UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR

FACULTAD DE INGENIERÍA Y GESTIÓN
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

“PROPUESTA PARA LA APLICACIÓN DEL FRAMEWORK OAUTH2 PARA LA SEGURIDAD DE UNA ARQUITECTURA BASADA EN MICROSERVICIOS EN LA EMPRESA EVERIS PERÚ S.A.C”

TRABAJO DE SUFICIENCIA PROFESIONAL
Para optar el Título Profesional de

INGENIERO DE SISTEMAS

PRESENTADO POR EL BACHILLER

DURAN CASTAÑEDA, BRYAN VINCENT

Villa El Salvador

2020

DEDICATORIA

El presente trabajo es dedicado a mis queridos padres Vicente y Ana María, por todo el apoyo incondicional brindado a lo largo de mi formación profesional.

AGRADECIMIENTOS

En primer lugar, dar el agradecimiento a mis padres, por darme de su apoyo incondicional y las fuerzas para nunca darme por vencido y continuar con mis estudios profesionales.

A la Universidad Nacional Tecnológica de Lima Sur (UNTELS) por darnos la oportunidad de tener acceso a educación profesional a los jóvenes de Villa El Salvador y por buscar siempre la excelencia de sus estudiantes.

ÍNDICE

LISTADO DE FIGURAS

LISTADO DE TABLAS

RESUMEN

En la actualidad cada vez más empresas optan por pasar de una arquitectura monolítica a una basada en microservicios, por lo cual es necesario que se implementen mecanismos para que solo los usuarios que se encuentran correctamente autorizados puedan acceder a la información que se almacena en los recursos internos.

Everis Perú S.A.es una de las empresas más aceptadas y con mayor facturación en Perú que brinda servicios de consultoría TI a destacadas entidades privadas y estatales en el Perú. Uno de estos servicios es el de desarrollar aplicaciones web las cuales se encargan de solucionar alguna problemática o la de realizar una nueva funcionalidad para lo cual es necesario que el desarrollo de las Apis backend utilicen mecanismos de seguridad para poder controlar el acceso de los usuarios o aplicaciones que solo puedan acceder a ellos.

Actualmente uno de los frameworks más famosos y de mayor aceptación para realizar flujos de autorización es el framework OAuth2 verificando lo que el usuario pueda hacer. Así mismo la aplicación de Openid Connect complementara el uso de OAuth permitiéndonos tener la información del usuario para validar su identidad.

Se espera que la aplicación del framework OAuth2 en el desarrollo de aplicaciones web pueda mejorar la seguridad de una arquitectura basada en microservicios permitiendo que haya ciertas reglas para poder acceder a los recursos internos de una aplicación.

INTRODUCCIÓN

La información es uno de los activos más valiosos para una empresa y asegurar que estos datos se encuentren debidamente resguardados es algo primordial. En la actualidad existen un gran número de microservicios los cuales son creados para poder administrar una gran cantidad de información necesaria para que cada uno de estos microservicios pueda realizar una tarea en específica como: actualizar datos de un cliente, enviar correos electrónicos, consultar datos del cliente, etc. Y su vez permitir que esta información pueda ser consultada por aplicaciones web internas o externas o inclusive por otros microservicios.

Para esto es necesario que esta información solo sea accedida por a aquellos usuarios que están autorizados correctamente. Al usar OAuth estamos delegando a la aplicación la capacidad para que esta pueda realizar acciones en su nombre y debido a que existen distintos tipos de aplicaciones OAuth nos brinda diferentes formas de obtener un token de acceso para que una aplicación acceda a un recurso.

El presente proyecto de investigación permite usar el protocolo de autorización OAuth2 para conceder acceso a la aplicación que está solicitando el acceso a los recursos almacenados en un servidor de aplicación, para esto el usuario debe de encontrarse debidamente autenticado para lo cual nos apoyaremos en el uso de OpenId Connect el cual nos permite obtener información del usuario que recibe la autorización y puede ser verificada.

El presente proyecto está dividido en 3 capítulos: el capítulo I se describe el objetivo general y los objetivos específicos a desarrollar a lo largo del proyecto de investigación. En el capítulo II hacemos referencia a los antecedentes de investigaciones anteriores que guardan relación con nuestra investigación así como desarrollamos el marco teórico el cual nos dará la base sobre la cual desarrollaremos el proyecto. Y finalmente en el capítulo II desarrollamos la metodología planteada para el desarrollo de nuestro trabajo de investigación así como las herramientas necesarias para el cumplimiento de los objetivos planteados en el capítulo I.

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

1.1. Objetivos

1.1.1. Objetivos General

- Proponer un modelo para la aplicación framework OAuth2 para mejorar la seguridad de una arquitectura basada en microservicios en la empresa Everis Peru S.A.C.

1.1.2. Objetivos Específicos

- Definir el alcance de autorización de las aplicaciones para mejorar la seguridad de una arquitectura basada en microservicios en la empresa Everis Peru S.A.C.

- Conceder a las aplicaciones internas acceso seguro a través del proceso de autenticación de una arquitectura basada en microservicios en la empresa Everis Peru S.A.C.

- Gestionar el ciclo de vida de token de acceso para mejorar la seguridad de una arquitectura basada en microservicios en la empresa Everis Peru S.A.C.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes

A continuación se citaran algunos antecedentes nacionales e internacionales que hacen referencia al tema de investigación propuesto.

2.1.1. Antecedentes Nacionales

En el contexto nacional se detallan las siguientes investigaciones:

- Según Feria (2018) en su investigación titulada *"Seguridad para el Control a Recursos de la Aplicación Web de Facturación Electrónica OPENFACT, AHREN Contratistas Generales - Ayacucho, 2017"*, esta investigación es del tipo aplicada a nivel descriptiva, la cual tiene como muestra el servicio web que se encarga de la generación de comprobantes electrónicos. Implemento un sistema de autenticación, autorización y trazabilidad permitiendo un adecuado control de acceso a los distintos recursos del sistema aplicando el protocolo de seguridad OAuth2; el cual tiene como principal objetivo identificar la identidad de los usuarios durante el proceso de autenticación para garantizar un adecuado control de acceso a recursos del sistema. Las conclusiones obtenidas demostraron que la implementación de la capa de seguridad en la aplicación web (protocolos OAuth2 y OpenID) garantizo un adecuado control de acceso a los recursos del sistema así como el uso de del protocolo OpenID ayudo a identificar la identidad de los usuarios que hacen uso de los componentes de la aplicación web colaborando con el control de acceso a los recursos del sistema. Usando el control de acceso basado en roles (autorización) se logró proteger los recursos de la aplicación, esto colaboro en evitar el uso no autorizado a los recursos del sistema.

La presente investigación aporta al proyecto ya que da una directriz de cómo implementar una capa de seguridad para tener un mejor control de acceso a los recursos del sistema por parte de terceros aplicando el protocolo OAuth2.

- Según Contreras & Vega (2019) en su tesis titulada *"Implementación de un Sistema de Gestión de Identidades Privilegiadas para el Control de Acceso en una Empresa Retail"*, investigación que es del tipo aplicada y se implementó el método

Plan-DO-Check-Act recomendado por la ISO/IEC 27001, el cual es un Sistema de Gestión de Seguridad de Información. Implemento un sistema de gestión de acceso privilegiado la cual permitió tener una gestión de las contraseñas efectiva, para poder controlar el acceso a los activos de la información, aplicando ciertas políticas para tener el control y registro de las actividades de los usuarios además de conocer la trazabilidad de los cambios realizados en los activos de la tecnología de la información (TI) mejorando la operatividad de los usuarios debido a que dentro de la empresa retail no se tiene una buena administración de las identidades haciendo que la empresa sea vulnerable ataques informáticos. Esta investigación tuvo como principal objetivo mejorar la gestión de identidades privilegiadas de los administradores de la plataforma de TI en la empresa retail, para evitar el acceso no autorizado a los sistemas de la infraestructura de TI de la empresa además esto permitió tener un acceso seguro de las identidades privilegiadas a los administradores de TI. De la presente investigación se concluyó que para reducir la extracción de la información por usuarios que no están autorizados es necesario tener una administración basada en roles además se logró reducir el tiempo de autenticación en los activos TI debido a que se maneja una función Single Sing-On para centralizar y administrar las credenciales de manera correcta.

Esta investigación aporta al proyecto ya que indica que para poder prevenir la extracción de información es necesario implementar de un sistema de seguridad para controlar el acceso no autorizado a los recursos internos del sistema para lo cual se implementó Single Sing-On pero podría utilizarse OAuth2 para realizar esta tarea además de proporcionar ciertos privilegios a los usuarios para recuperar solo la información a la que está autorizada visualizar.

2.1.2. Antecedentes Internacionales

Asimismo, en el contexto internacional se detallan las siguientes investigaciones:

- Según Castillo (2018), en su tesis titulada *"Diseño e Implementación de una Estrategia de Seguridad Mediante Políticas de Autenticación y Autorización para una empresa de Seguros",* implemento un sistema que provea un mecanismo para comunicarse y validar las peticiones de terceros, permitiendo asegurar el correcto

control de accesos a los recursos que se comparten a terceros, teniendo en cuenta que este mecanismo tiene que ser capaz de operar sobre aplicaciones móviles debido a que la empresa actual no hay un forma segura para que terceros puedan acceder a las APIs de la empresa de seguros por lo que no pueden acceder a los recursos que expone la API. Además de que cada aplicación de la empresa maneja distintas credenciales por lo que un usuario podría de manejar muchas credenciales para acceder a los recursos haciéndolo un proceso tedioso para el usuario. Esta investigación tiene como objetivo principal de esta investigación fue la de diseñar e implementar una estrategia de seguridad mediante el uso de políticas de autenticación y autorización para la empresa de seguros.

Este trabajo de investigación se apoyó en el uso del estándar OAuth2 para realizar la gestión de identidad y autorización de la cual se obtuvieron las siguientes conclusiones: la aplicación del protocolo OAuth2 permitió evitar la propagación de las credenciales por cada aplicación, además se logró limitar el acceso haciéndolo configurable y sobre todo se le otorgo la capacidad al usuario final de revocar el acceso en cualquier momento además se logró optimizar el proceso de autenticación a aplicaciones de terceros; para el caso de la autorización el usuario podrá utilizar las mismas credenciales para autenticarse en las aplicaciones de terceros para evitar tener distintas credenciales por cada aplicación de tercero.

La presente investigación aporta a nuestro proyecto una guía para realizar la implementación del protocolo OAuth2 para gestionar de manera eficaz el acceso de terceros a las APIs internas del sistema.

2.2. Bases teóricas

2.2.1. Seguridad

Según The SANS Institute (2020), define la seguridad de la información como el conjunto de procesos y metodologías diseñados e implementados para resguardar las distintas formas de información: impresa, electrónica u otro tipo de información o datos privados los cuales pueden ser usados, divulgados, destruidos, modificados o interrumpidos sin la debida autorización.

2.2.1.1. Tríada de la Seguridad de la Información

Según Siriwardena (2020), afirma que existen tres factores fundamentales para realizar una evaluación comparativa de la seguridad en los sistemas de información; confidencialidad, integridad y disponibilidad (CIA). Estos tres factores permiten diseñar un modelo de seguridad así como evaluar uno ya existente.

Según Andress (2011), afirma que la triada de la CIA nos proporciona un modelo para poder pensar y razonar conceptos de seguridad, esto por lo general se encuentra estrechamente relacionado con la seguridad de datos.

- **Confidencialidad**

  La confidencialidad tiene un concepto semejante, pero no igual a la privacidad. Siendo la confidencialidad un elemento imprescindible de la privacidad y hace referencia a nuestra facultad para poder resguardar nuestros datos de aquellos que no están autorizados para verlos, pudiendo esta implementarse en varios niveles de un proceso.

- **Integridad**

La integridad se define como la facultad de prevenir que nuestra información pueda ser modificada de forma no autorizada. Esto puede incluir modificar o eliminar información así sea autorizada, por eso la integridad también abarca no solo la capacidad para evitar la manipulación no deseada de nuestros datos sino que también poder ser capaz de revertir los cambios autorizados que se desean deshacer.

- **Disponibilidad**

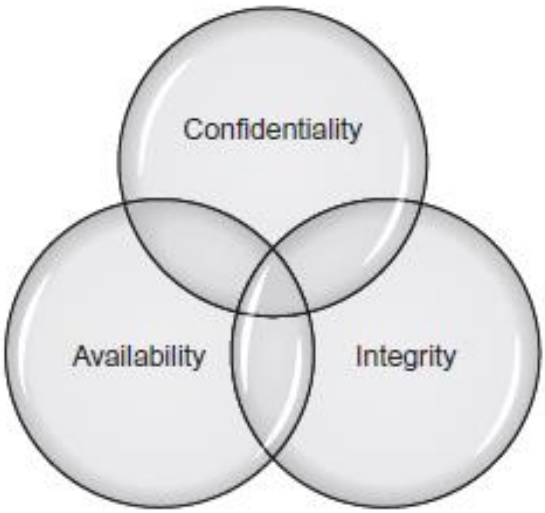

Figura 1. La Triada de la Seguridad de la Información

Fuente: (Andress, 2011)

2.2.2. Control de Accesos

El control de acceso es un proceso por el cual se reguardan los recursos para que solo puedan ser usados por aquellos usuarios que poseen el permiso correspondiente para acceder a ellos, protegiendo el recurso de accesos no autorizados. Estos controles definen cuatro puntos: quienes son los usuarios, que tienen permitido hacer estos usuarios, a que recursos pueden acceder y que operaciones pueden realizar sobre los recursos. Algunas tecnologías que utilizan estos controles de accesos incluyen contraseñas, tokens de hardware, datos biométricos y certificados (Kim & Solomon, 2012, pág. 142).

2.1.1.1. Tipos de Accesos

Según Pons (Pons Martorell, s.f) define que uno de los términos utilizados dentro del control de accesos es la direccionalidad del mismo, el control de accesos permite que haya unidereccionalidad cuando existe un control diferente para ir de A a B que para ir de B a A (ver Figura 1).

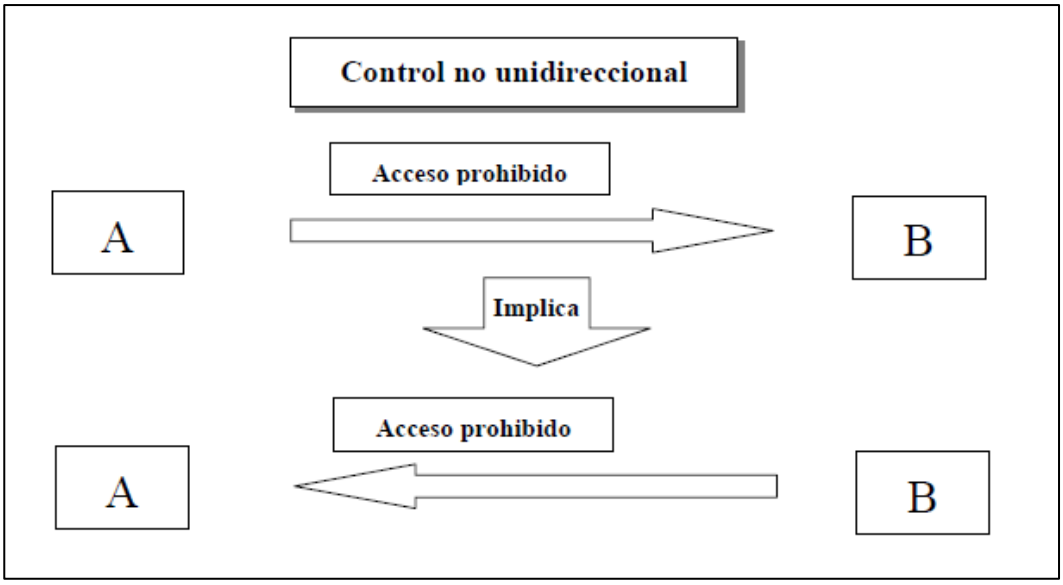

Figura 2. Control no Unidireccional

Fuente: (Pons Martorell, s.f)

De acuerdo a Pons (Pons Martorell, s.f), inclusive en algunos sistemas de control no permiten esta unidereccionalidad, es decir existe un control para ir de A a B y existe el mismo control para ir de B a A (ver Figura 2).

