta la latencia de respuesta y podría convertirse en un cuello de botella, lo que podría aliviarse replicando y distribuyendo geográficamente varios intermediarios. Tal como se muestra en la figura 12, el modo de intermediario está dividido en dos casos de uso:

- La aplicación accede con un testigo de acceso inválido. El intermediario intenta sin éxito resolver el testigo de acceso por medio del directorio de testigos de acceso global. El intermediario responderá con un mensaje HTTP no autorizado.
- La aplicación usa un testigo de acceso válido. El intermediario intenta resolver el testigo de acceso con su caché para acelerar el encaminamiento. Si el testigo de acceso no está almacenado en la caché, el intermediario intenta resolverlo por medio del directorio de testigos de acceso global, que devolvería el servidor de recursos apropiado guardando esta coincidencia temporalmente en la caché para futuras peticiones. Finalmente, se accederá al recurso mediante el intermediario.

El modo de redirección es el enfoque más eficaz porque la aplicación interactúa directamente con el servidor de recursos apropiado. Sin embargo, la lógica de aplicación es ligeramente más complicada porque necesita resolver el testigo de acceso por medio del directorio de testigos de acceso global, tal como se muestra en la figura 12. La aplicación almacenaría en caché localmente esta correlación para usos adicionales.

Si el punto terminal del servidor de recursos está autocontenido en el testigo de acceso, la resolución del testigo de acceso podría realizarse en el directorio de testigos de acceso global o directamente en el intermediario/aplicación.

• Revocar un testigo de acceso

El usuario necesita mecanismos para revocar la autorización a una aplicación, lo que significa revocar el testigo de acceso. La infraestructura de servicio local podría también recovar un testigo de acceso debido a las políticas específicas (por ejemplo, un testigo de acceso de un solo uso, una vez que la aplicación lo haya consumido).

El proceso para revocar un testigo de acceso, tal como se muestra en la figura 13, consiste en:

1. El usuario entra al portal web global para revocar la autorización a una aplicación. El portal delega esta revocación a la infraestructura de servicio global que actúa de intermediario con la infraestructura de servicio local apropiada.
2. La infraestructura de servicio local revocará el testigo de acceso, que se gestionó por el servidor de autorización local. La infraestructura de servicio global también elimina el testigo de acceso revocado de su caché.

Puesto que la revocación también puede realizarse directamente hacia la infraestructura de servicio local y puesto que los testigos pueden expirar sin tener que revocarse, con el fin de tener información actualizada sobre testigos de acceso válidos a nivel global, la infraestructura de servicio global recupera periódicamente la lista de testigos de acceso revocados desde cada infraestructura de servicio local para actualizar la caché de testigos de acceso. Esta sincronización no es crítica en cuanto al tiempo, porque no hay manera de acceder a los recursos con un testigo de acceso inválido, la sincronización simplemente elimina entradas inválidas a partir de la caché de la infraestructura de servicio global.

El mercado de operadores de telecomunicaciones es el punto central principal de esta invención. Cada operador de telecomunicaciones se convierte en el proveedor de servicios y cada usuario (o cliente) pertenece a un operador. Normalmente, un mercado de aplicaciones puede ofrecer aplicaciones construidas sobre los servicios del operador de telecomunicaciones. Sin embargo, el desarrollador de aplicaciones necesitaría implementar varias variantes de la misma aplicación para cubrir varios operadores de telecomunicaciones a menos que las interfaces de servicio, proporcionadas por cada operador, se normalicen.

La normalización de interfaces de servicio beneficiará a los desarrolladores de aplicaciones porque amplía el mercado de posibles usuarios, a los usuarios puesto que aumenta el número de aplicaciones y también su calidad, y a los operadores de telecomunicaciones porque sus ingresos crecen debido a un número mayor de transacciones. Todos los actores involucrados en el caso de uso se benefician de este enfoque.

Los proveedores de servicios (u operadores de telecomunicaciones) confían en una entidad global superior para la autenticación y autorización o proporcionan un portal web local para tal finalidad. El primer caso es válido si los usuarios confían en esta entidad global con la misma confianza que su operador de telecomunicaciones.

Sin embargo, otros casos de uso no pueden basarse en esta entidad global. La Wholesale Application Community (WAC) [15] tiene el objetivo de establecer un mercado de aplicaciones válido para cada operador y teléfono móvil. Sin embargo, los operadores no pueden abrir la información de sus usuarios, que es vital para su negocio, a una entidad diferente. Este último enfoque, en el que la autenticación y autorización se realizan localmente, serviría para resolver esta restricción, y los usuarios se beneficiarían de un amplio conjunto de aplicaciones que hacen uso de servicios de compañías de telecomunicaciones (para aquellos servicios que se hayan normalizado).

• Ventajas de la invención

Cuando se compara con enfoques tradicionales, OAuth distribuido presenta varios beneficios, tanto para usuarios finales como para proveedores de servicios. Algunos se resaltan a continuación:

- Escalabilidad y rendimiento. OAuth distribuido evita la necesidad de tener una infraestructura centralizada para emitir y ejecutar el uso de testigos de acceso, lo que a su vez permite acceder a las API de una manera completamente distribuida. Esto implica una gran mejora en términos de escalabilidad y rendimiento, ya que la infraestructura puede crecer horizontalmente para afrontar las demandas de servicios de escala de Internet. Los cuellos de botella pueden evitarse mientras se mantiene un control completo sobre el acceso a API.
- Potenciar API convencional. OAuth distribuido proporciona la flexibilidad requerida para que diferentes proveedores de servicios expongan las mismas API al tiempo que mantienen su autonomía. Siempre y cuando se adhieran a OAuth convencional y respeten los requisitos expuestos en este documento, los proveedores de servicios pueden implementar libremente su propio mecanismo de control de acceso a API local (autenticación de usuario, políticas de acceso, etc.) incluso cuando las API van a exponerse conjuntamente con otros proveedores de servicios en escenarios globales. El funcionamiento global del sistema es satisfactorio, mientras que las implementaciones locales no necesitan conocer nada unas de otras.
- Experiencia simplificada del desarrollador. Actualmente, los desarrolladores necesitan conocer los puntos terminales de API antes de poder acceder a ellos. Este hecho complica los escenarios mencionados anteriormente en los que hay varios puntos terminales y diferentes proveedores de servicios, ya que los desarrolladores necesitan afrontar esta diversidad por sí mismos.

OAuth distribuido mantiene una compatibilidad completa con las normas OAuth 1.0 o 2.0 y no requiere ningún comportamiento especial desde el lado de aplicación. Gracias a OAuth distribuido, los desarrolladores podrán acceder a API distribuidas, que pertenezcan potencialmente a diferentes proveedores de servicios, simplemente siguiendo OAuth convencional, sin tener que preocuparse por la diversidad subyacente.

Más precisamente, las aplicaciones no necesitarán hacer nada especial cuando se use el modo de intermediario, si se usa el modo de redirección, será necesario que la aplicación consuma el servicio de directorio de testigos de acceso global. Ésta es la única adición a la operación “convencional” en el lado de aplicación.

Un experto en la técnica puede introducir cambios y modificaciones en las realizaciones descritas sin apartarse del alcance de la invención tal como se define en las reivindicaciones adjuntas.

SIGLAS

API	Application Programming Interface; Interfaz de programación de aplicaciones
HTTP	HyperText Transfer Protocol; Protocolo de transferencia de hipertexto
IETF	Internet Engineering Task Force; Grupo de trabajo de ingeniería de Internet
REST	Representational State Transfer; Transferencia de estado representacional
SDP	Service Delivery Platform; Plataforma de prestación de servicios
SOAP	Simple Object Access Protocol; Protocolo simple de acceso a objetos
URL	Uniform Resource Locator; Localizador uniforme de recursos

REFERENCIAS

[1] OAuth 1.0, http://tools.ietf.org/html/rfc5849

[2] OAuth 2.0, http://tools.ietf.org/html/draft-ietf-oauth-v2-22

[3] http://www.sitepen.com/blog/2009/02/19/introducing-oauth-in-dojox/

[4] BlueVia developers OAuth guides, https://bluevia.com/en/knowledge/APIs.API-Guides.OAuth

[5] Twitter developers information, http://dev.twitter.com/pages/auth

[6] Google information on OAuth, http://code.google.com/p/oauth/

[7] Yahoo developers oauth information, http://developer.yahoo.com/oauth/

[8] Facebook developer information http://developers.facebook.com/docs/authentication/

[9] Google information on OAuth 2.0, http://code.google.com/intl/es-ES/apis/accounts/docs/OAuth2.html

[10] Federated Login for Google Account Users, http://code.google.com/intl/es-ES/apis/accounts/docs/OpenID.html

[11] OpenID OAuth Extension, http://step2.googlecode.com/svn/spec/openid_oauth_extension/latest/openid_oauth_extension.html

[12] Microsoft Session Directory Services Token Redirection Support in the Cisco Content Switching Module for the Cisco Catalyst 6500, http://www.cisco.com/en/US/prod/collateral/modules/ps2706/ps780/product_solution_overview0900aecd806fc547.pdf

[13] About IP Address and Token Redirection, http://technet.microsoft.com/en-us/library/cc732852.aspx

[14] NexTransit TM Interconnect Proxy, http://www.transnexus.com/Products/NexTransit.htm

[15] WAC - Wholesale Application Community, http://www.wacapps.net

REIVINDICACIONES

1. Un método para proporcionar acceso autorizado a una aplicación de servicio para usar un recurso protegido de un usuario final, siendo dicho recurso protegido una interfaz de programación de aplicación, API, y estando expuesto en unos puntos finales protegidos de protocolo OAuth de una pluralidad de dominios administrativos, perteneciendo cada dominio administrativo en dicha pluralidad de dominios administrativos a un proveedor de servicio y que es responsable de emitir y controlar individualmente el uso posterior de las autorizaciones para acceder a múltiples recursos protegidos, según se conceden por el usuario final que pertenece a dicho dominio administrativo, **caracterizado porque**:

- el método se aplica a escenarios OAuth distribuidos donde dichos puntos finales de dicha pluralidad de dominios administrativos se desconocen previamente por dicha aplicación de servicio;
- los puntos finales de protocolo OAuth son únicos y válidos para conseguir autorizaciones para acceder a algún punto final de API subyacente;
- una infraestructura global soporta registro de aplicación y gestión de credenciales;

y **porque** el método comprende adicionalmente:

i. seleccionar, por una entidad intermedia, un dominio administrativo de dicha pluralidad de dominios administrativos basándose en información de dicho usuario final y preferencias proporcionadas durante la fase de autenticación OAuth;
ii. encaminar, por dicha entidad intermedia, un procedimiento OAuth a dicho dominio administrativo seleccionado;
iii. realizar, por dicho dominio administrativo seleccionado, una autorización segura para conceder acceso a dicho usuario final por medio del protocolo OAuth, consistiendo el último al menos de:

- una fase de autenticación de dicho usuario final; y
- proporcionar dicha aplicación de servicio con un testigo de acceso para permitir dicho acceso a dicho recurso protegido; y

iv. proporcionar, dicho dominio administrativo seleccionado a dicha aplicación de servicio, una vez realizada dicha autorización segura, acceso directo o de intermediario a dicho recurso protegido mediante uno de dichos puntos terminales establecidos por dicha entidad intermedia, encaminando dicha entidad intermedia solicitudes de API autorizadas, enviadas por dicha aplicación a dicho dominio administrativo, basándose en dicho testigo de acceso,

en el que el método mantiene compatibilidad completa con normas OAuth 1.0 o 2.0 y no requiere ningún comportamiento especial del lado de la aplicación sino que sigue simplemente la realización de OAuth convencional.

