UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA DE FORMACIÓN PROFESIONAL DE INGENIERÍA DE SISTEMAS

"SEGURIDAD PARA EL CONTROL DE ACCESO A RECURSOS DE LA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA OPENFACT, AHREN CONTRATISTAS GENERALES - AYACUCHO, 2017"

Tipo de investigación: Aplicada

Presentado por:

Bach. FERIA VILA, Carlos Esteban

Asesor:

Ing. CARRILLO RIVEROS, ELINAR

Para optar el Título Profesional de:

INGENIERO DE SISTEMAS

Ayacucho, Julio del 2018

DEDICATORIA

A Dios porque todo lo bueno se lo debo a Él; a mis padres por darme la vida, por su ejemplo de vida, por siempre apoyarme inmerecidamente, y por todos sus esfuerzos invertidos en mi educación y formación como profesional y persona.

AGRADECIMIENTO

A todos mis docentes universitarios, en especial a la Ing. Elinar Carrillo; por su amistad, por compartir sus conocimientos, por sembrar en mí la pasión por el aprendizaje continuo, y por permitirme formarme como profesional.

CONTENIDO

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

CAPÍTULO IV

RESULTADOS DE LA INVESTIGACIÓN

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La Aplicación Web de Facturación Electrónica OPENFACT de la empresa Ahren Contratistas Generales S.A.C. es un software capaz de crear, procesar y almacenar comprobantes de pago electrónicos (boletas, facturas, notas de crédito, notas de débito, etc.) regulados por la Resolución de Superintendencia 097-2012/SUNAT; resolución que crea y norma los sistemas de emisión electrónica desarrollados desde los sistemas del contribuyente.

Debido a la decisión de distribuir la Aplicación Web de Facturación Electrónica OPENFACT de manera comercial surgió la necesidad de implementar un sistema capaz de asegurar un adecuado control de acceso a los recursos del software (servicios web). Los recursos del software (servicios web) debían de ser utilizados no sólo desde su propia interfaz sino también desde otros softwares independientes (softwares pertenecientes a otras empresas) lo que trajo como consecuencia la necesidad de rediseñar el sistema de seguridad del software.

Como resultado de la presente investigación se logró implementar la capa de seguridad, aplicando el protocolo de seguridad OAuth2, para la aplicación web de Facturación Electrónica OPENFACT; es decir, se implementó un sistema de autenticación, autorización y trazabilidad que permitió un adecuado control de acceso a los recursos en los diferentes escenarios en los que el software es usado.

PALABRAS CLAVE: Control de acceso a recursos, seguridad, OAuth2.

INTRODUCCIÓN

Las aplicaciones web que exponen sus recursos a través de servicios web para su uso por otros softwares, son cada vez más frecuentes; en la mayoría de los casos los servicios web son expuestos para su uso a través del protocolo HTTP (uno de los protocolos más usados). Por otro lado, los servicios web expuestos a través del protocolo HTTP pueden ser accedidos desde cualquier ordenador de la misma red de comunicación donde los servicios web estén alojados, por lo que al acceso a los mismos requieren métodos de control de acceso diferentes a los convencionales.

Las aplicaciones web que son publicadas en la nube requieren tener especial cuidado en cuanto a la seguridad de los recursos publicados; en todos ellos el problema común es: ¿Cómo autenticar, autorizar y monitorear para reducir el riesgo de ataques informáticos? La presente investigación pretende dar respuesta a la pregunta anterior y aplicarla a un software específico cuyo nombre es OPENFACT.

La presente investigación es de tipo aplicada y nivel descriptivo ya que aplicará herramientas y métodos para la solución de un problema específico y se limitará a describir las variables mas no a medirlas o manipularlas.

La principal herramienta de la presente investigación es el protocolo de seguridad OAuth2. OAuth2 es un protocolo de autorización que permite a terceros, clientes o sistemas, acceder a contenidos de propiedad de un usuario, alojados en un servidor de recursos sin que éstos tengan que manejar ni conocer las credenciales del usuario, es decir, usuario y contraseña.

CAPÍTULO I
PROBLEMA DE INVESTIGACIÓN

1.1 DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

Ahren Contratistas Generales S.A.C. es una empresa ayacuchana, fundada el año 2007, dedicada al servicio de consultoría, ejecución de obras, provisión de bienes y servicios desde la etapa de planificación, consultoría, ejecución y puesta en operación de los proyectos a su cargo.

La aplicación web de Facturación Electrónica OPENFACT, diseñada y desarrollado por la empresa Ahren Contratistas Generales S.A.C. en el año 2016 es un software capaz de crear, procesar y almacenar documentos electrónicos (boletas, facturas, notas de crédito, notas de débito, etc.) bajo el estándar UBL 2.1 (Universal Business Language); además cumple con los criterios, requisitos y características establecidas por la SUNAT.

El objetivo inicial de la creación del software OPENFACT fue crear un software, de uso exclusivamente interno. Al tratarse de un software de uso interno, el software solamente implementó un método de autenticación llamado *HTTP Basic*. El método de autenticación *HTTP Basic* es un método que consiste en enviar un usuario y contraseña, en la cabecera *Authorization* de cada petición HTTP, codificado en base64; por ejemplo:

Figura 1. Autorización mediante HTTP Basic.

Poco tiempo después de verificar la eficiencia, utilidad, y la necesidad de otras empresas por usar el OPENFACT, la empresa Ahren Contratistas Generales S.A.C. decidió abrir la posibilidad vender licencias para que otras empresas hagan uso del software.

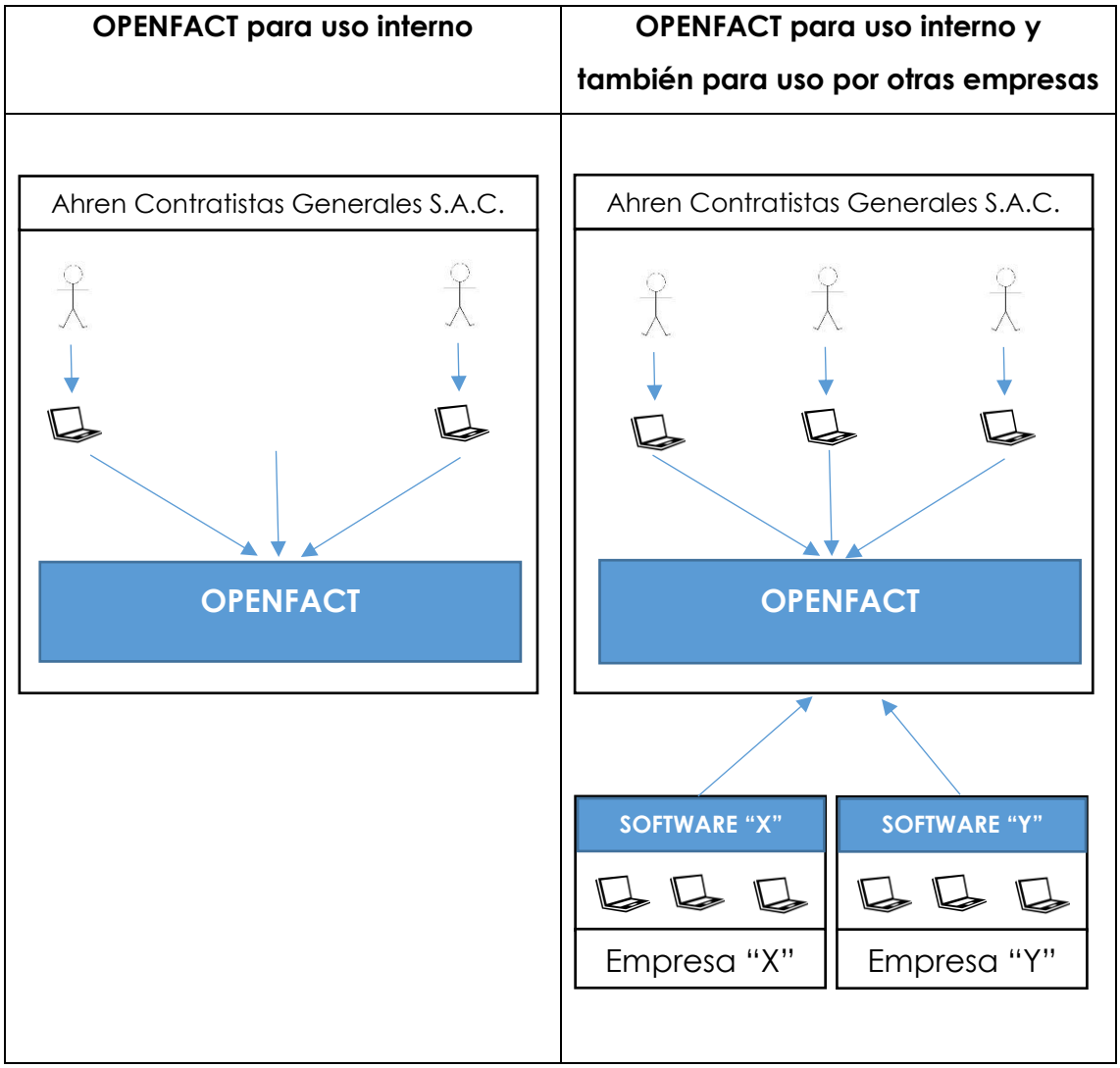


Figura 2. Uso interno y externo de OPENFACT.

A continuación, daré una breve descripción del software: OPENFACT es un software multiempresa, diseñado para ser usado durante el proceso de venta de una empresa; es decir cuando se vende un producto o servicio. El software está conformado por dos componentes básicos:

- **Openfact-web-console**: Consola de administración web (desarrollada usando el framework Angular 2).

- **Openfact-restful-services**: Componente que expone un conjunto de servicios web *restful* para el consumo de peticiones HTTP: GET, POST, PUT, DELETE (desarrollada usando Java EE).

Gracias a que OPENFACT expone servicios web *restful*, existe la posibilidad de integrar otros softwares (propiedad de una empresa externa) con el componente **openfact-restful-services** como se muestra en la figura 3.

OPENFACT es capaz de procesar peticiones de dos tipos de usuarios: usuarios que hacen uso del componente **openfact-web-console** y usuarios que hacen uso de otro software, propiedad de otra empresa externa como se muestra en la figura 3.

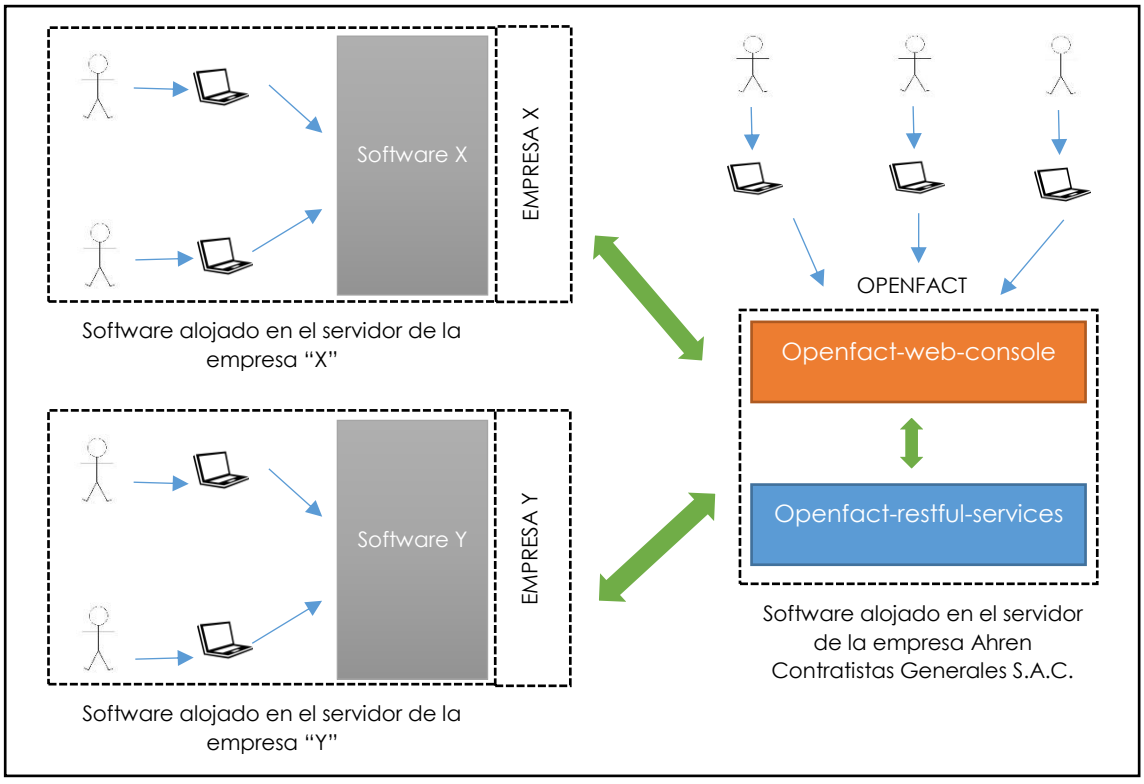


Figura 3. OPENFACT y los usuarios que interactúan con él.

OPENFACT es capaz de operar en dos modelos de negocios diferentes:

- Customer to Bussiness: Cuando los usuarios hacen uso del componente *openfact-web-console* (figura 3).
- Bussiness to Bussiness: Cuando los usuarios pertenecen a una empresa diferente y hacen uso de un software como intermediario entre su empresa y el componente *openfact-restful-services* de OPENFACT.

Además de ofrecer la funcionalidad básica de procesamiento de comprobantes de pago electrónicos exigido por la SUNAT, OPENFACT ofrece un software complementario, pero independiente, llamado OPENFACT+ (OPENFACT PLUS) cuya función es la de recolectar comprobantes de pago electrónicos asociados a las compras de una empresa, usando para esto diferentes fuentes de datos:

- Correo electrónico: Recolecta todos los comprobantes electrónicos de las bandejas de entrada del correo electrónico de los usuarios.
- Otras empresas: Otras empresas pueden enviar sus comprobantes de pago electrónico.
- Información proveniente desde los usuarios: OPENFACT+ ofrece la posibilidad de importar comprobantes electrónicos que el usuario los tenga almacenados en su computador.

En la actualidad OPENFACT+ se encuentra en versión *Beta* y da soporte para integración con el servidor de correos electrónicos Gmail (propiedad de la empresa Google); sin embargo, se planea añadir soporte para la integración con servidores de correos electrónicos Outlook (propiedad de la empresa Microsoft).

A continuación, se muestra un breve esquema sobre los componentes existentes en OPENFACT+ y cómo interactúan entre ellos. Nótese también

que es posible integrar softwares de empresas externas con OPENFACT+, como se muestra en la figura 4.

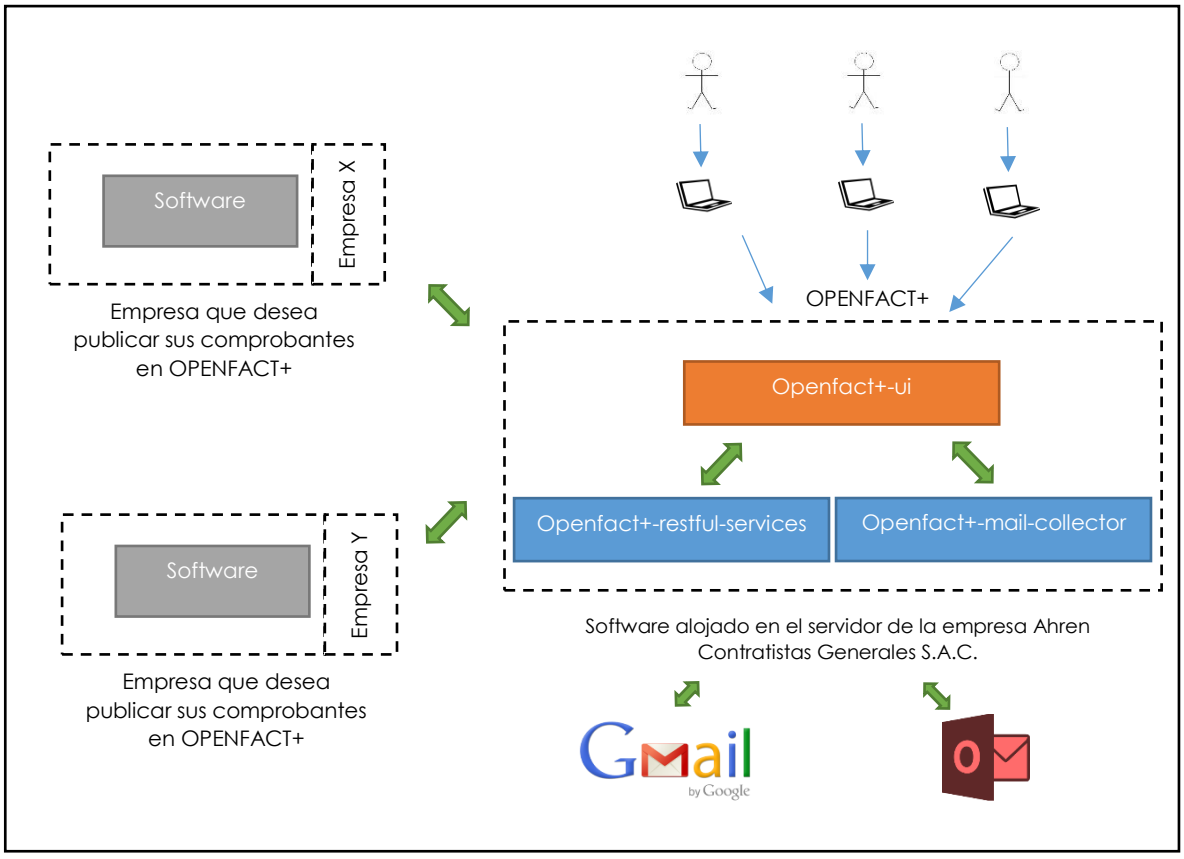


Figura 4. OPENFACT+ y los componentes con los que interactúa.

Finalmente, las empresas pueden hacer uso de OPENFACT para administrar sus ventas y OPENFACT+ para administrar sus compras, haciendo a OPENFACT una herramienta muy poderosa a la hora de administrar comprobantes de pago electrónicos.

Como se puede observar en las figuras 3 y 4, el software OPENFACT no sólo interactúa, con los usuarios, a través de su propia interfaz; sino que también interactúa con los usuarios a través del uso de otros softwares (comunicación software-software) propiedad de otras empresas. Además, los softwares de otras empresas ya tienen un conjunto de usuarios y contraseñas para sus trabajadores; por otro lado, cada empresa sólo tiene un usuario asignado en OPENFACT, por lo que es

necesario que todos los usuarios de empresas externas usen el mismo usuario y contraseña de OPENFACT.

Tabla 1

Posible distribución de usuarios y contraseñas en empresas externas.

Empresa	**Trabajador**	**Software de la empresa externa**	**OPENFACT**
Empresa "X"	Trabajador 1	Usuario/Contraseña 1	Usuario/Contraseña 1
	Trabajador 2	Usuario/Contraseña 2	
	Trabajador 3	Usuario/Contraseña 3	
Empresa "Y"	Trabajador 1	Usuario/Contraseña 1	Usuario/Contraseña 2
	Trabajador 2	Usuario/Contraseña 2	
	Trabajador 3	Usuario/Contraseña 3	

El software OPENFACT asigna únicamente un usuario a cada empresa; además, el software de cada empresa cuenta con un sistema de administración de usuarios independiente de OPENFACT. En consecuencia, cada vez que un trabajador de la empresa "X" desee usar el software de la empresa "X", este deberá autenticarse con el usuario y contraseña que la empresa externa le asignó; sin embargo, el software de la empresa "X" se comunicará con el software OPENFACT haciendo uso de un usuario y contraseña asignado por OPENFACT.

En la figura que se muestra a continuación (figura 5), se muestra el diagrama completo que describe el funcionamiento del software OPENFACT y OPENFACT+.

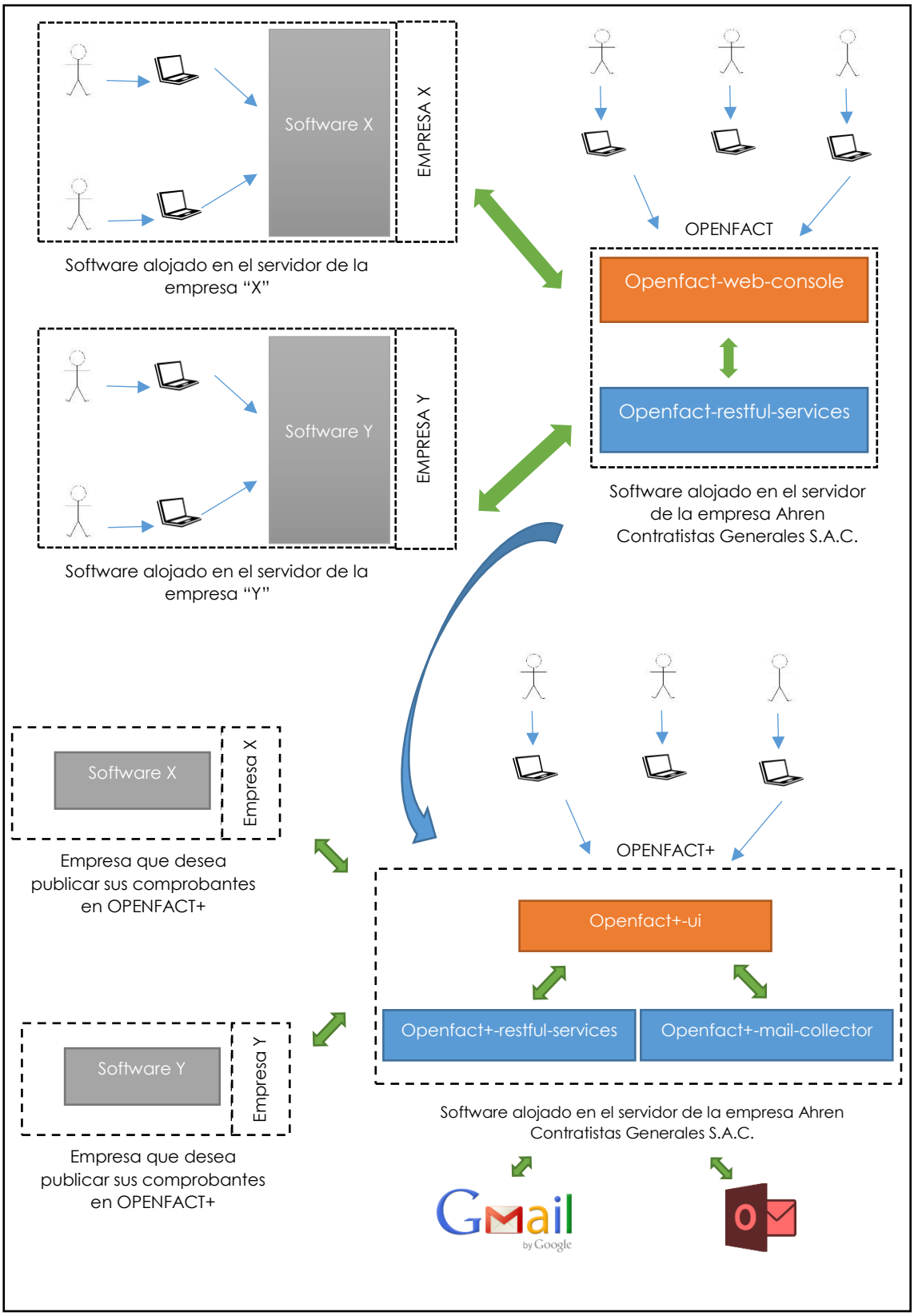


Figura 5. Diagrama general de OPENFACT.

Debido a que OPENFACT es un software que dejará de ser de uso exclusivamente interno y se convertirá en un software usado por varias empresas, se exige niveles de seguridad altos para proteger la manipulación no controlada de información por parte de las empresas externas o personas que puedan atacar al software; por lo que el típico sistema de seguridad *Login* (basado en usuario-contraseña), representado por el método *HTTP Basic*, es insuficiente para soportar los diferentes escenarios de transmisión de información en los que éste está implicado. A continuación, se lista algunos de las limitantes al usar un sistema de seguridad *Login (HTTP Basic)* basado en usuario y contraseña:

Tabla 2

Limitantes del método HTTP Basic en OPENFACT.