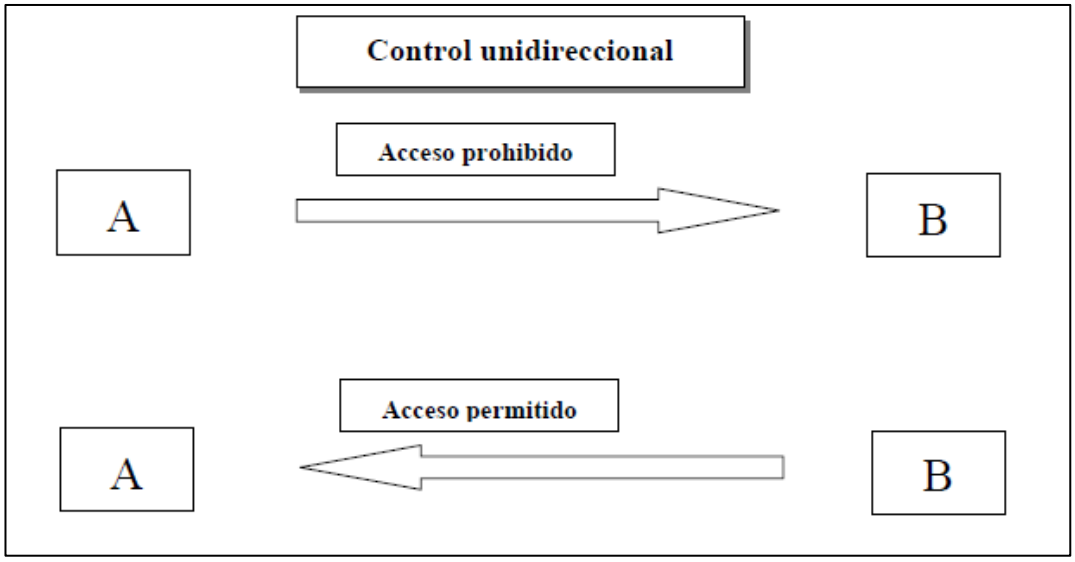

Figura 3. Control Unidireccional

Fuente: (Pons Martorell, s.f)

2.1.1.2. Clasificación

Según Pons (s.f), define que se pueden agrupar los sistemas de control en dos familias:

- **En el Servidor**

  Se instalan filtros en el servidor u ordenador que almacena la información a través de un software o en el mismo sistema operativo, esto permite realizar una selección de aquellos usuarios que pueden acceder físicamente al servidor.

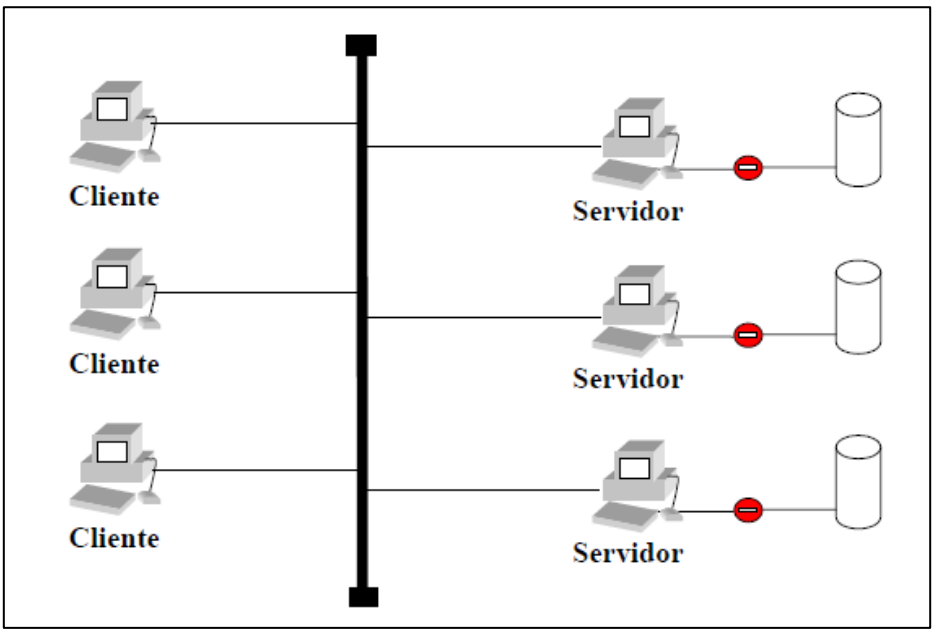

Figura 4. Control en el Servidor

Fuente: (Pons Martorell, s.f)

- **En la Red**

  Se colocan filtros en la red que realizan el control de accesos de máquinas remotas. Estos filtros son colocados en switchs LAN, routers o firewalls, con esto podemos filtrar grupos y zonas en el cual solo se filtran los accesos.

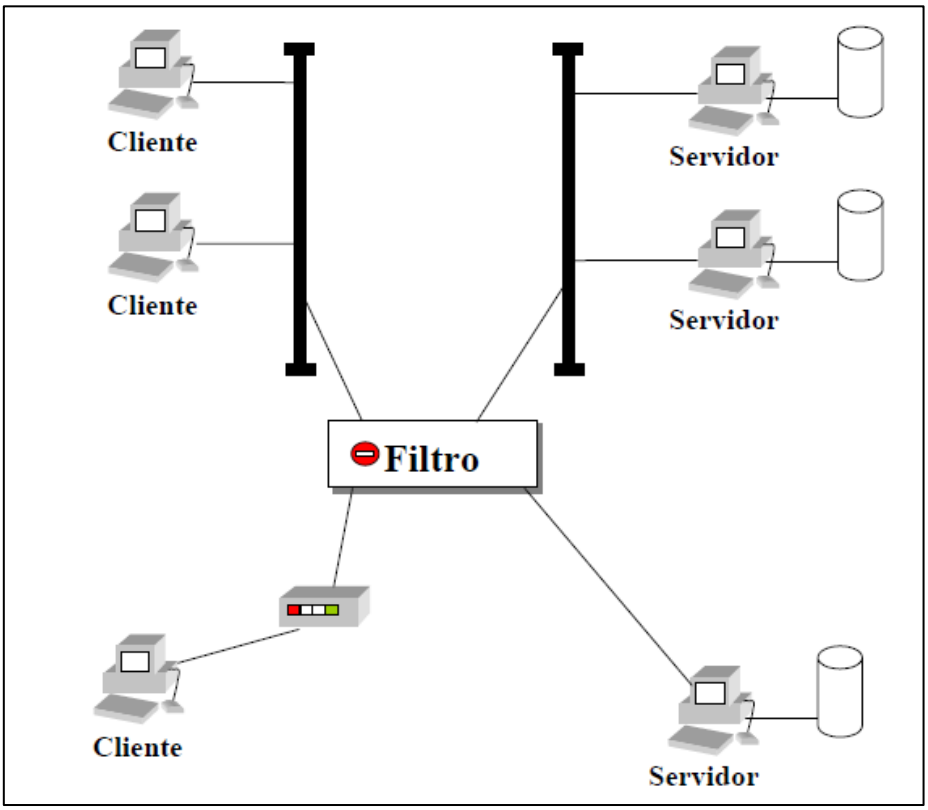

Figura 5. Control en la Red

Fuente: (Pons Martorell, s.f)

2.1.1.3. Tareas del Control de Accesos

Según Andress (2011), existen cuatro tareas básicas que se deben de implementar cuando se hace referencia al control de accesos:

- **Permitir el Acceso**

  Es otorgar acceso a una o varias partes en particular de un determinado recurso del sistema.

- **Denegar el Acceso**

  Es lo contrario a permitir el acceso de una o varias partes de algún recurso en particular. Es decir negar el acceso de usuarios no autorizados a los recursos del sistema.

- **Limitar el Acceso**

Es solo otorgar cierto acceso sobre los recursos hasta cierto punto. Es decir que el usuario no tiene acceso a toda la información sino que solo tiene acceso a lo que se le ha permitido acceder.

- **Revocar el Acceso**
  Es quitarle el acceso a un recurso determinado luego de haberle concebido el acceso.

2.2.3. Autenticación

Según Siriwardena (2020), la autenticación se define como un proceso en el cual se comprueba que un usuario, sistema o cosa es quien dice ser, la autenticación puede ser directa o intermediada según como sean las afirmaciones de autenticidad:

- **Autenticación Directa**
  Se define como una autenticación directa cuando un usuario otorga de manera directa sus credenciales (usuario y contraseña) a un sistema, es decir la entidad que desea autenticarse otorga las aseveraciones de autenticación al servicio al cual desea tener acceso.

- **Autenticación Intermedia**
  Se define como autenticación intermediada cuando hay un tercero (proveedor de identidad) involucrado en el proceso de autenticación. Por ejemplo si deseamos acceder a Yelp través de nuestra cuenta de Facebook este será el proveedor de identidad y el sitio web al que se desea acceder será el proveedor de servicio. Utilizando la autenticación intermediada se permitirá el acceso solo si el proveedor de identidad da una afirmación positiva al proveedor del servicio permitiendo así el acceso a su contenido.

2.1.1.4. Categorías de Autenticación

Según Siriwardena (2020), la autenticación se puede dar en uno o en múltiples factores (autenticación multifactor). Para ser considerado una autenticación multifactor al menos debe utilizarse una combinación de dos factores y no se considera una autenticación multifactor si dos técnicas pertenecen a la misma categoría (pin o un usuario y contraseña). A continuación se definen las categorías de autenticación:

- **Algo que sabes**

  Es una de las formas más populares de autenticación, incluye usuarios, contraseñas y números de PIN que el usuario conoce. A la vez es la forma de autenticación más vulnerable debido a que se puede usar ciertas combinaciones para dar con la contraseña, esto es conocido como ataque de fuerza bruta.

- **Algo que tienes**

  Es un forma de autenticación mucho más robusta que la autenticación basado en algo que se sabe (contraseñas), es algo que el usuario tiene o puede generar. Dentro de esta categoría se consideran los tokens, tarjetas de identificación y tarjetas de cajeros automáticos.

- **Algo que eres**

  Esta categoría comprende a todas las técnicas de autenticación basadas en biometría (huellas dactilares, biometría facial, retina ocular, etc.) siendo esta la forma más segura de autenticación.

2.2.4. Autorización

Según Ur Rehamn (2008), define autorización como un proceso en el cual se concede o rechaza el acceso a los recursos informáticos, es decir se habilita el control de acceso únicamente a aquellos que poseen una necesidad genuina de obtener acceso y que por lo general se da luego del proceso de autenticación.

Lo primero que se valida en una aplicación web es la identidad al iniciar sesión para asegurar que solo se conceda el acceso a los datos y servicios para los cuales está autorizado mediante una lista de control de accesos para cada operación (Boyd, 2012).

2.2.5. Microservicios

Los microservicios son elementos independientes que se encuentran construidos bajo un dominio empresarial para cumplir con tareas determinadas, estos se comunican entre sí brindando varias opciones para poder resolver problemas. Una arquitectura basada en microservicios se refiere a varios microservicios cumpliendo ciertas tareas determinas y comunicándose entre sí de manera colaborativa (Newman, 2019).

2.2.6. OAuth 2.0

OAuth 2.0 es un protocolo de delegación, un medio para permitir que alguien que controla un recurso permita que una aplicación de software acceda a ese recurso en su nombre sin hacerse pasar por él (Richer & Sanso, 2017).

El marco de autorización de OAuth 2.0 permite que una aplicación de terceros obtenga acceso limitado a un servicio HTTP, ya sea en nombre del propietario de un recurso al orquestar una interacción de aprobación entre el propietario del recurso y el servicio HTTP, o al permitir que la aplicación de un tercero para obtener acceso en su propio nombre (IETF, 2012).

OAuth 2.0 allows an application to access resources (typically personal information) protected by a resource server on behalf of the resource owner, through the consumption of an access token issued by an authorization server (Li, Mitchell, & Chen, 2018).

2.2.6.1. Roles de OAuth

Dentro del marco de OAuth se definen 4 roles básicos: el propietario del recurso, el cliente, el servidor de recursos y el servidor de autorización.

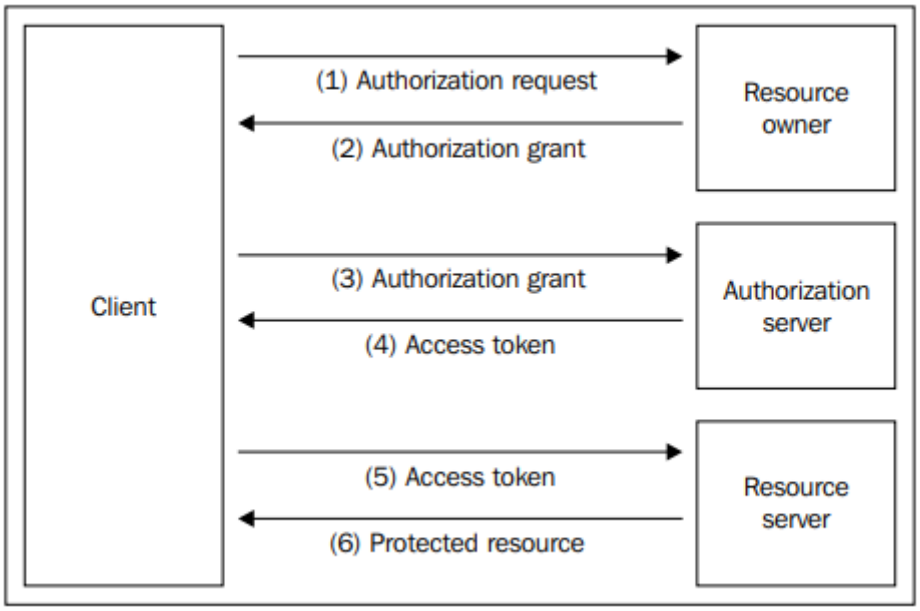

Figura 6. Roles utilizados en la arquitectura OAuth

Fuente: (Torroglosa-García, Pérez-Morales, Martinez-Julia, & Lopez, 2013)

2.2.6.1.1. Propietario del Recurso

El propietario de un recurso es una entidad que puede otorgar acceso a un recurso protegido. Cuando esta entidad es una persona, también se puede utilizar el término usuario final o usuario (Spasovski, 2013).

2.2.6.1.2. Cliente

Un cliente es una aplicación que realiza solicitudes a recursos protegidos en el servidor de recursos, en nombre del propietario del recurso (el usuario).

Una aplicación cliente puede ser una aplicación web, una aplicación de escritorio, una aplicación móvil, etc. La especificación OAuth 2.0 no impone limitaciones con respecto al entorno en el que debe ejecutarse la aplicación cliente (Spasovski, 2013).