2. Un método de acuerdo con la reivindicación 1, que comprende gestionar interacción entre dicha entidad intermedia y dicho usuario final por medio de una página de enlaces.

3. Un método de acuerdo con la reivindicación 1, que comprende usar dicho testigo de acceso como un puntero a una decisión de encaminamiento tomada durante dicha fase de autenticación y que proporciona, a dicha entidad intermedia, una base de datos que asocia un testigo de acceso a un dominio administrativo de dicha pluralidad de dominios administrativos.

4. Un método de acuerdo con la reivindicación 3, que comprende realizar dicho encaminamiento de dichas peticiones autorizadas de API por medio de una redirección que proporciona a dicha aplicación un localizador de recurso uniforme, o URL, de dominio administrativo basándose en dicho testigo de acceso que consulta dicha base de datos, en el que dicha aplicación a continuación accede a dicho recurso protegido con dicho URL de dominio administrativo y una ruta a dicho recurso protegido.

5. Un método de acuerdo con la reivindicación 4, que comprende realizar dicho encaminamiento de dichas peticiones autorizadas de API por medio de un intermediario que extrae dicho testigo de acceso y obtiene dicho punto final de API asociado a dicho testigo de acceso de dicha infraestructura intermedia.

6. Un método de acuerdo con cualquiera de las reivindicaciones anteriores, en el que dicha pluralidad de dominios administrativos usa una página web o wap común que permite la interacción con el usuario final para realizar al menos parte de dicha fase de autenticación.

7. Un método de acuerdo con cualquiera de las reivindicaciones anteriores, en el que cada uno de dicha pluralidad de dominios administrativos usa una página web o wap local que permite la interacción con el usuario final para realizar al menos parte de dicha fase de autenticación.

8. Un método de acuerdo con cualquiera de las reivindicaciones anteriores, en el que dicho protocolo abierto es OAuth 1.0, que comprende conceder un testigo de petición a dicha aplicación, usándose dicho testigo de petición para solicitar que dicho usuario final autorice a dicha aplicación a acceder a dicha API, realizado durante dicha fase de autenticación.

9. Un método de acuerdo con la reivindicación 8, que comprende, dicha aplicación de servicio, obtener dicho testigo de petición de dicha entidad intermedia.

10. Un método de acuerdo con la reivindicación 9, que comprende, dicha entidad intermedia, conseguir dicho testigo de petición de dicho dominio administrativo o establecer dicho testigo de petición en dicho dominio administrativo.

11. Un método de acuerdo con cualquiera de las reivindicaciones anteriores 9 a 10 cuando dependen de la reivindicación 3, en el que dicho procedimiento OAuth comprende adicionalmente:

- conseguir dicha aplicación dicho testigo de petición;
- reenviar dicho usuario a dicha página de enlaces para averiguar dicho dominio administrativo;
- usar dicha entidad intermedia dicha información de dicho usuario final para averiguar dicho dominio administrativo y reenviar dicho usuario final a la página web o wap común o local correspondiente en consecuencia;
- proporcionar dicho usuario credenciales, cuando no se están autenticando previamente por otros medios, a dicha página web o wap común o local para realizar dicha fase de autenticación;
- reenviar dicha página web o wap común o local una petición de autenticación a un servidor de autenticación de dicho dominio administrativo con dichos credenciales;
- autorizar dicho usuario final dicha aplicación para acceder a dicho recurso protegido;
- generar dicho servidor de autorización de dicho dominio administrativo un código de autorización y enviarlo a dicha aplicación mediante dicha entidad intermedia;
- pedir dicha aplicación y conseguir dicho testigo de acceso mediante dicha entidad intermedia, que obtiene un testigo de acceso desde dicho dominio administrativo; y
- usar dicha aplicación dicho testigo de acceso para conseguir acceso a dicho recurso protegido.

12. Un método de acuerdo con cualquiera de las reivindicaciones anteriores 1 a 7, en el que dicho protocolo abierto es OAuth 2.0.

13. Un método de acuerdo con la reivindicación 12, en el que dicho procedimiento OAuth 2.0 comprende adicionalmente:

- reenviar dicho usuario final a dicha página de enlaces para averiguar dicho dominio administrativo;
- usar dicha entidad intermedia dicha información de dicho usuario final para averiguar dicho domino administrativo apropiado y reenviar dicho usuario final a la página web o wap común o local apropiada en consecuencia;
- proporcionar dicho usuario credenciales cuando no se están autenticando previamente por otros medios en dicha página web o wap común o local para realizar dicha fase de autenticación;
- reenviar dicha página web o wap común o local una solicitud de autenticación a un servidor de autorización de dicho domino administrativo con dichas credenciales;
- autorizar dicho usuario final dicha aplicación para acceder a dicho recurso protegido;
- generar dicho servidor de autorización un código de autorización o un testigo de acceso y enviar dicho código de autorización o dicho testigo de acceso a dicha aplicación de servicio mediante dicha entidad intermedia;
- pedir dicha aplicación y conseguir dicho testigo de acceso mediante dicha entidad intermedia en caso de recibir dicho código de autorización, en el que dicha entidad intermedia consigue un testigo de acceso de dicho dominio administrativo; y
- usar dicha aplicación dicho testigo de acceso para conseguir acceso a dicho recurso protegido.

14. Un método de acuerdo con cualquiera de las reivindicaciones anteriores, que comprende, dicho servidor de autorización, revocar dicho testigo de acceso a petición de dicho usuario final o revocar dicho testigo de acceso de acuerdo con políticas específicas de dicha API.

15. Un método de acuerdo con la reivindicación 14, que comprende, dicho usuario final, pedir dicha revocación de dicho testigo de acceso a dicha página común o local y reenviar dicha solicitud a dicho servidor de autorización de dicho dominio administrativo mediante dicha entidad intermedia.

16. Un método de acuerdo con cualquiera de las reivindicaciones anteriores 14 a 15, en el que dicha entidad intermedia recupera periódicamente una lista de testigos de acceso revocados de dichos puntos finales de API.

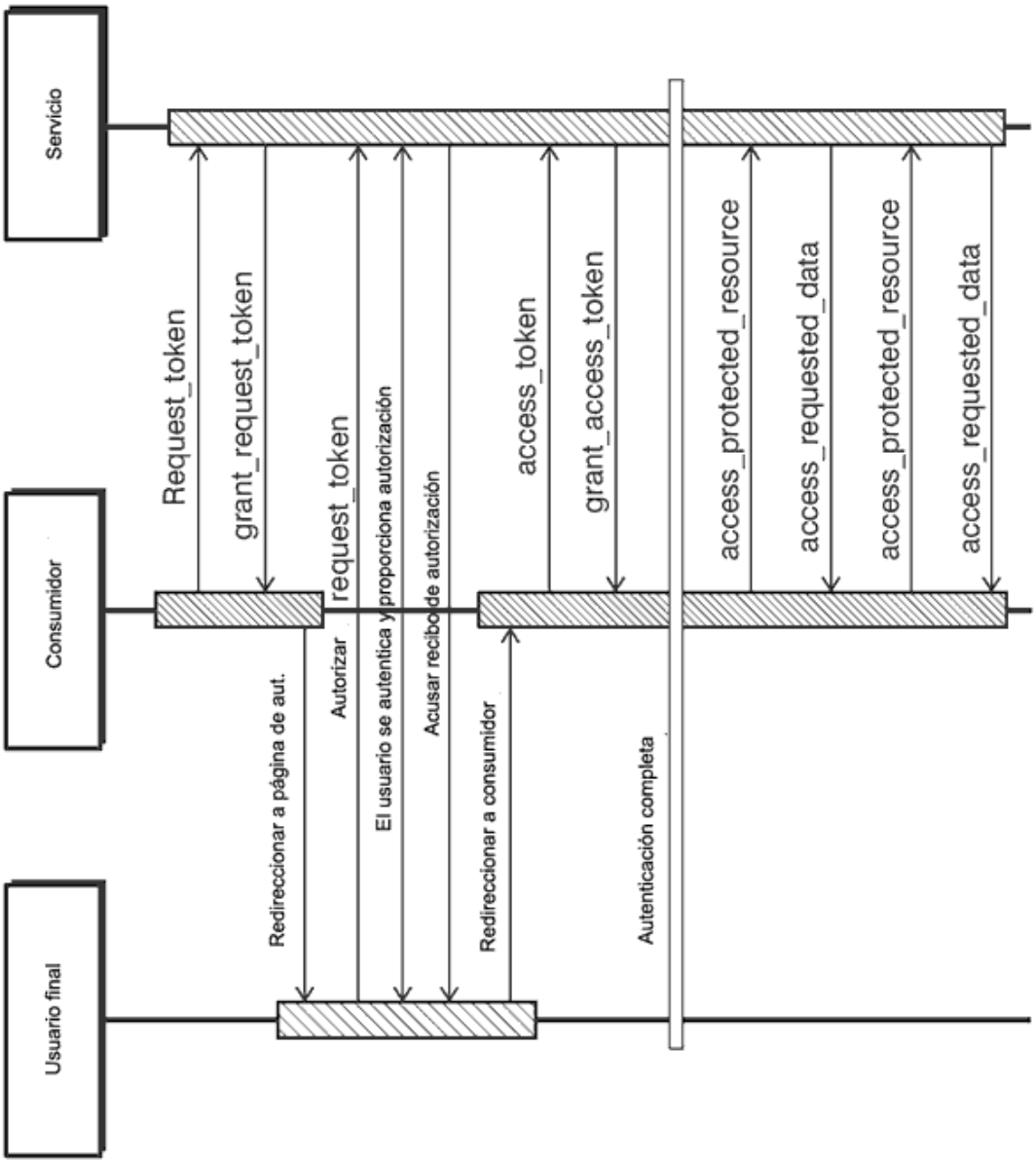
Servicio
Consumidor
Usuario final
Request_token
grant_request_token
request_token
El usuario se autentica y proporciona autorización
access_token
grant_access_token
access_protected_resource
access_requested_data
access_protected_resource
access_requested_data
Redireccionar a página de aut.
Autorizar
Acusar recibo de autorización
Redireccionar a consumidor
Autenticación completa