Escenario	**Solución usando Login (usuario-contraseña)**	**Limitantes o problemas**
Un usuario requiere enviar una petición POST hacia el componente *openfact-restful-services* haciendo uso del software de la empresa "X" como intermediario. (Figura 1.2)	El software de la empresa "X" debe de almacenar un usuario y contraseña válidos y enviar estos datos en cada petición hacia el componente *openfact-restful-services*	Si la transferencia de datos no se realiza a través de una conexión segura HTTPS el riesgo del robo de las credencias es alto, ya que basta con usar un *sniffer* para recolectar información y obtener las credenciales de un usuario, dado que un *sniffer* adecuado puede proporcionar información detallada

		que podría incluso transcribir una conversación de chat por completo.
La empresa "X" (figura 3) utiliza su propio software y almacena un usuario y contraseña (perteneciente a OPENFACT) en algún archivo de configuración o base de datos. Debido a alguna razón la empresa "X" desea cambiar la contraseña del usuario usado para conectarse al componente openfact-restful-services (OPENFACT).	La persona dueña del usuario (*username*) debe de realizar el cambio de su contraseña usando el componente *openfact-web-console (OPENFACT)*; como resultado, la contraseña del usuario será actualizada en los sistemas de OPENFACT, pero no en los softwares (propiedad de otras empresas) que hacen uso de OPENFACT. Finalmente, debido a que la contraseña cambió, una persona deberá actualizar la contraseña del usuario (almacenada en archivos de configuración o bases de datos	Los softwares (propiedad de otras empresas) que hacen uso de OPENFACT, no pueden ser notificadas ante un cambio de las credenciales de un usuario de OPENFACT; por lo tanto, existe la posibilidad tener tiempos muertos en los softwares propiedad de otras empresas.

	correspondientes al software de la empresa "X") y reiniciar el software de ser necesario.	
Un usuario desea saber si alguien está haciendo uso de sus credenciales sin su autorización.	El software deberá implementar su propio módulo de seguridad que se encargue de registrar cada *Login*.	El módulo de seguridad no es reutilizable para otros proyectos de software que la empresa tenga en mente.
Un usuario desea robar la identidad de otro usuario sin la necesidad de robar su usuario y contraseña.	Robar la cadena asociada a la cookie generalmente llamado SESSION_ID y usarla en su propio navegador.	Robar los datos de una cookie es relativamente sencillo ya que solo requiere usar un *Sniffer*.
Un usuario desea enviar datos al componente *openfact-restful-services* (figura 3) a través del uso de una aplicación móvil.	La aplicación móvil debe de almacenar un usuario y contraseña en el disco de almacenamiento del dispositivo móvil; y después usar el usuario y contraseña almacenados para interactuar con OPENFACT.	Los dispositivos móviles no cuentan con bases de datos por lo que los datos almacenados pueden ser accedidos desde cualquier otra aplicación móvil instalada en el mismo aparato, por lo que las credenciales tienen riesgo de ser robadas.
La aplicación web OPENFACT desea enviar comprobantes	OPENFACT debe de almacenar un usuario y contraseña válidas y	Las credenciales pueden ser robadas

electrónicos a la aplicación web OPENFACT+	enviarlas en cada petición.	escaneando el flujo de datos de la red.

La presente tabla lista algunos de las limitantes del uso del método HTTP Basic en los diferentes escenarios en los que funciona OPENFACT.

En consecuencia, el sistema de seguridad *Login* (basado en usuario-contraseña, *HTTP Basic*) presenta muchas limitantes y hace muy difícil el escalamiento del sistema de seguridad y del mismo sistema.

1.2 DEFINICIÓN DEL PROBLEMA DE INVESTIGACIÓN

1.2.1 PROBLEMA GENERAL

¿Cómo controlar el acceso a recursos de la aplicación web de Facturación Electrónica OPENFACT, Ahren Contratistas Generales S.A.C., 2017?

1.2.2 PROBLEMAS ESPECÍFICOS

1. ¿Cómo la autenticación permite un control de acceso a recursos de la aplicación web de Facturación Electrónica OPENFACT, Ahren Contratistas Generales S.A.C.?
2. ¿Cómo la autorización permite un control de acceso a recursos de la aplicación web de Facturación Electrónica OPENFACT, Ahren Contratistas Generales S.A.C.?
3. ¿Cómo la trazabilidad permite un control de acceso a recursos de la aplicación web de Facturación Electrónica OPENFACT, Ahren Contratistas Generales S.A.C.?

1.3 OBJETIVOS DE LA INVESTIGACIÓN

1.3.1 OBJETIVO GENERAL

Implementar la capa de seguridad para la aplicación web de Facturación Electrónica OPENFACT, Ahren Contratistas Generales S.A.C., utilizando los protocolos de seguridad OAuth2 y OpenID, y la herramienta Keycloak con el propósito de garantizar un adecuado control de acceso a recursos en los contextos en los que la aplicación es usada y la finalidad de reducir el riesgo de manipulación de datos.

1.3.2 OBJETIVOS ESPECÍFICOS

1. Identificar la identidad de los usuarios durante el proceso de autenticación para garantizar un adecuado control de acceso a recursos del sistema.

2. Proteger los recursos del sistema, permitiendo que estos sean solamente usados por aquellos usuarios (personas u otro software) a los que se les ha concedido autorización para ello.
3. Registrar las actividades de los usuarios para una adecuada trazabilidad durante el proceso de control de acceso a recursos.

1.4 JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.4.1 IMPORTANCIA Y JUSTIFICACIÓN

En la actualidad la empresa Ahren Contratistas Generales S.A.C., distribuye su Software de Facturación Electrónica OPENFACT, comercialmente. El software está diseñado para que sea usado por dos tipos de usuarios: usuarios(personas) que hacen uso directo del software a través de un navegador, y usuarios (otro software) que hacen uso del software a través del consumo del conjunto de servicios web expuestos por el sistema. La arquitectura del software exige un control de acceso a recursos que garantice un adecuado proceso de autorización y autenticación que soporte las necesidades de seguridad del sistema; es decir, el software requiere implementar un sistema de control de accesos (autenticación y autorización) para los dos tipos de usuarios antes mencionados.

La definición de la capa de seguridad en el Software de Facturación Electrónica permitió reducir el riesgo de ataques informáticos (ataques que involucren acceder a servicios web no autorizados), crear confianza (en cada uno de las empresas que hacen uso del software) y añadir valor comercial al software.

1.4.2 DELIMITACIÓN

La investigación implementó la capa de seguridad de la aplicación web de Facturación Electrónica OPENFACT de la empresa

Ahren Contratistas Generales S.A.C. según el estado actual del sistema en el año 2017.

CAPITULO II
REVISIÓN DE LITERATURA

2.1 ANTECEDENTES DE LA INVESTIGACIÓN

Varona (2014) en su tesis *"Implementación de un proveedor de autorizaciones OAuth 2.0 con Scala",* cuyo objetivo principal fue el de realizar un estudio detallado sobre OAuth2 para después desarrollar un software (servidor OAuth2) basado en Scala que cumpla con las especificaciones de OAuth2, concluye que: primero, es muy difícil para una sola persona conocer todos los contextos en los que se usa OAuth2 y por ello piensa que desarrollar un servidor OAuth2 requiere muchas personas y tiempo suficiente para llevar a cabo la tarea; segundo, concluye que OAuth2 mantiene el balance entre seguridad y usabilidad, ya que OAuth2 es relativamente sencillo de comprender y aplicar, esta es una de las razones por la que muchas organizaciones optan por usar OAuth2 como principal herramienta a la hora de asegurar la comunicación de sus softwares con el exterior.

Lozano (2013) en su tesis "Sistema centralizado de autenticación de usuarios para la Subdirección de Seguridad de la Información UNAM-CERT" concluye que OAuth ofrece grandes mejoras sobre modelos tradicionales de autenticación y propone su implementación en la Subdirección de Seguridad de la Información-México para mejorar el proceso de autenticación de los usuarios que acceden a los sistemas mantenidos por la misma organización.

2.2 MARCO TEÓRICO

2.2.1 CONTROL DE ACCESO

Whitman y Mattord (2016) afirman que el control de acceso es el proceso de verificar la identidad de una entidad y ofrecer la posibilidad de acceder a recursos limitados según restricciones establecidas

previamente. El control de acceso regula la admisión de usuarios a partes seguras de una organización (ya sea acceso lógico para sistemas de información o acceso físico para el acceso a instalaciones de la organización). El control de acceso es mantenido usando recursos como: una colección de políticas, programas para llevar a cabo estas políticas, y tecnologías que hagan cumplir las políticas. El control de acceso tiene cuatro procesos: obtener la identidad de la entidad que solicita acceso a un área física o lógica (identificación), confirmar la identidad de la entidad que solicita acceso a un área lógica o física (autenticación), determinar qué acciones puede ejecutar la entidad en el área física o lógica (autorización), y documentar las actividades de los individuos o sistemas autorizados. Un satisfactorio sistema de control de acceso aborda-sin importar si se trata de acceso lógico o físico-siempre los cuatro elementos.

Kim y Solomon (2014) afirman que el control de acceso son métodos usados para restringir y permitir el acceso a determinados elementos, como un automóvil, casa, computadoras, incluso un celular. El control de acceso es el proceso de proteger un recurso para que este sea usado sólo por aquellos que tengan permitido usarlo. El control de acceso protege los recursos de uso no autorizado. Las empresas usan el control de acceso para administrar qué empleados pueden o no pueden hacer algo. El control de acceso define quiénes son los usuarios (personas o procesos de computadoras), que pueden hacer los usuarios, a que recursos tienen acceso, y que operaciones pueden realizar. Los sistemas de control de acceso usan una gran variedad de tecnologías, incluyendo contraseñas, tokens de hardware, biometría, y certificados. El acceso puede ser concedido a instalaciones físicas, como edificios o habitaciones. El acceso puede ser concedido también a sistemas de computada y datos.

2.2.2 AUTENTICACIÓN

Whitman y Mattord (2016) afirman que la autenticación es el proceso de validar la identidad de una entidad. Es estar seguro de que la entidad es quien dice ser. Existen cuatro tipos de autenticación: autenticación por contraseña, autenticación por tokens encriptados o tarjetas inteligentes, autenticación biométrica, autenticación por patrones de identificación (la voz, por ejemplo).

Sinn (2008) afirma que la autenticación es la garantía de que una entidad es quien dice ser. Existen en general dos tipos de autenticación. *Autenticación de Entidad*, que provee información de la entidad involucrada en una actividad que la entidad desea realizar; la forma más común de realizar este tipo de autenticación es con el uso de un usuario y contraseña. El segundo tipo de autenticación es la *Autenticación de Origen de Datos*; también llamado, autenticación de no repudio. Este proceso identifica a una entidad como el autor u origen de datos.

2.2.3 AUTORIZACIÓN

La RAE (2017) afirma que la autorización es el acto de una autoridad por el cual se permite a alguien una actuación en otro caso prohibida. Además, no especifica su definición necesariamente en contextos informáticos.

Sankar y Sundaralingam (2004) afirman que la autorización, en sistemas informáticos, establece qué cosas se te están permitidas hacer después de que te hayas identificado a ti mismo; las acciones permitidas generalmente están basadas en propiedades de la entidad, políticas, identidad, etc. La autorización está estrechamente ligada a la autenticación en la mayoría de sistemas informáticos y es usada para evitar el uso no autorizado de recursos; sin embargo, la autorización no siempre requiere una autenticación previa.

2.2.4 TRAZABILIDAD

Se entiende la trazabilidad como el conjunto de aquellos procedimientos preestablecidos y autosuficientes que permiten conocer el histórico, la ubicación y la trayectoria de un producto a lo largo de la cadena de un proceso y momento dado, a través del uso de herramientas determinadas (AECOC, 2017).

Pinzon (2010) afirman que la trazabilidad puede ser definida como la capacidad de seguir la historia, la aplicación o la localización de todo aquello que está bajo consideración. Al considerar un producto, la trazabilidad puede estar relacionada con el origen de las partes, el procesamiento y distribución.

2.2.5 SERVICIOS WEB

Un Servicio Web es una interfaz de red, ubicada entre dos aplicaciones, que hace accesible la funcionalidad de una aplicación informática, construida usando estándares de tecnologías de internet. En otras palabras, si una aplicación puede ser accedida a través de una red usando una combinación de protocolos como HTTP, XML, SMTP, o Jaber, entonces estamos hablando de un Servicio Web. Un Servicio Web es una capa de abstracción que separa la plataforma y el lenguaje de programación haciendo posible la integración de diferentes sistemas informáticos independientemente del lenguaje de programación en el que estén definidos (Snell, Tidwell, y Kulchenko, 2002).

Tabor (2002) afirma que los Servicios Web son aplicaciones modulares diseñadas para hacer posible la interacción de dos o más sistemas independientemente del lenguaje de programación en el que estén definidos. Los servicios web surgieron por la necesidad de hacer posible la comunicación entre organizaciones haciendo uso de tecnologías de internet como estándar de comunicación y superar así diferencias de software y hardware.

2.2.6 SERVICIOS WEB RESTFUL

Los Servicios Web Restful son un tipo específico de servicios web basados en el estilo de arquitectura REST (Representational State Transfer) que define una interfaz uniforme que hace posible que los servicios web trabajen bien en el internet. La arquitectura REST es considera una arquitectura cliente-servidor y está diseñada para usar los protocolos de comunicación HTTP (Oracle, 2017).

Patni (2017) afirma que los servicios web restful son una forma de proveer interoperabilidad entre sistemas de computadora ubicadas en el internet; estos servicios permiten a los sistemas acceder y manipular recursos usando un uniforme y predefinido grupo de operaciones (GET, POST, PUT, DELETE). Una de las características más importantes de los servicios web restful es que no son capaces de guardar estados, es decir una vez realizada una operación entre el cliente y el servidor, el servidor no es capaz de recordar la interacción anterior.

2.2.7 PROTOCOLO HTTP

Gourley (2002) precisa que el protocolo HTTP (Hypertext Transfer Protocol) es el protocolo más usado por los sistemas para comunicarse a través del *World Wide Web* (www). Este protocolo es muy famoso por ser capaz de permitir comunicaciones de doble sentido entre un navegador y un servidor web. El protocolo HTTP define: cómo el cliente y servidor ser comunican, el formato de los mensajes usados durante la comunicación, la forma de identificar los tipos de mensaje y recursos asociados a este.

Consortium (1999) afirma que el Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como

el World Wide Web. Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libres de errores.

2.2.8 MODELO DE NEGOCIOS BUSINESS TO BUSINESS

Christopth (2003) afirma que, en el ámbito del software, empresa-a-empresa (B2B) es una arquitectura de software diseñada para conectar diferentes sistemas informáticos y hacer que estos interactúen entre sí; permite el flujo de datos (lectura y escritura) entre ellos haciendo uso de un proceso llamado integración. Estos modelos de arquitectura necesitan del uso de un estándar o lenguaje común que haga posible su comunicación.

Hall (2017) afirma que, en el ámbito comercial el término empresa-a-empresa (B2B) es un modelo de interacción comercial entre dos o más empresas que interactúan entre sí con el fin de realizar transacciones conjuntas; haciendo uso, generalmente, de tecnologías de la información. El término empresa-a-empresa es también usado en el comercio electrónico, en el que interactúan diferentes empresas; por ejemplo, un banco y las empresas que realizan transacciones a través de ella.

2.2.9 OAUTH2

Richer y Sanso (2017) afirman que OAuth2 es un protocolo de autorización que permite a terceros (clientes) acceder a contenidos propiedad de un usuario (alojados en aplicaciones de confianza, servidor de recursos) sin que éstos tengan que manejar ni conocer las

credenciales del usuario. Es decir, aplicaciones de terceros pueden acceder a contenidos propiedad del usuario, pero estas aplicaciones no conocen las credenciales de autenticación.

OAuth es un estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas. Se trata de un protocolo que permite autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, móviles y Web. OAuth2 permite a un usuario del sitio A compartir su información en el sitio A (proveedor de servicio) con el sitio B (llamado consumidor) sin compartir toda su identidad. Para desarrolladores que consumen servicios, OAuth es un método con el que se puede interactuar con datos protegidos. Para desarrolladores de proveedores de servicio, OAuth proporciona a los usuarios un acceso a sus datos al mismo tiempo que protege las credenciales de su cuenta (RFC6749, 2012).

2.2.9.1 QUÉ NO ES OAUTH

OAuth2 es usado en una gran variedad de proyectos de software que exponen sus servicios a través del uso de APIs. OAuth2 participa en el proceso de asegurar los datos de un software; sin embargo, algunas veces se tiende a pensar en OAuth2 como un sistema ubicuo (que todo lo puede), de allí la importancia de tener claro qué cosa no es OAuth2.

Richer y Sanso (2017) afirman:

1. *OAuth2 no puede ser definido fuera del contexto del uso del protocolo HTTP:* OAuth2 no brinda documentación oficial para su uso en protocolos que no sean HTTP.
2. *OAuth2 no es un protocolo de autenticación*: OAuth2 por sí sólo no brinda ninguna información acerca de quién es el usuario.
3. *OAuth2 no define un mecanismo de delegación usuario-a-usuario:* OAuth delega mecanismos de delegación usuario-a-cliente.

4. *OAuth2 no define mecanismos para procesar los datos productos del proceso de autorización:* OAuth2 no define la estructura ni el contenido de cada uno de los datos productos del proceso de autorización.
5. *OAuth2 no define el formato de un token*: La estructura de un token no es definido por OAuth2.
6. *OAuth2 no define métodos de encriptado*: OAuth2 no define como encriptar los tokens, en su lugar deja abierta la posibilidad de reusar métodos externos de encriptado como por ejemplo JOSE (Javascript Object Signing and Encryption).
7. *OAuth2 no es un sólo protocolo sino la unión de varias especificaciones*.

2.2.9.2 PROCESO DE AUTORIZACIÓN ANTES DE OAUTH2

Richer y Sanso (2017) afirman que el problema de conectar múltiples servicios en la nube es relativamente nuevo debido a los cambios y avances del mundo del software durante los últimos años. Antes de la existencia de OAuth2 se solían usar dos métodos para comunicar dos o más softwares entre sí:

1. **Copiar las credenciales de un usuario y replicarlas en el cliente (un software tercero)**: Este escenario requiere que la aplicación cliente guarde el usuario y contraseña del usuario; este método sólo puede ser usado en aplicaciones que pertenecen a la misma empresa ya que un usuario no compartiría su contraseña con una empresa que no conoce. Esta técnica expone la contraseña del usuario y además el sistema no es capaz de diferenciar entre una petición del mismo usuario y una petición de un cliente.

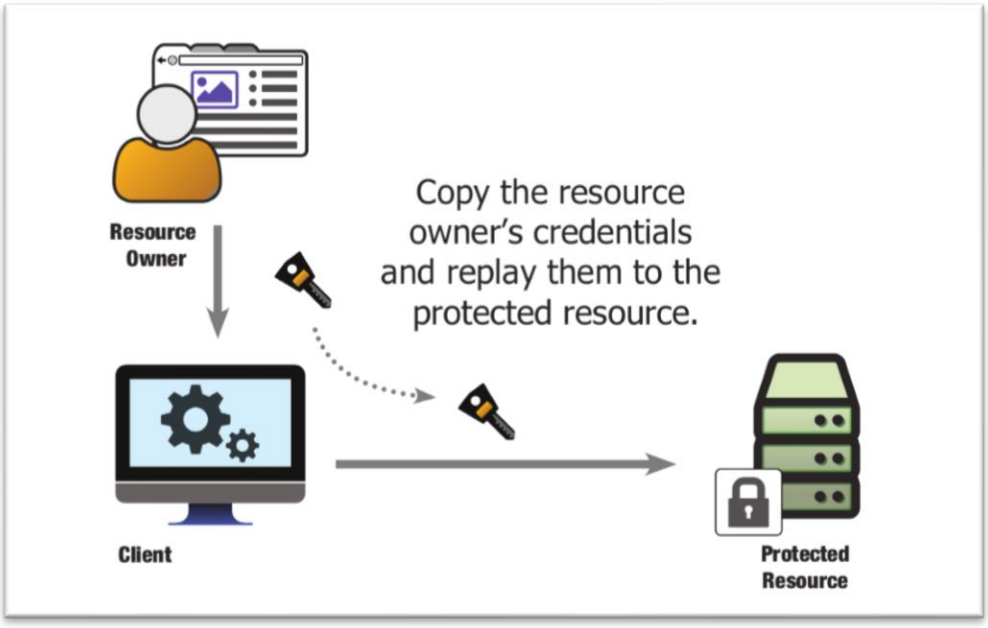


Figura 6. Proceso de autorización antes de OAuth2 (Richer y Sanso, 2017)

2. **Dar a los usuarios una contraseña especial que sólo debe de ser usada por softwares de terceros:** Este escenario requiere que el usuario cree una contraseña especial para cada aplicación cliente; requiere además que alguien se ocupe de administrar estas contraseñas.

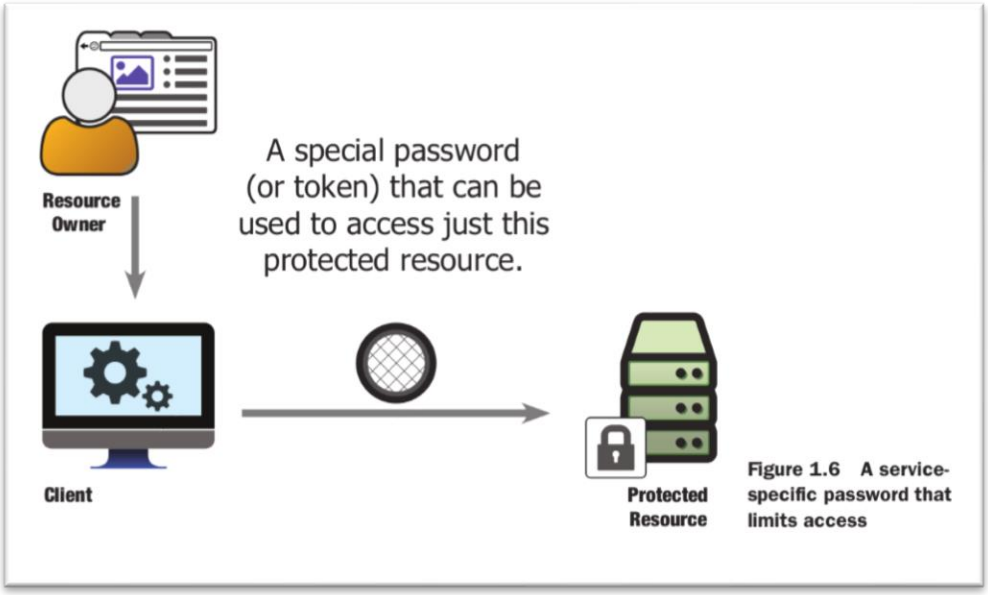


Figura 7. Método de dar a los usuarios una contraseña especial que sólo debe de ser usada por softwares de terceros (Richer y Sanso, 2017)

Finalmente, el almacenamiento de las credenciales de un usuario para su posterior envío en cada transacción es un anti patrón llamado *password-sharing;* OAuth2 reemplaza este anti patrón mediante un protocolo de delegación que es simultáneamente más seguro y más fácil de usar.

2.2.9.3 DELEGACIÓN DE ACCESO

Richer y Sanso (2017) afirman que una de las partes más fundamentales de OAuth2 es el concepto de "*Delegación*". La Delegación es el proceso de otorgar a un cliente la autoridad para acceder a recursos protegidos en su representación; para que esto ocurra OAuth2 introduce otro componente en el sistema llamado "Servidor de autorización" cuya función es la de otorgar los permisos necesarios para acceder a los recursos protegidos.

2.2.9.4 ACTORES DE OAUTH2

Richer y Sanso (2017) afirman que existen cuatro principales actores en un sistema OAuth2: Cliente (client), Servidor de autorización (Authorization Server), Propietario del recurso (Resource Owner) y Recurso protejido (Protected Resource); cada uno de estos componentes es responsable de diferentes partes del protocolo y todos trabajan juntos para hacer que OAuth2 funcione.