2.2.6.1.3. Servidor de Recursos

Un servidor de recursos es el que atiende los recursos protegidos, a los que se debe acceder mediante solicitudes autorizadas desde una aplicación cliente (Spasovski, 2013).

2.2.6.1.4. Servidor de Autorización

Para que un cliente pueda acceder a los recursos protegidos, esto primero debe ser autorizado por el propietario del recurso.

Esto es lo que hace el servidor de autorización: le pide al usuario (propietario del recurso) que confirme que el cliente debe estar autorizado para tener acceso. Por cada solicitud de autorización exitosa, el servidor emite al cliente un "token de acceso". Es un token que el cliente usa para especificar para qué usuario realiza las solicitudes (Spasovski, 2013).

2.2.6.2. OAuth Grant Types

2.2.6.2.1. Authorization Code

Este tipo de concesión es más apropiado para aplicaciones web del lado del servidor. Una vez que el propietario del recurso ha autorizado el acceso a sus datos, se le redirige a la aplicación web con un código de autorización como parámetro de consulta en la URL. La aplicación cliente debe intercambiar este código por un token de acceso. Este intercambio se realiza de servidor a servidor y requiere tanto el client_id como el client_secret, lo que evita que incluso el propietario del recurso obtenga el token de acceso (Boyd, 2012).

2.2.6.2.2. Client Credentials

El tipo de concesión de credenciales de cliente permite que una aplicación obtenga un token de acceso para los recursos que son propiedad del cliente o cuando la autorización se ha "acordado previamente con un servidor de autorización". Este tipo de concesión es apropiado para aplicaciones que necesitan acceder a API, como servicios de almacenamiento o bases de datos, en nombre de sí mismas y no en nombre de un usuario específico (Boyd, 2012).

2.2.6.2.3. Device Code

El perfil de dispositivo se creó para permitir el uso de OAuth en dispositivos que no tienen navegadores web integrados o tienen opciones de entrada limitadas, como una consola de juegos o un marco de fotos electrónico. El usuario normalmente inicia el flujo en el dispositivo y luego se le dice que use una computadora para acceder a un sitio web y aprobar el acceso para el dispositivo escribiendo un código

de autorización que se muestra en el dispositivo. Facebook tiene un gran ejemplo de este flujo al que se hace referencia en su documentación (Boyd, 2012).

2.2.6.2.4. Refresh Token

Un token de actualización de OAuth es similar en concepto al token de acceso, ya que el servidor de autorización lo emite al cliente y el cliente no sabe ni le importa qué hay dentro del token. Sin embargo, lo que es diferente es que el token nunca se envía al recurso protegido. En cambio, el cliente usa el token de actualización para solicitar nuevos tokens de acceso sin involucrar al propietario del recurso (Richer & Sanso, 2017).

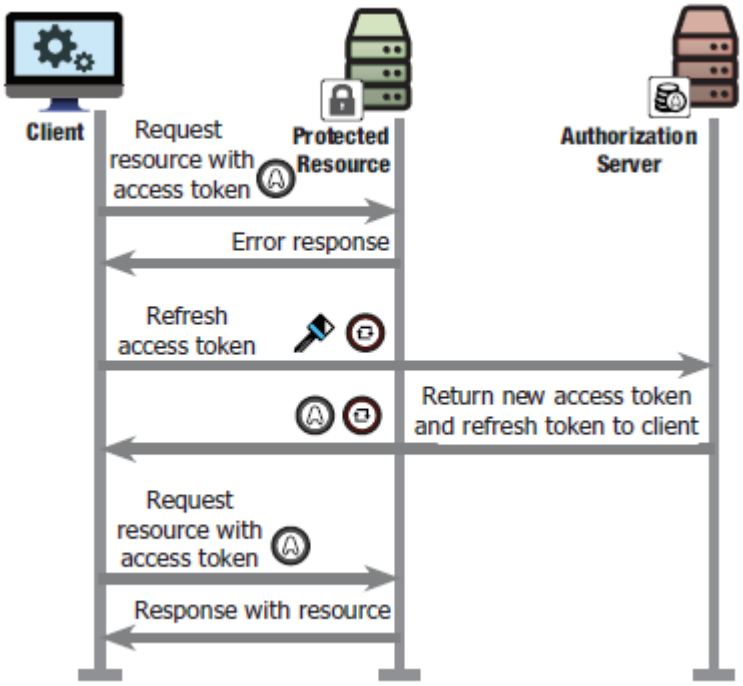

Figura 7. Flujo para refrescar el token de acceso

Fuente: (Richer & Sanso, 2017)

2.2.7. OpenId Connect

OpenID Connect es un estándar abierto publicado por OpenID Foundation en febrero de 2014 que define una forma interoperable de utilizar OAuth 2.0 para realizar la autenticación de usuarios (Richer & Sanso, 2017).

OIDC se diseñó como una capa sobre el protocolo OAuth 2.0 para proporcionar información en un formato estándar a las aplicaciones sobre la identidad de un usuario autenticado. Esto proporcionó una solución para las aplicaciones para la autenticación de usuarios, así como la autorización de API (Wilson & Hingnikar, 2019).

2.2.8. JSON WEBTOKEN (JWT)

JSON Web Token (JWT) es un medio compacto y seguro para URL para representar reclamaciones que se transferirán entre dos partes. Las afirmaciones en un JWT están codificadas como un objeto JSON que se utiliza como carga útil de una estructura JSON Web Signature (JWS) o como texto sin formato de una estructura JSON Web Encryption (JWE), lo que permite que las reclamaciones estén firmadas digitalmente o protegidas por integridad. con un código de autenticación de mensajes (MAC) y / o cifrado ((IETF), 2015).

2.3. Definición de términos básicos

- **OAuth2:** Es una capa protectora establecida en un servicio para que las aplicaciones clientes puedan acceder a los recursos del servicio o a los recursos que pertenecen a un usuario que utiliza el sistema, (Spasovski, 2013)**.**

- **Cliente:** Es una aplicación que envía solicitudes a recursos que se encuentran resguardados en un servidor de recursos, en nombre del dueño de los recursos, el usuario (Spasovski, 2013)**.**

- **Access Token:** Es solicitado ya se por una aplicación del lado del servidor, una aplicación web del lado del cliente o una aplicación nativa para poder hacer solicitudes de API en nombre del usuario o de la propia aplicación (Boyd, 2012).

CAPÍTULO III: METODOLOGÍA DEL TRABAJO PROFESIONAL

3.1. Delimitación del Proyecto

3.1.1. Teórica

El presente proyecto de investigación se enfocara en hacer más seguro la invocación de los microservicios, tanto para aplicaciones internas como aplicaciones externas utilizando el protocolo de autorización OAuth2 para el desarrollo del presente trabajo.

- **OAuth2:** Concede a una aplicación de tercero acceso limitado a un servicio HTTP, ya sea en nombre del propietario del recurso al orquestar la relación de aprobación entre el propietario del recurso y el servicio HTTP, o al conceder el acceso a una aplicación tercera en su propio nombre (IETF, 2012).

3.1.2. Temporal

El desarrollo del proyecto de investigación tendrá una duración de 4 meses, la cual está comprendido desde agosto del 2020 hasta diciembre del 2020.

3.1.3. Espacial

La presente investigación fue realizada dentro de la empresa Everis Perú S.A.C., Av. Defensores del Morro 1868 - Chorrillos, Lima.

3.2. Determinación y análisis del problema

3.2.1. Formulación del Problema

3.2.1.1. Problema General

- ¿De qué manera el modelo propuesto para la aplicación del framework OAuth2 mejora la seguridad de una arquitectura basada en microservicios en la empresa Everis Perú S.A.C.?

3.2.1.2. Problema Específico

- ¿Cómo la definición del alcance de la autorización permite mejorar el acceso seguro de aplicaciones internas o externas a los recursos a través del proceso de autorización?

- ¿Cómo la aplicación de OAuth2 concederá acceso seguro de aplicaciones internas o externas a los recursos a través del proceso de autenticación?

- ¿Cómo gestionar el ciclo vida de un token de acceso mejorar la seguridad de acceso de aplicaciones internas o externas a los recursos a través del proceso de autenticación?

3.2.1.3. Descripción de la realidad problemática

Asegurar que los datos de una empresa solo puedan ser consultados solo por los usuarios que se encuentran correctamente autenticados y que poseen los permisos necesarios para esto, es indispensable para asegurar los datos y que toda organización debería de implementar.

Actualmente no se controla el alcance que pueden tener las aplicaciones a los recursos, pudiendo una aplicación acceder a toda la informacion que expone un servidor de recursos por lo cual se plante definir el alcance para cada aplicación. Así como es muy fácil que una aplicación maliciosa pueda interceptar una petición y pueda hacerse pasar por una aplicación legitima asi robando informacion valiosa del servidor. Por tal motivo se implementa un método que otorga el framework OAuth2 para evitar esto y poder gestionar de manera eficiente los token que con esta extensión se generan.

Dentro de una arquitectura basada en microservicios, cada uno de estos microservicios tiene la función de realizar una tarea específica (consultar, actualizar, crear, eliminar, etc.) o la de realizar algún flujo especifico (registrar un usuario, enviar notificaciones, generar una clave, validar datos del cliente, etc.) por lo que es indispensable que estos microservicios se comuniquen entre sí para poder compartir información, para poder cumplir con el objetivo para los cuales fueron creados.

Pero no todos los microservicios o aplicaciones web deben de poder acceder a toda la información que expone cada uno de estos, sino que solamente a la información necesaria para que esta pueda ser procesada y expuesta a través de un api y para que otro microservicio o aplicación web pueda utilizarla.

Por lo ya antes mencionado esta investigación tiene como finalidad aplicar el framework de autorización OAuth2 para mejorar la seguridad y gestiona de manera correcta el acceso de las distintas aplicaciones web tanto internas como externas hacia los microservicios incluyendo estos mismos.

3.2.1.4. Justificación del problema

Proteger la información para que solo aquellas aplicaciones (internas o externas) que se encuentran debidamente autorizadas es primordial dentro de una arquitectura basada en microservicios, cada microservicio cumple con una tarea específica y es independiente pero es necesario que puedan comunicarse entre sí.

Usar OAuth2 para poder gestionar el acceso de cada aplicación a los recursos del sistema es muy útil, este framework proporciona distintos flujos de autorización dependiendo el tipo de autorización que se le requiera dar a quien solicita el acceso sea una aplicación web, una aplicación móvil o un api.

3.3. Modelo de solución propuesto

3.3.1. Definir Alcance de Autorización

Dentro de flujo de autorización se puede definir el alcance de las aplicaciones hacia los recursos del usuario. Podemos distinguir entre dos tipos distintas de solicitudes para la obtención de un token de acceso:

Token de Aplicación

La aplicación para poder acceder a los recursos del usuario sin necesidad de enviar datos de autenticación de un usuario, basta con enviar las credenciales de la aplicación al servidor de autenticación para que este pueda validar si todos los datos son correctos retornar un token de acceso para que la aplicación pueda acceder a los recursos de la aplicación.

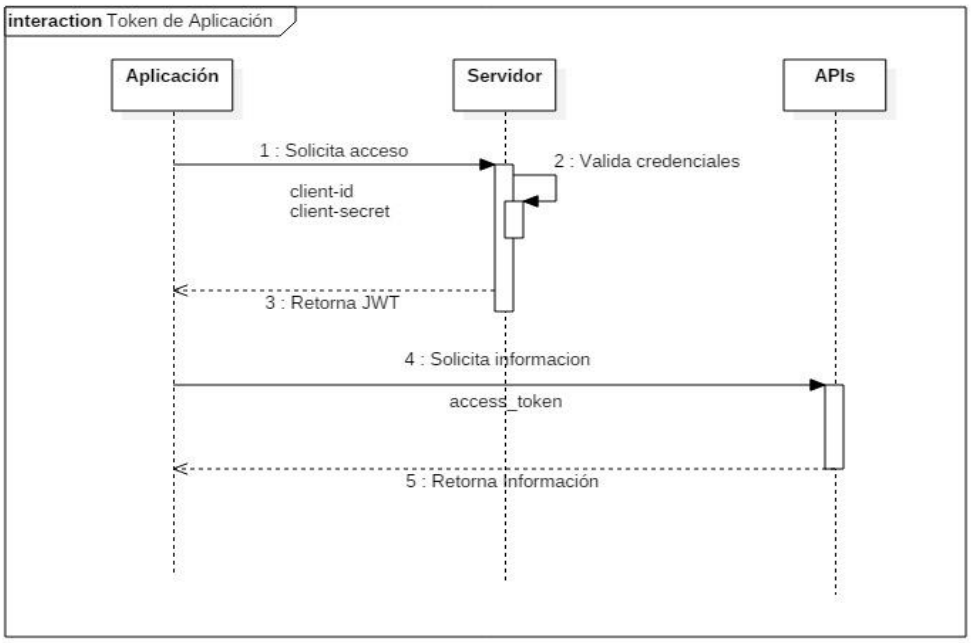

Figura 8. Diagrama de Secuencia Token de Aplicación

Fuente: Elaboración propia

Token de Usuario

La aplicación para acceder a los recursos del usuario primero necesita de autenticarse debidamente con un usuario y contraseña a través de una página de login, esto con el motivo de acceder a los recursos del usuario (APIs). Si las credenciales de autenticación son las correctas entonces se le brindara un código de autorización para poder hacer el intercambio por un token de acceso.

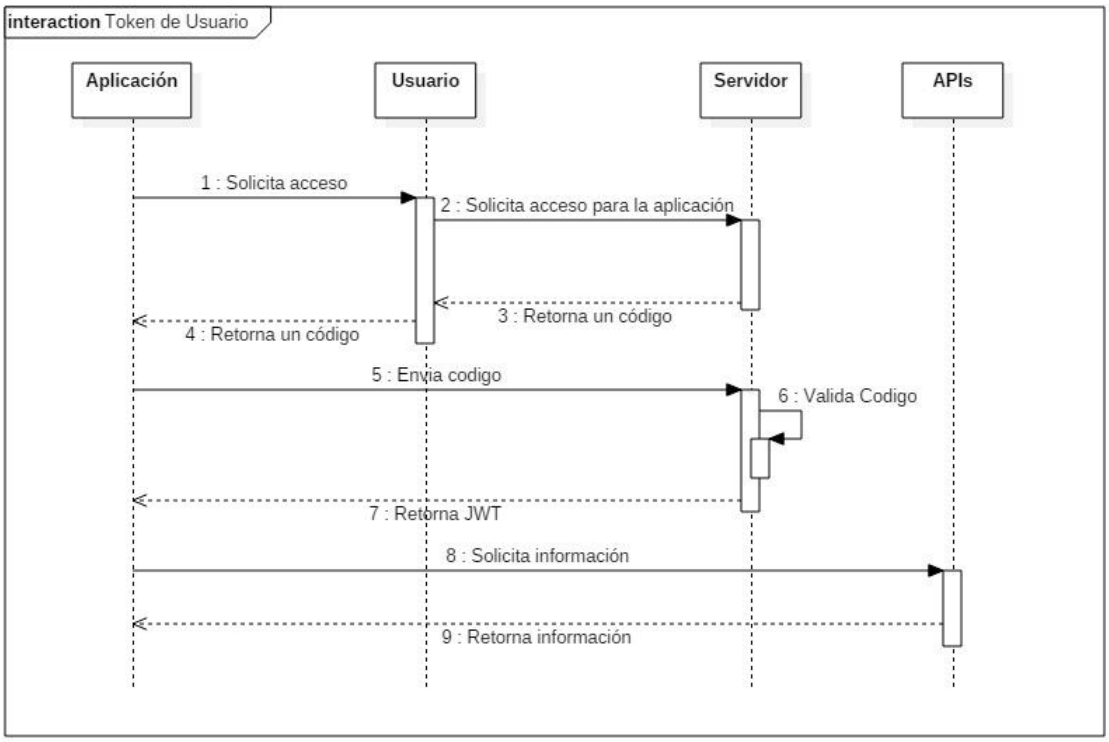

Figura 9. Diagrama de Secuencia Token de Usuario

Fuente: Elaboración propia

3.3.2. Configuración de Scopes

La configuración de los scopes es muy importante para poder limitar el alcance de la solicitud de acceso pudiendo una aplicación solicitar uno o más scopes. El scope se puede definir dentro de una lista separada por espacios o comas.