FIG. 1

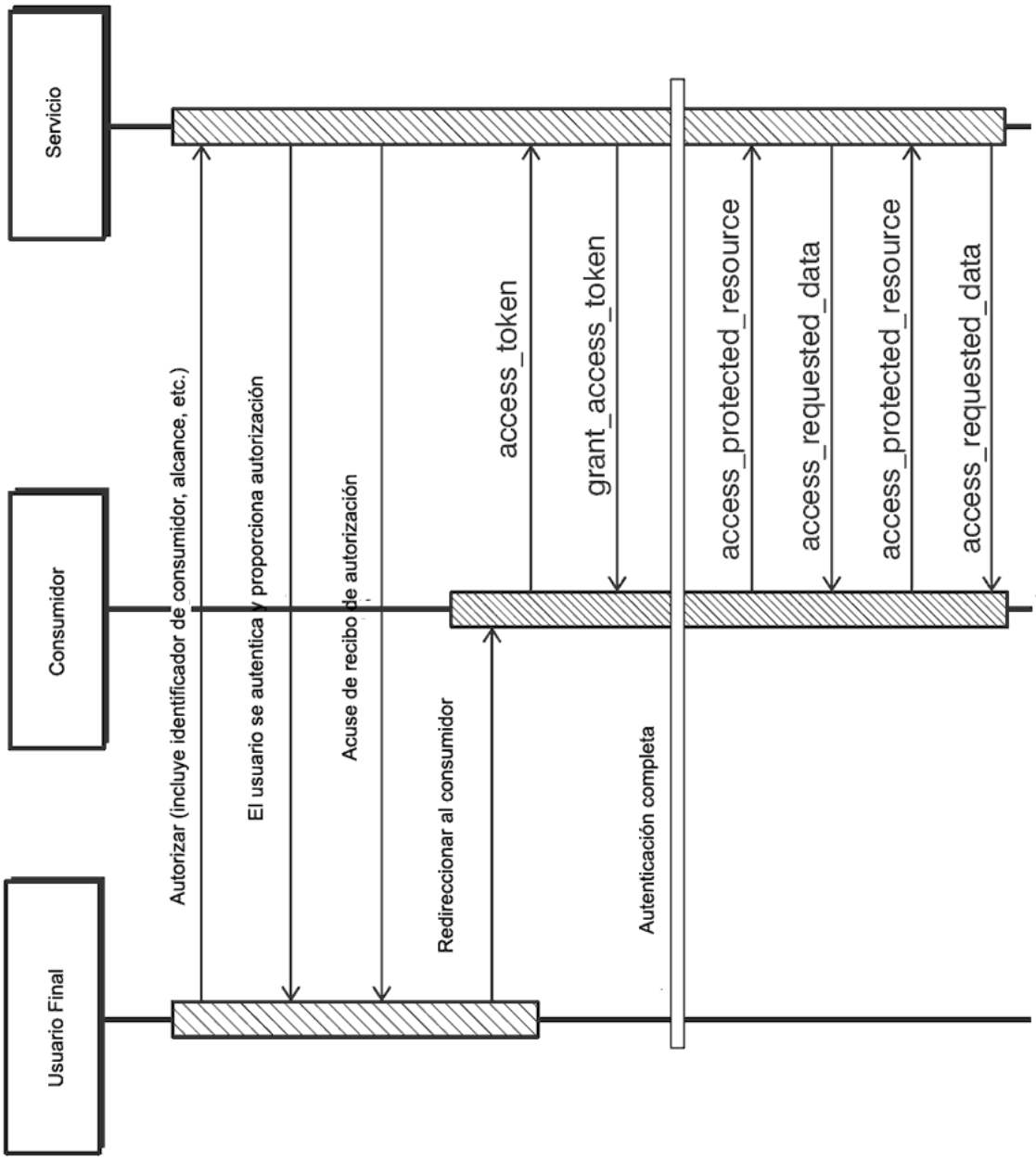
Servicio
Consumidor
Usuario Final
Autorizar (incluye identificador de consumidor, alcance, etc.)
El usuario se autentica y proporciona autorización
Acuse de recibo de autorización
Redireccionar al consumidor
Autenticación completa
access_token
grant_access_token
access_protected_resource
access_requested_data
access_protected_resource
access_requested_data

FIG. 2

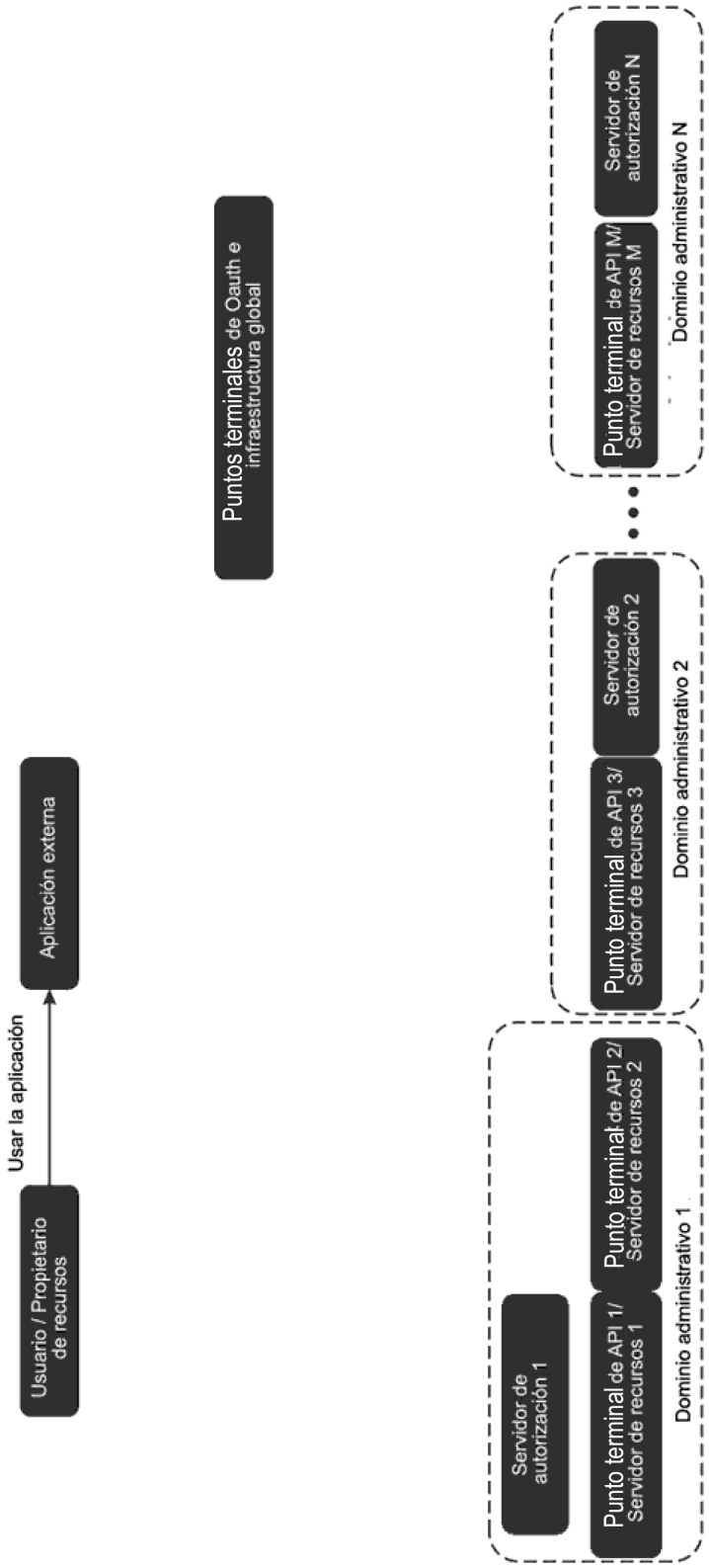
Usuario / Propietario de recursos
Usar la aplicación
Aplicación externa
Puntos terminales de Oauth e infraestructura global
Servidor de autorización 1
Punto terminal de API 1/ Servidor de recursos 1
Punto terminal de API 2/ Servidor de recursos 2
Dominio administrativo 1
Servidor de autorización 2
Punto terminal de API 3/ Servidor de recursos 3
Dominio administrativo 2
Servidor de autorización N
Punto terminal de API M/ Servidor de recursos M
Dominio administrativo N