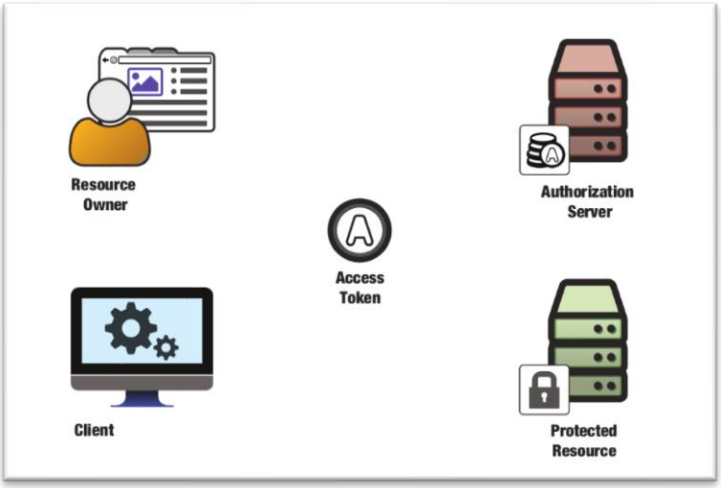


Figura 8. Principales actores del protocolo OAuth2 (Richer y Sanso, 2017)

2.2.9.4.1 CLIENTE (CLIENT)

Boyd (2012) afirma que un cliente es una aplicación que hace peticiones a una API para solicitar acciones sobre recursos protegidos en representación del propietario de los recursos y con su autorización.

Richer y Sanso (2017) afirman que en OAuth2, un cliente es una pieza de software que intenta tener acceso a recursos protegidos en representación del propietario de los recursos. Gracias al diseño de OAuth2, el cliente es el componente más simple y sus responsabilidades están centradas en obtener un token de un servidor de autorización y usar el token en los recursos protegidos. El cliente no necesita entender el contenido del token.

2.2.9.4.2 SERVIDOR DE AUTORIZACIÓN (AUTHORIZATION SERVER)

Boyd (2012) afirma que un servidor de autorización obtiene el consentimiento del propietario del recurso y después distribuye Access Tokens hacia los clientes para que estos puedan acceder a los recursos protegidos alojados en un servidor de recursos.

Richer y Sanso (2017) afirman que un servidor de autorización es un servidor HTTP que actúa como componente central en un sistema OAuth2. El servidor de autorización autentica al propietario de los recursos y a los clientes, provee mecanismos para permitir que los propietarios de los recursos autoricen a los clientes, y distribuye tokens a los clientes. Algunos servidores de autorización brindan también funcionalidades adicionales tales como la capacidad de recordar decisiones de autorización.

2.2.9.4.3 PROPIETARIO DEL RECURSO (RESOURCE OWNER)

Boyd (2012) afirma que el *propietario del recurso (resource owner)* es típicamente el usuario de una aplicación; el propietario de un recurso

tiene la capacidad de otorgar acceso a los datos (de su propiedad) que están alojados en un servidor de recursos.

Richer y Sanso (2017) afirman que el propietario de los recursos es una entidad que tienen la autoridad de delegar acceso al cliente. A diferencia de las otras partes de un sistema OAuth2, el propietario de los recursos no es una pieza de Software. En la mayoría de los casos el propietario de los recursos es una persona que interactúa con un navegador (un usuario).

2.2.9.4.4 RECURSO PROTEGIDO (PROTECTED RESOURCE)

Boyd (2012) afirma que es un servidor que almacena recursos pertenecientes a un usuario y que son protegidos por OAuth2. Éste es típicamente un API que almacena y protege datos como fotos, calendarios videos, o contactos (Boyd, 2012).

Richer y Sanso (2017) afirman que un recurso protegido debe de estar disponible a través del acceso a un servidor HTTP. Estos recursos requieren un token para poder ser accedidos. Los recursos protegidos requieren validar los tokens presentados y determinar cómo responder a las peticiones. En una arquitectura OAuth2, los recursos protegidos tienen la última palabra al determinar si aceptar o no un token.

2.2.9.5 COMPONENTES DE OAUTH2

2.2.9.5.1 ACCESS TOKEN

Finnigan (2017) afirma que un Access Token es un objeto que describe el contexto de seguridad de un proceso. Un Access Token contiene información de la identidad y privilegios de una cuenta de usuario asociada con el proceso. En otras palabras, un Access Token es una credencial que puede ser usada por un cliente para acceder a los recursos expuestos por un API.

Richer y Sanso (2017) afirman que en el ámbito de OAuth2, un *Access Token* es una cadena de caracteres, generalmente cifrados, que son usados como una llave para acceder a recursos protegidos de un sistema. El uso de este tipo de token evita la exposición de las credenciales del usuario durante la transmisión de información. Para poder conseguir un Access Token, los sistemas clientes deben de pasar por un proceso de autenticación y autorización dentro de un servidor en el que confían todos los miembros involucrados en el proceso de control de acceso.

2.2.9.5.2 REFRESH TOKEN

Richer y Sanso (2017) afirman que un Refresh Token tiene un concepto similar al Access Token; lo que lo hace diferente es que este token nunca es enviado al recurso protegido, en vez de eso el cliente usa el Refresh Token para obtener un nuevo Access Token en el Servidor de Autorización sin la necesidad de pedir las credenciales del usuario.

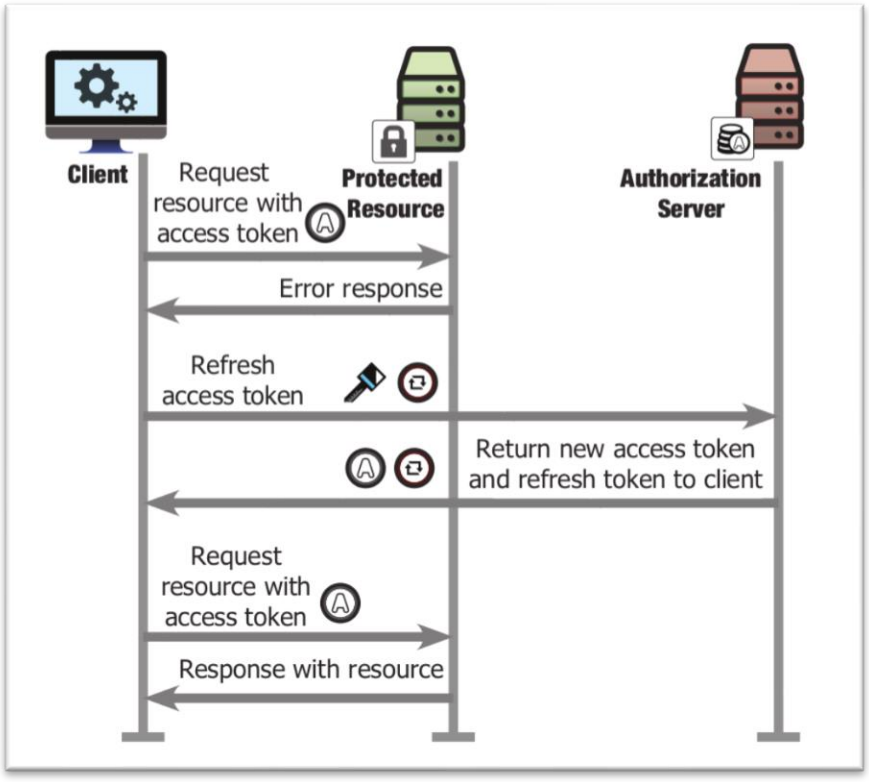


Figura 9. Cómo usar un Refresh Token (Richer y Sanso, 2017)

Spasovski (2013) afirma que un Refresh Token es un tipo de token que solo puede ser conseguido del servidor de autorización. A diferencia del Access Token, la existencia de este tipo de token, en el proceso de control de acceso, es opcional. El objetivo de este token es dar a los clientes la posibilidad de renovar o extender la autorización obtenida en el pasado a través de la generación de un nuevo Access Token. Estos tokens deben de ser almacenados en lugares seguros debido a que su exposición significa un peligro para el control seguro de datos.

2.2.9.5.3 SCOPE

Richer y Sanso (2017) afirman que un Scope es una representación de permisos a un recurso protegido. Los Scopes son representaciones de cadenas que pueden ser combinadas en una lista de permisos separados por comas. El formato y estructura de un Scope no está definido por OAuth2. Los Scopes son un mecanismo importante para limitar el acceso de un cliente. Los Scopes son definidos por el recurso protegido basado en los APIs que este ofrece.

Bihis (2015) afirna que un Scope sólo puede ser definido en el contexto de un Token. Un Scope representa una lista de recursos protegidos a los que el cliente puede acceder. los Scopes pueden abarcar una gran variedad de recursos como por ejemplo: la abilidad de leer y/o modificar datos con la cuenta de un usuario en específico; o puede ser tan específico como la abilidad de sólo acceder al nombre y apellido de un usuario. Los permisos de un Scope se adquieren al mismo tiempo en el que se adquiere un token.

2.2.9.5.4 AUTHORIZATION GRANT

Richer y Sanso (2017) afirman que un Authorization Grant es el medio por el cual un Cliente obtiene acceso a un recurso protegido usando el protocolo OAuth2, y si el proceso es satisfactorio se termina por obtener un token. Es un término que produce confusión debido a que el

mismo termino es usado para definir el mecanismo por el cual el usuario delega la autorización, así como como el acto de delegar por sí mismo.

Un Authorization Grant es una credencial que representa la autorización del propietario del recurso (para brindar acceso a recursos protejidos) usado por el cliente para obtener un Access Token. OAuth2 define cuatro tipos: Authorization Code, Implicit, Resource Owner Password Credentials, y Client Credentials (RFC6749, 2012).

2.2.9.6 AUTHORIZATION GRANT TYPES

Richer y Sanso (2017) afirman que Grant Types son métodos por los que un cliente puede obtener un Access Token; gracias a la existencia de más de un método es posible aplicar OAuth en diferentes escenarios. En OAuth2 existen cuatro métodos de obtener un Access Token:

a. Code Grant Type.
b. Implicit Grant Type.
c. Credentials Grant Type.
d. Resource Owner Credentials Grant Type.

2.2.9.6.1 CODE GRANT TYPE

Richer y Sanso (2017) afirman que en este método el cliente envía una solicitud de acceso al propietario del recurso (generalmente un usuario), a través del servidor de autorización; el servidor de autorización envía un código de autorización (Authorization Code) al cliente a través de la URL que está usando un navegador; finalmente el cliente envía este código al servidor de autorización para recibir un Access Token.

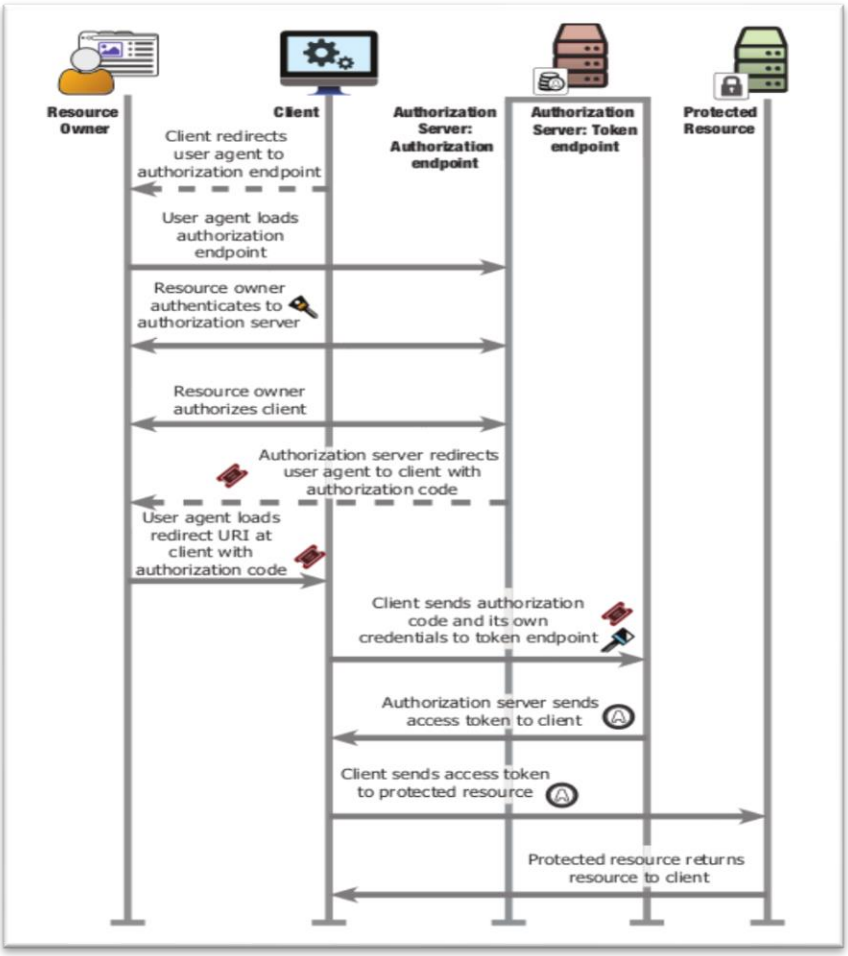


Figura 10. Cómo obtener un Access Token usando el método Code Grant Type (Richer y Sanso, 2017)

Keycloak (2018) afirma que este es un protocolo recomendado en escenarios en los que se usa un navegador. Los pasos para obtener un Access Token usando este método son:

1. En navegador ingresa a una aplicación, la aplicación nota que el usuario no está logeado por lo que el navegador redirige la peticion al servidor de autorización. El navegador envía un parámetro indicando a donde debe de dirigirse el navegador despues de que el servidor de autorización termine su trabajo.
2. El servidor de autorización autentica al usuario crea un código unico que tiene tiempo limitado de vida y además sólo puede ser usado una única vez. Una vez autenticado el usario, el servidor de autorización redirige la petición indicada en el paso 1.

3. La aplicación cliente extrae el código unico que es retornado del servidor y luego invoca una peticion POST en el servidor de autorización solicitando un Access Token (esta solicitud debe de incluir el codigo obtenido en el paso 2).

2.2.9.6.2 IMPLICIT GRANT TYPE

Richer y Sanso (2017) afirman que este método obtiene el Access Token, solicitado por el cliente, en la primera respuesta del servidor de autorización evitando pasos adicionales.

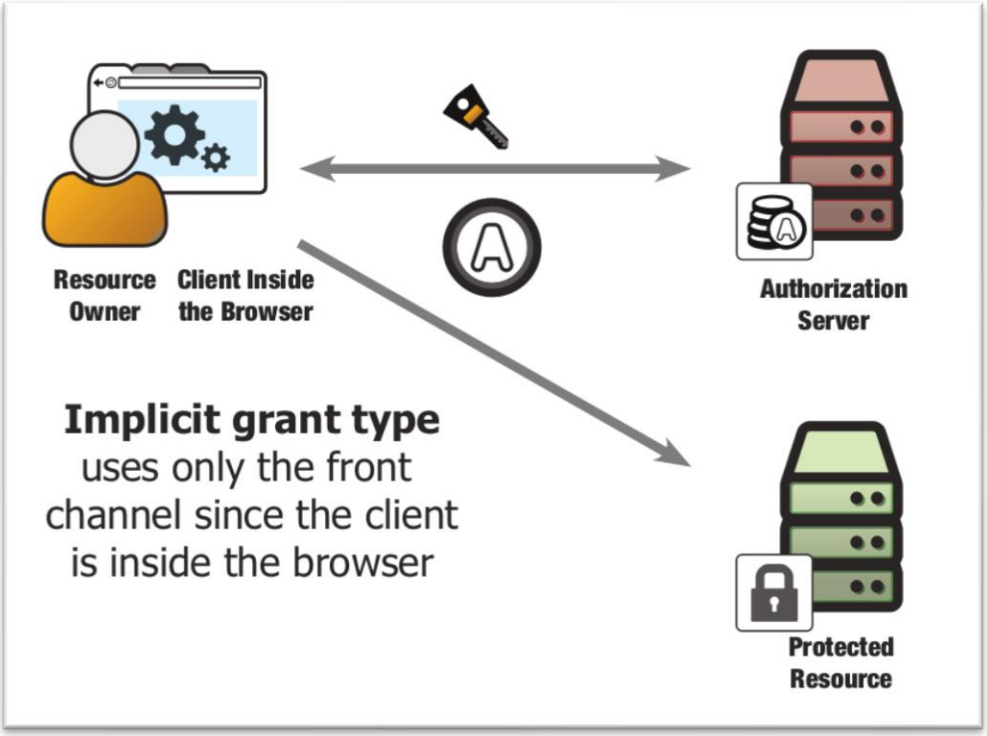


Figura 11. Método Code Grant Type (Richer y Sanso, 2017)

Keycloak (2018) afirma que este es un método basado en el uso de un navegador, similar al método Code Grant Type con la diferencia que este método usa menos pasos y no existe un Refresh Token involucrado. Este método no es recomendado ya que da la posibilidad de crear tokens de larga duración y que son vulnerables al robo. Los pasos para usar este método son:

1. El navegador ingresa a la aplicación, la aplicación identifica que el usuario no está autenticado por lo que redirige la petición hacia el servidor de autorización. La aplicación envía un parámetro indicando el siguiente paso después de que el servidor de autorización haya terminado su trabajo.
2. El Servidor de autorización autentica al usuario y crea un Access Token. El servidor de autorización redirige la petición a la dirección URL indicada en el paso 1.

2.2.9.6.3 CLIENT CREDENTIALS GRANT TYPE

Richer y Sanso (2017) afirman que este método implica la no existencia de un propietario del recurso explicito, es decir el dueño de los recursos no es un usuario sino un software. Este caso es común en contextos donde dos o más aplicaciones desean comunicarse entre sí sin la intervención de un usuario.

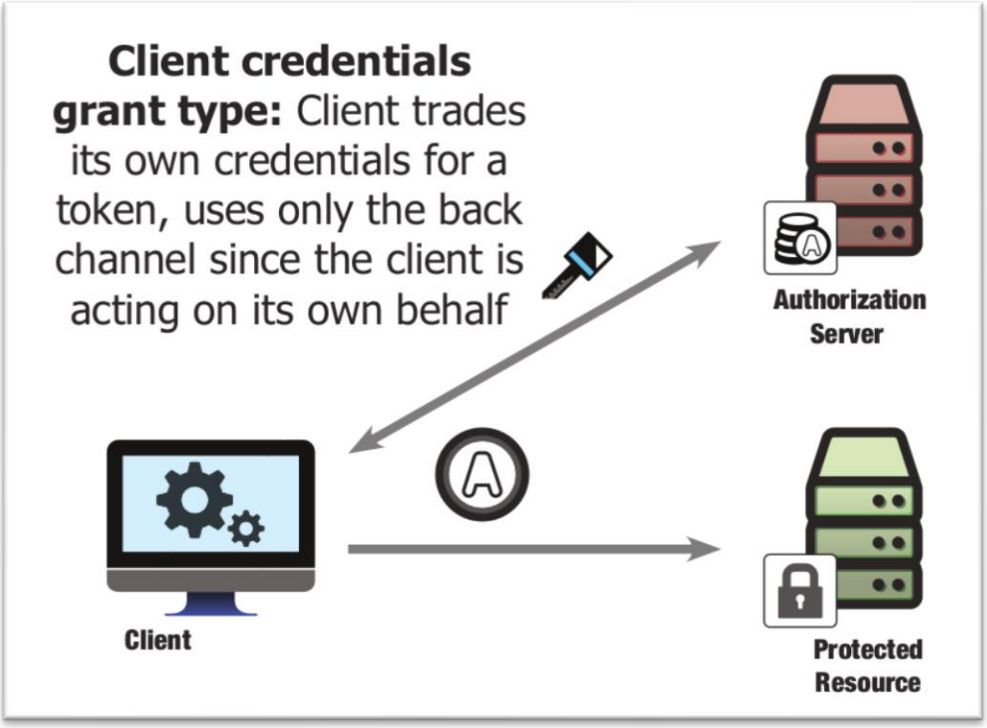


Figura 12. Método Client Credentials Grant Type (Richer y Sanso, 2017)

Keycloak (2018) afirma que este método es usado generalmente por clientes REST ya que en vez de obtener un token que represente la autorización de un usuario, se crea un token basado en los permisos que un servicio asociado a la aplicación cliente tiene.

2.2.9.6.4 RESOURCE OWNER CREDENTIALS GRANT TYPE

Richer y Sanso (2017) afirman que este método es usado cuando la aplicación tiene las credenciales (username, password) de un usuario y desea obtener un token haciendo uso de estas credenciales. Este método es similar al método usando antes de la existencia de OAuth2 donde el cliente enviaba las credencias del usuario en cada transacción. (Richer & Sanso, 2017)

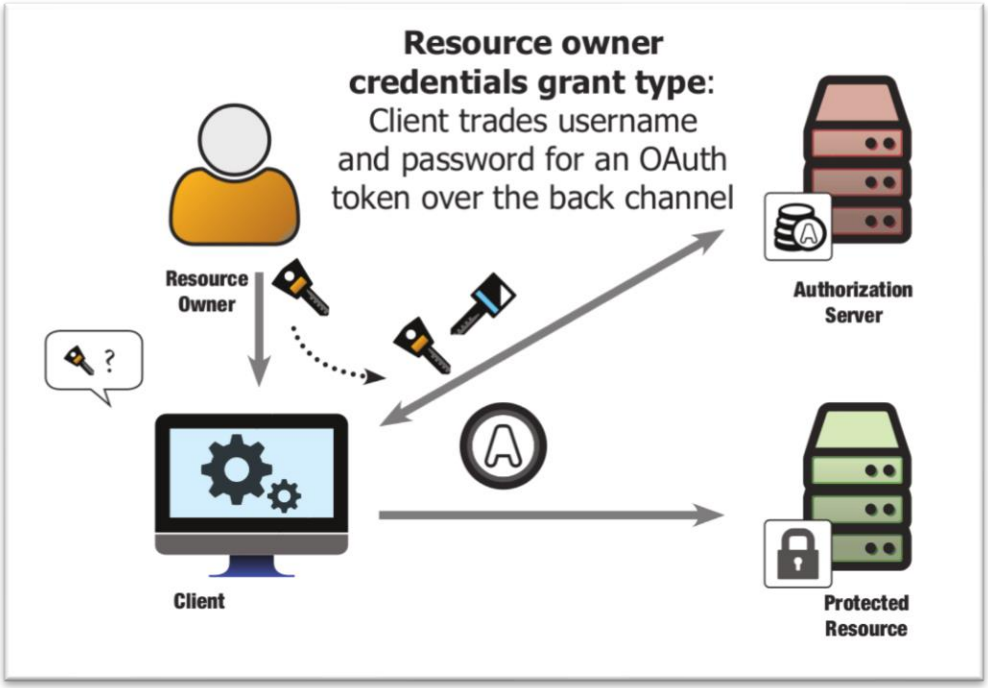


Figura 13. Método Resource Owner Credentials Grant Type (Richer y Sanso, 2017)

Keycloak (2018) afirma que este método es también conocido como Direct Access Grant ya que la aplicación cliente solicita un token por medio del usuario y contraseña del usuario por lo que la cantidad de pasos es inferior al resto de métodos existentes.

2.2.10 JSON WEB TOKEN (JWT)

Richer y Sanso (2017) afirman que OAuth2 no especifica la estructura de un token debido a que OAuth2 puede ser usado en una gran variedad de escenarios con diferentes características, requerimientos, y perfiles de riesgo. Los tokens pueden expirar, revocados, ser indefinidos, o una combinación de las tres características anteriores dependiendo de las circunstancias. Esta flexibilidad y modularidad permite a OAuth2 ser adaptable en escenarios complejos. Es en este escenario que JWT aparece como un estándar respecto a la estructura de un Token.

Figura 14. Estructura de un JWT sin cifrar pero codificada usando Base64URL-encoded (Richer y Sanso, 2017)

A simple vista puede parecer que un JWT es una cadena de caracteres aleatorios, pero se debe de tomar atención a que existe dos secciones separas por un "punto". Cada parte separada por comas no es una cadena aleatoria sino la representación Base64URL-encoded de un objeto JSON. Si decodificamos la primera parte del JWT obtendremos un objeto como el siguiente:

Figura 15. Estructura de un JWT sin cifrar pero con Base64URL decodificado (Richer y Sanso, 2017)

2.2.11 PROTECCIÓN CRIPTOGRÁFICA DE TOKEN: JSON OBJECT SIGNING AND ENCRYPTION (JOSE)

Richer y Sanso (2017) afirman que JSON Object Signing and Encryption (JOSE) es un estándar que provee herramientas para encriptar objetos JSON simétrica (HMAC) y asimétricamente (RSA) haciendo uso de llaves públicas y privadas.

2.2.12 OPENID

Richer y Sanso (2017) afirman que *OpenID Connect* es un estándar abierto, que define la manera de usar OAuth2 en el proceso de autenticación de manera descentralizada. OpenID está diseñado para usar objetos JSON cifrados y encriptados que permitan transmitir datos de manera segura a través de diferentes aplicaciones.