La siguiente tabla muestra los scopes que se están implementando para cada una de las aplicaciones consumidoras:

Tabla 1

Definición de scopes para flujo de autorización y autenticación

Aplicación	Scope
Api Gateway	apigw
Canales Móviles	can-mov
Canales Web	can-web

Nota. Fuente: Elaboración propia.

3.3.3. Configuración de Servicios

Es necesario realizar una configuración por cada aplicación que se va a autenticar o va a solicitar autorización dentro del flujo, para lo cual dentro de las fuentes que contienen el código donde se implementa el framework OAuth2 se configura un archivo JSON por cada aplicación. Como se muestra en la siguiente tabla:

Tabla 2

Configuración de los servicios para los flujos de autorización y autenticación

Aplicación	Archivo	client-id	client-Secret	scope
Api Gateway	ApiGateway-100001.json	…633f71	…e551be	apigw
Canales Móviles	CanalMobile-1000002.json	…c7c8b4	…b0fb7b	can-mov
Canales Web	CanalWeb-1000003.json	…f75cb5	…ecacf2	can-web

Nota. Fuente: Elaboración propia.

Tanto el client-id y el client-secret son usados por cada aplicación como credenciales para realizar su autenticación y autorizacion, el framework devolverá un JWT que cada aplicación podrá usar para acceder a las APIs internas.

Estos valores son colocados dentro de cada archivo JSON para ser leído por el framework y conceder la autorización correspondiente, estos valores de coincidir con los que la aplicación envía en cada una de las peticiones caso contrario mostrara un error.

3.3.4. Configuración de Grant Types

El flujo actual de autenticación y autorización soporta los siguientes Grant Types de OAuth2:

3.3.4.1. Flujo Client Credentials

La aplicación utiliza este tipo de Grant Type para obtener un token de acceso solo autenticando la aplicación más no el usuario. La aplicación lo utiliza generalmente para acceder a recursos de sí mismos y no recursos de un usuario.

Atributos de una solicitud Client Credentials

Para realizar una petición del tipo client credentials se deben tener en cuenta los siguientes atributos:

Tabla 3
Parámetros de una Solicitud Client Credentials

Parámetro	Necesario	Descripción	Valor
grant_type	Si	Tipo del flujo OAth	authorization_code
scope	Si	Alcance de la autorización	apigw
client_id	Si	Identificador único del cliente	…633f71
client_secret	Si	Clave secreta de la aplicación	…e551be

Nota. Fuente: Elaboración propia.

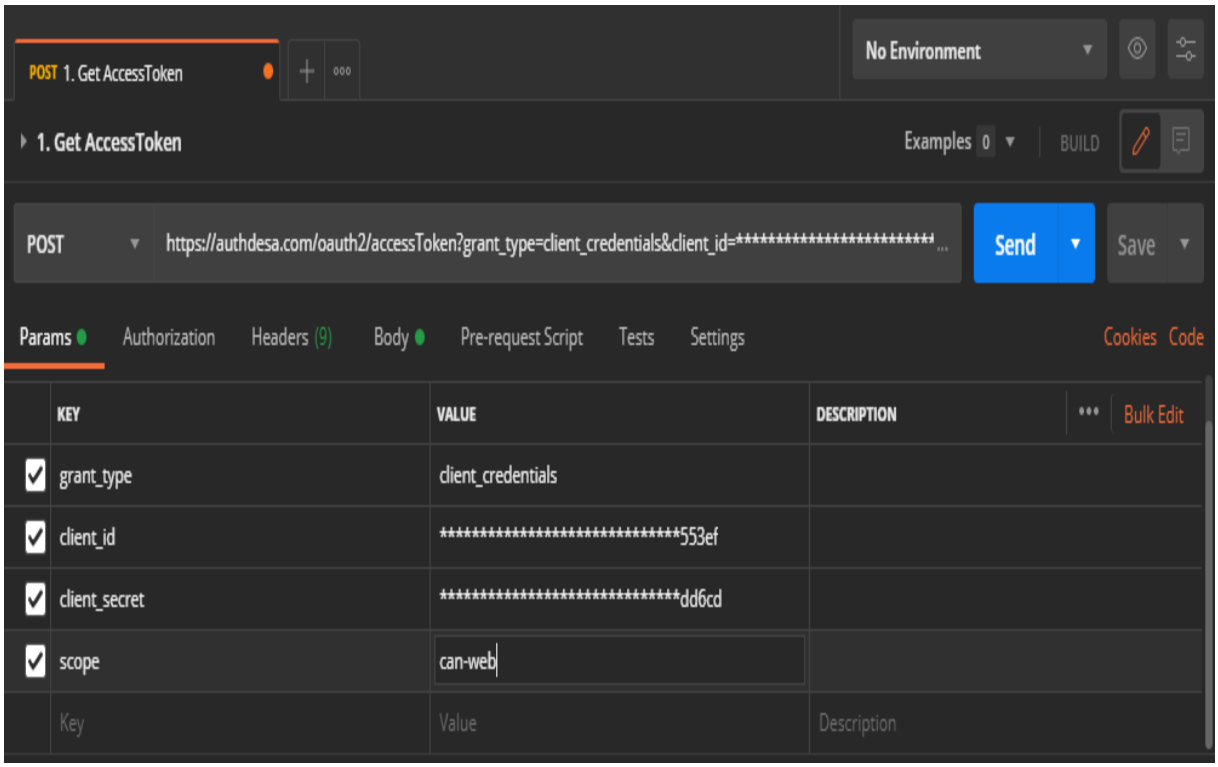

Figura 10. Request de una solicitud client credentials

Fuente: Elaboración propia

Ejemplo:

- Method: POST
- URL**:** https://authdesa.com/ouath2/accessToken
- grant_type: client_credentials
- client-id: **********553ef
- client-secret: **********dd6cd
- scope: can-web

Respuesta de la Petición:

Si todos los parámetros enviados son correctos, el servicio responderá un access token en formato JWT.

Figura 11. Response de una solicitud client credentials

Fuente: Elaboración propia

Atributos de respuesta de una solicitud Client Credentials:

La siguiente tabla muestra los parámetros de respuesta de la petición del flujo client credentials.

Tabla 4

Parámetros de Respuesta una Solicitud Client Credentials

Parámetro	Descripción	Valor
access_token	Token de acceso	eyJhbGc..
id_token	Token con información de autenticación del usuario	eyJhbGciOiJ..
token_type	Tipo del token de acceso	bearer

expires_in	Tiempo de duración del token	86400
scope	Alcance de la autorización	apigw

Nota. Fuente: Elaboración propia.

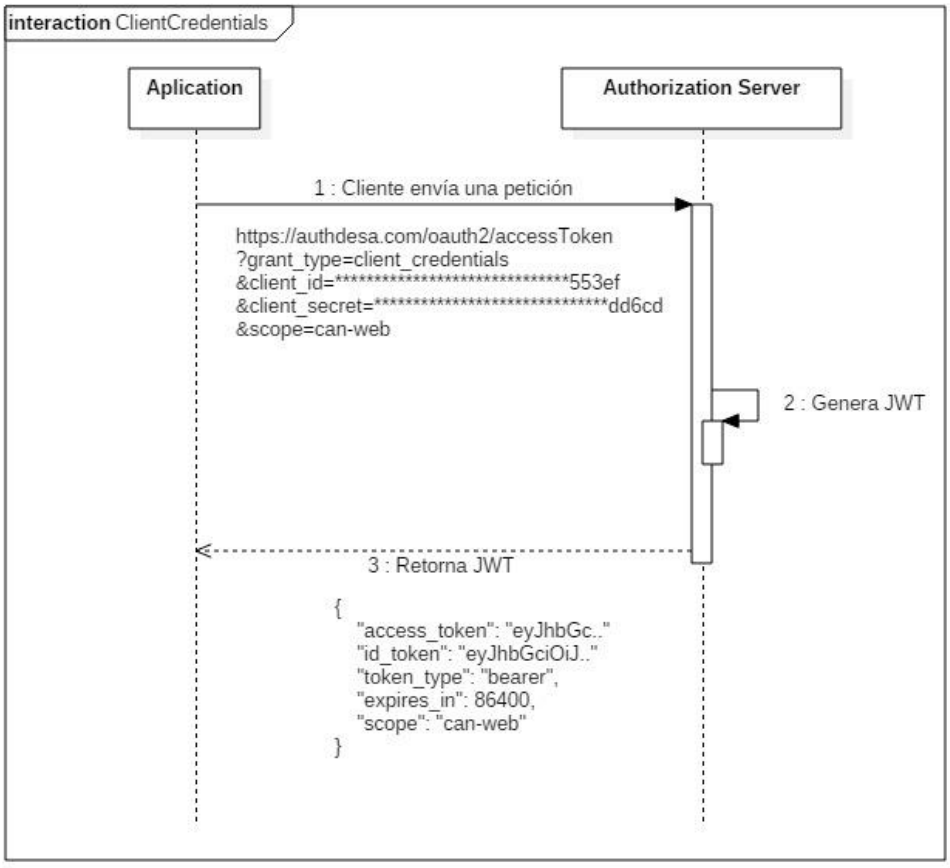

Figura 12. Diagrama de secuencia - Flujo Client Credentials

3.3.4.2. Flujo Authorization Code

La aplicación utiliza este tipo de Grant Type esperando que el usuario autorice a la aplicación otorgándole un código de intercambio temporal devuelto en la URL, el cual la aplicación utilizara para poder intercambiarlo por un token de acceso.

Aplicar extension PKCE (Proof Key Code Exchange)

Se utiliza la extensión para poder mitigar la amenaza de que algún atacante malicioso pueda interceptor el código de autorización que devuelve el servidor de autorización y realizar el intercambio por el token de acceso.

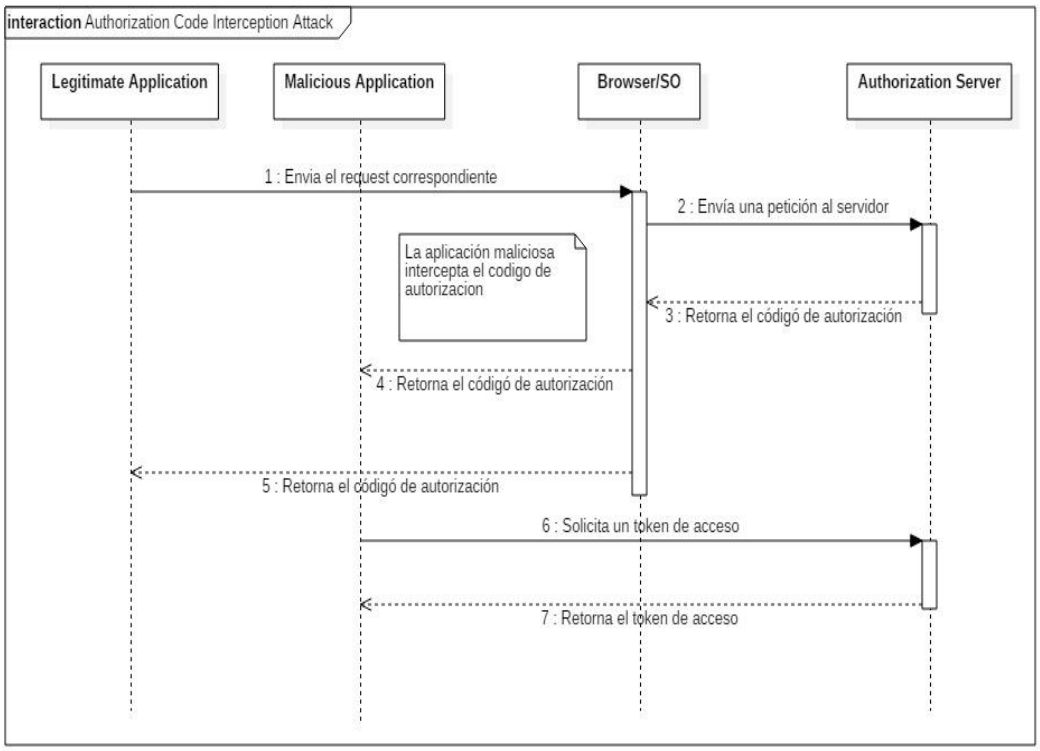

Figura 13. Diagrama de secuencia - Intercepción del Código de Autorización
Fuente: Elaboración propia

Mitigar la Intercepción del Código de Autorización

Para poder mitigar los ataques por aplicaciones maliciosas utilizamos una clave aleatoria “code verifier”. Esta clave se crea para cada una de las peticiones de autorización y el valor transformado llamado “code challenge” lo enviamos al servidor de autorización para obtener un token de acceso.

Luego el código de autorización se envía al endpoint del token con “code verifier”, y el servidor de autorizacion lo compara con el código de solicitud recibido previamente para realizar la prueba de posesión del “code verifier” por parte del cliente.