FIG. 3

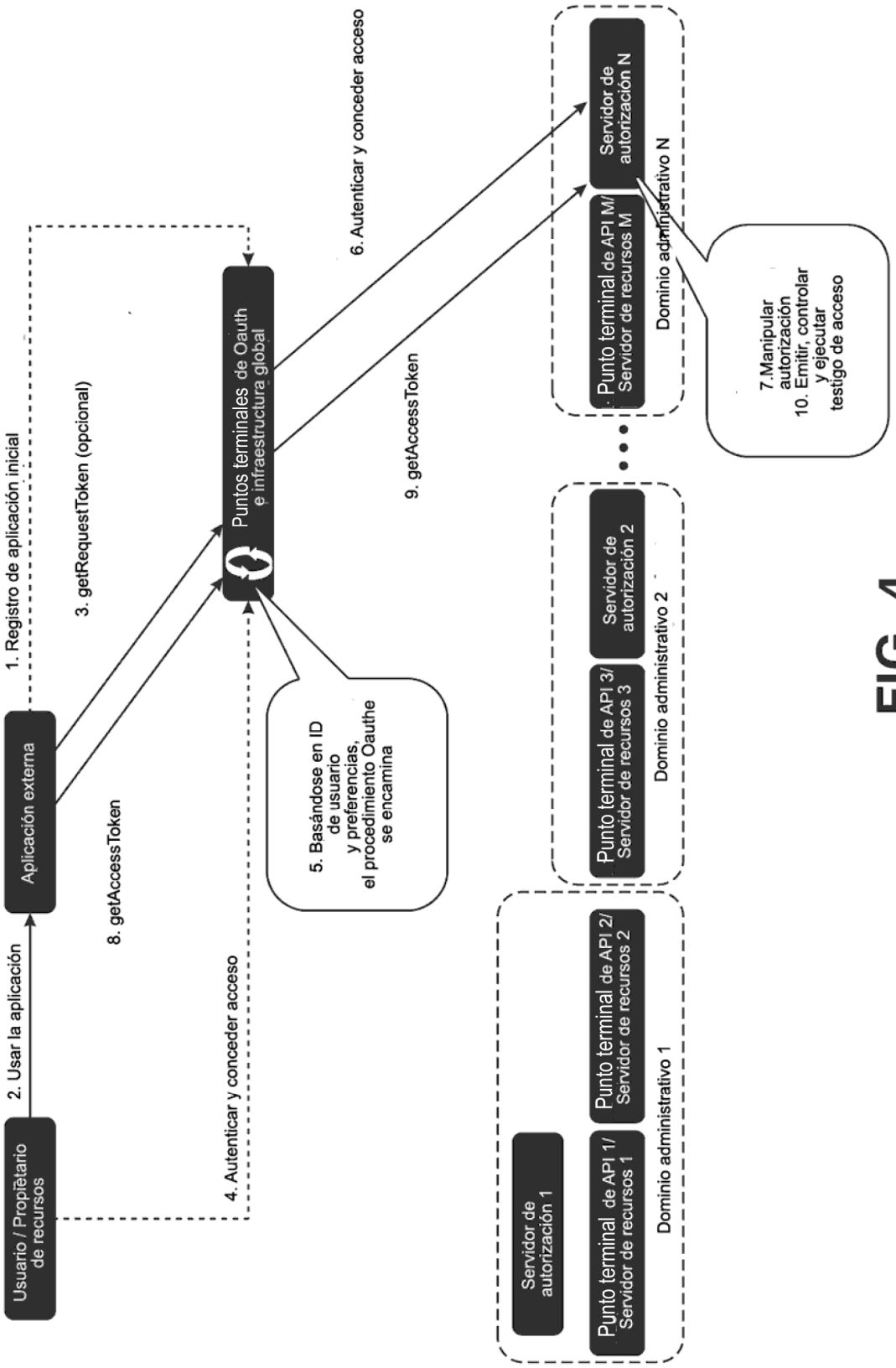
Usuario / Propietario de recursos
2. Usar la aplicación
Aplicación externa
1. Registro de aplicación inicial
3. getRequestToken (opcional)
8. getAccessToken
4. Autenticar y conceder acceso
Puntos terminales de Oauth e infraestructura global
5. Basándose en ID de usuario y preferencias, el procedimiento Oauthe se encamina
6. Autenticar y conceder acceso
9. getAccessToken
Servidor de autorización 1
Punto terminal de API 1/ Servidor de recursos 1
Punto terminal de API 2/ Servidor de recursos 2
Dominio administrativo 1
Servidor de autorización 2
Punto terminal de API 3/ Servidor de recursos 3
Dominio administrativo 2
Punto terminal de API M/ Servidor de recursos M
Servidor de autorización N
Dominio administrativo N
7.Manipular autorización
10. Emitir, controlar y ejecutar testigo de acceso

FIG. 4

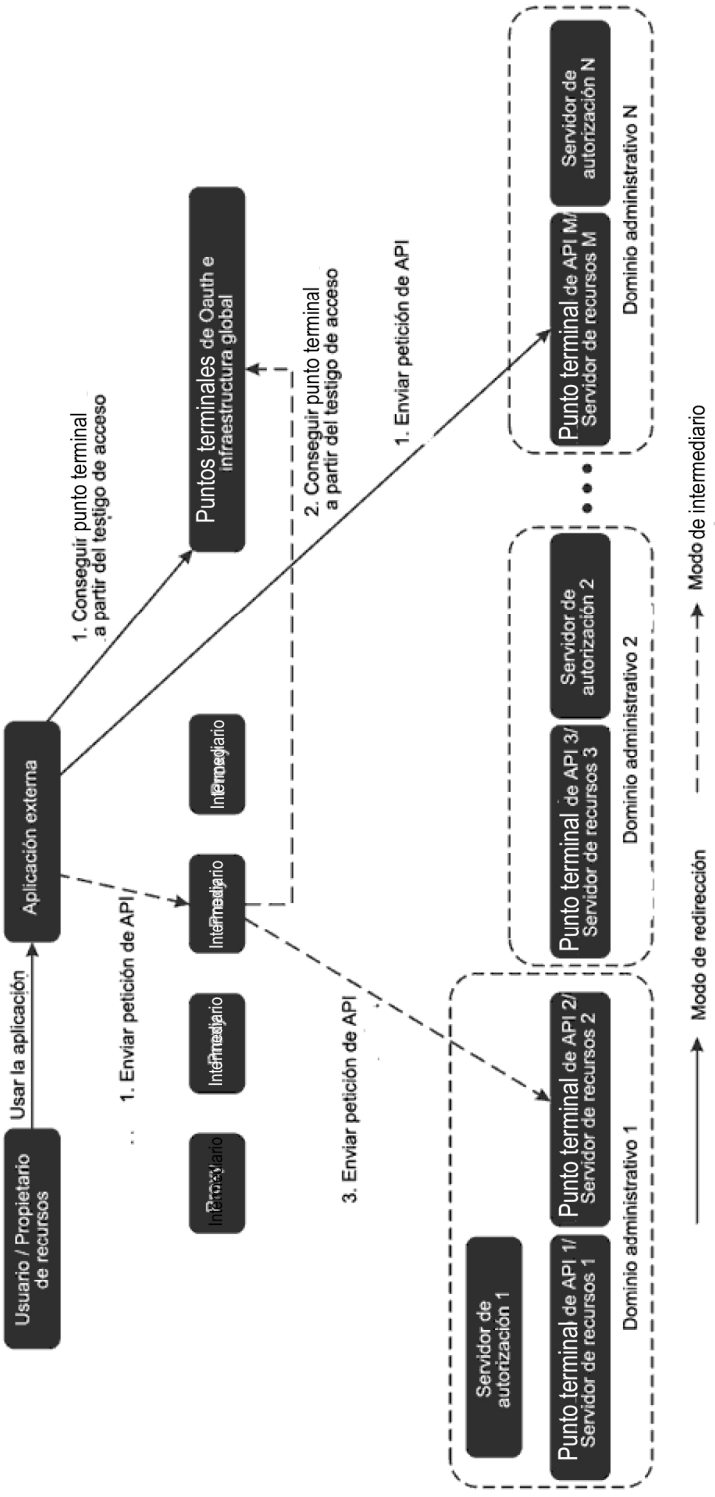
Usuario / Propietario de recursos
Usar la aplicación
Aplicación externa
1. Enviar petición de API
Intermediario
Intermediario
Intermediario
1. Conseguir punto terminal a partir del testigo de acceso
Puntos terminales de Oauth e infraestructura global
2. Conseguir punto terminal a partir del testigo de acceso
3. Enviar petición de API
1. Enviar petición de API
Servidor de autorización 1
Punto terminal de API 1/ Servidor de recursos 1
Punto terminal de API 2/ Servidor de recursos 2
Dominio administrativo 1
Servidor de autorización 2
Punto terminal de API 3/ Servidor de recursos 3
Dominio administrativo 2
Servidor de autorización N
Punto terminal de API M/ Servidor de recursos M
Dominio administrativo N
Modo de redirección
Modo de intermediario

FIG. 5

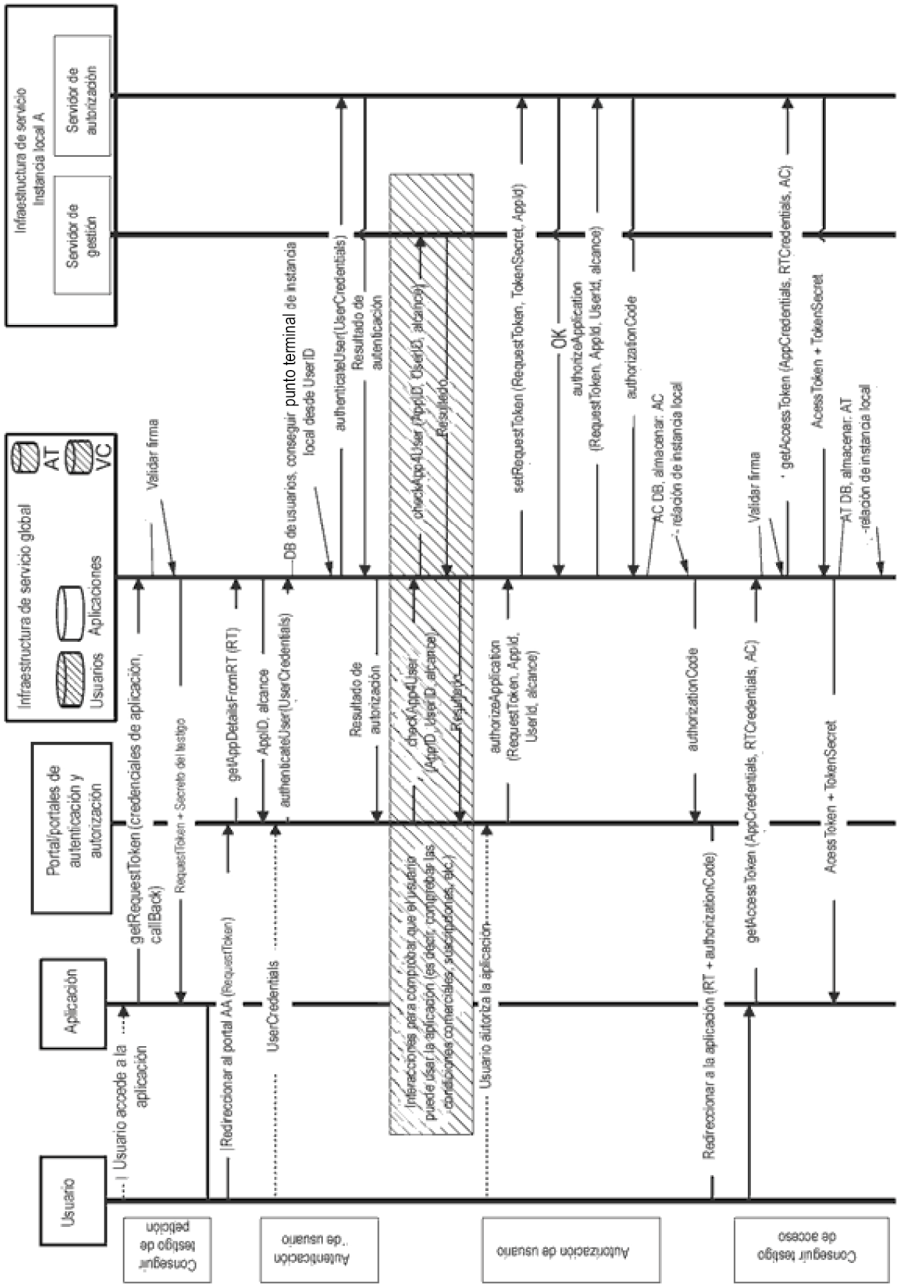
Usuario
Aplicación
Portal/portales de autenticación y autorización
Infraestructura de servicio global
Usuarios
Aplicaciones
AT
VC
Infraestructura de servicio Instancia local A
Servidor de gestión
Servidor de autorización
Usuario accede a la aplicación
getRequestToken (credenciales de aplicación, callBack)
RequestToken + Secreto del testigo
Validar firma
DB de usuarios, conseguir punto terminal de instancia local desde UserID
authenticateUser(UserCredentials)
Resultado de autenticación
checkApp4User (AppID, UserID, alcance)
Resultado
setRequestToken (RequestToken, TokenSecret, AppId)
OK
authorizeApplication (RequestToken, AppId, UserId, alcance)
authorizationCode
AC DB, almacenar: AC - relación de instancia local
Validar firma
getAccessToken (AppCredentials, RTCredentials, AC)
AcessToken + TokenSecret
AT DB, almacenar: AT - relación de instancia local
Redireccionar al portal AA (RequestToken)
getAppDetailsFromRT (RT)
AppID, alcance
authenticateUser(UserCredentials)
Resultado de autenticación
checkApp4User (AppID, UserID, alcance)
Resultado
authorizeApplication (RequestToken, AppId, UserId, alcance)
authorizationCode
getAccessToken (AppCredentials, RTCredentials, AC)
Acess Token + TokenSecret
UserCredentials
Interacciones para comprobar que el usuario puede usar la aplicación (es decir, comprobar las condiciones comerciales, suscripciones, etc.)
Usuario autoriza la aplicación
Redireccionar a la aplicación (RT + authorizationCode)
Conseguir testigo de petición
Autenticación de usuario
Autorización de usuario
Conseguir testigo de acceso