Rafeeq (2008) afirma que OpenID es un protocolo que permite a una persona usar una URL como una identidad y usar la misma identidad en múltiples aplicaciones web que soporten este protocolo. OpenID permite: acceder a diferentes aplicaciones web sin la necesidad de ingresar un usuario y contraseña como información. Implementar sistemas de única autenticación (Single Sign On-SSO) para múltiples aplicaciones dentro de una organización. Reduce el riesgo de errores durante el proceso de autenticación.

2.2.12.1 ID TOKEN

Richer y Sanso (2017) afirman que ID Token solo puede ser definido en el ámbito de OPENID. El ID Token es un token firmado (JWT) que es enviado al cliente junto al *Access Token* que es creado durante el proceso de autorización OAuth2. El ID Token contienen un conjunto de datos relacionados a la sesión de autenticación incluyendo la identificación del usuario.

2.2.13 SCRUM

Scrum es un marco de trabajo para desarrollar, entregar y mantener productos complejos; a la vez, ayuda a entregar productos de máximo valor posible productiva y creativamente. Scrum es liviano, fácil de entender, y difícil de dominar. Scrum no es un proceso, una técnica o método definitivo; en lugar de eso es sólo un marco de trabajo en el cual se pueden emplear varios procesos y técnicas (Sutherland & Schwaber, 2017).

Scrum es un Framework que se aplica a la colaboración efectiva de equipos de trabajo en contextos complejos. Scrum no es una metodología, sino que aplica el método científico empírico para lidiar con situaciones no predecibles y resolver problemas complejos (SCRUM, 2018).

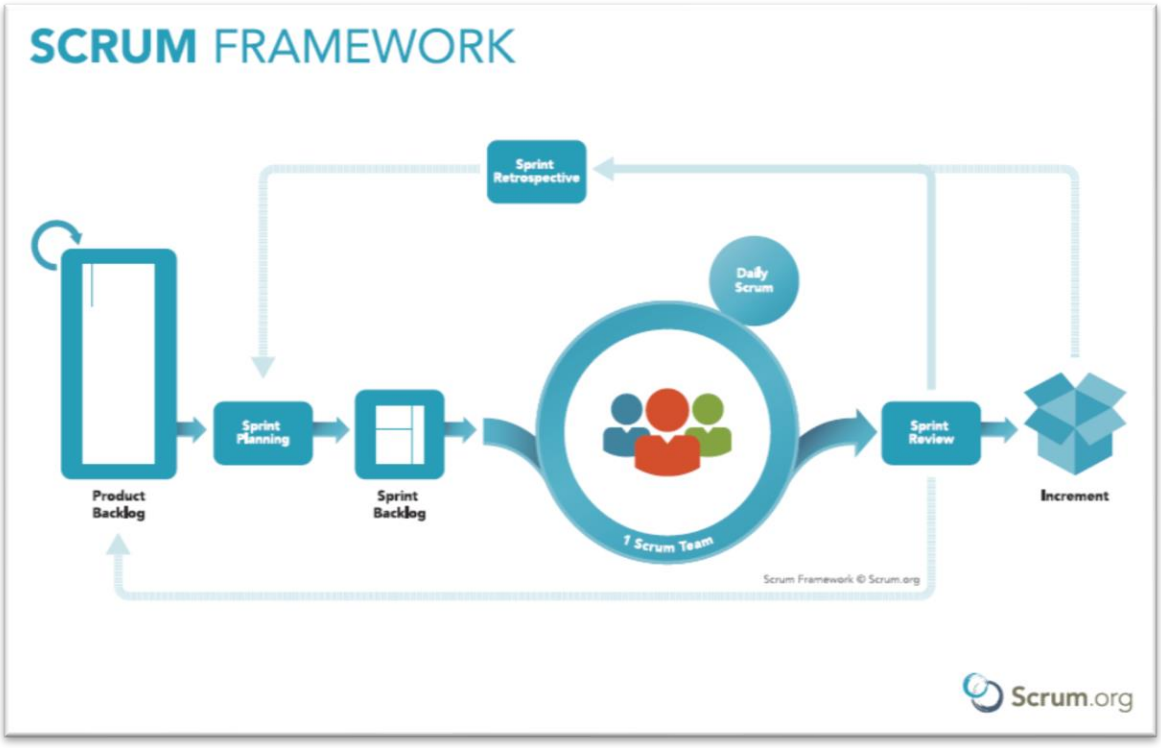


Figura 16. El marco de trabajo Scrum (SCRUM, 2018)

2.2.13.1 PILARES DE SCRUM

Scrum se basa en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. Los tres pilares que soportan toda la implementación del control de procesos empírico son: transparencia, inspección y adaptación (Sutherland y Schwaber, 2017).

2.2.13.1.1 TRANSPARENCIA

"Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. La transparencia requiere que dichos aspectos sean definidos por un estándar común, de tal modo que los observadores compartan un entendimiento común de lo que se están viendo." (Sutherland y Schwaber, 2017, pág. 4)

2.2.13.1.2 INSPECCIÓN

"Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones indeseadas. Su inspección no debe ser tan frecuente como para que interfiera en el trabajo. Las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos en el mismo lugar de trabajo." (Sutherland & Schwaber, 2017, pág. 5)

2.2.13.1.3 ADAPTACIÓN

"Si un inspector determina que uno o más aspectos de un proceso se desvían de límites aceptables y que el producto resultante será inaceptable, el proceso o el material que está siendo procesado deben ajustarse. Dicho ajuste debe realizarse cuanto antes para minimizar desviaciones mayores." (Sutherland & Schwaber, 2017, pág. 5)

2.2.13.2 EL EQUIPO SCRUM

"El Equipo Scrum consiste en un Dueño de Producto (Product Owner), y el Equipo de Desarrollo multifuncionales. Los equipos autoorganizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los Equipos Scrum entregan productos de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación." (Sutherland & Schwaber, 2017, pág. 6)

2.2.13.2.1 EL DUENO DE PRODUCTO (PRODUCT OWNER)

Sutherland y Schwaber (2017) afirman que El Dueño de Producto es el responsable de maximizar el valor del producto resultante del trabajo del equipo de desarrollo. El cómo se lleva a cabo esto podría variar ampliamente entre distintas organizaciones. El Dueño de Producto es la única persona responsable de gestionar la Lista del Producto (Product Backlog). La gestión de la Lista del Producto incluye:

- Expresar claramente los elementos de la Lista del Producto; Ordenar los elementos en la Lista del Producto para alcanzar los objetivos y misiones de la mejor manera posible.
- Optimizar el valor del trabajo que el equipo de desarrollo realiza
- Asegurar que la Lista del Producto es visible, transparente y clara para todos y que muestra aquello en lo que el equipo trabajará a continuación.
- Asegurar que el Equipo de Desarrollo entiende los elementos de la Lista del Producto al nivel necesario.

2.2.13.2.2 EL EQUIPO DE DESARROLLO (DEVELOPMENT TEAM)

Sutherland y Schwaber (2017) afirman que el equipo de desarrollo consiste en los profesionales que realizan el trabajo de entregar un incremento de producto "terminado" que potencialmente se pueda poner en producción al final de cada Sprint. Un Incremento "Terminado"

es obligatorio en la Revisión del Sprint. Solo los miembros del equipo de desarrollo participan en la creación del incremento o iteración. La organización es la encargada de estructurar y empoderar a los equipos de desarrollo para que estos organicen y gestionen su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Equipo de Desarrollo. Los Equipos de Desarrollo tienen las siguientes características:

- Son autoorganizados. Nadie (ni siquiera el Scrum Master) indica al equipo de desarrollo cómo convertir elementos de la *Lista del Producto* en Incrementos de funcionalidad potencialmente desplegables.
- Los equipos de desarrollo son multifuncionales, esto es, como equipo cuentan con todas las habilidades necesarias para crear un Incremento de producto.
- Scrum no reconoce títulos para los miembros de un equipo de desarrollo independientemente del trabajo que realice cada persona.
- Scrum no reconoce subequipos en los equipos de desarrollo, no importan los dominios que requieran tenerse en cuenta, como pruebas, arquitectura, operaciones o análisis de negocio.
- Los miembros individuales del equipo de desarrollo pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el equipo de desarrollo como un todo.

2.2.13.2.3 SCRUM MASTER

"El Scrum Master es responsable de promover y apoyar Scrum como se define en la Guía de Scrum. Los Scrum Masters hacen esto ayudando a todos a entender la teoría, prácticas, reglas y valores de Scrum." (Sutherland & Schwaber, 2017, pág. 8)

2.2.13.3 EVENTOS SCRUM

Sutherland y Schwaber (2017) afirman que en Scrum existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo (time-boxes), de tal modo que todos tienen una duración máxima. Scrum prescribe cuatro eventos formales, contenidos dentro del Sprint, para la inspección y adaptación:

- Planificación del Sprint (Sprint Planning)
- Scrum Diario (Daily Scrum)
- Revisión del Sprint (Sprint Review)
- Retrospectiva del Sprint (Sprint Retrospective)

2.2.13.3.1 PLANIFICACIÓN DE SPRINT (SPRINT PLANNING)

Sutherland y Schwaber (2017) afirman que el trabajo a realizar durante el Sprint se planifica en la planificación de Sprint. Este plan se crea mediante el trabajo colaborativo del Equipo Scrum completo. La Planificación de Sprint tiene un máximo de duración de ocho horas para un Sprint de un mes. Para Sprints más cortos el evento es usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña al Equipo Scrum a mantenerse dentro del bloque de tiempo. La Planificación de Sprint responde a las siguientes preguntas:

- ¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?
- ¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?

2.2.13.3.2 SCRUM DIARIO (DAILY SCRUM)

"El Scrum Diario es una reunión con un bloque de tiempo de quince minutos para el Equipo de Desarrollo. El Scrum Diario se lleva a cabo cada

día del sprint. En él, el Equipo de Desarrollo planea el trabajo para las siguientes 24 horas." (Sutherland & Schwaber, 2017, pág. 12)

2.2.13.3.3 REVISIÓN DEL SPRINT (SPRINT REVIEW)

"Al final del Sprint se lleva a cabo una Revisión de Sprint para inspeccionar el Incremento y adaptar la *Lista de Producto,* si fuese necesario. Durante la Revisión de Sprint, el Equipo Scrum y los interesados colaboran acerca de lo que se hizo durante el Sprint." (Sutherland & Schwaber, 2017, pág. 13)

2.2.13.3.4 RETROSPECTIVA DEL SPRNT (SPRINT RETROSPECTIVE)

Sutherland y Schwaber (2017) afirman que La Retrospectiva de Sprint es una oportunidad para el Equipo Scrum de inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente Sprint. La Retrospectiva de Sprint tiene lugar después de la Revisión de Sprint y antes de la siguiente Planificación de Sprint. Se trata de una reunión de, a lo sumo, tres horas para Sprints de un mes. Para Sprints más cortos el evento es usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El propósito de la Retrospectiva de Sprint es:

- Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas.
- Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras.
- Crear un plan para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo.

2.2.13.4 ARTEFACTOS SCRUM

Los artefactos de Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la

información clave, necesaria para asegurar que todos tengan el mismo entendimiento del artefacto (Sutherland & Schwaber, 2017).

2.2.13.4.1 LISTA DE PRODUCTO (PRODUCT BACKLOG)

La *Lista de Producto* es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El dueño del producto (Product Owner) es el responsable de la *Lista de Producto*, incluyendo su contenido, disponibilidad y ordenación. La *Lista de Producto* evoluciona a medida que el producto y el entorno en el que se usará también lo hacen. La Lista de Producto es dinámica; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil (Sutherland & Schwaber, 2017).

2.2.13.4.2 LISTA DE PENDIENTES DEL SPRINT (SPRINT BACKLOG)

La Lista de Pendientes del Sprint es el conjunto de elementos de la *Lista de Producto* seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. La Lista de Pendientes del Sprint es una predicción hecha por el Equipo de desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento "Terminado". La Lista de Pendientes del Sprint es un plan con un nivel de detalle suficiente como para que los cambios en el progreso se puedan entender en el Scrum Diario (Sutherland & Schwaber, 2017).

CAPÍTULO III
METODOLOGÍA DE LA INVESTIGACIÓN

3.1 TIPO Y NIVEL DE INVESTIGACIÓN

3.1.1 TIPO DE INVESTIGACIÓN

Caballero (2009), afirma que una investigación es aplicada cuando se tiene como objetivo solucionar un problema específico haciendo uso de técnicas o métodos.

Por lo expuesto, la presente investigación es de tipo aplicada ya que el objetivo no es generar nuevo conocimiento sino la de aplicar el conocimiento ya existente para solucionar un problema específico.

3.1.2 NIVEL DE INVESTIGACIÓN

Fernandez, Baptista, y Hernandez (2014) afirman que, los estudios descriptivos tienen el objetivo de describir, medir o recoger información de un fenómeno o proceso sin manipular las variables involucradas.

Por lo expuesto en el párrafo anterior, la presente investigación es una investigación de nivel descriptiva.

3.2 DISEÑO DE LA INVESTIGACIÓN

La presente investigación es no experimental y de diseño transversal, ya que se centra en el análisis de la situación de una o diversas variables en un único punto en el tiempo.

Un diseño no experimental, porque no se manipulan deliberadamente las variables y solo se observarán fenómenos tal y como se dan en su contexto natural, para después analizarlos (Fernandez, Baptista, y Hernandez, 2014).

3.3 POBLACIÓN Y MUESTRA

3.3.1 POBLACIÓN

La población está conformada por todos los servicios web que el software OPENFACT expone.

3.3.2 MUESTRA

Se usó una muestra no probabilística con juicio de experto. Se tomó como muestra el servicio web encargado de la creación de comprobantes de pago electrónico.

3.4 VARIABLES E INDICADORES

3.4.1 DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

VARIABLE

CONTROL DE ACCESO: Proceso de verificación de la identidad de un usuario, dando la posibilidad de acceder a operaciones y recursos limitados según lo permitido.

INDICADORES

AUTENTICACIÓN: Procedimiento que permite asegurar que un usuario es auténtico o quien dice ser.

AUTORIZACIÓN: Conjunto de permisos u operaciones permitidas a un usuario con el propósito de proteger los recursos de un sistema.

TRAZABILIDAD: Conjunto de procedimientos o pasos que permite seguir el proceso de evolución de un producto en sus diferentes etapas.

3.4.2 DEFINICIÓN OPERACIONAL

VARIABLE

X: Control de acceso.

INDICADORES

X1: Autenticación.

X2: Autorización.

X2: Trazabilidad.

3.5 TÉCNICAS E INSTRUMENTOS

3.5.1 TÉCNICAS PARA RECOLECTAR INFORMACIÓN

Se empleó la técnica de la observación para determinar si es posible acceder a un servicio web, expuesto por el software OPENFACT, usando los parámetros de seguridad determinados por el protocolo de seguridad OAuth2.

Adicionalmente se empleó la técnica de análisis documental para recolectar información acerca de la correcta forma de aplicar el protocolo OAuth2.

3.5.2 INSTRUMENTOS PARA RECOLECTAR INFORMACIÓN

Los instrumentos utilizados para recolectar datos fueron:

1. Ficha de observación (Anexo A): utilizado para registrar intentos (simulaciones manuales) de consumo de servicios web, utilizando diferentes parámetros durante el envío de peticiones HTTP.
2. Ficha de registro documental (Anexo B): utilizado para registrar información sobre cómo implementar el protocolo de seguridad OAuth2 en un software.

3.5.3 HERRAMIENTAS PARA LA RECOLECCIÓN DE DATOS

A continuación, se listan el conjunto de herramientas tecnológicas utilizadas durante la investigación.

Tabla 3

Herramientas tecnológicas utilizadas durante la investigación.

SOFTWARE	FABRICANTE	SERVICIO
Postman	Postdot Technologies Inc.	Software para enviar peticiones HTTP (GET, POST, PUT, DELETE) a servicios web de tipo restful.
Keycloak	Red Hat	Sistema SSO y OAuth2 que permite la administración de la seguridad de un sistema.

La presente tabla lista los softwares más importantes durante el proceso de la investigación.

3.5.4 TÉCNICA PARA APLICAR OAUTH2

No existe una técnica para aplicar OAuth2 en un proyecto de software; sin embargo, sugiero seguir los siguientes pasos como guía:

Tabla 4

Conjunto de pasos para aplicar OAuth a un proyecto de software.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Identificar y/o definir los actores del sistema	Diagrama de componentes del sistema	Realizar un dibujo que represente la interacción de los componentes entre sí, incluyendo los componentes que aún no existen, pero se	Desarrolladores

		espera tenerlos en el futuro	
Definir el alcance del sistema de seguridad	Lista de requerimientos de seguridad que se desea implementar	Listar todos los requerimientos de seguridad que se desea tener.	Desarrolladores
Desarrollar o decidir usar un servidor de seguridad OAuth2	Servidor se seguridad desplegado	Realizar un análisis comparativo de los diferentes productos en el mercado.	Desarrolladores
Definir reglas adicionales que el vendedor del servidor OAuth2 permita.	Configuración del servidor de seguridad	Analizar la documentación del servidor y seleccionar las mejores opciones para el sistema	Desarrolladores
Implementar la seguridad en cada uno de los componentes	Código fuente del módulo de seguridad de cada uno de los componentes	Escribir código que permita interactuar con el servidor se seguridad	Desarrolladores

La presente tabla (elaboración propia) representa un conjunto de pasos que un desarrollador puede seguir si éste desea aplicar el protocolo OAuth a su propio software; sin embargo, cada desarrollador puede definir sus propios pasos y métodos.

3.5.5 TÉCNICA PARA APLICAR SCRUM

La presente investigación utiliza Scrum como marco de trabajo para organizar las tareas y actividades involucradas en el proceso de implementación de requerimientos.

Tabla 5

Conjunto de pasos para planear un proyecto.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
asignar los roles de equipo.	Lista de roles	Identificar los roles del proyecto.	- Product Owner - Scrum master
Identificación de requerimientos.	Product backlog	Listar los requerimientos del sistema.	- Product owner
Priorización de requerimientos	Product backlog priorizado	Asignar prioridades a cada uno de los requerimientos.	- Product owner
Escribir historias de usuario	Descripción de historias de usuario	Descripción de las historias de usuario con la plantilla genérica como(rol), quiero(función) y para(propósito).	- Product owner - Scrum master - Desarrolladores
Revisión y/o demostración del sprint.	Revisión y/o demostración del sprint.	Presentar mediante imágenes u otras herramientas lo avanzado en el sprint.	- Scrum master - Desarrolladores

Esta tabla describe un conjunto de tareas necesarias para iniciar un proyecto de software basado en Scrum.

Tabla 6

Conjunto de actividades para ejecutar un proyecto basado en Scrum.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Reunión diaria (Daily meeting)	Lista de obstáculos	Realizar reuniones cortas al inicio de cada día para informar lo que cada miembro del equipo está haciendo y qué obstáculos encontraron el día anterior.	- Scrum master - Desarrolladores
Desarrollo	Código fuente	Escribir código fuente para satisfacer las historias de usuario.	- Desarrolladores
Review Metting	Review meeting	Verificar el cumplimiento de las historias de usuario del Sprint.	- Scrum master - Desarrolladores
Mejorar el proceso de desarrollo	Retrospectiva del scrum	Listar las actividades del Sprint y verificar su cumplimiento.	- Scrum master - Desarrolladores

La presente tabla representa el conjunto de actividades que se realizarán durante el proceso de aplicación de Scrum durante la presente investigación.

CAPITULO IV
RESULTADOS DE LA INVESTIGACIÓN

4.1 ARQUITECTURA DEL OPENFACT

Después de evaluar las limitantes del método HTTP Basic como método de autorización, se decidió utilizar el protocolo de seguridad OAuth2 para asegurar los servicios web restfull del software OPENFACT; como consecuencia se tomó la decisión de utilizar un servidor OAuth2 ya existente en el mercado cuyo nombre es Keycloak. A continuación, se muestra la arquitectura final de OPENFACT:

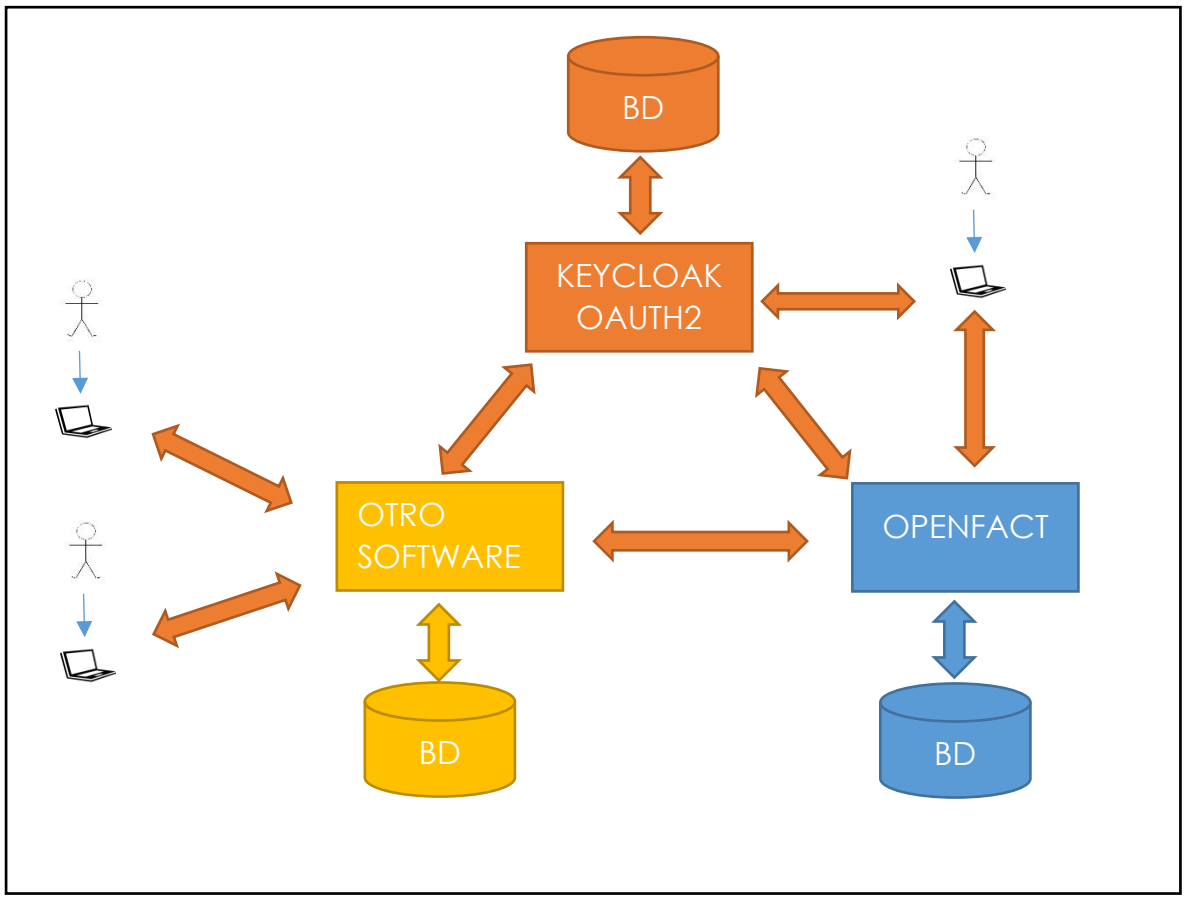


Figura 17. Arquitectura OAuth2 de OPENFACT.

4.2 RESULTADOS DE LA OBSERVACIÓN

La presente sección representa los resultados obtenidos a partir del uso de la ficha de observación (Anexo A) cuyo objetivo y descripción se encuentra en la sección 3.5.2 de la presente investigación.

Durante esta etapa se simularon peticiones HTTP al servicio web de OPENFACT para la creación de comprobantes de pago. Se utilizó el software *Postman* para simular las cabeceras y el cuerpo de las peticiones HTTP.

Tabla 4.1

Resultado del análisis documental, después de hacer simulaciones HTTP.

SERVICIO WEB	*http://midominio/api/organizations/{ miempresa}/sunat/documents/invoices*
ESCENARIO	RESULTADO
Peticiones sin la cabecera: ***Authorization***	Todas las peticiones retornaron errores HTTP con código 404 (forbidden).
Peticiones HTTP con cabeceras cuya estructura fue: ***Authorization: Bearer miToken***	Las peticiones HTTP con tokens no válidos; es decir, tokens que no fueron generados por el sistema de seguridad; retornaron errores HTTP 404 (forbidden).
	Las peticiones HTTP con tokens expirados retornaron errores HTTP 404 (forbidden).
	Las peticiones HTTP con tokens válidos retornaron

La presente la tabla representa los resultados obtenidos después de realizar simulaciones HTTP, utilizando el software Postman.