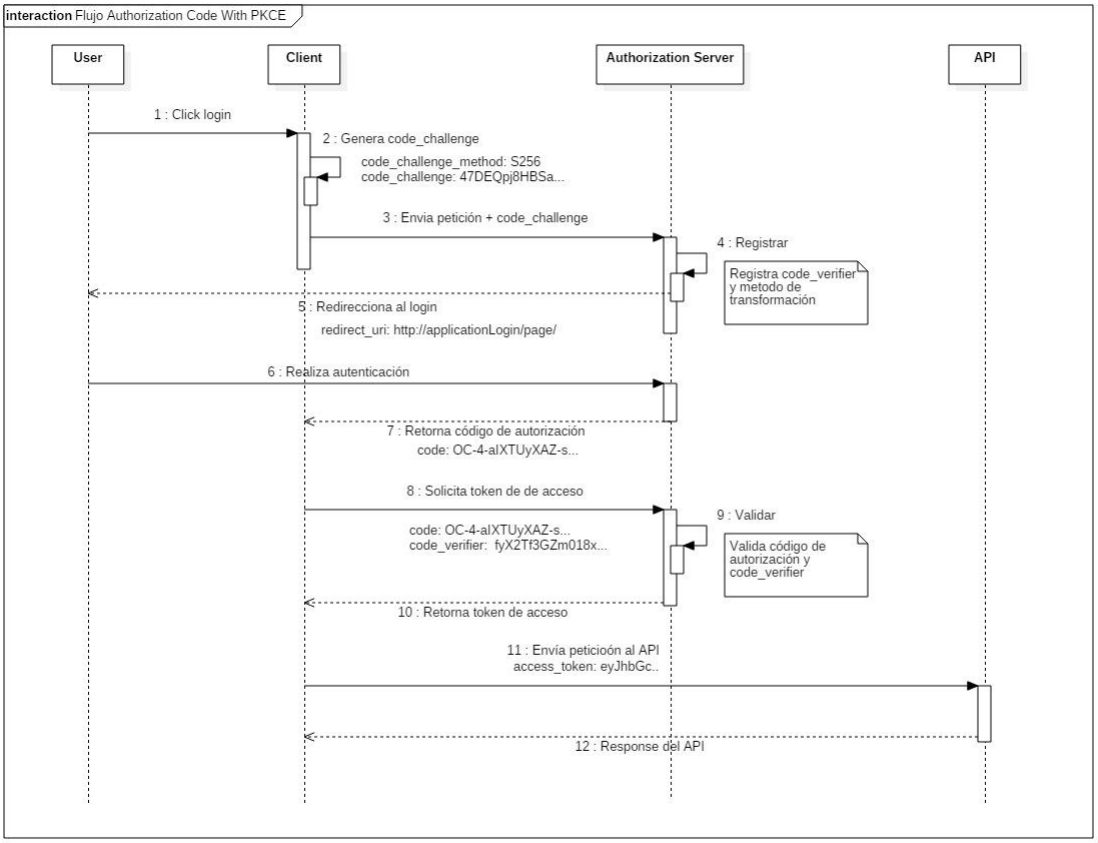

Figura 14. Diagrama de secuencia - Flujo Authorization Code con PKCE

Fuente: Elaboración propia

Atributos de una solicitud Authorization Code

Para realizar una petición del tipo client credentials se deben tener en cuenta los siguientes atributos:

Tabla 5

Parámetros de una solicitud Authorization Code con PKCE

Parámetro	Necesario	Descripción	Valor
response_type	Si	Especifica el tipo de respuesta que se espera del servidor de autorización	code
redirect_uri	Si	Especifica la URL de la página a la que se redirige al usuario después de una autenticación e inicio de sesión exitosos.	http://applicationLogin/page/
client_id	Si	Identificador único de una aplicación.	**********cdceb11
state	No (Pero recomendado	Una cadena aleatoria que ayuda a protegerse contra la falsificación de solicitudes entre sitios (CSRF).	**********dugdGIzTUng
scope	Si	Define el alcance de una petición.	can-web

code_challenge_method	Si	Algoritmo de hash utilizado para generar el valor del parámetro code_challenge.	47DEQpj8HBSa...
code_challenge	Si	Valor hash y codificado generado por el cliente.	S256
nonce	No (Pero recomendado	Ayuda a garantizar que el token de identidad que recibe sea el mismo token de identidad que solicitó	V6yq7YFK297nD...

Nota. Fuente: Elaboración propia.

La siguiente imagen muestra una petición enviada desde POSTMAN:

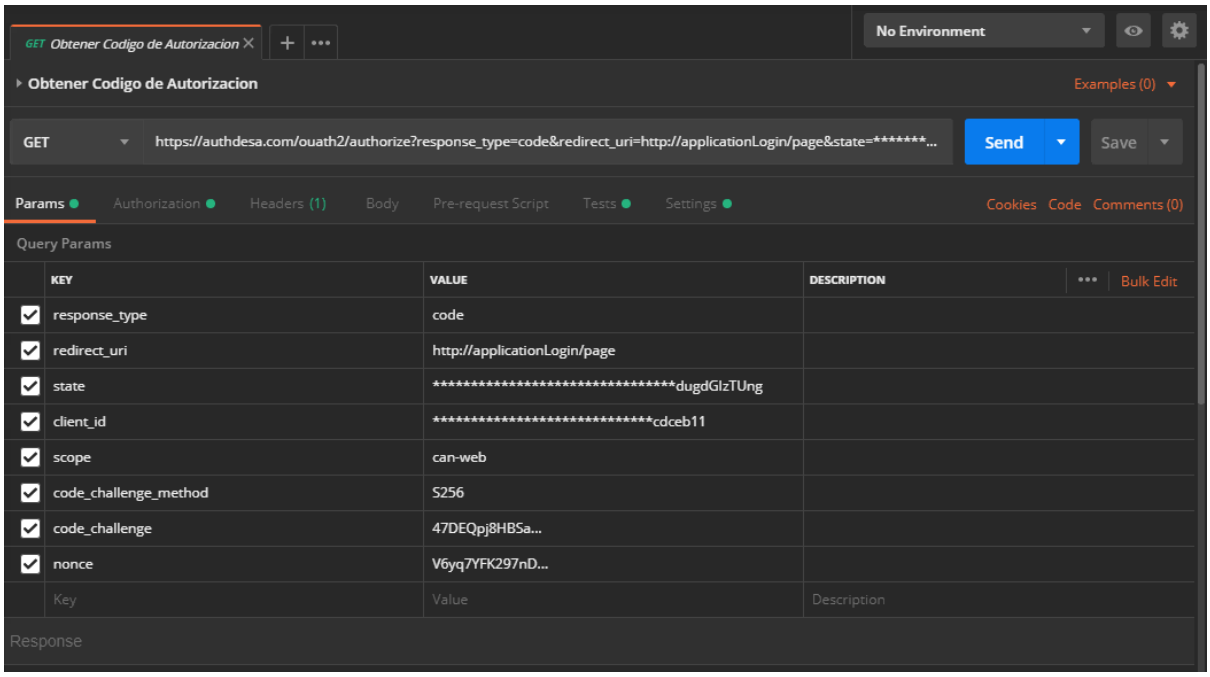

Figura 15. Request para Generar Authorization Code.

Fuente: Elaboración propia

Para simular una prueba del intercambio del código de autorización por un token de acceso se envía los siguientes datos.

Ejemplo:

- Method: POST
- URL**:** https://authdesa.com/ouath2/authorize
- response_type: code
- redirect_uri: http://applicationLogin/page/
- state: **********dugdGIzTUng
- client-id: **********cdceb11
- scope: can-web
- code_challenge_method: S256
- code_challenge: 47DEQpj8HBSa...

- nonce: V6yq7YFK297nD...

3.3.4.2.1. Obtener un Código de Autorización

Para poder acceder a los recursos del usuario es necesario primero realizar la autenticación a través del siguiente formulario:

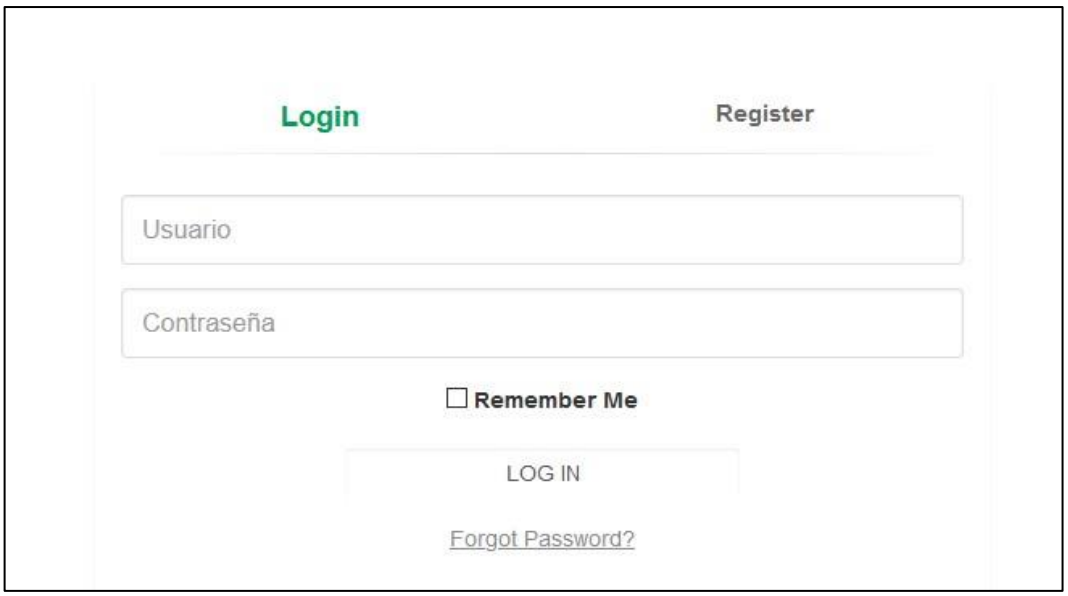

Figura 16. Login de Acceso de las Aplicaciones.
Fuente: Elaboración propia

Si las credenciales ingresadas en la ventana del login son las correctas, entonces el servidor de autorización devolverá un código de autorización en la url (redirect_url) para luego poder ser intercambiado por un token de acceso:

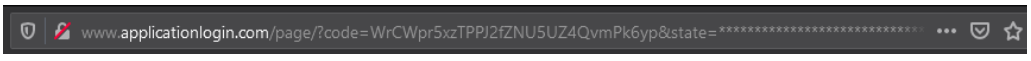

Figura 17. Obtención de un Código de Autorización.
Fuente: Elaboración propia

3.3.4.2.2. Obtener un Token de Acceso

Luego de obtener el código de autorización, se envía este código junto con algunos parámetros adicionales para que el servidor de autorización devuelva un token de acceso.

Tabla 6

Parámetros para obtener un token de acceso

Parámetro	Necesario	Descripción	Valor
code	Si	Código de autorización	**********JRexU6BUdxCU427
grant_type	Si	Tipo del flujo OAuth	code
redirect_uri	Si	URL de redireccionamiento	http://applicationLogin/page/
client_id	Si	Identificador único del cliente	**********cdceb11
code_verifier	Si	Verificador de código único para cada solicitud de autorización	fyX2Tf3GZm018xCCnGFRMg2IXXfrJkZez…

Nota. Elaboración propia.

La siguiente imagen muestra una petición enviada desde POSTMAN:

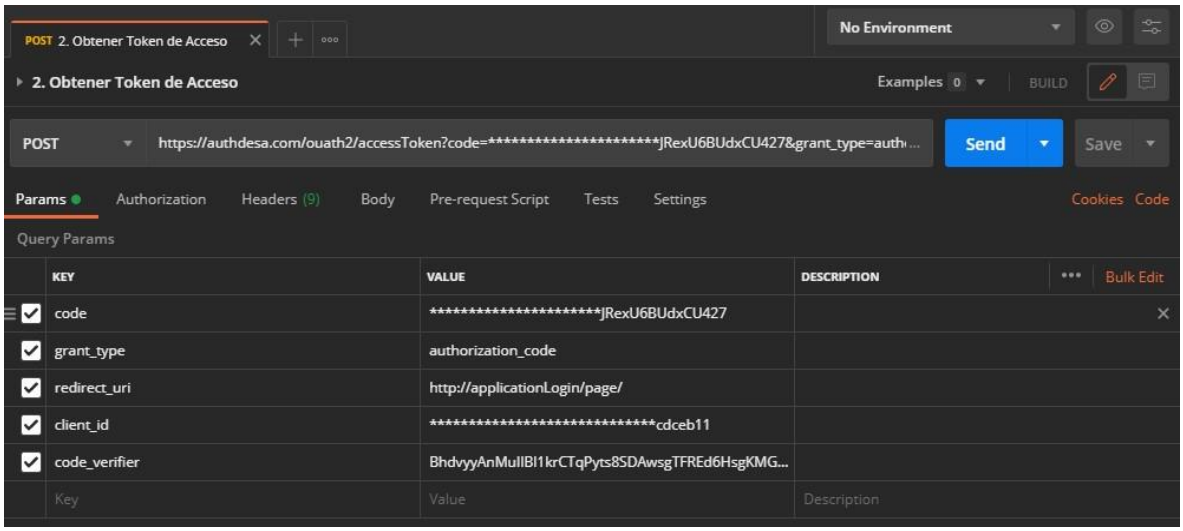

Figura 18. Intercambiar Código de Autorizacion por Token de Acceso

Fuente: Elaboración propia

Para simular una prueba del intercambio del código de autorización por un token de acceso se envía los siguientes datos.

Ejemplo:

- Method: POST
- URL**:** https://authdesa.com/ouath2/accessToken
- code: **********JRexU6BUdxCU427
- grant_type: authorization_code
- redirect_uri: http://applicationLogin/page/
- client-id: **********cdceb11
- code_verifier: fyX2Tf3GZm018xCCnGFRMg2IXXfrJkZez…

Respuesta de una Petición

Si todos los parámetros enviados son correctos, el servicio responderá un token de acceso en formato JWT.

Figura 19. Response de una Solicitud Authorization Code

Atributos de respuesta de una solicitud Authorization Code:

La siguiente tabla muestra los parámetros de respuesta de la petición del flujo client credentials.

Tabla 7

Parámetros de Respuesta una Solicitud Authorization Code

Parámetro	Descripción	Valor
access_token	Token de acceso	eyJhbGciOiJ…
id_token	Token con información de autenticación del usuario	eyJhbGciOiJ…
token_type	Tipo del token de acceso	bearer
expires_in	Tiempo de duración del token	86400

scope	Alcance de la autorización	can-web

Nota. Fuente: Elaboración propia.

3.3.5. Invocación a APIs Internas

Luego de obtener el token de acceso, cada aplicación puede acceder a los recursos a los cuales se les ha concedido la autorizacion correspondiente. Para el caso de los scopes “can-web” y “can-mov” requieren de acceder a recursos del cliente como datos de personas, datos de las cuentas etc. Para el caso del scope “apigw” solo necesita acceder a datos de la misma aplicación sin necesidad de autenticarse.

Obtener los Datos de una Persona

Se usa el token de acceso para invocar al api persons-v1, el cual obtiene los datos de una persona con solo enviar el DNI de la persona.

Request de la petición:

El token de acceso se puede enviar de dos maneras distintas:

1. Se puede enviar en la petición de POSTMAN en el parámetro Authorization, y se selecciona el TYPE como “Bearer Token” y colocando en el campo Token el token de acceso generado en el flujo Authorization Code.