FIG. 6

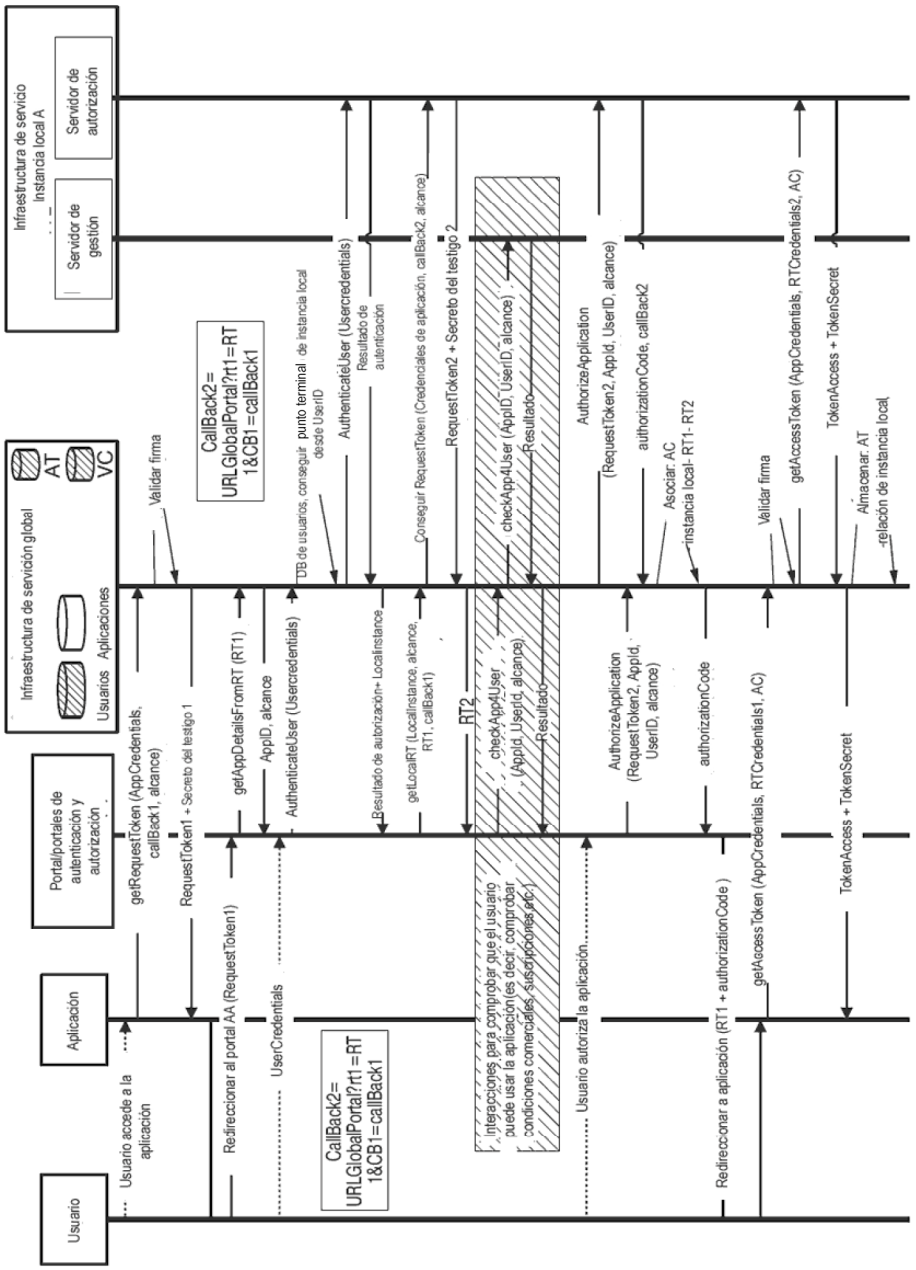
Usuario
Aplicación
Portal/portales de autenticación y autorización
Infraestructura de servición global
Usuarios
Aplicaciones
AT
VC
Infraestructura de servicio instancia local A
Servidor de gestión
Servidor de autorización
Usuario accede a la aplicación
getRequestToken (AppCredentials, callBack1, alcance)
RequestToken1 + Secreto del testigo 1
Redireccionar al portal AA (RequestToken1)
getAppDetailsFromRT (RT1)
AppID, alcance
AuthenticateUser (Usercredentials)
UserCredentials
Validar firma
CallBack2= URLGlobalPortal?rt1=RT 1&CB1=callBack1
DB de usuarios, conseguir punto terminal de instancia local desde UserID
AuthenticateUser (Usercredentials)
Resultado de autenticación
Resultado de autenticación+ LocalInstance
getLocalRT (LocalInstance, alcance, RT1, callBack1)
Conseguir RequestToken (Credenciales de aplicación, callBack2, alcance)
RequestToken2 + Secreto del testigo 2
RT2
checkApp4User (AppID, UserID, alcance)
Resultado
Interacciones para comprobar que el usuario puede usar la aplicación(es decir, comprobar condiciones comerciales, suscripciones,etc.)
checkApp4User (AppId, UserId, alcance)
Resultado
AuthorizeApplication (RequestToken2, AppId, UserID, alcance)
authorizationCode, callBack2
Asociar: AC
instancia local- RT1- RT2
AuthorizeApplication (RequestToken2, AppId, UserID, alcance)
authorizationCode
Usuario autoriza la aplicación
Validar firma
getAccessToken (AppCredentials, RTCredentials2, AC)
TokenAccess + TokenSecret
Almacenar: AT
relación de instancia local
CallBack2= URLGlobalPortal?rt1=RT 1&CB1=callBack1
Redireccionar a aplicación (RT1 + authorizationCode)
getAccessToken (AppCredentials, RTCredentials1, AC)
TokenAccess + TokenSecret