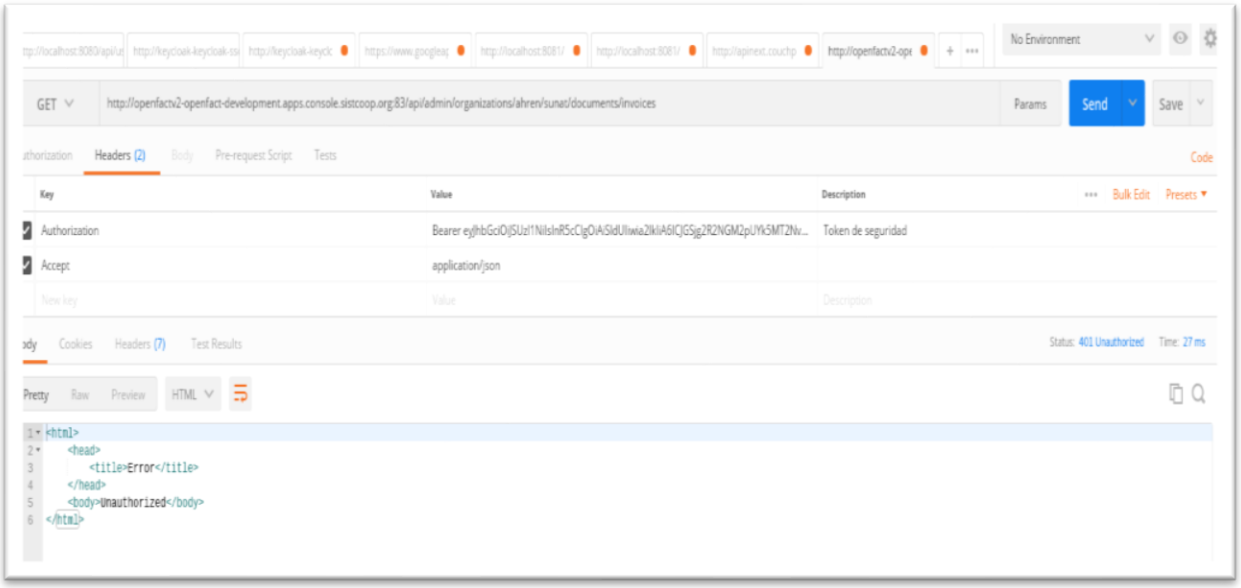


Figura 18. Simulación de una petición HTTP.

4.3 RESULTADOS DEL ANÁLISIS DOCUMENTAL

La presente sección representa los resultados obtenidos a partir del uso de la ficha de análisis documental (Anexo C).

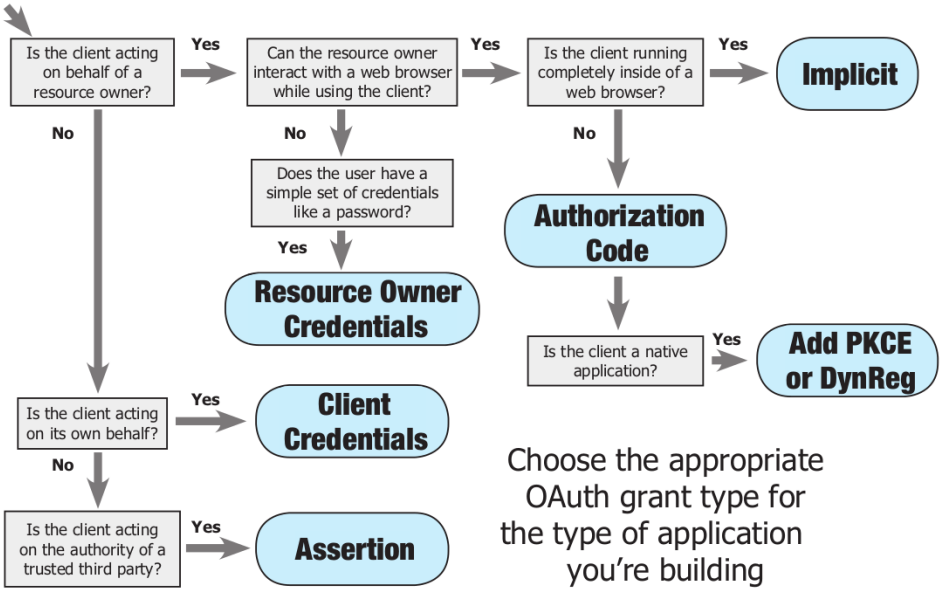
A continuación, se muestran los principales resultados encontrados después de realizar el análisis documental:

Tabla 7

Pasos a seguir cuando se desea escoger el adecuado Grant Type.

¿CÓMO ESCOGER EL APROPIADO GRANT TYPE?	
FECHA	25/06/2018
FUENTE	OAuth2 in Action (Ritcher y Sanso, 2017)
OBJETIVO	Determinar los métodos de obtención de tokens para cada uno de los componentes del sistema.
RESULTADO	
Pasos para escoger un adecuado método:	

1. ¿El Cliente está actuando en representación de un propietario de recursos particular? Y si es así, ¿tienes la habilidad de enviar al usuario a una página web dentro de un navegador? Entonces deberás de usar los métodos: Code Grant Type o Implicit Grant Type; la selección entre estos dos métodos dependerá del Cliente.
2. ¿El Cliente es usado completamente en un navegador web? Es decir, se trata de una aplicación que no tiene partes que se ejecutan en un servidor sino por el contrario la aplicación nace y muere completamente en el navegador. Si la respuesta es Si, entonces deberías usar el método Implicit Grant Type debido a su optimización para este tipo de casos. Si la la respuesta es No, entonces deberías de usar el método Code Grant Type ya que brinda las mejores propiedades de seguridad y flexibilidad.
3. ¿El Cliente es una aplicación nativa? Lo recomendable es usar el método Code Grant Type y adicionalmente a ello se recomienda incluir extensiones de seguridad como registro dinámico (DynReg) o intercambio de llaves seguras (PKCE)
4. ¿El Cliente actúa en su propia representación? Esto incluye el proceso de acceder APIs que no necesariamente con un usuario. Si la respuesta es SI, entonces se debe de usar el método Client Credentials Grant Type.
5. ¿El Cliente no es capaz de usar un navegador web? Si la respuesta es SI, entonces se debe de usar el método Resource Owner Credentials Type; sin embargo, este método es muy peligroso ya que envía las credenciales del usuario en cada petición.



Cómo escoger el adecuado Grant Type. (Richer & Sanso, 2017)

La presente tabla representa un aproximado del conjunto de pasos que se sugieren al momento de escoger el adecuado Grant Type durante el proceso de implementación del protocolo OAuth2.

Tabla 8

Pasos para obtener un token usando el método Code Grant Type.

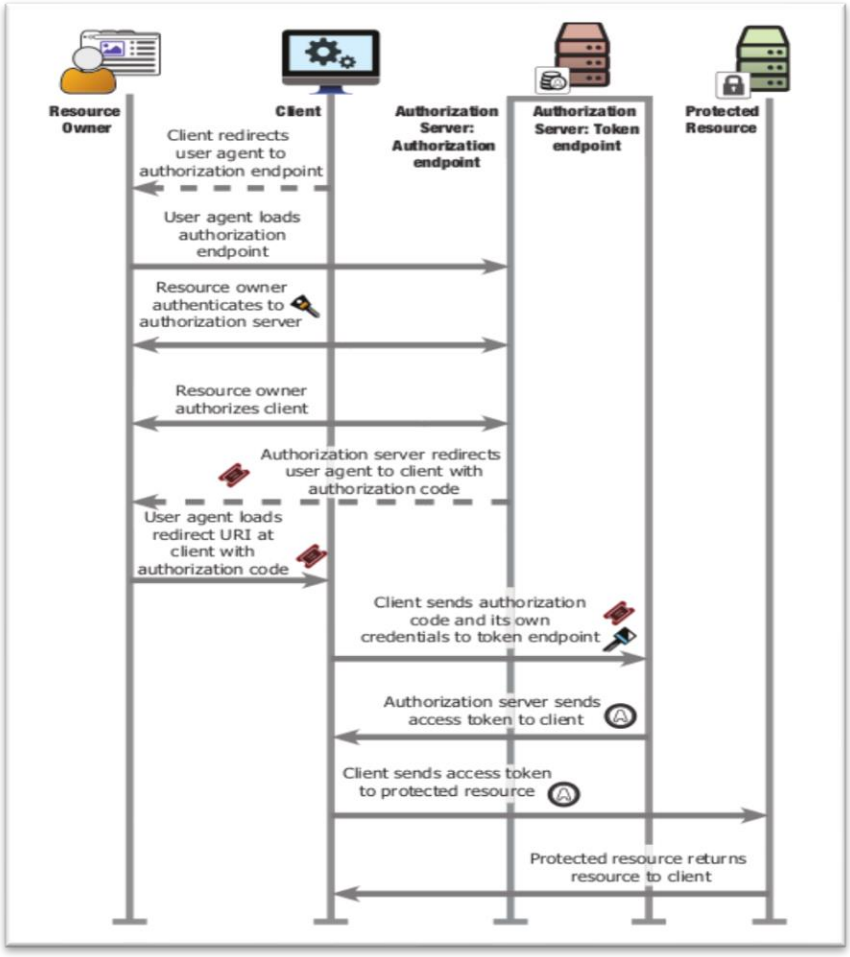
MÉTODO CODE GRANT TYPE	
FECHA	25/06/2018
FUENTE	OAuth2 in Action (Ritcher y Sanso, 2017)
OBJETIVO	Determinar los pasos para obtener un token válido usando.
RESULTADO	

Pasos para escoger un adecuado método:

1. En navegador ingresa a una aplicación, la aplicación nota que el usuario no está logeado por lo que el navegador redirige la peticion al servidor de autorización. El navegador envía un parámetro indicando a donde debe de dirigirse el navegador despues de que el servidor de autorización termine su trabajo.
2. El servidor de autorización autentica al usuario crea un código unico que tiene tiempo limitado de vida y además sólo puede ser

usado una única vez. Una vez autenticado el usario, el servidor de autorización redirige la petición indicada en el paso 1.

3. La aplicación cliente extrae el código unico que es retornado del servidor y luego invoca una peticion POST en el servidor de autorización solicitando un Access Token (esta solicitud debe de incluir el codigo obtenido en el paso 2).



Cómo obtener un token con el método Code Grant Type. (Richer & Sanso, 2017)

La presente tabla representa el conjunto de pasos a seguir cuando se desea obtener un token (OAuth2) utilizando el método Code Grant Type.

4.4 RESULTADOS DE LOS ARTEFACTOS DE OAUTH2

Como ya lo mencioné, no existe un método definido para la aplicación de OAuth; sin embargo, en la sección 3.5.4 sugiero un conjunto de pasos para la aplicación de OAuth2. La presente sección

representa los resultados obtenidos a partir de la técnica sugerida (elaboración propia donde se sugieren un conjunto de tareas).

4.4.1 DIAGRAMA DE COMPONENTES DEL SISTEMA

A continuación, se muestra el diagrama de componentes de todos los actores involucrados en el sistema (tarea 1 descrita en la sección 3.5.4).

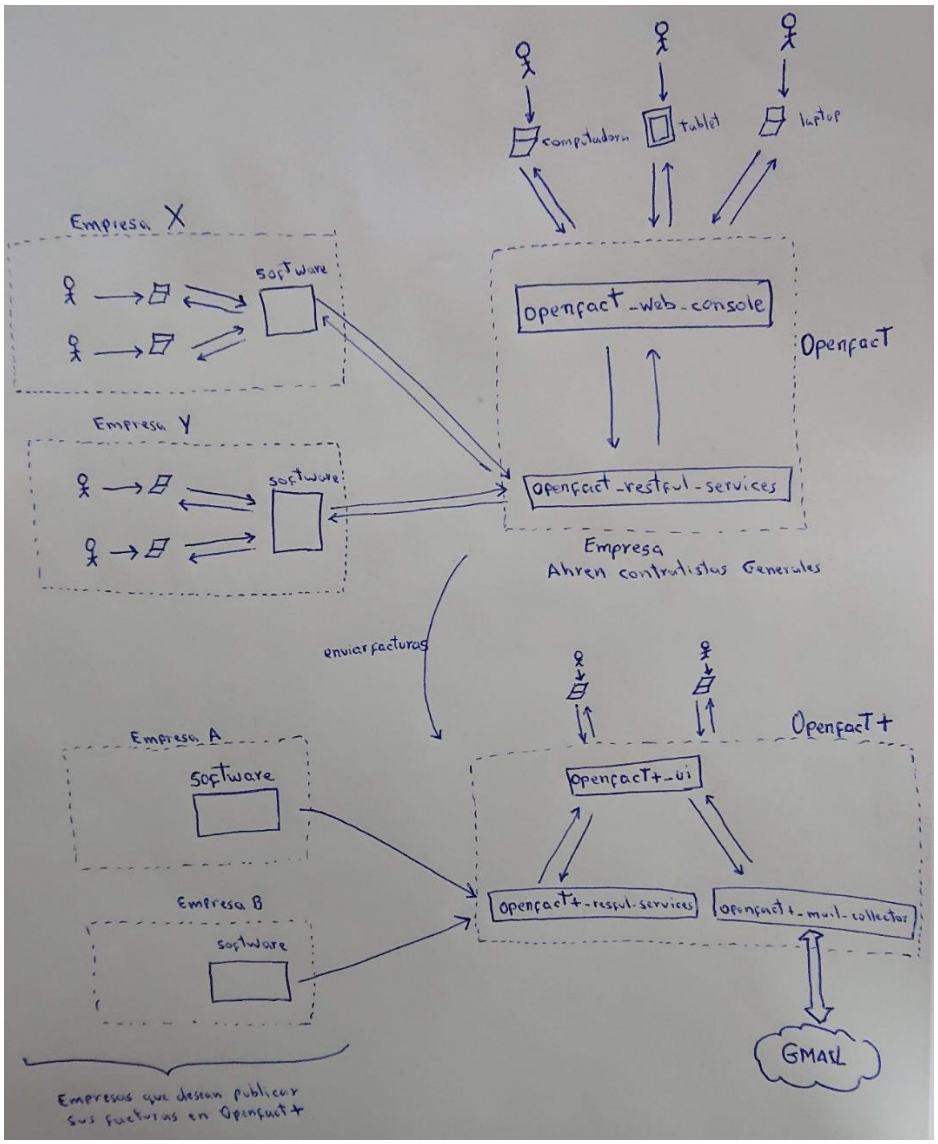


Figura 19. Diagrama de componentes del sistema de los actores involucrados en el sistema OPENFACT.

1. **OPENFACT**: Sistema de facturación electrónica. Está compuesto por dos componentes:

a. **Openfact-web-console**: Aplicación Front End de la aplicación web OPENFACT cuya función es la de presentar interfaces gráficas para la emisión de comprobantes de pago electrónico. Hecho utilizando el Framework Angular 2.
b. **Openfact-restful-services**: Aplicación Back End de la aplicación web OPENFACT cuya función es la de exponer Servicios Web Restful que después podrán ser utilizados por *openfact-web-console* u otro software. Hecho utilizando el lenguaje de programación Java.

2. **OPENFACT+**
   a. **Openfact+-ui**: Aplicación Front End de la aplicación web OPENFACT+ cuya función es la de presentar interfaces gráficas para acceder a las funcionalidades de "*openfact+-restful-services*". Hecho utilizando el Framework Angular 2.
   b. **Openfact+-restful-services:** Aplicación Back End de la aplicación OPENFACT+ cuya función es la de exponer Web Service Restful que después podrán ser utilizados por "*openfact+-ui*" u otros softwares.
   c. **Openfact+-mail-collector**: Aplicación Back End de la aplicación OPENFACT+ cuya función es la de recolectar comprobantes de pago de la bandeja de entrada de los correos electrónicos de los usuarios, para después enviar los comprobantes a "*openfact+-restful-services*".
3. **SOFTWARES QUE USAN OPENFACT**: Representa al grupo de otros softwares (propiedad de empresas terceras) que interactúan, creando transacciones con el componente OPENFACT.
4. **SOFTWARES QUE USAN OPENFACT+**: Representa al grupo de otros softwares (propiedad de empresas terceras) que interactúan, creando transacciones con el componente OPENFACT+.

4.4.2 LISTA DE REQUERIMIENTOS DE SEGURIDAD

La presente lista de requerimientos es el resultado de las especificaciones deseadas por el equipo de desarrollo y administradores de la empresa (tarea 2 descrita en la sección 3.5.4).

Tabla 8

Lista de requerimientos de seguridad.

NRO	REQUERIMIENTOS
1	Los usuarios que hagan uso de OPENFACT deben de ser creados manualmente por el administrador del sistema; sin embargo, el sistema OPENFACT+ debe de dar la posibilidad de que cada usuario se cree a sí mismo.
2	Los usuarios del OPENFACT no deben de ser los mismos que los usuarios del OPENFACT+; sin embargo, debe de existir la posibilidad de asociar una cuenta de OPENFACT con una cuenta de OPENFACT+
3	Los componentes openfact-web-console y openfact+-ui deben de estar aseguradas y permitir el ingreso sólo a usuarios autorizados.
4	El componente "*openfact-restful*-services" y "*openfact+-restful-services*" solo deben de aceptar peticiones HTTP válidas y autorizadas por un token.
5	Debe de existir la posibilidad de enviar peticiones seguras desde "*openfact-mail-collector*" a "*openfact+-restful-services*"
6	Los usuarios deben de ser capaces de administrar y monitorear los tokens que fueron creados en su nombre.
7	Se debe de poder trazar las actividades de seguridad respecto a los usuarios del sistema.
8	Configurar la aplicación web SAHREN para que éste pueda enviar boletas y facturas al OPENFACT.

La presente tabla fue diseñada por el equipo de desarrolladores del software OPENFACT.

4.4.3 SERVIDOR DE SEGURIDAD DESPLEGADO

El servidor de seguridad seleccionado para la implementación del protocolo OAuth2 es ***Keycloak***, que es un software de código abierto propiedad de la empresa RedHat. A continuación, se presenta algunas de las especificaciones del servidor **Keycloak** (tarea 3 descrita en la sección 3.5.4).

Tabla 9

Descripción del servidor de seguridad OAuth utilizado.

Version	3.4.3-Final
Licencia	Apache 2.0
Código fuente	https://github.com/keycloak/keycloak
Adaptadores para JavaScript	Si
Adaptadores para java	Si
Nivel de Personalización	SPI (Service Provider Interface) para personalización de funcionalidades
OAuth2	Si
OpenID	Si
Federated Autentication	Si
Adaptadores Javascript	Si
Adaptadores Java	Si
Adaptadores .Net	Si (soportado por la comunidad)
Adaptadores Node Js	Si
Implementación de JWT	Si
Consola web de administración	Si

La presente tabla muestra una breve explicación de las características del software Keycloak, que es el software utilizado para dar soporte OAuth2 durante la presente investigación.

4.4.3.1 DESPLEGAR KEYCLOAK LOCALMENTE

El servidor de seguridad Keycloak puede ser descargado desde la siguiente dirección URL: https://www.keycloak.org/downloads.html. Para poner iniciar el servidor se debe de ejecutar el archivo standalone.sh o standalone.bat dependiendo del sistema operativo que se pretende usar.

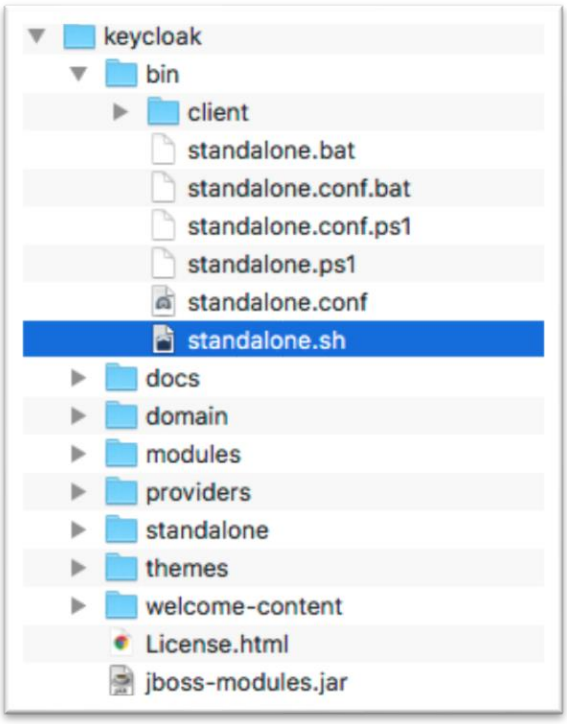


Figura 20. Estructura del servidor Keycloak.

Después de iniciar el servidor de seguridad, podemos acceder al servidor desde la URL: http://localhost:8080 donde podremos visualizar las siguientes imágenes:

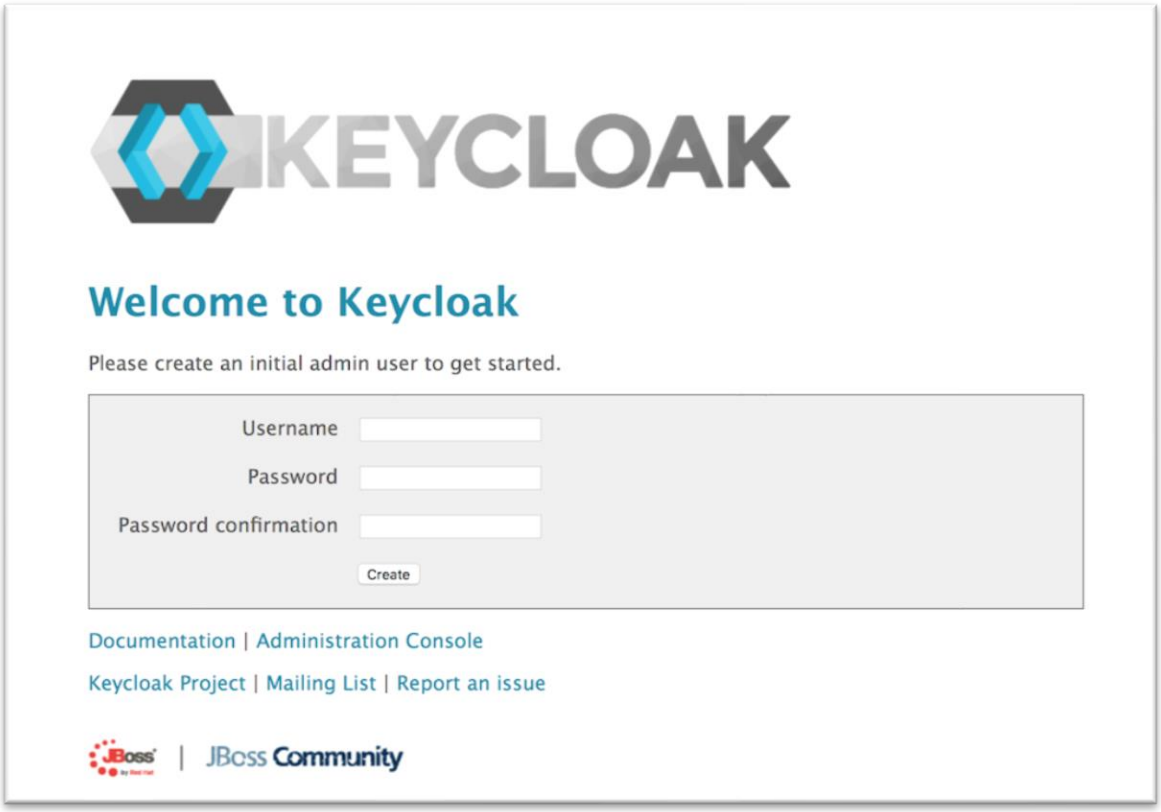


Figura 21. Creación del usuario administrador del servidor Keycloak.

Figura 22. Login del servidor de seguridad Keycloak.