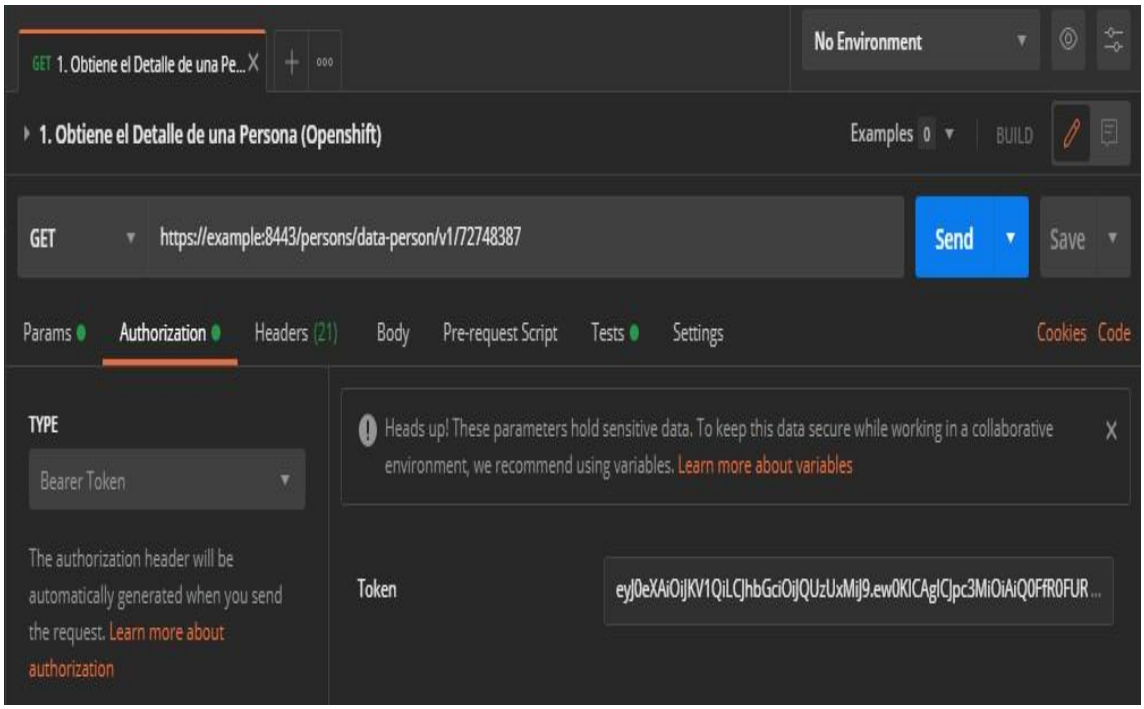

Figura 20. Solicitud Enviando Parámetro Autorization

Fuente: Elaboración propia

2. También se puede enviar el token de acceso en la cabecera de la petición haciendo uso del header de la petición, se crea un parámetro con el nombre “Authorization” y en el valor se envía la cadena “bearer” concatenado con el valor del token de acceso obtenido del flujo authorization code.

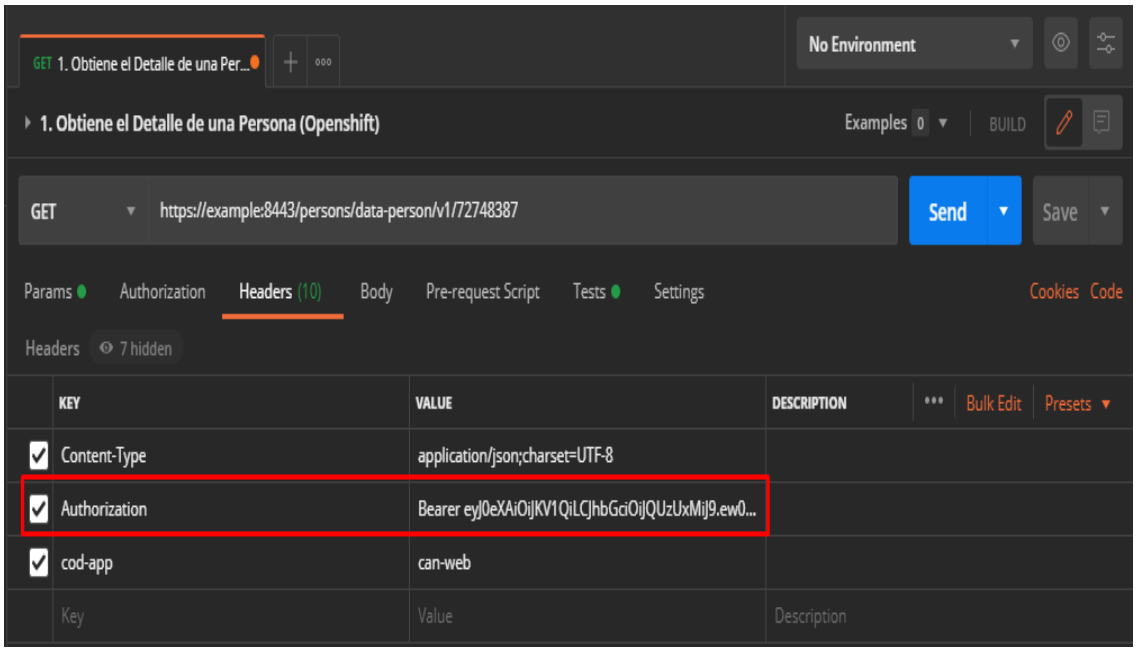

Figura 21. Solicitud Enviando Header Autorization

Fuente: Elaboración propia

Response de la petición:


```
{
    "personData": {
        "idPerson": "985635471",
        "typeDocument": "DNI",
        "document": 72748387,
        "names": "Bryan Vincent",
        "phaterLastName": "Castañeda",
        "motherLastName": "Duran",
        "birthdate": "30-10-1993",
        "addresses": [
            {
                "type": "HOME",
                "value": "St. 3 Gp.14 Mz. A Lt.2 - Villa el Salvador"
            },
            {
                "type": "MAIL",
                "value": "vbryan.dc@hotmail.com"
            }
        ],
        "phone": "983480173"
    }
}
```


Figura 22. Response api persons-v1 (Datos de una persona)

Fuente: Elaboración propia

La imagen anterior muestra el mensaje de respuesta al realizar una petición al api persons-v1, la cual obtiene los datos de una persona:

3.3.5. Solicitar Actualización de Token de Acceso (Refresh Token)

Los token de acceso generado por el servidor de autorización tienen un tiempo de vida determinado por lo cual es necesario que cada vez que el token ya no sea válido podamos solicitar un nuevo token de acceso. La siguiente imagen muestra el flujo para solicitar una actualización del token de acceso:

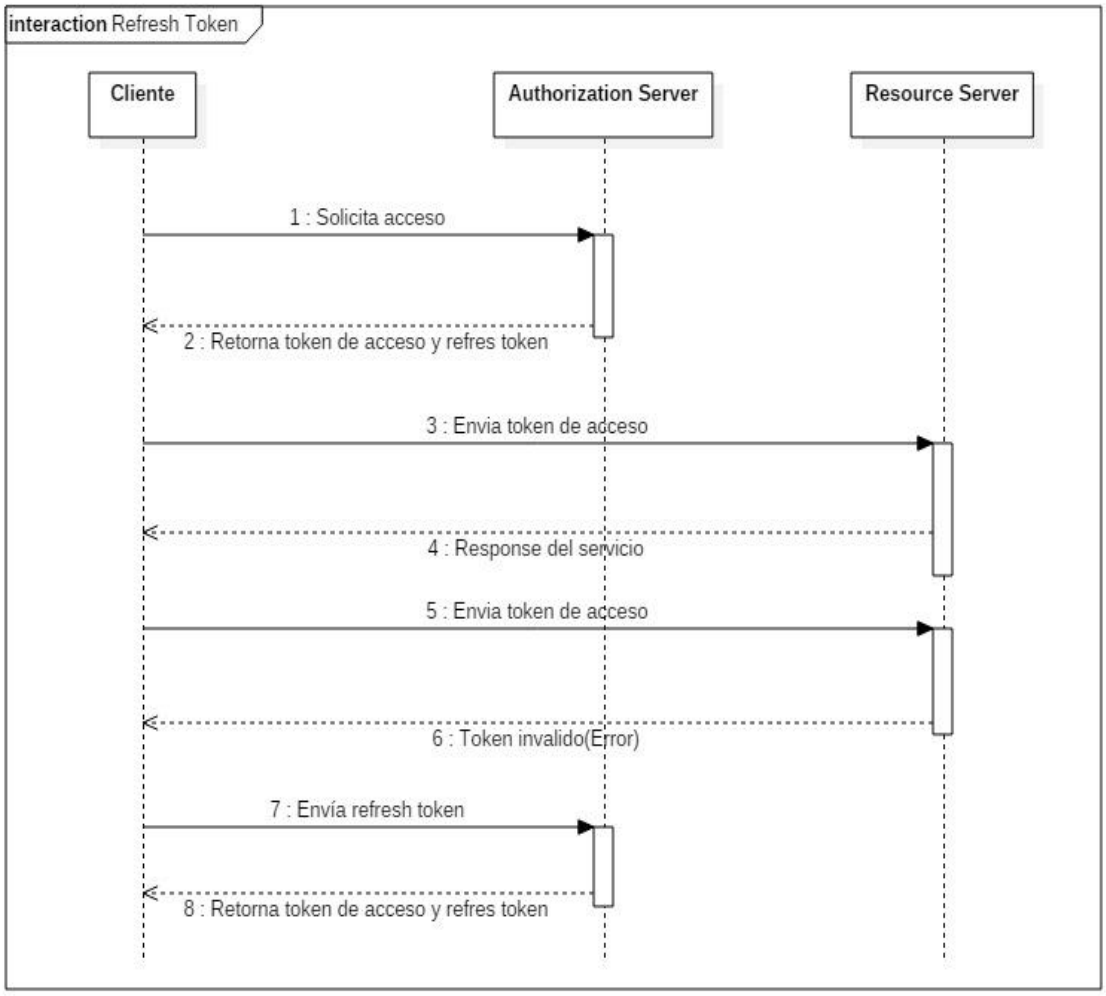

Figura 23. Diagrama de Flujo Refresh Token de la Aplicación
Fuente: Elaboración propia

Al realizar una petición para obtener un token de acceso, en el response la petición también se puede obtener un refresh token que de ser necesario se puede utilizar para poder otro token en caso el token retornado sea invalido.

Figura 24. Response con Refresh Token
Fuente: Elaboración propia

Cuando el token de acceso es invalido y es usado para poder acceder a los recursos del usuario, el servidor devolverá un error indicando que el token de acceso ya no es válido.

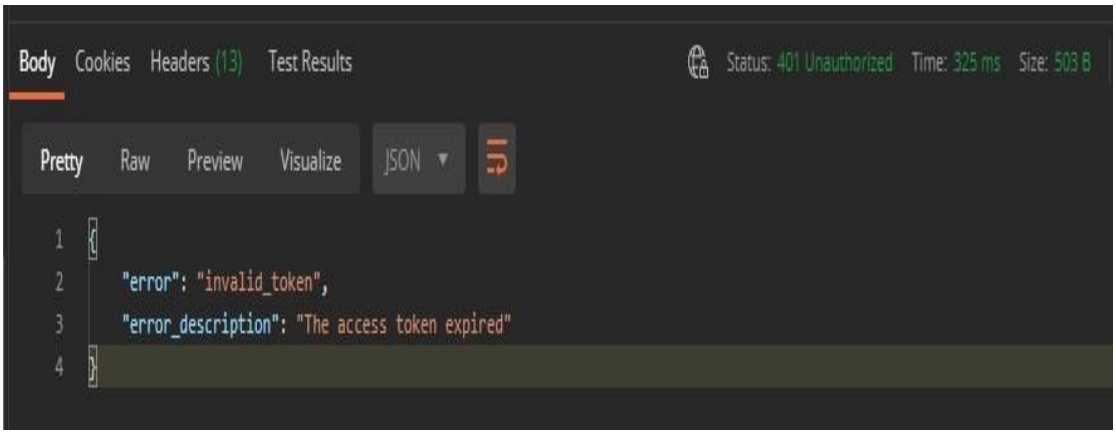

Figura 25. Response con Refresh Token
Fuente: Elaboración propia

Cuando se reconoce este error se puede usar el token de actualización de la petición anterior para poder obtener un nuevo token de acceso para poder

reintentar la solicitud. Para lo cual realizamos una nueva petición utilizando el token de actualización y las credenciales del cliente.

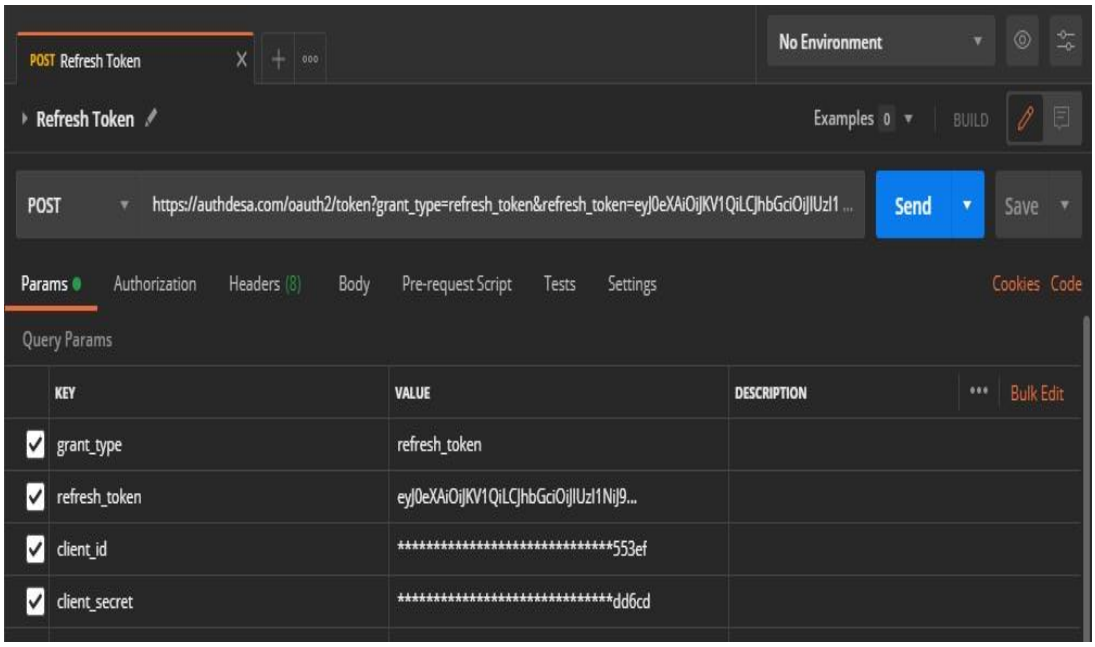

Figura 26. Request de una Petición Refresh Token
Fuente: Elaboración propia

3.3.6. Determinar el Estado de un Token (Introspection)

Para poder validar los token de acceso emitidos por el servidor de autorización cuando un cliente realiza alguna solicitud al servidor de recursos, es necesario qeue el servidor de autorización nos devuelva la información correspondiente del token de acceso. La siguiente imagen muestra cómo se define el flujo para la validación del token de acceso:

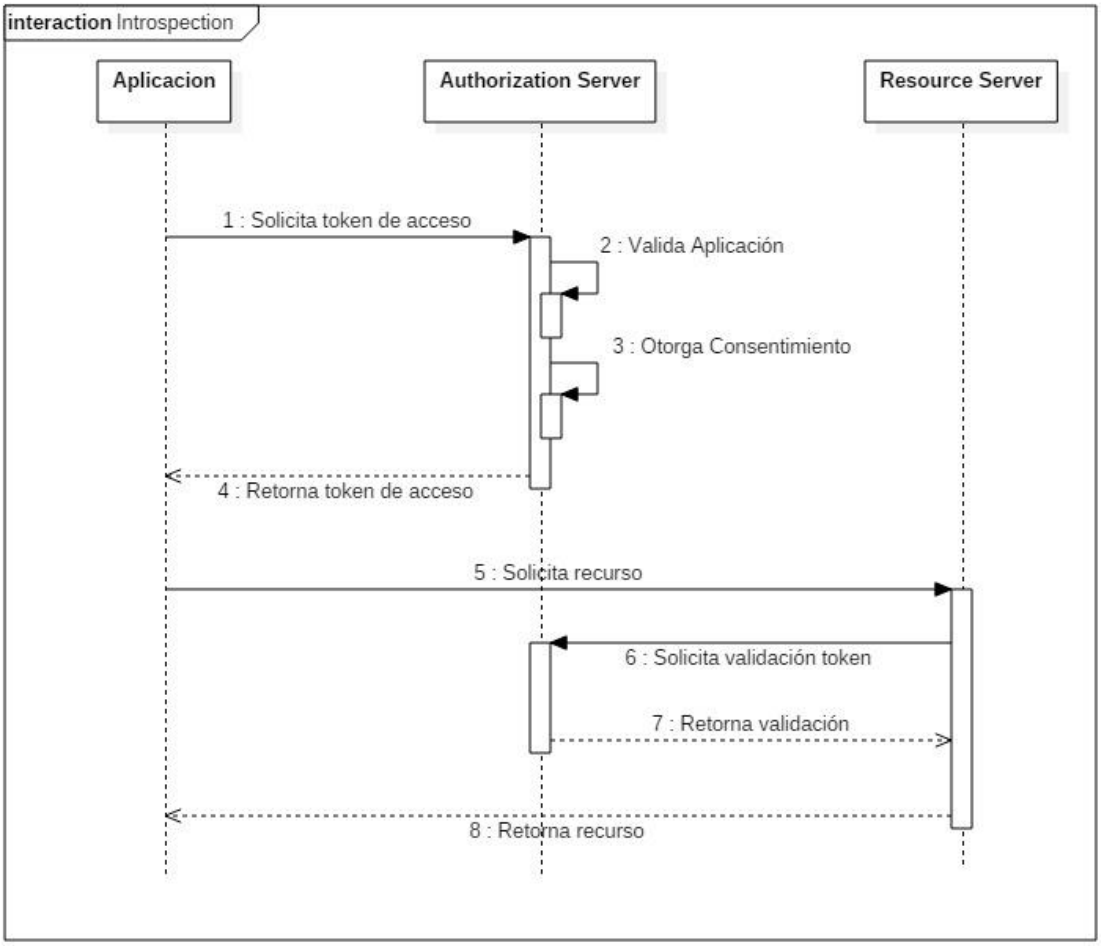

Figura 27. Diagrama de Secuencia Validar Token de Acceso (Introspection)
Fuente: Elaboración propia

Para realizar una validación del token de acceso se envían los siguientes parámetros en el POSTMAN:

Ejemplo:

- Method: POST
- URL**:** https://authdesa.com/ouath2/instrospect
- token: eyJhbGciOiJSUzI1NiIsInR5cCI6Ik**********
- Authorization: Bearer dXN1YXJpbzpjb**********

Es necesario enviar como parámetro header las credenciales de la aplicación con las cuales se generó el token de acceso (client_id y client_secret).

Request de una Petición Introspect

La siguiente imagen muestra el parámetro en donde se envía el token de acceso generado por el servidor de autorización para poder ser validado.

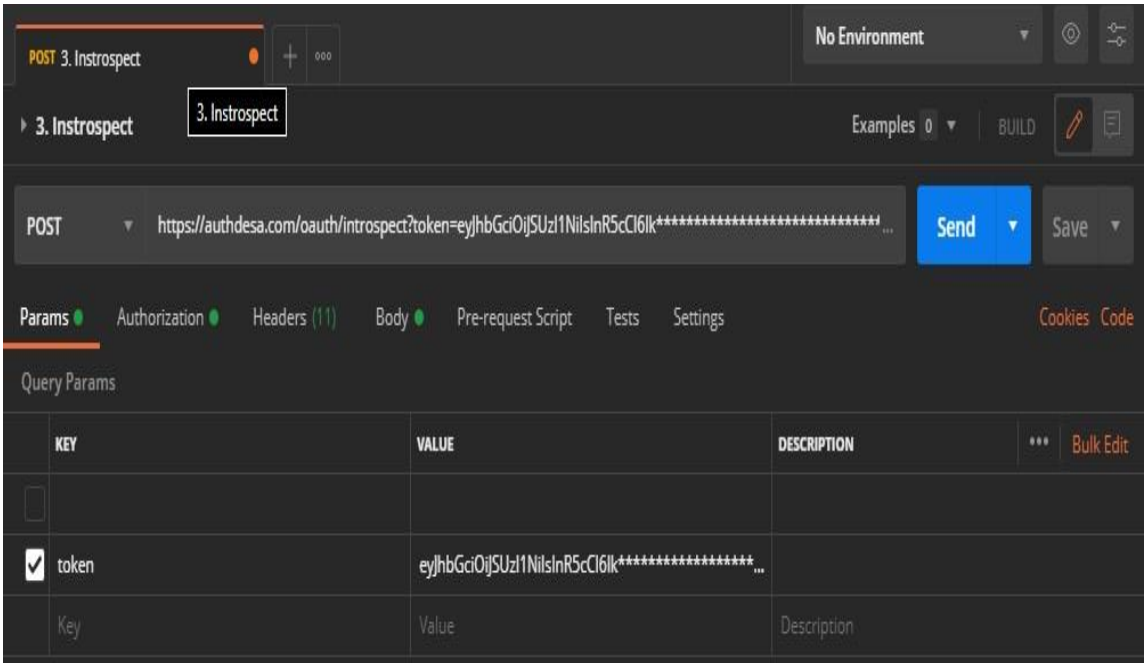

Figura 28. Petición Introspect Parámetro Token
Fuente: Elaboración propia

La siguiente imagen muestra el parámetro en donde se envían las credenciales del cliente encriptadas en base 64.

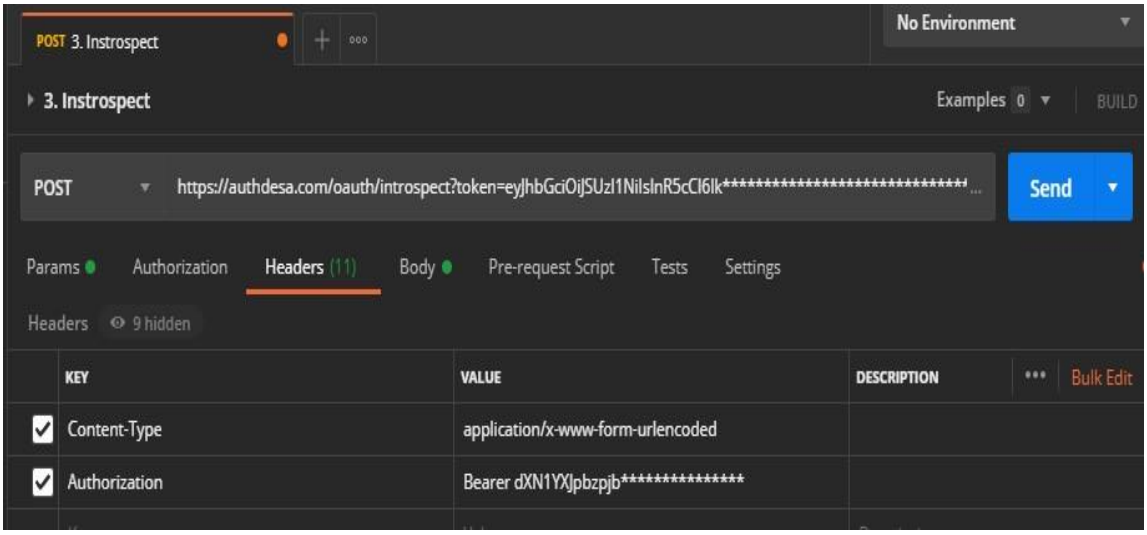

Figura 29. Petición Introspect Parámetro Authorization
Fuente: Elaboración propia

Response de una Petición Introspect (Éxito)

La siguiente imagen muestra las propiedades que devuelve una petición exitosa de introspect:

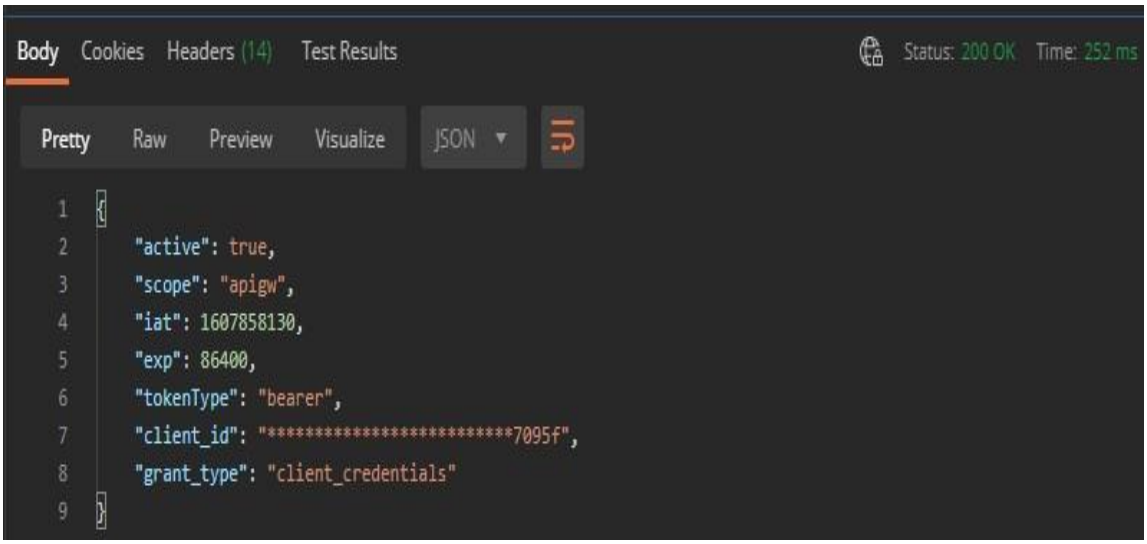

Figura 30. Response Invocación Exitosa (Introspect)
Fuente: Elaboración propia

Los atributos de respuesta de la petición se detallan en la siguiente tabla:

Tabla 8
Parámetro de Respuesta Introspection

Parámetro	Descripción	Valor
active	Representa si el token está activo	true o false
scope	Alcance de la autorización	apigw
iat	Fecha y hora de la generación del token	1607858130
exp	Tiempo de vida del token	86400
tokenType	Tipo de token	bearer
client_id	Identificador unico	**********7095f
grant_type	Flujo de autorización	client_credentials

Nota. Fuente: Elaboración propia

Response de una Petición Introspect (Error)

En el caso de que la autenticación no se valida la petición devolverá un código de estado 401 y una respuesta errónea como muestra la imagen:

Figura 31. Response Invocación Erronea (Introspect)
Fuente: Elaboración propia

La respuesta puede ser exitosa pero el campo “active” puede devolver un valor “false” debido a:

- El token solicitado no existe o no es válido
- El token expiró
- El token se envió a un cliente diferente al que realiza esta solicitud

3.3.7. Revocar el Token de Acceso

Luego de que la aplicación uso el token de acceso para obtener los recursos del usuario se puede notificar al servidor de autorización que ya no se necesita el token de acceso o la actualización del token que se ha obtenido previamente.

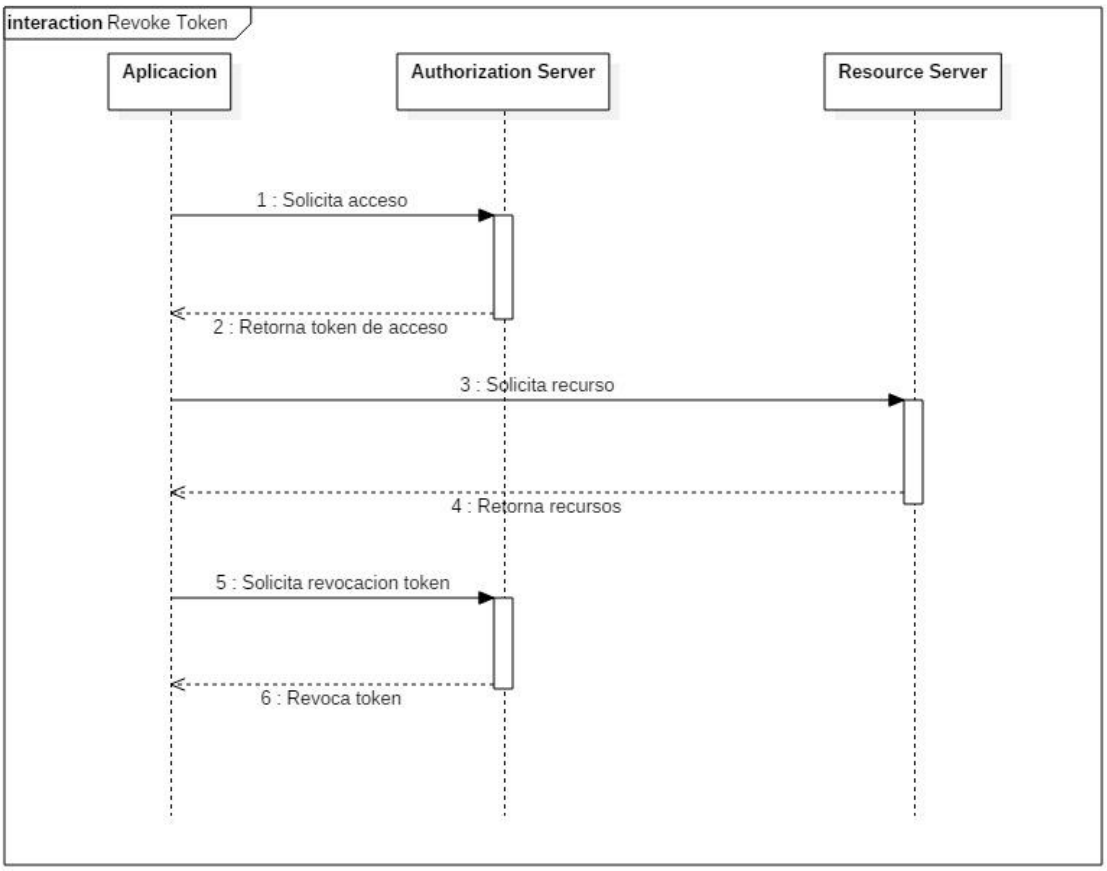

Figura 32. Diagrama de Secuencia Revoke Token
Fuente: Elaboración Propia

Para realizar una validación del token de acceso se envían los siguientes parámetros en el POSTMAN:

Ejemplo:

- Method: POST
- URL**:** https://authdesa.com/ouath2/revoke
- token: eyJhbGciOiJSUzI1NiIsInR5cCI6Ik**********
- Authorization: Bearer dXN1YXJpbzpjb**********

Request de una Petición Revoke Token

La siguiente imagen muestra el parámetro en donde se envía el token de acceso para que el servidor de autorización revoke el token.