FIG. 7

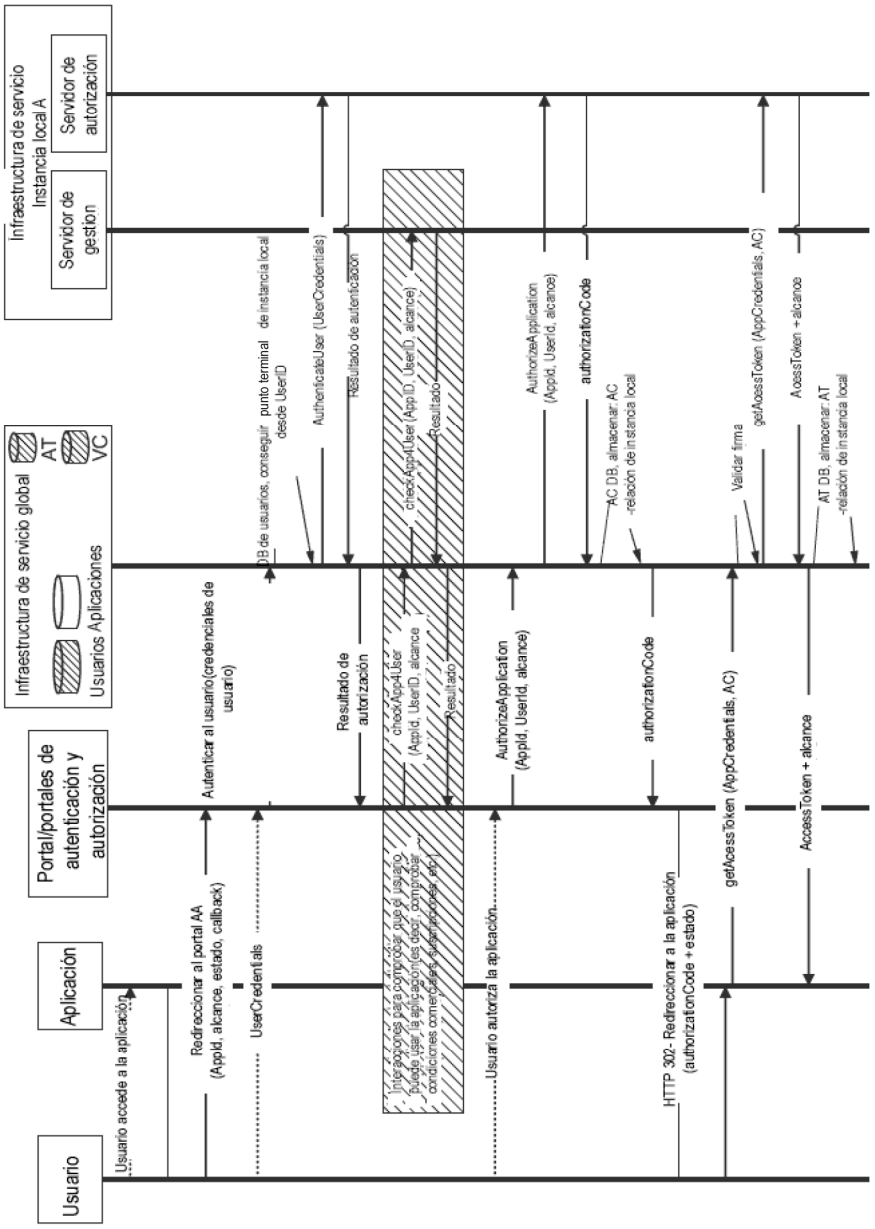
Usuario
Aplicación
Portal/portales de autenticación y autorización
Infraestructura de servicio global
AT
VC
Usuarios Aplicaciones
Infraestructura de servicio Instancia local A
Servidor de gestion
Servidor de autorización
Usuario accede a la aplicación
Redireccionar al portal AA (AppId, alcance, estado, callback)
UserCredentials
Autenticar al usuario(credenciales de usuario)
DB de usuarios, conseguir punto terminal de instancia local desde UserID
AuthenticateUser (UserCredentials)
Resultado de autenticación
Resultado de autorización
checkAppAUser (AppID, UserID, alcance)
Resultado
Interacciones para comprobar que el usuario puede usar la aplicación (es decir, comprobar condiciones como edad, suscripciones, etc.)
Resultado
AuthorizeApplication (AppId, UserId, alcance)
AuthorizeApplication (AppId, UserId, alcance)
authorizationCode
AC DB, almacenar AC
-relación de instancia local
authorizationCode
Usuario autoriza la aplicación
HTTP 302- Redireccionar a la aplicación (authorizationCode + estado)
getAccessToken (AppCredentials, AC)
Validar firma
getAccessToken (AppCredentials, AC)
AccessToken + alcance
AccessToken + alcance
AT DB, almacenar AT
-relación de instancia local

FIG. 8

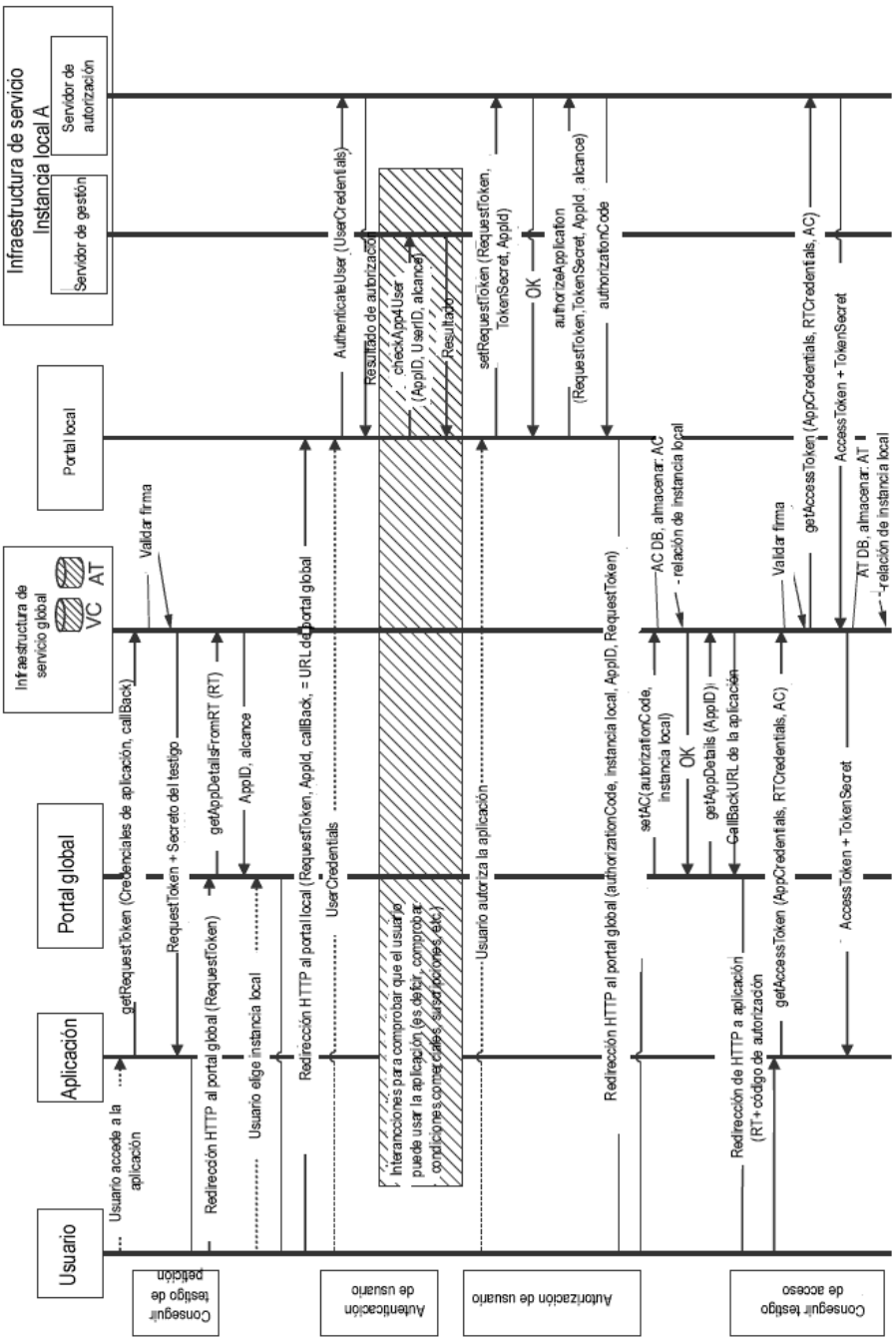
Usuario
Aplicación
Portal global
Infraestructura de servicio global
VC AT
Portal local
Infraestructura de servicio Instancia local A
Servidor de gestión
Servidor de autorización
Usuario accede a la aplicación
getRequestToken (Credenciales de aplicación, callBack)
RequestToken + Secreto del testigo
Redirección HTTP al portal global (RequestToken)
getAppDetailsFromRT (RT)
AppID, alcance
Usuario elige instancia local
Validar firma
Redirección HTTP al portal local (RequestToken, AppId, callBack, = URL del portal global
UserCredentials
Interacciones para comprobar que el usuario puede usar la aplicación (es decir, comprobar condiciones como cuotas, suscripciones, etc.)
AuthenticateUser (UserCredentials)
Resultado de autorización
check/App4User (AppID, UserID, alcance)
Resultado
Usuario autoriza la aplicación
setRequestToken (RequestToken, TokenSecret, AppId)
OK
authorizeApplication (RequestToken,TokenSecret, AppId , alcance)
authorizationCode
Redirección HTTP al portal global (authorizationCode, instancia local, AppID, RequestToken)
setAC(authorizationCode, instancia local)
AC DB, almacenar: AC - relación de instancia local
OK
getAppDetails (AppID)
CallBackURL de la aplicación
Redirección de HTTP a aplicación (RT + código de autorización
Validar firma
getAccessToken (AppCredentials, RTCredentials, AC)
getAccessToken (AppCredentials, RTCredentials, AC)
AccessToken + TokenSecret
AccessToken + TokenSecret
AT DB, almacenar: AT - relación de instancia local
Conseguir testigo de petición
Autenticación de usuario
Autorización de usuario
Conseguir testigo de acceso