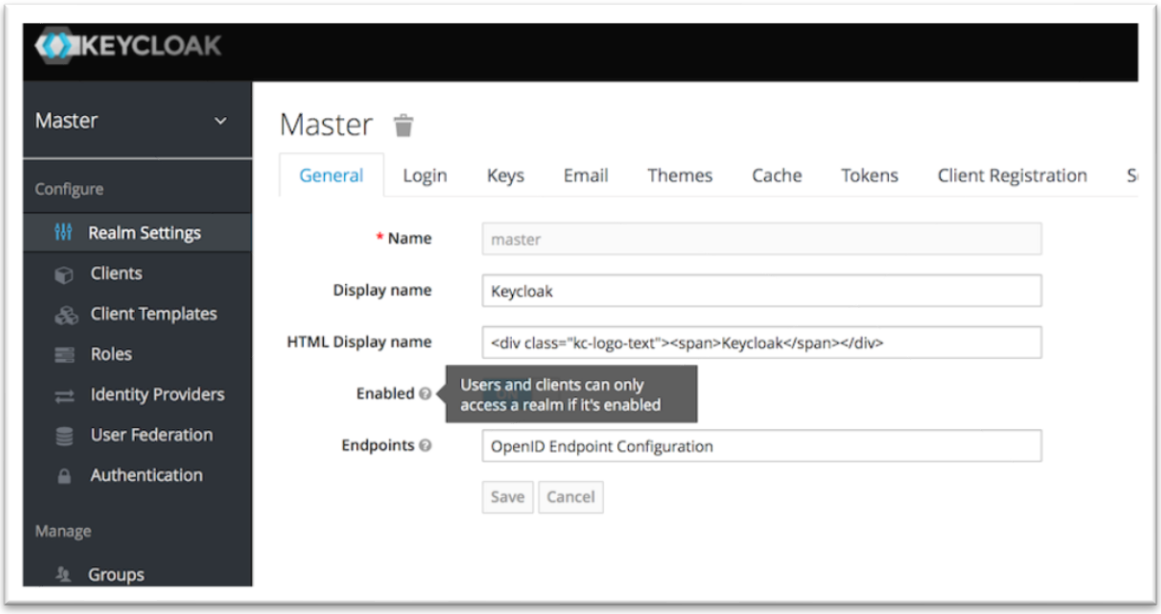


Figura 23. Página de administración del sistema de seguridad Keycloak.

4.4.3.2 DESPLEGAR KEYCLOAK EN OPENSHIFT

Si bien, el servidor Keycloak puede ser desplegado localmente se decidió utilizar la plataforma Pass Openshift para el despliegue del servidor de seguridad. Para desplegar el servidor Keycloak se debe de utilizar la plantilla que puede ser descargada en la URL: https://github.com/openfact/openfact-openshiftfiles/blob/master/templates/keycloak/server/keycloak-ephemeral.yaml

```yaml
kind: Template
apiVersion: v1
labels:
  template: keycloak-ephemeral
metadata:
  annotations:
    description: Application template for Keycloak 3.1.0.Final
    iconClass: icon-wildfly
    tags: keycloak,java,wildfly,jboss
    version: 1.0.0.Final
    openshift.io/display-name: Keycloak 3.1.0.Final
  name: keycloak
message: "A new Keycloak service has been created in your project.\n\n Keycloak Server (Master Realm)\n--------------------\n- Username: ${
objects:
  - kind: Route
    apiVersion: v1
    metadata:
      name: ${APPLICATION_NAME}
      labels:
        app: ${APPLICATION_NAME}-ephemeral
    spec:
      to:
        kind: Service
        name: ${APPLICATION_NAME}
  - kind: Service
    apiVersion: v1
    metadata:
      name: ${APPLICATION_NAME}
      labels:
```

Figura 24. Plantilla Openshift para el despliegue del servidor Keycloak

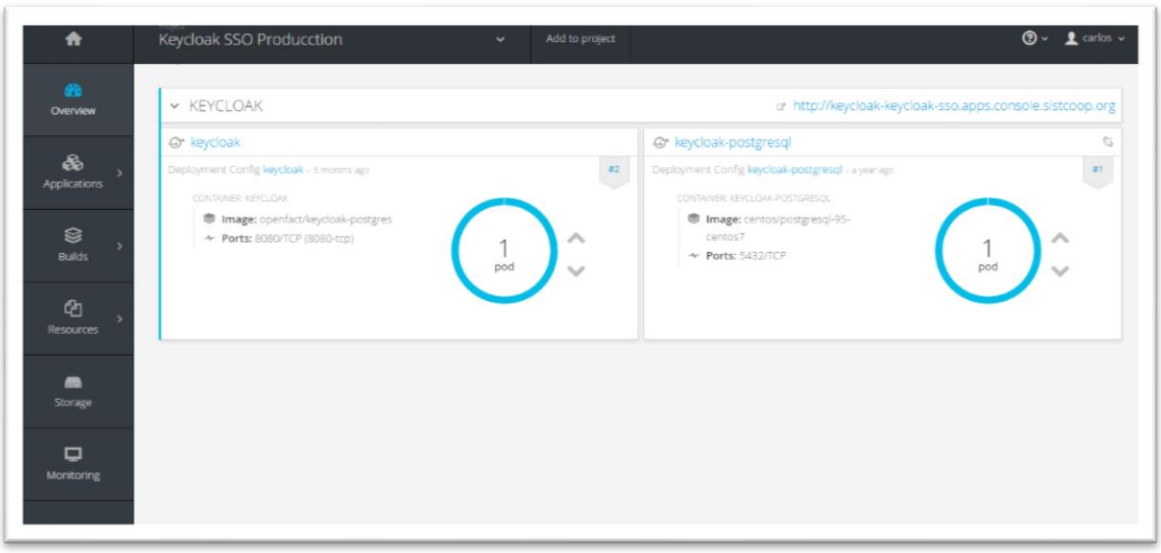


Figura 25. Servidor Keycloak Desplegado en plataforma Openshift.

4.4.4 DEFINIR REGLAS ADICIONALES DE SEGURIDAD

La siguiente lista de requerimientos adicionales fue sugerida por el equipo de desarrollo después de analizar las bondades y limitaciones del servidor de seguridad Keycloak (tarea 4 descrita en la sección 3.5.4)

Tabla 10

Reglas adicionales de seguridad.

NRO	REQUERIMIENTO
1	El sistema openfact+ debe de permitir el uso de alguna red social como forma de autenticación.
2	Se debe de permitir recuperar la contraseña cuando un usuario lo olvida.
3	Se debe de dar la posibilidad de tener autenticación de doble fase en el sistema.
4	Se debe de tener un registro de los eventos de actividades del usuario.
5	Se deben de configurar políticas de seguridad respecto a las contraseñas de los usuarios.

La presente tabla representa el conjunto de reglas adicionales de seguridad determinados por el equipo de desarrollo y administrativos involucrados en el software OPENFACT.

4.4.5 CÓDIGO FUENTE PARA ASEGURAR LOS COMPONENTES

Al tratarse de una actividad larga y definida en varias etapas, este paso será detallado en el siguiente ítem llamado: Resultados de la implementación.

4.5 RESULTADOS SCRUM

La presente sección representa los resultados obtenidos a partir de la aplicación de la técnica descrita en la sección 3.5.5 (técnica para aplicar Scrum) cuya descripción se encuentra en la sección ya mencionada.

Los siguientes son los resultados de la implementación aplicando el proceso scrum:

4.5.1 ASIGNACIÓN DE ROLES

Se asignaron los siguientes roles dentro del equipo involucrado con el desarrollo del software OPENFACT:

Tabla 11

Asignación de roles Scrum.

ROL	RESPONSABILIDADES	EQUIPO
Product Owner	Responsable de la generación del Product Backlog.	- Pabel J. Quispe Congacha.
Scrum Master	Responsable de Guiar al equipo con la metodología scrum.	- Carlos E. Feria Vila.
Desarrolladores	Equipo con distintas responsabilidades: Analistas, Desarrolladores, control de calidad, diseñadores, etc.	- Carlos Feria Vila. - Chano Huamán Landa. - Pavel E. Tueros Cárdenas. - Alex Palomino Pariona.

Tabla que describe la asignación de roles dentro del equipo de desarrollo.

4.5.2 PRODUCT BACKLOG

La lista de requerimientos para el proceso scrum es la combinación de requerimientos obtenidos en la sección 4.4.2 y 4.4.4:

Tabla 12

Backlog Scrum.

NRO	REQUERIMIENTOS
1	Los usuarios que hagan uso de OPENFACT deben de ser creados manualmente por el administrador del sistema; sin embargo, el sistema OPENFACT+ debe de dar la posibilidad de que cada usuario se cree a sí mismo.
2	Los usuarios del OPENFACT no deben de los mismos que los usuarios del OPENFACT+; sin embargo, debe de existir la posibilidad de asociar una cuenta de OPENFACT con una cuenta de OPENFACT+
3	Los componentes de Front End openfact-web-console y openfact+-ui deben de estar aseguradas y permitir el ingreso solo a usuarios autorizados.
4	El componente "*openfact-restful*-services" y "*openfact+-restful-services*" solo deben de aceptar peticiones HTTP válidas y autorizadas por un token.
5	Debe de existir la posibilidad de enviar peticiones seguras desde "*openfact-mail-collector*" a "*openfact+-restful-services*"
6	Los usuarios deben de ser capaces de administrar y monitorear los tokens que fueron creados en su nombre.
7	Se debe de poder trazar las actividades de seguridad respecto a los usuarios del sistema.
8	Configurar la aplicación web SAHREN para que éste pueda enviar boletas y facturas al OPENFACT.
9	El sistema openfact+ debe de permitir el uso de alguna red social como forma de autenticación.

10	Se debe de permitir recuperar la contraseña cuando un usuario lo olvida.
11	Se debe de dar la posibilidad de tener autenticación de doble fase en el sistema.
12	Se debe de tener un registro de los eventos de actividades del usuario.
13	Se deben de configurar políticas de seguridad respecto a las contraseñas de los usuarios.

La presente tabla describe el conjunto de requerimientos recopilados durante el proceso de la investigación.

4.5.3 PRIORIZACIÓN DE REQUERIMIENTOS

La priorización de requerimientos se realiza para identificar que tareas se realizarán antes que las otras. A continuación, se presenta el cuadro de priorización de requerimientos.

Tabla 13

Priorización de requerimientos.

NRO	REQUERIMIENTOS	PRIORIDAD
1	Los usuarios que hagan uso de OPENFACT deben de ser creados manualmente por el administrador del sistema; sin embargo, el sistema OPENFACT+ debe de dar la posibilidad de que cada usuario se cree a sí mismo.	Alta
2	Los usuarios del OPENFACT no deben de los mismos que los usuarios del OPENFACT+; sin embargo, debe de existir la posibilidad de asociar una cuenta de OPENFACT con una cuenta de OPENFACT+	Baja

3	Los componentes de Front End openfact-web-console y openfact+-ui deben de estar aseguradas y permitir el ingreso solo a usuarios autorizados.	Alta
4	El componente "*openfact-restful-services*" y "*openfact+-restful-services*" solo deben de aceptar peticiones HTTP válidas y autorizadas por un token.	Alta
5	Debe de existir la posibilidad de enviar peticiones seguras desde "*openfact-mail-collector*" a "*openfact+-restful-services*"	Media
6	Los usuarios deben de ser capaces de administrar y monitorear los tokens que fueron creados en su nombre.	Alta
8	Se debe de poder trazar las actividades de seguridad respecto a los usuarios del sistema.	Alta
9	Configurar la aplicación web SAHREN para que éste pueda enviar boletas y facturas al OPENFACT.	Alta
10	El sistema OPENFACT+ debe de permitir el uso de alguna red social como forma de autenticación.	Media
11	Se debe de permitir recuperar la contraseña cuando un usuario lo olvida.	Baja
12	Se debe de dar la posibilidad de tener autenticación de doble fase en el sistema.	Baja
13	Se debe de tener un registro de los eventos de actividades del usuario.	Alta

14	Se deben de configurar políticas de seguridad respecto a las contraseñas de los usuarios.	Media

Tabla que describe la asignación de prioridades a los requerimientos.

4.5.4 SCRUM DIARIO (DAILY MEETING)

Las reuniones del equipo de trabajo de realizaron todos los días, durante 10 minutos, al iniciar el día. Durante las reuniones se realizaron feedbacks que sirvieron para conocer el avance de cada miembro y las dificultades que afrontaron el día anterior a cada reunión.

4.5.5 REVISIÓN O DEMOSTRACIÓN DEL SPRINT

En esta etapa se muestra todo lo avanzado y logrado dentro del ultimo Sprint. Lo mostrado durante esta fase representa sólo las tareas terminadas y probadas al 100%. En la presente investigación se realizó sólo una iteración y todas las tareas fueron terminadas en una sola iteración.

La revisión y demostración del primer sprint incluye:

1. Configuraciones
    a. Configuración del "Identity Provider" para asociar cuentas de Google.
    b. Configuración de Keycloak para recuperar contraseñas.
    c. Configuracion de políticas de seguridad en el sistema Keycloak.
2. Demostración de código
    a. Muestra del código Javascript para asegurar los componentes openfact-web-console y openfact+-ui.
    b. Muestra del código Java para asegurar los componentes "*openfact-restful-*services" y "*openfact+-restful-services*"
    c. Código para enviar comprobantes electrónicos desde el SAHREN al OPENFACT usando un token.

d. Muestra del código de autenticación del sistema "*openfact-mail-collector*" a "*openfact+-restful-services*".

3. Demostración de uso del software
    a. Creación de un usuario en OPENFACT.
    b. Creación de un usuario OPENFACT+
    c. Asociación de una cuenta en OPENFACT y OPENFACT+
    d. Monitorear sesiones de cada usuario.
    e. Cómo configurar la autenticación de doble fase.
    f. Cómo realizar la trazabilidad de las actividades de los usuarios.
    g. Cómo administrar el proceso de autorización mediante la asignación de roles.
    h. Cómo crear facturas usando el componente openfact-web-console.
    i. Cómo crear facturas en el SAHREN y enviarlos al OPENFACT.

4.5.5.1 CONFIGURACION DEL IDENTITY PROVIDER PARA AUTENTICACIÓN CON GOOGLE

Se realizó la configuración del *Identity Provider* en el servidor Keycloak con el objetivo de permitir asociar una cuenta de usuario a su cuenta personal de Google+. Esta configuración requiere configurar una cuenta Google y crear un cliente web y después configurar el servidor Keycloak con los datos del cliente web.

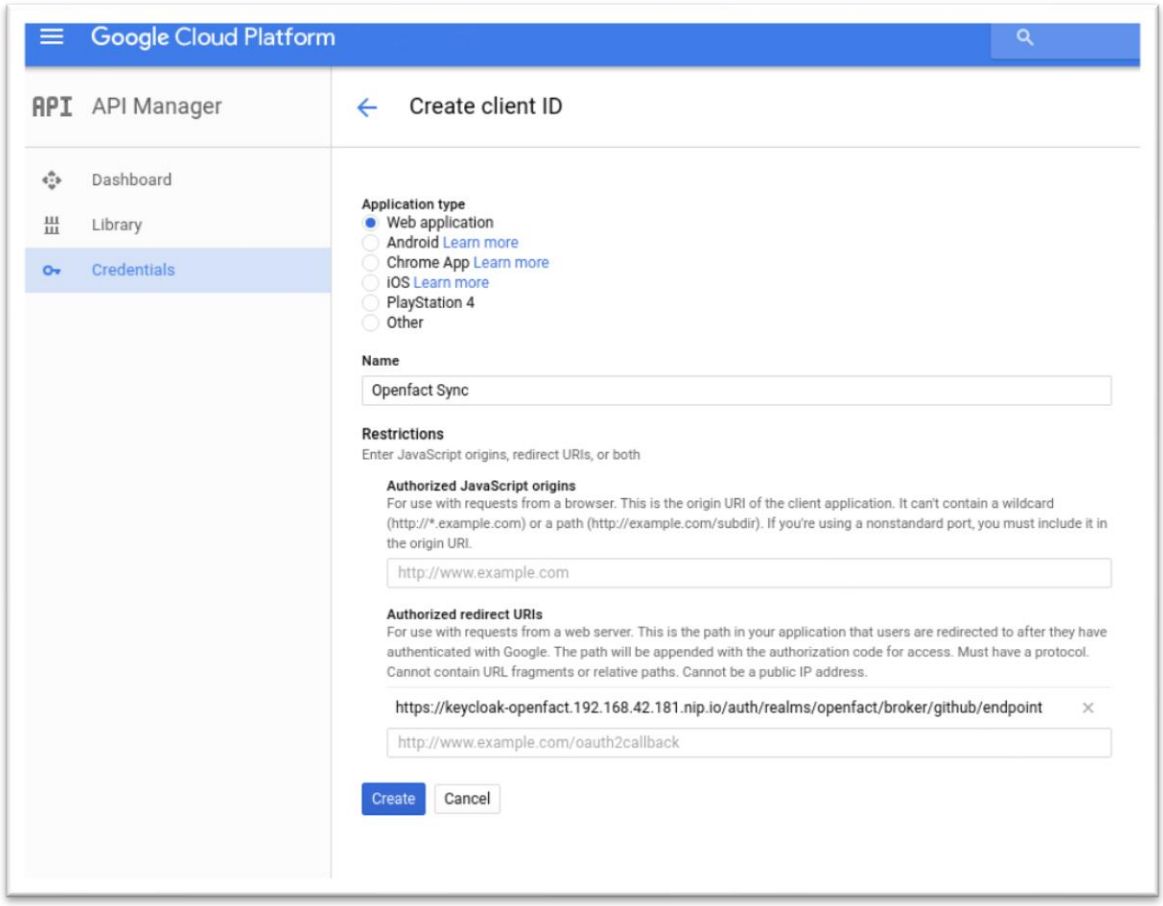


Figura 26. Creación de una aplicación web en la consola de administración de Google.

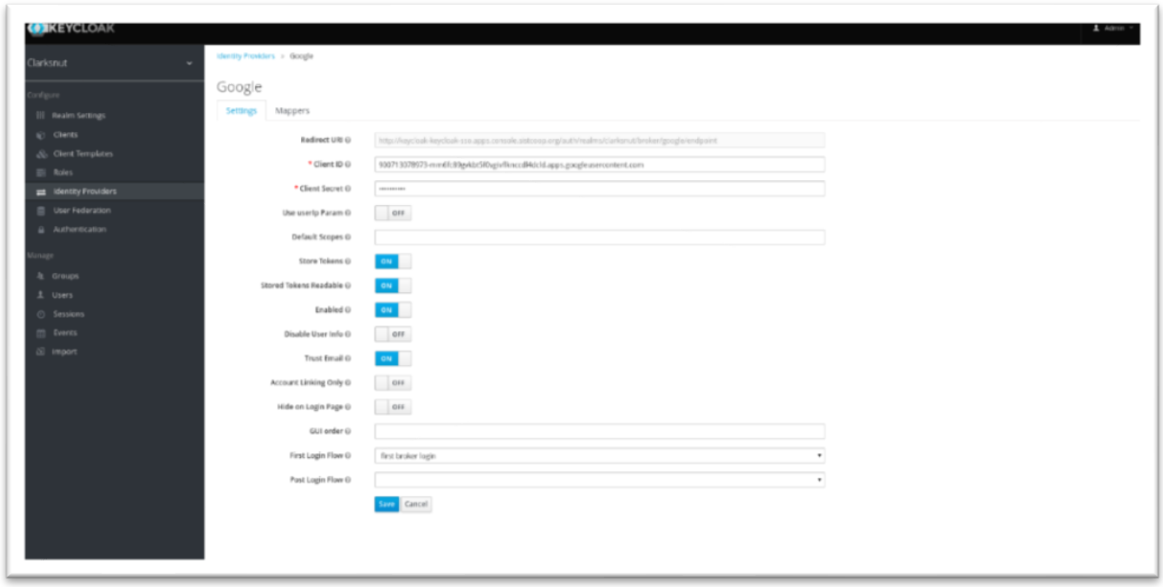


Figura 27. Configuración de Keycloak para asociar la API de autenticación de Google.

4.5.5.2 CONFIGURACIÓN DEL KEYCLOAK PARA RECUPERAR CONTRASEÑAS

Para permitir recuperar las contraseñas se hicieron las siguientes configuraciones en el servidor Keycloak:

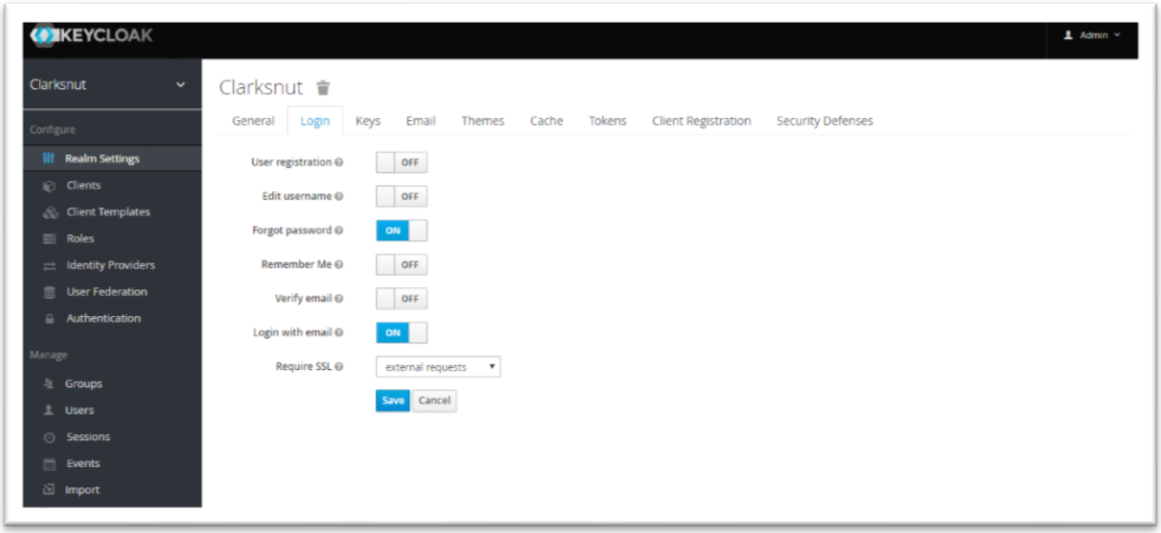


Figura 28. Habilitación de la recuperación de contraseñas.

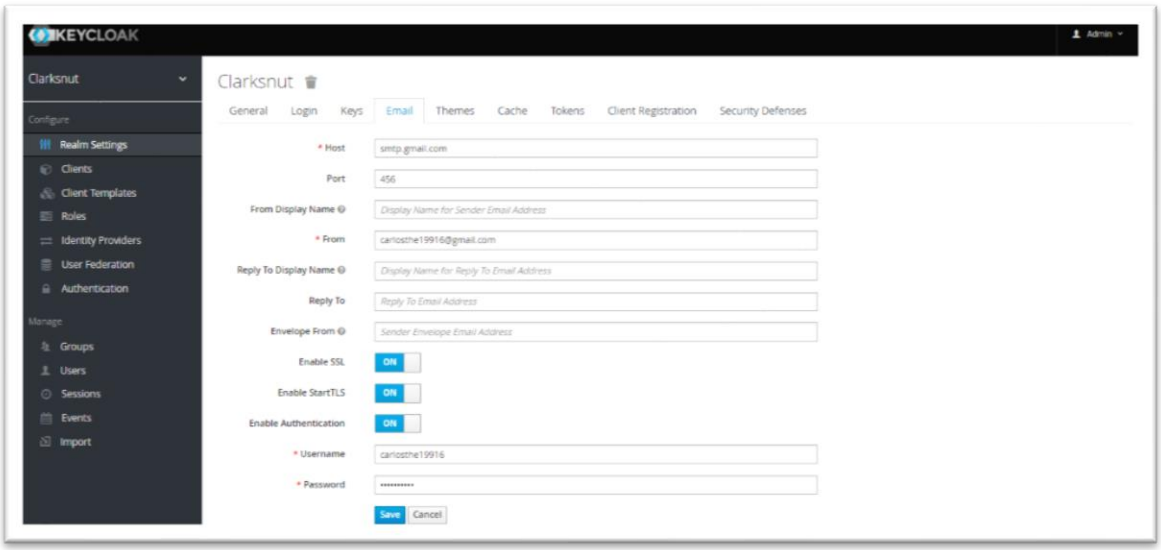


Figura 29. Configuración del servidor de correos para envió de recuperación de contraseñas.

4.5.5.3 CONFIGURACIÓN DE POLÍTICAS DE SEGURIDAD EN KEYCLOAK

Se realizó la configuración de protección contra ataques de Logeo con fuerza bruta y también se configuró las políticas de contraseña de los usuarios.