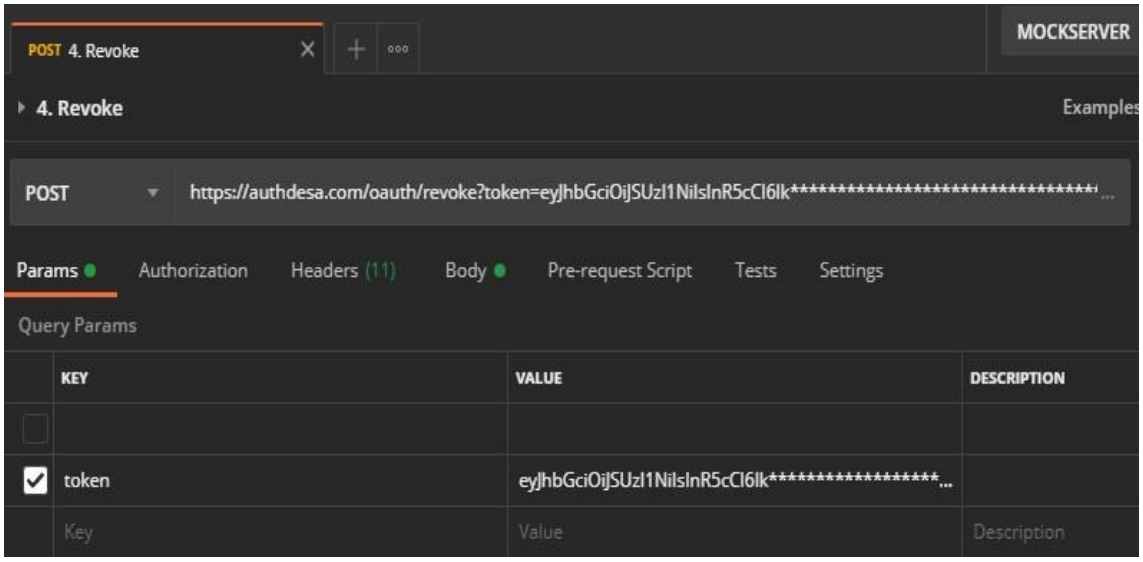

Figura 33. Petición Revoke Parámetro Token
Fuente: Elaboración propia

La siguiente imagen muestra el parámetro en donde se envían las credenciales del cliente encriptadas en base 64 para realizar el revoke, tienen que coincidir con las credenciales enviadas para generar el token de acceso.

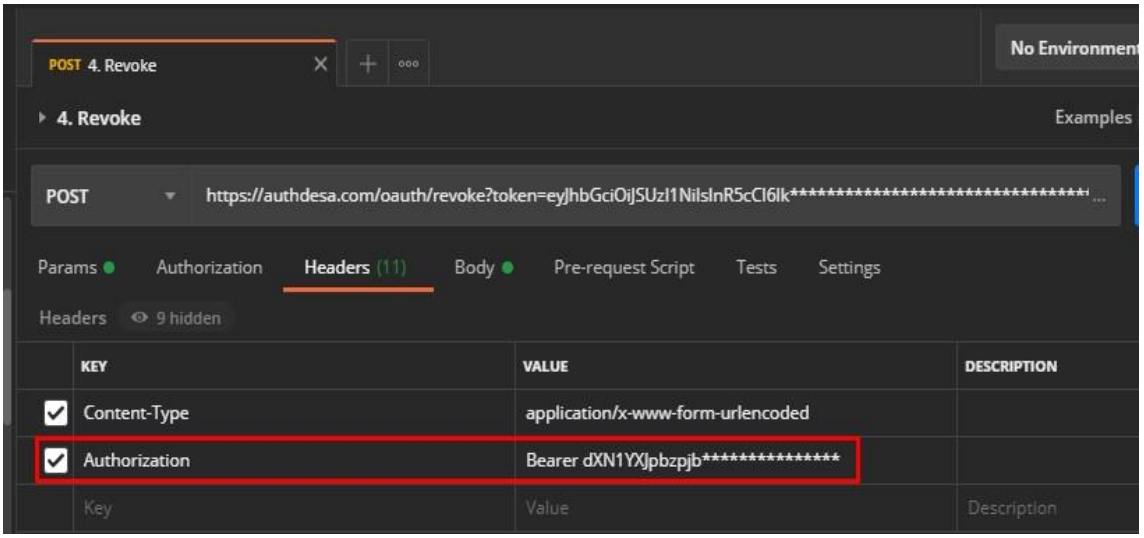

Figura 34. Petición Revoke Parámetro Authorizacion
Fuente: Elaboración propia

Response de una Petición Revoke

La siguiente imagen una respuesta exitosa con un código de HTTP 200 de exito que devuelve una petición exitosa de revoke:

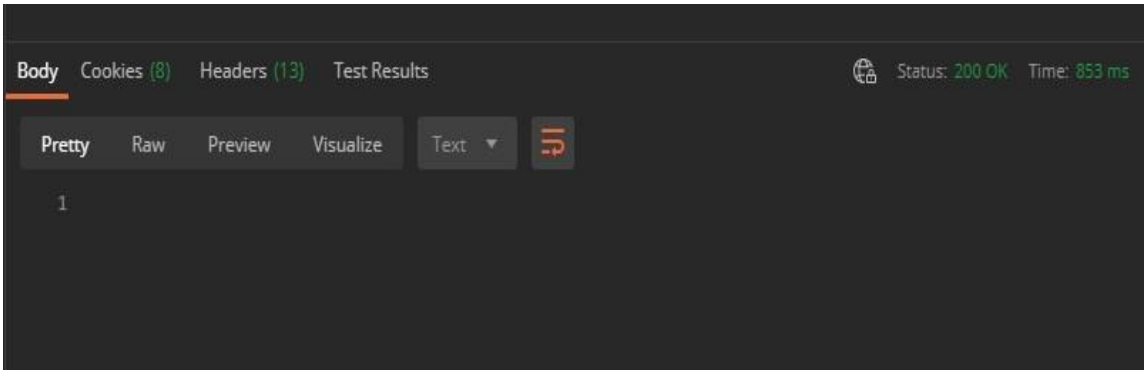

Figura 35. Response Petición Revoke
Fuente: Elaboración propia

Esta petición no devuelve ningún cuerpo en el reponse del servicio, pero basta con verificar que el código HTTP sea 200 para revocar el token de acceso. Ahora si volvemos a realizar un introspection al token de acceso revocado cambia el estado “active” a false.

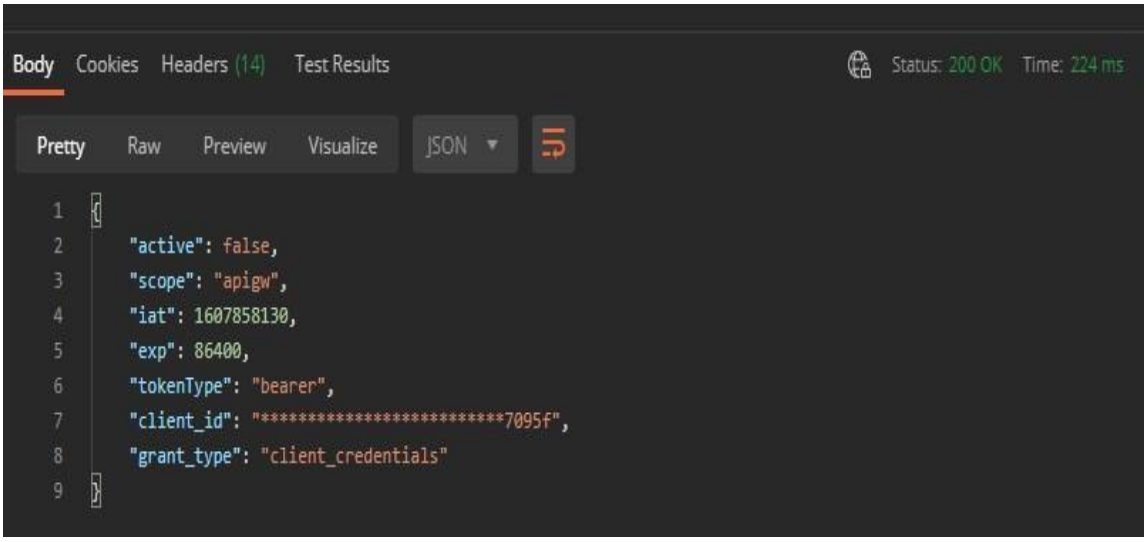

Figura 36. Validacion Revoke del Token de Acceso
Elaboración: Fuente Propia

De esta manera podemos de gestionar todo el flujo de vida de un token de acceso y mejorar la seguridad de las apis del usuario.

3.4. Resultados

En base a los objetivos planteados se propone un modelo para la aplicación del framework OAuth2 siguiendo los pasos para su implementación ya que esto permitirá:

- Definir el alcance de las aplicaciones así como el mejor flujo a elegir para según los requerimientos, pudiendo obtener un token de aplicación para acceder a sus propios recursos o un token de usuario para acceder a los recursos del usuario mediante una autenticación previa.

- Mejorar el control de acceso a los distintos microservicios que se encuentran alojados en el servidor de recursos a través del envió de un token de acceso en formato JWT que será leído por el servidor y validando la autenticidad del mismo.

- Mejorar el nivel de seguridad para que aplicaciones maliciosas no puedan interceptar los códigos de autorización y puedan solicitar token de acceso y acceder a los recursos del usuario, esto añadiendo parámetros que tanto la aplicación como la aplicación conocen.

CONCLUSIONES

- A travez de la aplicación del framework OAuth2 se logró que el acceso a los microservicios sea más seguro ya que se tiene una capa más de seguridad para su control.

- La aplicación del framework OAuth2 permitió definir el alcance de la autorización que se le otorga a cada una de las aplicaciones a través de la definición de scopes, los cuales será utilizados como identificadores de alcance.

- La aplicación del framework OAuth2 minimiza el riesgo de que terceros puedan recuperar un token de accso para acceder a los recursos del usuario a través de la aplicación de la extensión PKCE.

- A través de la gestión de los token de acceso se mejora la seguridad al momento de que las aplicaciones quieran acceder a los recursos ya que se pueden aplicar distintas políticas para cada una de las aplicaciones.

RECOMENDACIONES

- Se recomienda que el departamento de seguridad gestione las credenciales de cada aplicación y sean solo ellos los que puedan conocer el valor de cada una de ellas, para evitar que terceros puedan tener acceso a estas credenciales y puedan acceder a los recursos del usuario en nombre de alguna aplicación.

- Se recomienda utilizar algún protocolo de encriptación que permita hacer que las credenciales sean lo suficientemente seguras para evitar que terceros puedan adivinar o dar con las credenciales.

- Se recomienda que tanto el servidor de autorización como el servidor de recursos tengan los recursos suficientes para soportar una alta carga de transacciones.

- Se recomienda manejar distintas credenciales para cada aplicación en cada uno de los distintos ambientes: desarrollo, certificación y producción.

- Se recomienda implementar por cada tipo de servidores de recursos y servidor de recursos un balanceador de peticiones para poder evitar la sobrecarga de los servidores.

BIBLIOGRAFIA

Montoya S., J., & Restrepo R., Z. (2012). GESTIÓN DE IDENTIDADES Y CONTROL DE ACCESO DESDE UNA PERSPECTIVA ORGANIZACIONAL. *Ing. USBMed,, 3(1)*. Obtenido de http://revistas.usbbog.edu.co/index.php/IngUSBmed/article/view/261

(IETF), I. E. (May de 2015). *Json Web Tokens*. Obtenido de https://tools.ietf.org/html/rfc7519#page-6

Andress, J. (2011). *The Basics of Information Security.* Waltham, United States of America: Syngress. Obtenido de https://b-ok.lat/book/1162907/8abd04

Boyd, R. (2012). *Getting Started with OAuth 2.0.* United States of America: O'Reilly Media, Inc.

Castillo Gutiérrez, M. A. (2018). *DISEÑO E IMPLEMENTACIÓN DE UNA ESTRATEGIA DE SEGURIDAD MEDIANTE POLÍTICAS DE AUTENTICACIÓN Y AUTORIZACIÓN PARA UNA EMPRESA DE SEGUROS.* Tesis de Maestría, Universidad de Chile, FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION, Santiago de Chile.

Cevallos Teneda, A. S. (2016). *SISTEMA DE FEDERACIONES DE IDENTIDADES PARA LA FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL USANDO SOFTWARE DE CÓDIGO ABIERTO.* FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL, Ambato - Ecuador.

Contreras Pérez, L. A., & Vega Ortega, R. A. (2019). *Implementación de un Sistema de Gestión de Identidades Privilegiadas para el Control de Acceso en una Empresa Retail.* Tesis de Pregrado, Universidad San Martín de Porres, Facultad de Ingeniería y Arquitectura, Lima.

Feria Vila, C. E. (2018). *SEGURIDAD PARA EL CONTROL DE ACCESO A RECURSOS DE LA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA OPENFACT, AHREN CONTRATISTAS GENERALES - AYACUCHO, 2017.* Ayacucho.

gutierreaz duran, A. j. (2015). jhbhj. *2016*.

IETF, I. E. (2012). *OAuth 2.0*. Obtenido de https://tools.ietf.org/html/rfc6749

Kim, D., & Solomon, M. (2012). *Fundamentals of Information Systems Security.* Massachusetts, United States of America: World Headquarters.

Li, W., Mitchell, C., & Chen, T. (2018). Mitigating CSRF attacks on OAuth 2.0 Systems. *IEEE 2018 16th Annual Conference on Privacy, Security and Trust (PST)*. Obtenido de https://booksc.xyz/book/73427172/9ef549

Lucena López, M. J. (2001). *Criptografía y Seguridad en Computadoras.* Universidad de Jaén, Jaén.

Milián, V. (2010). SEGURIDAD EN ASP.NET: AUTENTICACIÓN Y AUTORIZACIÓN. *1er. CONGRESO (IBEROAMÉRICANO DE INGENIERÍA DE PROYECTOS)*. Obtenido de http://www.ijopm.org/index.php/IJOPM/article/view/29

Mino Pérez, K. A. (2017). *IMPLEMENTACIÓN SSO (SINGLE SIGN-ON) PARA OPTIMIZAR EL ACCESO A LOS SERVICIOS WEB EN LA UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO.* Universidad Nacional Pedro Ruiz Gallo, Escuela de Posgrado de la Universidad Nacional Pedro Ruiz Gallo, Lambayeque.

Newman, S. (2019). *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith.* United States of America: O'Reilly Media.

Pons Martorell, M. (s.f). Control de Accesos. Mataró. Obtenido de https://b-ok.lat/book/1404772/c04d97

Richer, J., & Sanso, A. (2017). *OAuth 2 in Action.* United States of America: Manning Publications Co.

Sagástegui Chigne, H., Alva Obeso, M., Labra Gayo, J., & Fernández Lanvín, D. (2005). MODELO DE CONTROL DE ACCESO X-RBAC PARA UN FRAMEWORK DE APRENDIZAJE COLABORATIVO. *Conferência IADIS Ibero-Americana WWW/Internet 2005*. Obtenido de https://www.academia.edu/486983/MODELO_DE_CONTROL_DE_ACCESO_X_RBAC_PARA_UN_FRAMEWORK_DE_APRENDIZAJE_COLABORATIVO

Siriwardena, P. (2020). *Advanced API Security.* San Jose, States United of America: Apress. Obtenido de https://b-ok.lat/book/5319807/0ddf29

Spasovski, M. (2013). *OAuth 2.0 Identity and Access Management Patterns.* United Kingdom: Packt Publishing.

Spasovski, M. (2013). *OAuth 2.0 Identity and Access Management Patterns.* United Kingdom: Packt Publishing.

The SANS Institute. (2020). *The SANS Institute*. Obtenido de Information Security Resources: https://www.sans.org/information-security/

Torroglosa-García, E., Pérez-Morales, A., Martinez-Julia, P., & Lopez, D. (2013). Integration of the OAuth and Web Service family security. *Computer Networks, 57*, 2233-2249.

Ur Rehamn, R. (2008). *Get Ready For OpenId.* Conformix Technologies. Obtenido de https://b-ok.lat/book/694399/c426c2

Wilson, Y., & Hingnikar, A. (2019). *Solving Identity Management In Modern Applications: Demystifying OAuth 2.0, OpenID Connect, And SAML 2.0.* Estados Unidos: Apress Media LLC.