FIG. 9

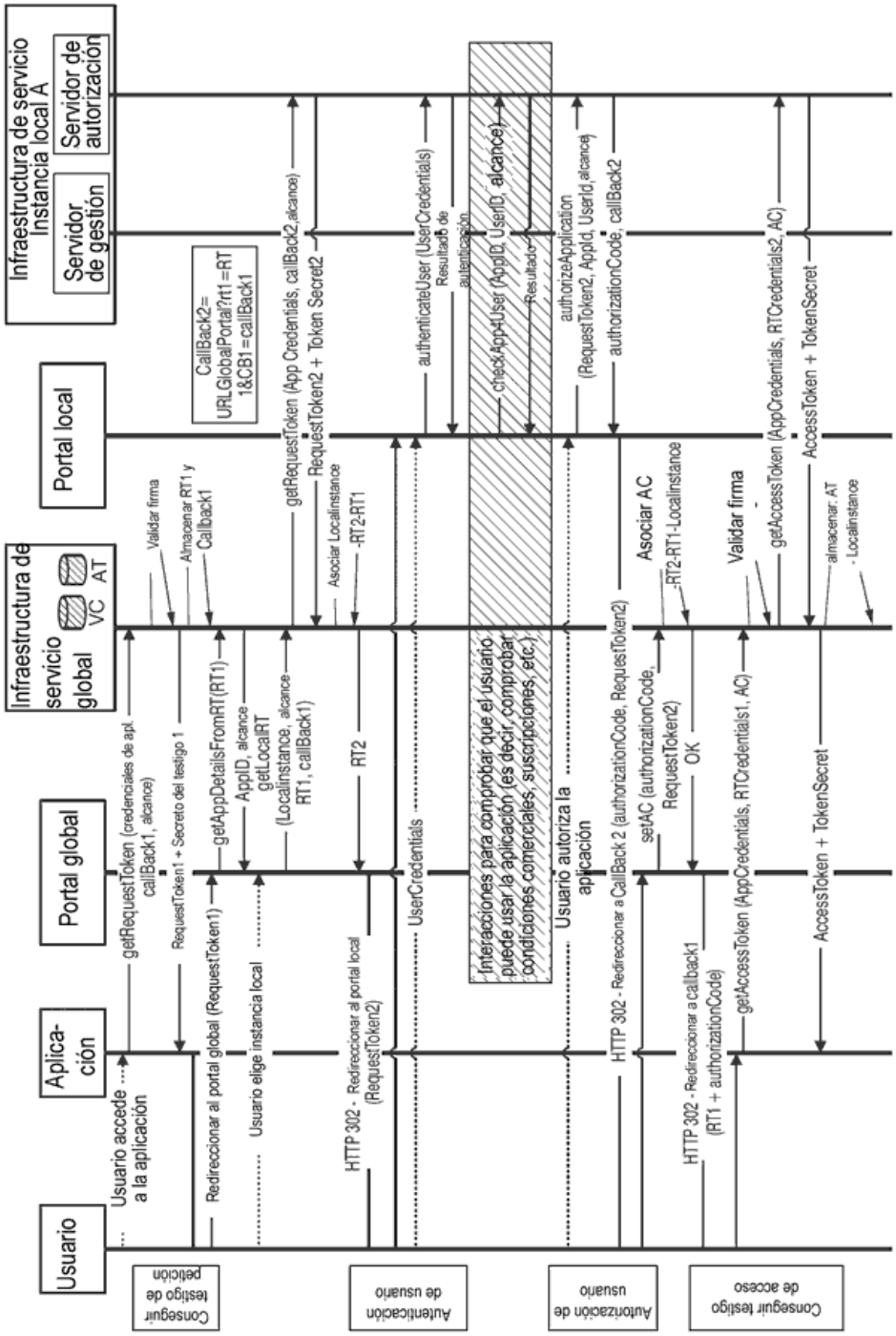
Usuario
Aplica-ción
Portal global
Infraestructura de servicio global
VC
AT
Portal local
Infraestructura de servicio Instancia local A
Servidor de gestión
Servidor de autorización
Usuario accede a la aplicación
getRequestToken (credenciales de apl. callBack1, alcance)
Validar firma
Almacenar RT1 y CallBack1
CallBack2= URLGlobalPortal?rt1=RT 1&CB1=callBack1
getRequestToken (App Credentials, callBack2,alcance)
RequestToken1 + Secreto del testigo 1
Redireccionar al portal global (RequestToken1)
getAppDetailsFromRT(RT1)
AppID, alcance
RequestToken2 + Token Secret2
Asociar LocalInstance
-RT2-RT1
authenticateUser (UserCredentials)
Resultado de autenticación
checkAppHUser (AppID, UserID, alcance)
Resultado
authorizeApplication (RequestToken2, AppId, UserId, alcance)
authorizationCode, callBack2
Usuario elige instancia local
getLocalRT (LocalInstance, alcance RT1, callBack1)
Asociar AC
-RT2-RT1-LocalInstance
Validar firma
-
getAccessToken (AppCredentials, RTCredentials2, AC)
AccessToken + TokenSecret
HTTP 302 - Redireccionar al portal local (RequestToken2)
RT2
UserCredentials
Interacciones para comprobar que el usuario puede usar la aplicación (es decir, comprobar condiciones comerciales, suscripciones, etc.)
almacenar: AT
- LocalInstance
Usuario autoriza la aplicación
HTTP 302 - Redireccionar a CallBack 2 (authorizationCode, RequestToken2)
setAC (authorizationCode, RequestToken2)
OK
HTTP 302 - Redireccionar a callback1 (RT1 + authorizationCode)
getAccessToken (AppCredentials, RTCredentials1, AC)
AccessToken + TokenSecret
Conseguir testigo de petición
Autenticación de usuario
Autorización de usuario
Conseguir testigo de acceso

FIG. 10

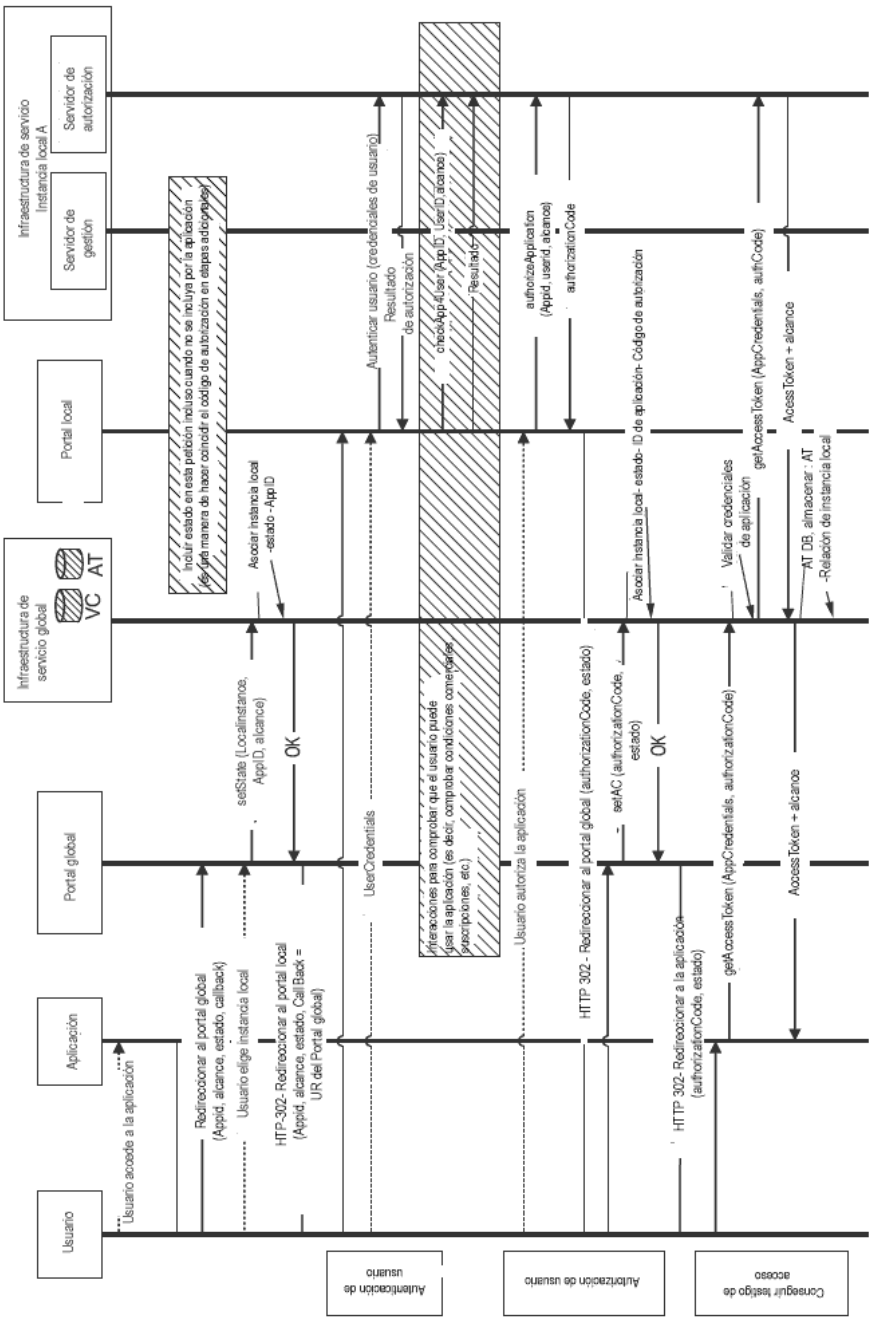
Usuario
Aplicación
Portal global
Infraestructura de servicio global
VC
AT
Portal local
Infraestructura de servicio Instancia local A
Servidor de gestión
Servidor de autorización
Usuario accede a la aplicación
Redireccionar al portal global (Appid, alcance, estado, callback)
Usuario elige instancia local
HTTP-302- Redireccionar al portal local (Appid, alcance, estado, Call Back = UR del Portal global)
setState (LocalInstance, AppID, alcance)
OK
Asociar instancia local -estado - AppID
UserCredentials
Autenticar usuario (credenciales de usuario)
Resultado de autorización
Incluir estado en esta petición incluso cuando no se incluya por la aplicación de esta manera de hacer comodín el código de autorización en etapas adoptadas
Interacciones para comprobar que el usuario puede usar la aplicación (es decir, comprobar condiciones comerciales, suscripciones, etc.)
checkAppUser (AppID, UserID, alcance)
Resultado
Usuario autoriza la aplicación
HTTP 302 - Redireccionar al portal global (authorizationCode, estado)
setAC (authorizationCode, estado)
OK
authorizeApplication (Appid, userid, alcance)
authorizationCode
Asociar instancia local- estado- ID de aplicación- Código de autorización
HTTP 302- Redireccionar a la aplicación (authorizationCode, estado)
getAccessToken (authorizationCode)
AccessToken + alcance
Validar credenciales de aplicación
getAccessToken (AppCredentials, authCode)
AccessToken + alcance
AT DB, almacenar : AT -Relación de instancia local
Autenticación de usuario
Autorización de usuario
Consegur testigo de acceso