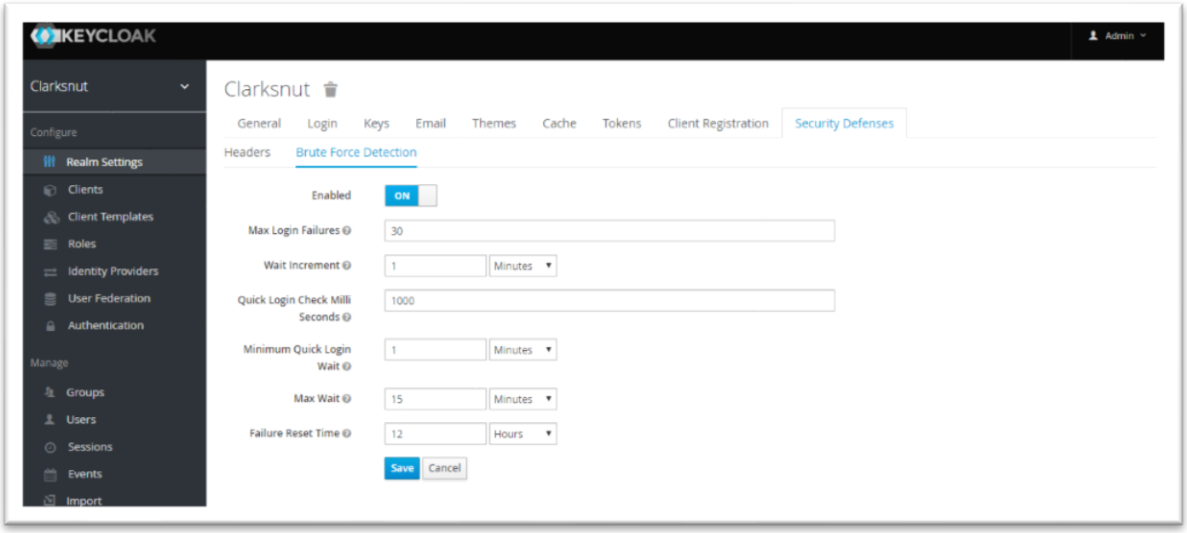


Figura 30. Configuración contra ataque de fuerza bruta.

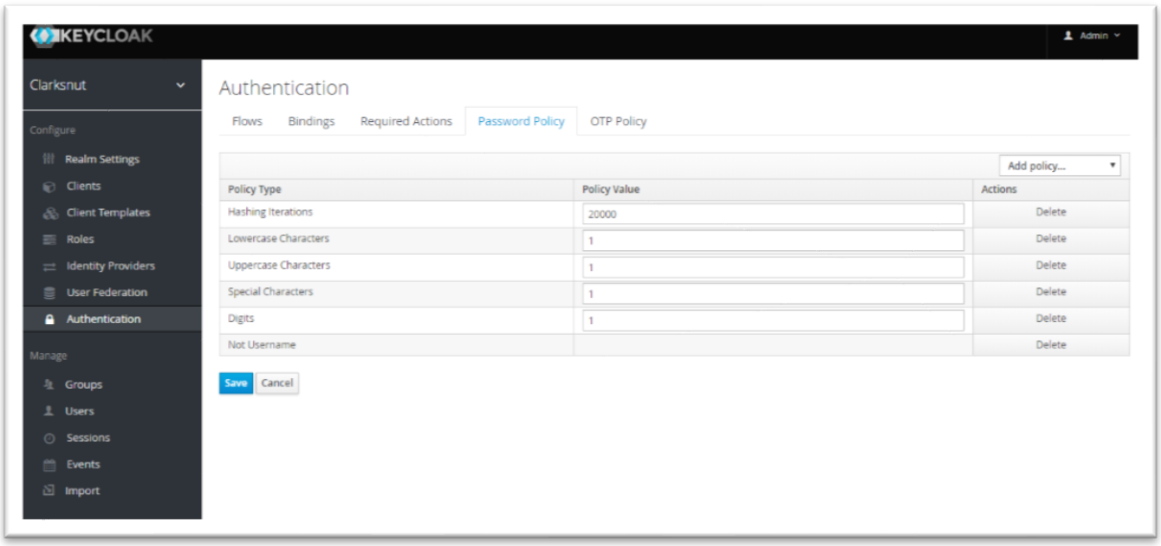
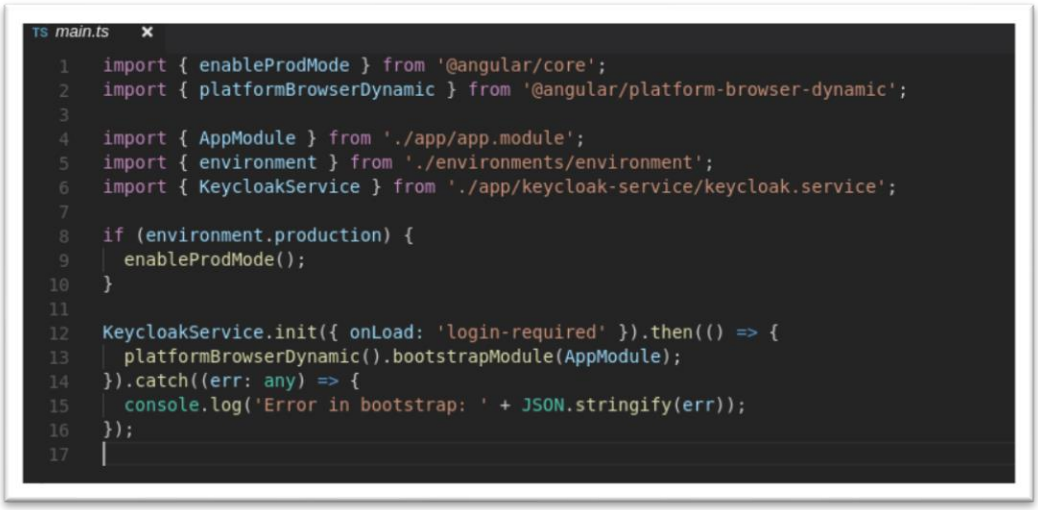


Figura 31. Configuración de políticas de seguridad de contraseñas.

4.5.5.4 MUESTRA DEL CÓDIGO JAVASCRIPT PARA ASEGURAR LOS COMPONENTES OPENFACT-WEB-CONSOLE Y OPENFACT+-UI

Al tratarse de aplicaciones Front End, se utilizó la librería keycloak.js para asegurar el acceso a esos componentes.

La librería keycloak.js puede ser descargada desde la siguiente url: https://github.com/keycloak/keycloak-js-bower



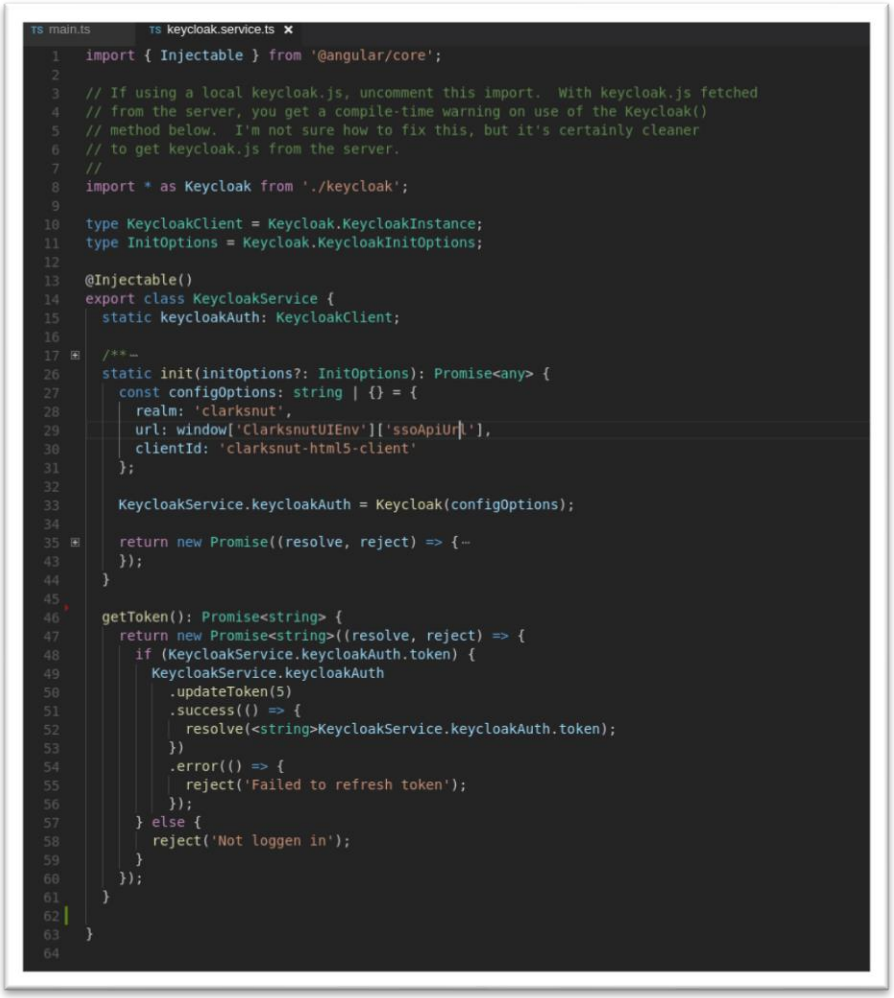
```ts
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';
import { KeycloakService } from './app/keycloak-service/keycloak.service';

if (environment.production) {
  enableProdMode();
}

KeycloakService.init({ onLoad: 'login-required' }).then(() => {
  platformBrowserDynamic().bootstrapModule(AppModule);
}).catch((err: any) => {
  console.log('Error in bootstrap: ' + JSON.stringify(err));
});
```

Figura 32. Código para autenticar usuarios.



```ts
import { Injectable } from '@angular/core';

// If using a local keycloak.js, uncomment this import.  With keycloak.js fetched
// from the server, you get a compile-time warning on use of the Keycloak()
// method below.  I'm not sure how to fix this, but it's certainly cleaner
// to get keycloak.js from the server.
//
import * as Keycloak from './keycloak';

type KeycloakClient = Keycloak.KeycloakInstance;
type InitOptions = Keycloak.KeycloakInitOptions;

@Injectable()
export class KeycloakService {
  static keycloakAuth: KeycloakClient;

  /**…
  static init(initOptions?: InitOptions): Promise<any> {
    const configOptions: string | {} = {
      realm: 'clarksnut',
      url: window['ClarksnutUIEnv']['ssoApiUrl'],
      clientId: 'clarksnut-html5-client'
    };

    KeycloakService.keycloakAuth = Keycloak(configOptions);

    return new Promise((resolve, reject) => {…
    });
  }

  getToken(): Promise<string> {
    return new Promise<string>((resolve, reject) => {
      if (KeycloakService.keycloakAuth.token) {
        KeycloakService.keycloakAuth
          .updateToken(5)
          .success(() => {
            resolve(<string>KeycloakService.keycloakAuth.token);
          })
          .error(() => {
            reject('Failed to refresh token');
          });
      } else {
        reject('Not loggen in');
      }
    });
  }

}
```

Figura 33. Servicio para obtención de tokens.

4.5.5.5 MUESTRA DEL CÓDIGO JAVA PARA ASEGURAR LOS COMPONENTES OPENFACT-RESTFUL-SERVICES Y OPENFACT+-RESTFUL-SERVICES

Las aplicaciones del Backend están hechas con el lenguaje de programación Java y desplegados en servidores de aplicación Wildfly, por lo que se determinó usar los adaptadores por defecto que brinda el servidor Keycloak.

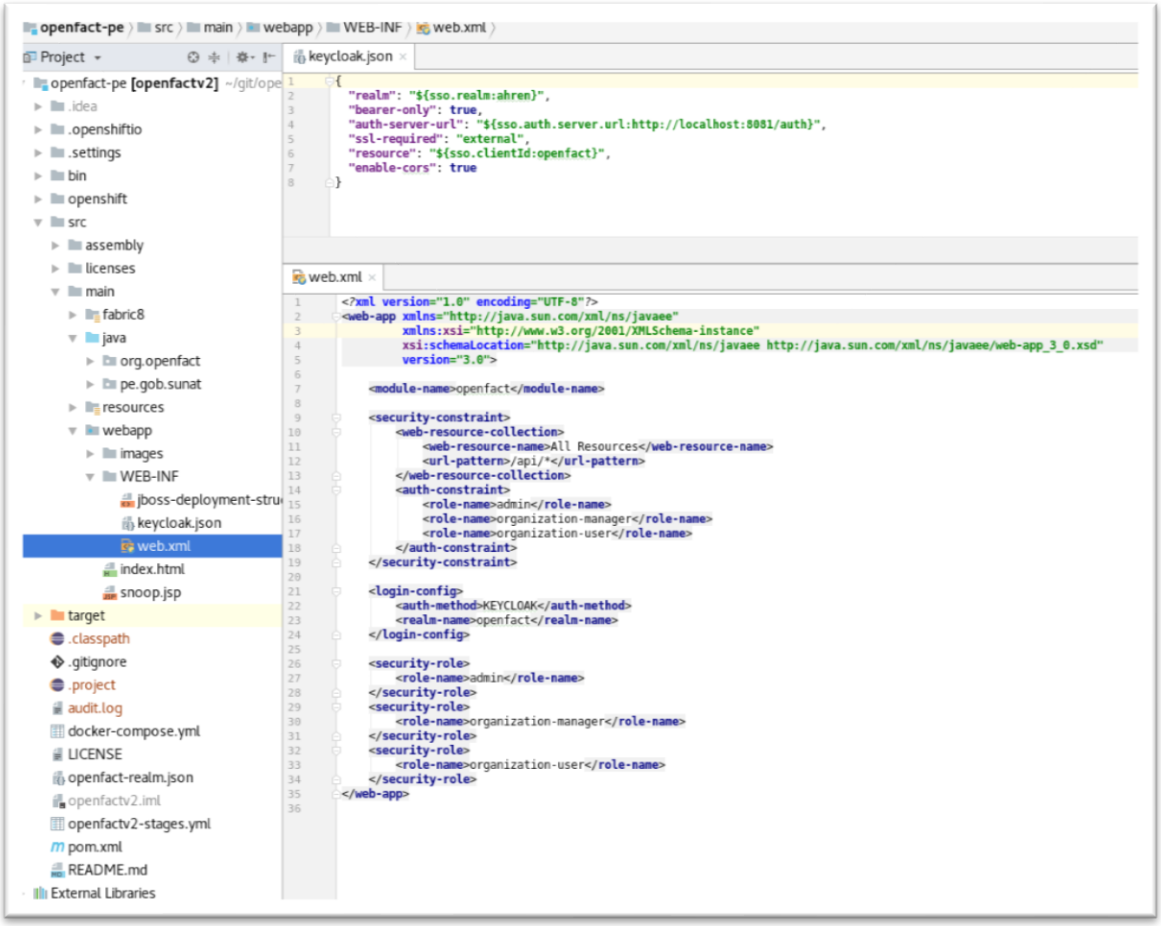


Figura 34. Configuración de la integración del Back End con Keycloak.

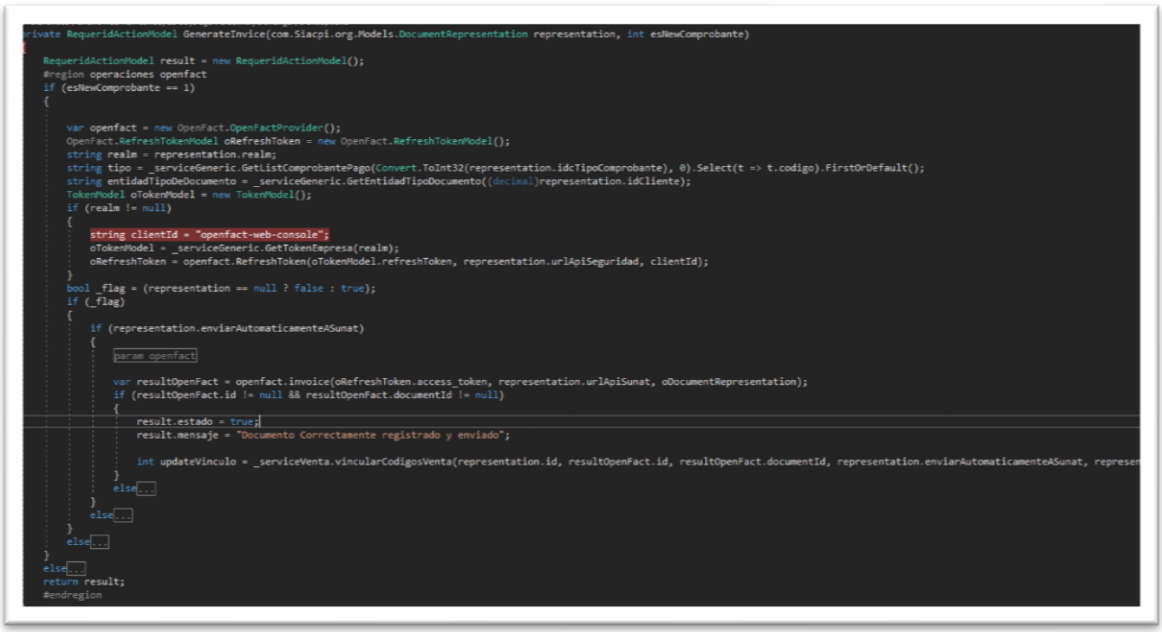
Además, fue necesario importar el adaptador del Keycloak para java:

```
<dependency>
    <groupId>org.wildfly.swarm</groupId>
    <artifactId>keycloak</artifactId>
</dependency>
```

Figura 35. Librería Maven para la importación del adaptador Keycloak para Java.

4.5.5.6 CÓDIGO PARA ENVIAR COMPROBANTES ELECTRÓNICOS DESDE EL SAHREN AL OPENFACT USANDO UN TOKEN.

El software SAHREN es un ERP perteneciente a la empresa Ahren Contratistas Generales S.A.C. y cuyo código fuente está integrado con los servicios web que expone la aplicación web OPENFACT. A continuación, se muestra el código que emite una factura y luego la envía al OPENFACT.



```
private RequeridActionModel GenerateInvice(com.Siacpi.org.Models.DocumentRepresentation representation, int esNewComprobante)
{
    RequeridActionModel result = new RequeridActionModel();
    #region operaciones openfact
    if (esNewComprobante == 1)
    {

        var openfact = new OpenFact.OpenFactProvider();
        OpenFact.RefreshTokenModel oRefreshToken = new OpenFact.RefreshTokenModel();
        string realm = representation.realm;
        string tipo = _serviceGeneric.GetListComprobantePago(Convert.ToInt32(representation.idcTipoComprobante), 0).Select(t => t.codigo).FirstOrDefault();
        string entidadTipoDeDocumento = _serviceGeneric.GetEntidadTipoDocumento((decimal)representation.idCliente);
        TokenModel oTokenModel = new TokenModel();
        if (realm != null)
        {
            string clientId = "openfact-web-console";
            oTokenModel = _serviceGeneric.GetTokenEmpresa(realm);
            oRefreshToken = openfact.RefreshToken(oTokenModel.refreshToken, representation.urlApiSeguridad, clientId);
        }
        bool _flag = (representation == null ? false : true);
        if (_flag)
        {
            if (representation.enviarAutomaticamenteASunat)
            {
                param openfact

                var resultOpenFact = openfact.invoice(oRefreshToken.access_token, representation.urlApiSunat, oDocumentRepresentation);
                if (resultOpenFact.id != null && resultOpenFact.documentId != null)
                {
                    result.estado = true;
                    result.mensaje = "Documento Correctamente registrado y enviado";

                    int updateVinculo = _serviceVenta.vincularCodigosVenta(representation.id, resultOpenFact.id, resultOpenFact.documentId, representation.enviarAutomaticamenteASunat, represen
                }
                else...
            }
            else...
        }
        else...
    }
    else...
    return result;
    #endregion
```

Figura 36. Código del ERP SAHREN para enviar facturas al OPENFACT.

4.5.5.7 MUESTRA DEL CÓDIGO DE AUTENTICACIÓN DEL SISTEMA OPENFACT-MAIL-COLLECTOR A OPENFACT+-RESTFUL-SERVICES

La comunicación entre estos dos componentes es de software a software, por lo que usó el método de obtención de tokens Client Credentials Grant Type.

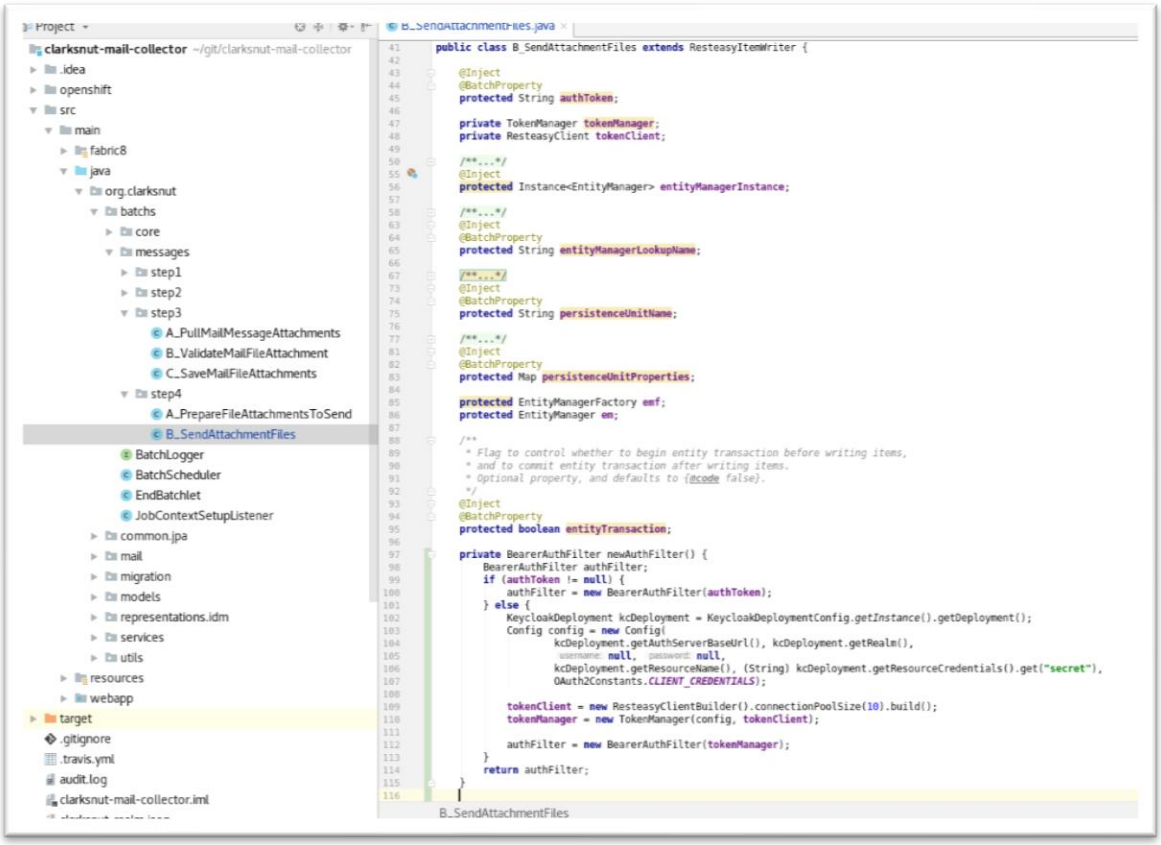


Figura 37. Código Java para crear peticiones seguras usando el método Client Credential Grant Type.