FIG. 11

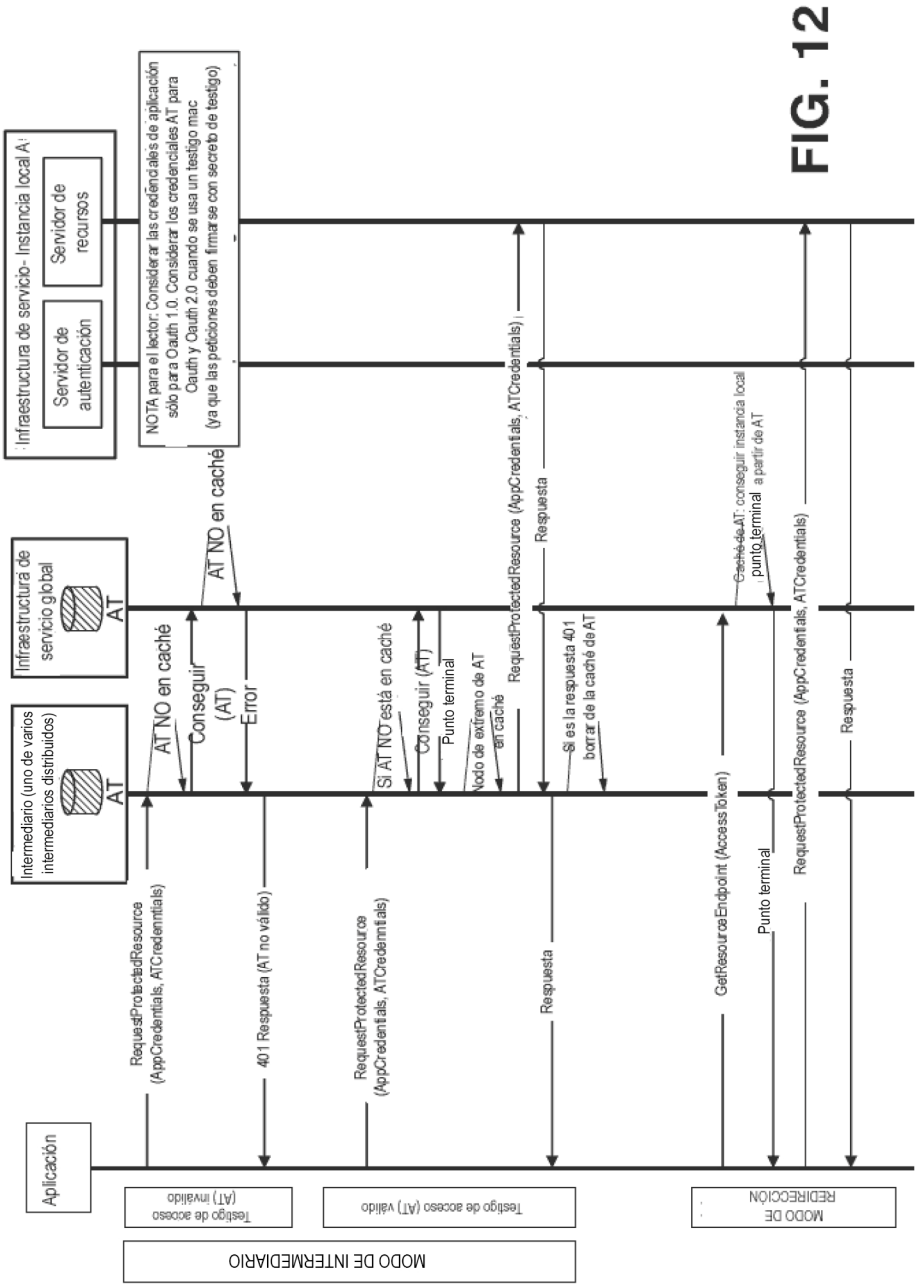
Aplicación
Intermediario (uno de varios intermediarios distribuidos)
AT
Infraestructura de servicio global
AT
Infraestructura de servicio- Instancia local A
Servidor de autenticación
Servidor de recursos
NOTA para el lector: Considerar las credenciales de aplicación sólo para Oauth 1.0. Considerar los credenciales AT para Oauth y Oauth 2.0 cuando se usa un testigo mac (ya que las peticiones deben firmarse con secreto de testigo)
RequestProtectedResource (AppCredentials, ATCredentials)
401 Respuesta (AT no válido)
RequestProtectedResource (AppCredentials, ATCredentials)
Respuesta
GetResourceEndpoint (AccessToken)
Punto terminal
RequestProtectedResource (AppCredentials, ATCredentials)
Respuesta
AT NO en caché
Conseguir (AT)
Error
AT NO en caché
Si AT NO está en caché
Conseguir (AT)
Punto terminal
Nodo de extremo de AT en caché
RequestProtectedResource (AppCredentials, ATCredentials)
Respuesta
Si es la respuesta 401 borrar de la caché de AT
Caché de AT: conseguir instancia local a partir de AT
punto terminal
Testigo de acceso (AT) inválido
Testigo de acceso (AT) válido
MODO DE REDIRECCIÓN
MODO DE INTERMEDIARIO

FIG. 12

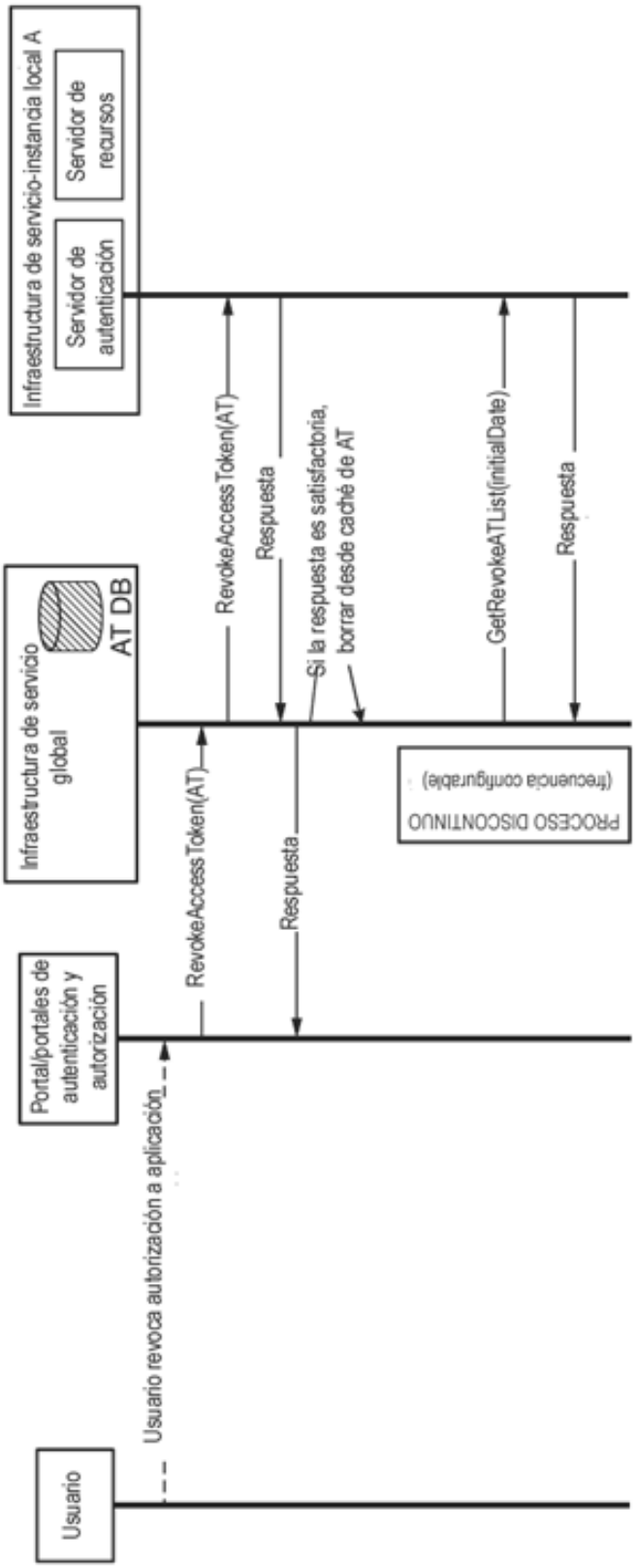
Usuario
Portal/portales de autenticación y autorización
Infraestructura de servicio global
AT DB
Infraestructura de servicio-instancia local A
Servidor de autenticación
Servidor de recursos
Usuario revoca autorización a aplicación
RevokeAccessToken(AT)
Respuesta
RevokeAccessToken(AT)
Respuesta
Si la respuesta es satisfactoria, borrar desde caché de AT
GetRevokeATList(initialDate)
Respuesta
PROCESO DISCONTINUO
(frecuencia configurable)

FIG. 13

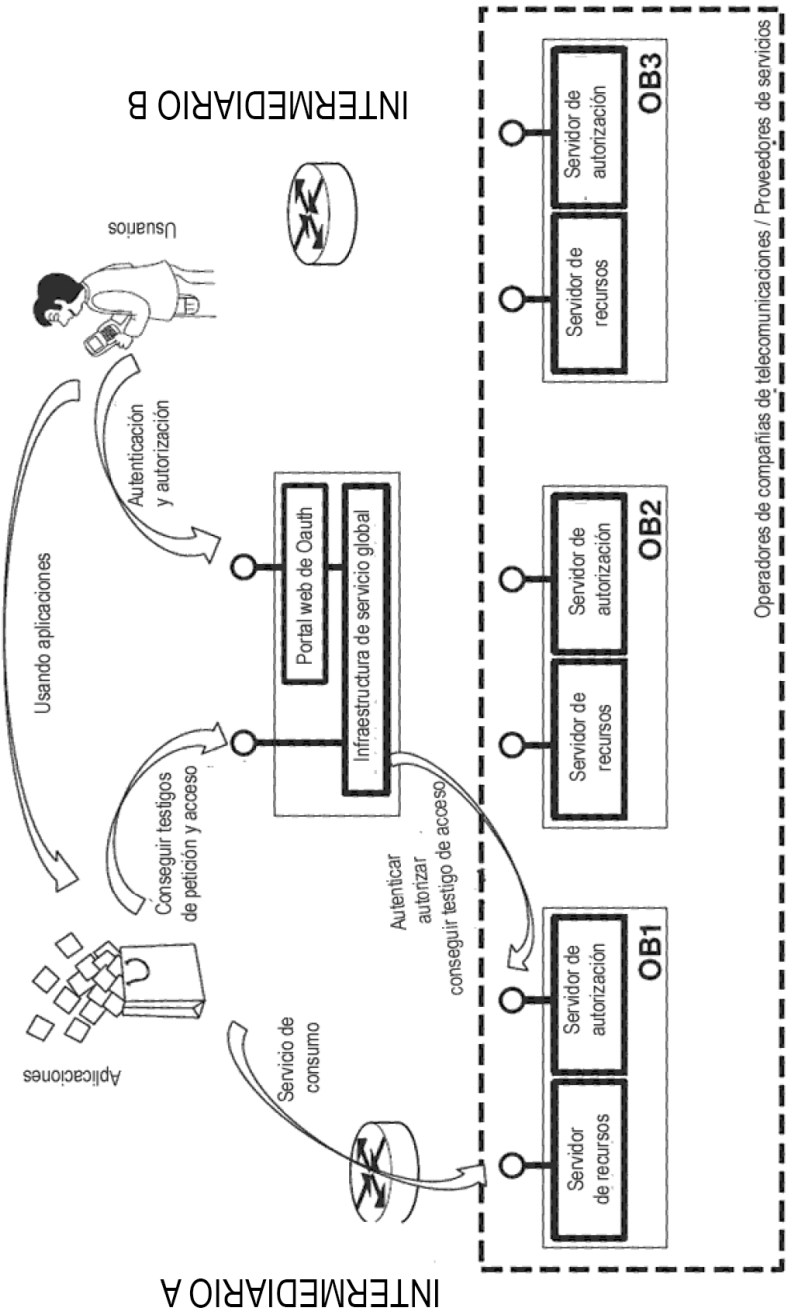
INTERMEDIARIO B
Usuarios
Autenticación y autorización
Usando aplicaciones
Portal web de Oauth
Infraestructura de servicio global
Conseguir testigos de petición y acceso
Autenticar autorizar conseguir testigo de acceso
Aplicaciones
Servicio de consumo
INTERMEDIARIO A
Operadores de compañías de telecomunicaciones / Proveedores de servicios
OB1
OB2
OB3
Servidor de autorización
Servidor de recursos

FIG. 14