4.5.5.8 CREACIÓN DE UN USUARIO EN OPENFACT.

La creación de usuarios en OPENFACT se realiza de manera manual, como se muestra en la siguiente imagen:

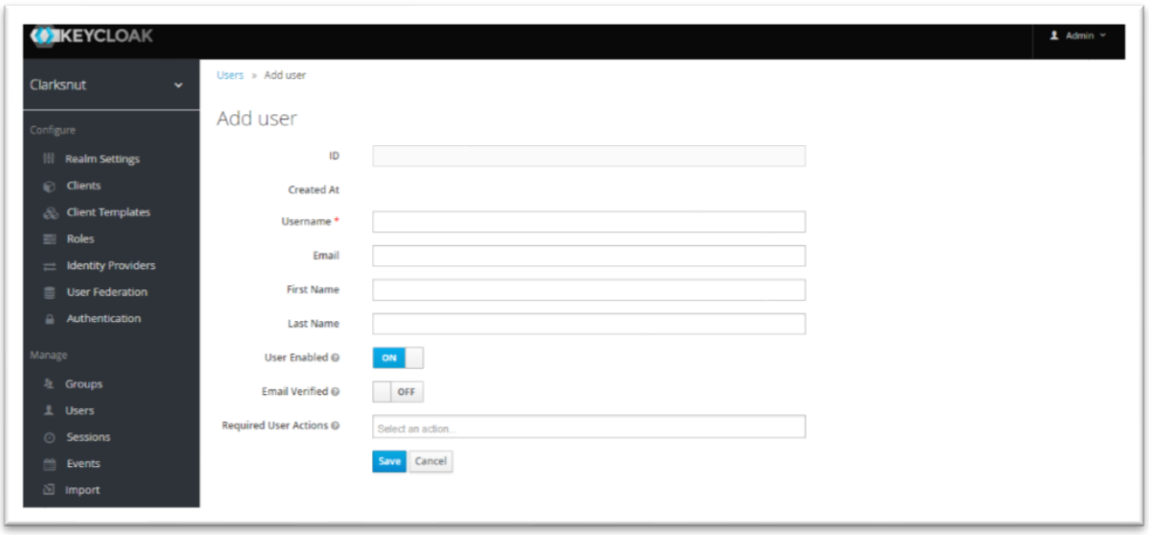


Figura 38. Cómo crear un usuario en OPENFACT.

4.5.5.9 CREACIÓN DE UN USUARIO OPENFACT+

La creación de usuarios en OPENFACT+ no requiere la creación manual de los usuarios ya que éste puede ingresar haciendo uso de sus cuentas en Google+.

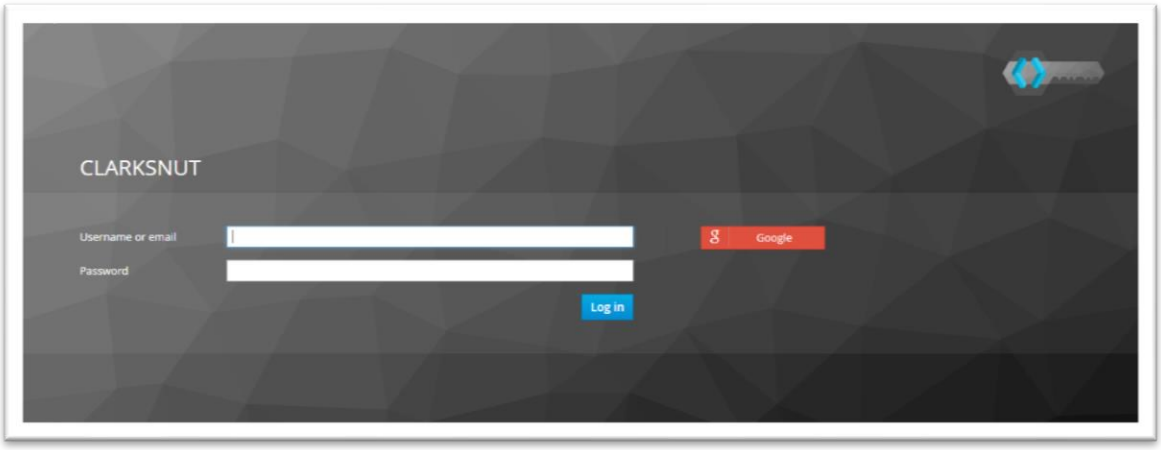


Figura 39. Cómo crear un usuario en OPENFACT+.

4.5.5.10 MONITOREAR SESIONES DE CADA USUARIO

Todos los usuarios pueden monitorear sus sesiones abiertas como se muestra en la siguiente imagen:

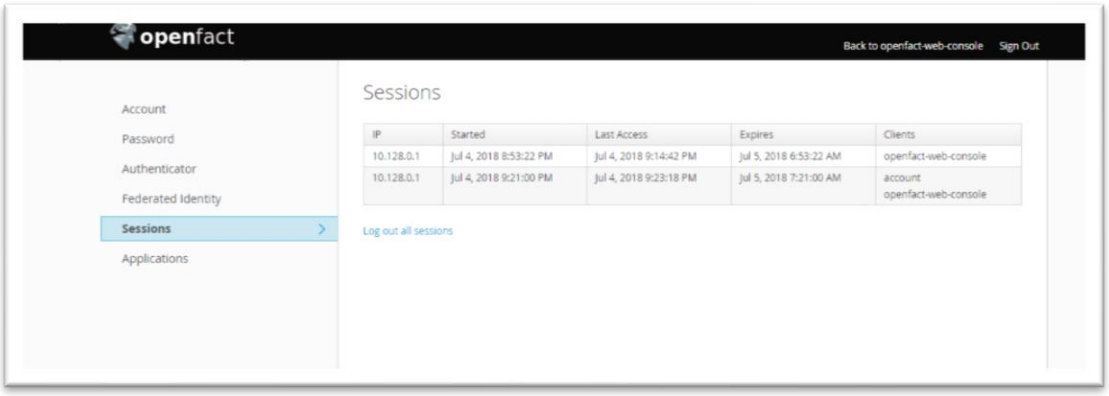


Figura 40. Monitorear las sesiones de cada usuario.

4.5.5.11 AUTENTICACIÓN DE DOBLE FASE

Para configurar la autenticación de doble fase cada usuario debe de acceder a la ventana de administración de cuentas y utilizar un lector de códigos QR como se muestra en la siguiente imagen:

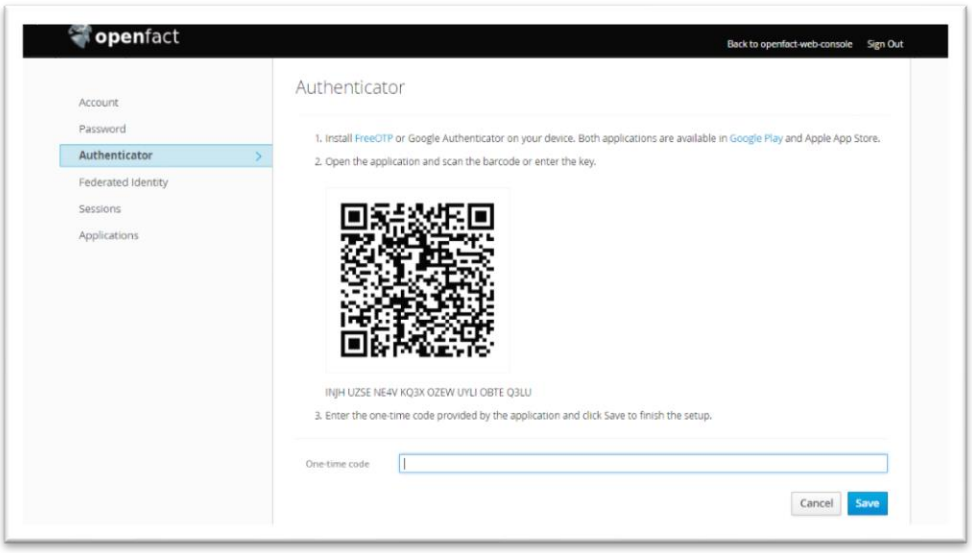


Figura 41. Implementación de la autenticación de doble fase.

4.5.5.12 TRAZABILIDAD DE LAS ACTIVIDADES DE LOS USUARIOS

El Servidor Keycloak permite monitorear las actividades de los usuarios en la pestaña de eventos del sistema.

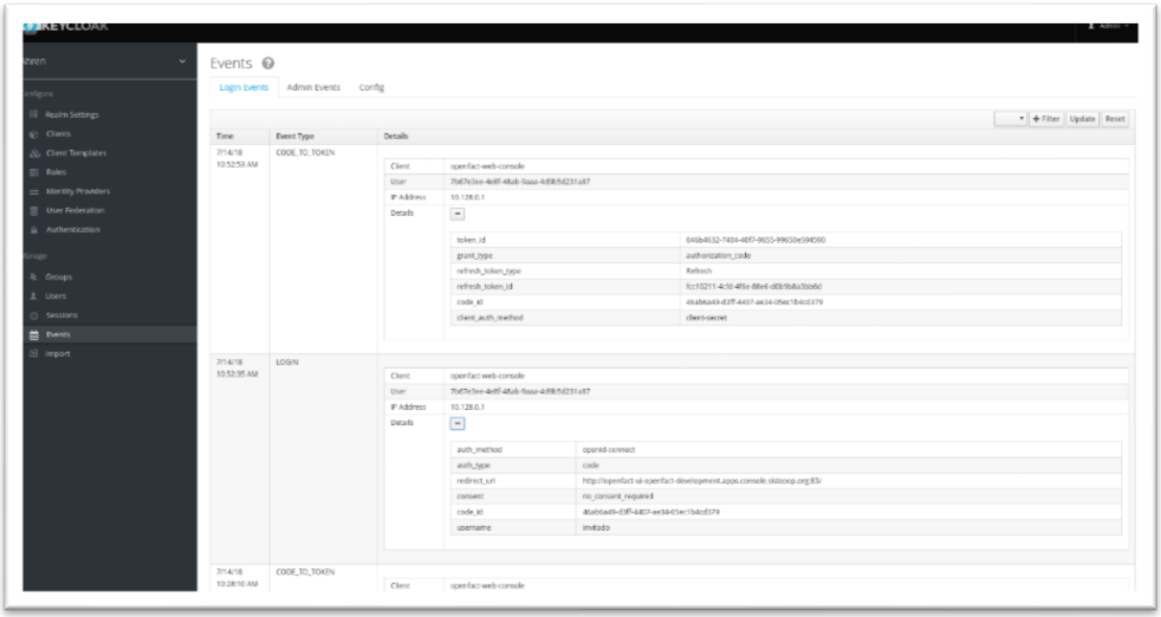


Figura 42. Trazar las actividades de los usuarios mediante Keycloak.

4.5.5.13 ADMINISTRAR EL PROCESO DE AUTORIZACIÓN MEDIANTE LA ASIGNACIÓN DE ROLES

El Servidor keycloak permite la administración de roles por cada usuario, como se muestra en la siguiente imagen:

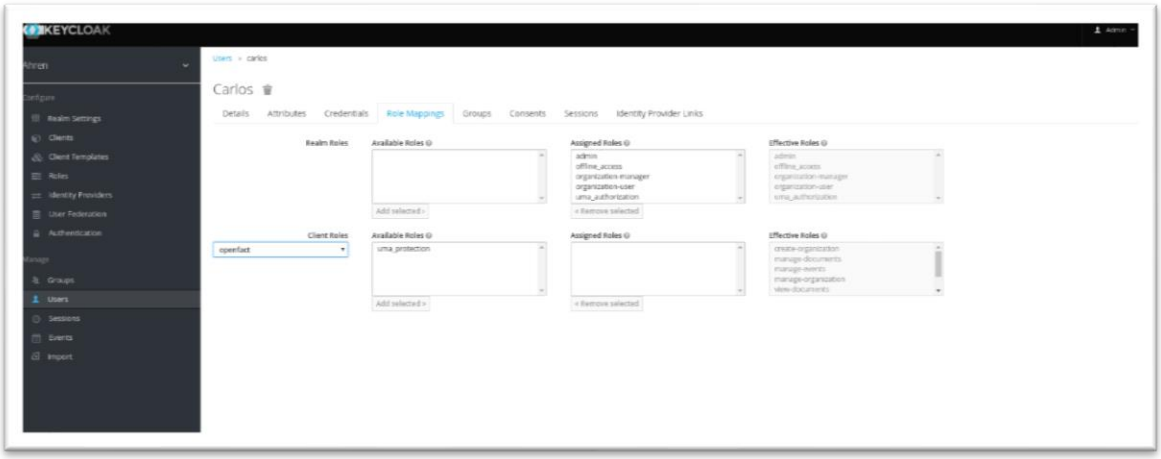


Figura 43. Asignación de roles por usuario mediante Keycloak.

4.5.5.14 CREAR FACTURAS USANDO EL COMPONENTE OPENFACT-WEB-CONSOLE

La creación de facturas desde el componente openfact-web-console utiliza tokens para el envío seguro de peticiones como se muestra en las siguientes imágenes:

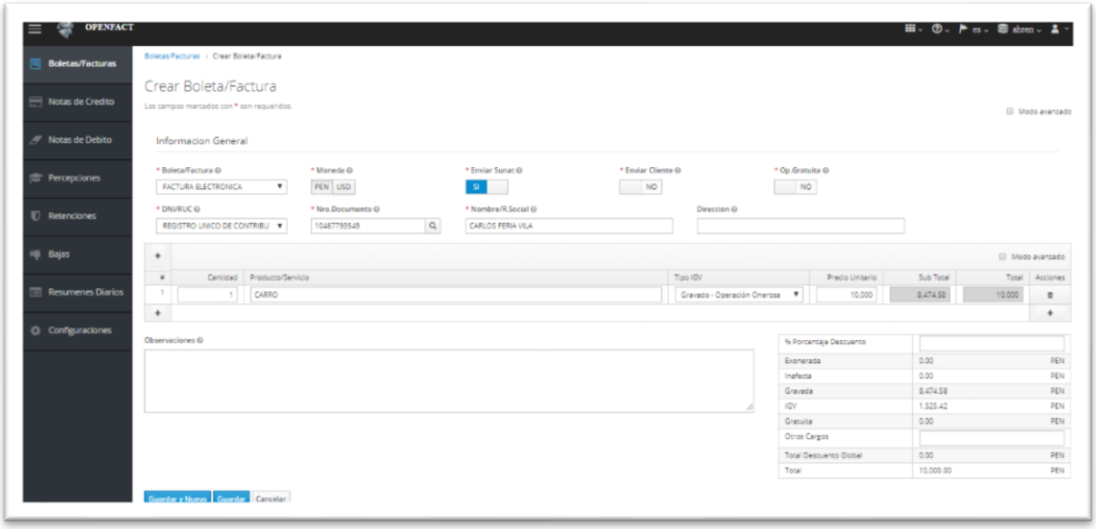


Figura 44. Creación de facturas mediante OPENFACT.

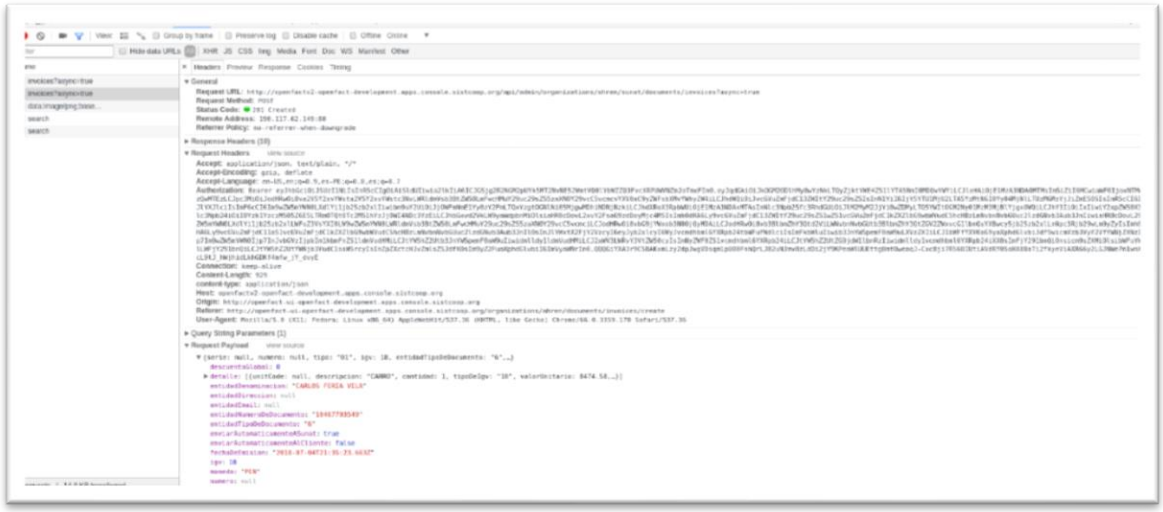


Figura 45. Petición HTTP generada para crear una factura electrónica. La petición incluye un token expuesto a través de la cabecera llamada "Authorization"

4.5.5.15 CREAR FACTURAS EN EL SAHREN Y ENVIARLOS AL OPENFACT

El Software SAHREN envía comprobantes de pago usando los servicios web de la aplicación web OPENFACT.

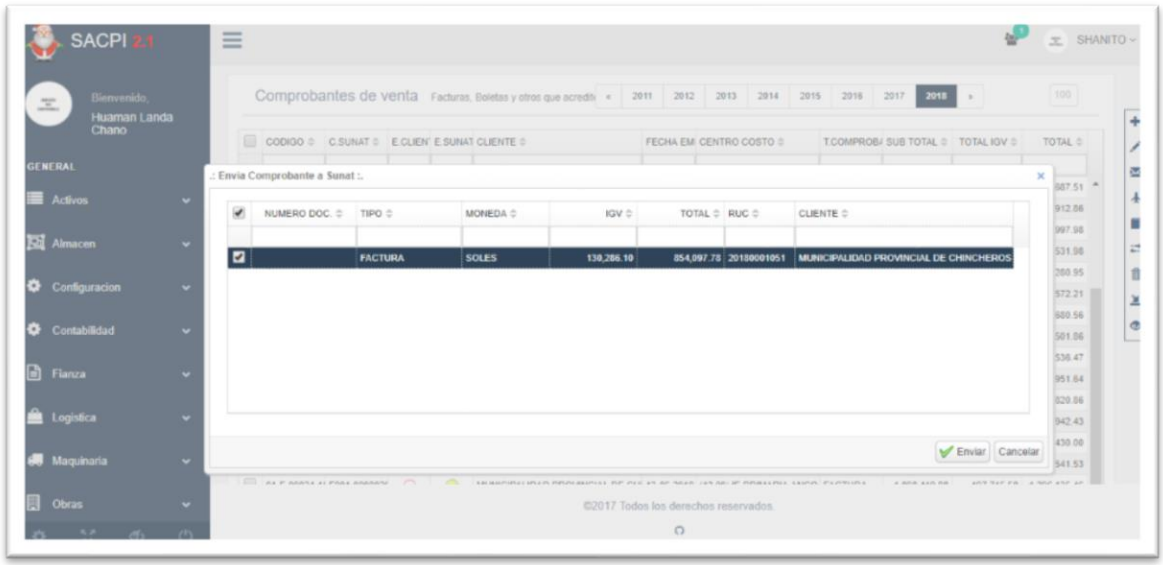


Figura 46. GUI del software SAHREN para envío de facturas al OPENFACT.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

a) Se logró implementar la capa de seguridad de la aplicación web OPENFACT mediante la aplicación de los protocolos OAuth2 y OpenID descrita en la sección 3.5.4 y cuya implementación está descrita en la sección 4.4; garantizando de esta manera un adecuado control de acceso a recursos y reducir el riesgo de manipulación no controlada de datos en todos los contextos en los que la aplicación web es utilizada.

b) Se logró identificar la identidad de cada uno de los usuarios (a través del uso del protocolo OpenID) que hacen uso de los componentes de la aplicación web OPENFACT, tal y como se muestra en la sección 4.5.5; de esta forma, la identificación de los usuarios colaboró con un mejor control de acceso a recursos.

c) Se logró proteger los recursos de la aplicación OPENFACT usando un control de accesos basado en roles (autorización), como se muestra en la sección 4.5.5.13; de esta manera, la autorización (basada en roles) colaboró con evitar el uso no autorizado de recursos.

d) Se logró trazar (registrar y monitorear) las actividades de los usuarios en el sistema de seguridad, como se muestra en la sección 4.5.5.10 y 4.5.5.12; de esta forma, la trazabilidad colabora con el proceso de control de acceso a recursos ya que permite a los usuarios y al administrador del sistema observar en tiempo real las sesiones existentes en el software.

5.2 RECOMENDACIONES

a) En caso de querer aplicar el protocolo OAuth2 en otros proyectos de software, se recomienda utilizar servidores de seguridad de

empresas terceras y no desarrollar su propia versión ya que el protocolo OAuth2 es muy popular y existen soluciones incluso gratuitas en la web.

b) Se recomienda utilizar el protocolo HTTPS en todas las aplicaciones desplegadas en la web ya que aún usando OAuth2 se puede ser víctima de ataques de robo de identidad. HTTPS garantiza el cifrado de los datos transmitidos entre el navegador y el servidor web.

BIBLIOGRAFÍA

AECOC. (1 de Noviembre de 2017). *Comité de Seguridad Alimentaria*. Obtenido de http://www.aecoc.es/servicios/implantacion/trazabilidad/#/login

Bihis, C. (2015). *Matering OAuth 2.0.* Estados Unidos: Packt Publications.

Boyd, R. (2012). *Getting Started with OAuth2.0.* Estados Unidos: O'Reilly.

Caballero, R. A. (2009). *Innovaciones en las guías metodológicas para los planes y tesis de maestría y doctorado.* Perú: Instituto Metodológico Alen Caro E.I.R.L.

Christoph, B. (2003). *B2B Integration: Concepts and Architecture.* Estados Unidos: Springer.

Consortium, T. I. (1999). *Hypertext Transfer Protocol HTTP 1.0 Specifications.* Estados Unidos.

Fernandez, C., Baptista, P., & Hernandez, R. (2014). *Metodología de la investigación.* México DF, México: McGRAW-HILL/INTERAMERICANA EDITORES, S.A. DE C.V.

Finnigan, K. (2017). *Enterprise Java Microservices.* Estados Unidos: Manning Publications.

Gourley, D. (2002). *HTTP: The definitive Guide: The Definitive Guide.* Estados Unidos: O'Reilly.

Hall, S. (2017). *Innovative B2B Marketing: New Models, Processes and Theory.* Estados Unidos: KoganPage.

Keycloak. (05 de Febrero de 2018). *Securing Applications and Services*. Obtenido de https://www.keycloak.org/documentation.html

Kim, D., & Solomon M. (2014). *Fundamentals of Information Systems Security (2th edition).* México: Universidad Nacional Autónoma de México.

Lozano, C. (2013). *SIstema Centralizado de autenticacion de usuarios para la subdirección de seguridad de la informacion UNAM-CERT.* México: Universidad Nacional Autónoma de México.

Oracle. (02 de Noviembre de 2017). *Java Enterprise Edition Documentation*. Obtenido de https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

Patni, S. (2017). *Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS.* Estados Unidos: Apress.

Pinzon, R. (2010). *Trazabilidad.* Colombia: Universidad Industrial de Santander.

RAE. (1 de Noviembre de 2017). *Real Academia de a Lengua Española*. Obtenido de http://dle.rae.es/?id=4UXxkoH

Rafeeq, U. (2008). *Get Ready for OpenID.* Estados Unidos: Confirmix Books.

RFC6749. (2012). *The OAuth 2.0 Authorization Framework.* Obtenido de https://tools.ietf.org/html/rfc6749

Richer, J., & Sanso, A. (2017). *OAuth 2 in Action.* Estados Unidos: Manning Publications Co.

Sankar, K., & Sundaralingam, S. (2004). *Cisco Wireless LAN Security.* Estados Unidos: Ciscopress.

SCRUM. (01 de Junio de 2018). *scrum.org*. Obtenido de https://www.scrum.org/resources

Sinn, R. (2008). *Software Security Technologies.* Estados Unidos: Centage Learning.

Snell, J., Tidwell, D., & Kulchenko, P. (2002). *Programming Web Services with SOAP: Building Distributed Applications.* Estados Unidos: O'Reilly & Associates.

Spasovski. (2013). *OAuth 2.0 Identity and Access Management Patterns.* Estados Unidos: Packt Publications.

Sutherland, J., & Schwaber, K. (2017). *The Scrum Guide*. Estados Unidos.

Tabor, R. (2002). *Microsoft .NET XML Web Services.* Estados Unidos: Sams.

Varona, P. (2014). *Implementación de un proveedor de autorizaciones OAuth 2.0 con Scala.* Madrid: Universidad Autónoma de Madrid.

Whitman, M., & Mattord, H. (2016). *Management of information Security*. Estados Unidos: Universidad Estatal de Kennesaw.

ANEXOS

ANEXO A

FICHA DE OBSERVACIÓN	
FECHA:	…/…/.....
SOFTWARE UTILIZADO:	.....
OBJETIVO DE LA FICHA:	….....
PETICION HTTP (REQUEST)	**Cabeceras:** **Cuerpo:**
RESULTADO HTTP (RESPONSE)	
DESCRIPCIÓN	**OBSERVACIONES Y/O CONCLUSIONES**

ANEXO B

REGISTRO DOCUMENTAL	
FECHA:	…/…/.....
FUENTE:	.....
OBJETIVO DE LA FICHA:	….....
DESCRIPCIÓN	**OBSERVACIONES Y/O CONCLUSIONES**