
**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE INGENIERÍA**


# Sistema de reportes con información consumida por una API de un DLP

**INFORME DE ACTIVIDADES PROFESIONALES**

Que para obtener el título de

**Ingeniero en Computación**

**P R E S E N T A**


Esteban Palacio Nieto


**ASESORA DE INFORME**

M.I. Tanya Itzel Arteaga Ricci


**Ciudad Universitaria, Cd. Mx., 2023**

Índice

**Resumen global.**

El desarrollo que aquí presento cubre la necesidad de descargar información concerniente a métricas, telemetría, y configuraciones que se encuentra contenida en documentos de procesamiento automatizado (JSON) que son solicitados a la API de un servidor que administra un fabricante que desarrolla un producto DLP.

Posteriormente este desarrollo procesa estos datos para generar conclusiones significativas agrupando de forma lógica los datos, para proceder a la generación de un reporte de hoja de cálculo en Excel que permite al operador de la solución de resguardar reportes concernientes al funcionamiento del producto DLP para conformar una memoria técnica.

Adicionalmente tiene la funcionalidad de configurar opciones y características en la plataforma, para configurar parámetros de funcionamiento de la solución DLP, como: creación de grupos y aprobación de solicitudes de extracción de información en forma no cifrada por parte del usuario final. De esta forma se consigue el propósito de personalizar el funcionamiento de la misma, posibilitando a los clientes poder ajustar a su medida la utilización de la herramienta DLP.

Posteriormente el usuario administrativo que opera el desarrollo tiene la libertad de editar el reporte usando una herramienta ofimática para informar estas métricas a las áreas gerenciales.

Este proyecto responde a las necesidades de la industria de generar reportes que ayuden a la mejor toma de decisiones y que documenten el uso de sus herramientas.

**Introducción.**

En tiempos recientes es notable el aumento en la utilización del modelo SaaS (software as a service) como forma de resolver la problemática de brindar servicios que cubran diversos casos de uso en la industria. Este modelo a diferencia del modelo tradicional, conocido como "bajo demanda" donde el software y los servicios funcionan dentro de servidores locales en la infraestructura de cada organización, permite una mayor flexibilidad, comodidad, y mejoras en el flujo de trabajo.

Algunos ejemplos de aplicaciones que cimientan su funcionalidad en la nube son: plataformas de gestión de cuentas para la venta (Salesforce, Zoho, etc.), plataformas de seguridad Informática (antivirus, DLP, filtro de contenido, filtrado de tráfico, etc.), plataformas de comunicación (email, mensajería interna, plataformas de colaboración).

Aunque existen muchos beneficios para las organizaciones que optan por adoptar un sistema basado en la funcionalidad SaaS, existen también algunas desventajas, puesto que este modelo provoca una dependencia de los servicios de un tercero, teniendo que ceder control en el manejo de datos en una transferencia del riesgo hacia el proveedor del servicio. Esta característica puede ser disuasiva para aquellos que son cautelosos en el manejo la información, pero puede ser atractiva para aquellos que quieren simplificar su operación.

Siguiendo el modelo SaaS se tienen sistemas que permiten la gestión de la seguridad de los datos por medio de la gestión de permisos y el cifrado de estos para evitar la pérdida de información. Tales sistemas conocidos como DLP (Data Loss Prevention) son descritos por Gartner como:

*"Aquellas aplicaciones que brindan visibilidad del uso de datos en una organización para un amplio conjunto de casos de uso y la aplicación dinámica de políticas basadas en el contenido y el contexto para datos en reposo y en uso".*

Estas herramientas buscan abordar las amenazas relacionadas con los datos, incluidos los riesgos de pérdida de datos accidental o inadvertida, y la exposición de datos confidenciales mediante la

supervisión, el filtrado, el bloqueo y otras funciones relacionadas al resguardo de los datos y al control de acceso por parte de los usuarios autorizados.

Trabajé desempeñando actividades relacionadas a mi carrera Ingeniería en Computación en una empresa con oficinas en Ciudad de México y Mérida Yucatán. Esta empresa se dedica a brindar soluciones de seguridad informática y de la información para organizaciones de distintos rubros con el objetivo de aligerar la carga de trabajo del personal técnico y proveer a las organizaciones de las herramientas necesarias para minimizar incidencias que afecten la continuidad de operaciones. Para tal fin, la compañía para la que laboré comercializa productos (software) de diversos fabricantes, aportando valor añadido a los mismos para subsanar las necesidades particulares de cada cliente. En esta empresa comencé a laborar en agosto de 2018.

El presente trabajo se enfoca en describir los esfuerzos realizados en uno de los proyectos de la empresa con la cual colaboré, que sirve como interfaz de control de una herramienta tipo DLP de un fabricante externo.

El objetivo de este trabajo es de describir el proceso de diseño y desarrollo de un sistema de reportes y configuraciones que mediante la integración con una herramienta DLP que se encuentra en el catálogo de productos que se comercializan por parte de la compañía.

Este desarrollo que aquí presento permite a los administradores la obtención de información editable que contiene además una representación gráfica del análisis de los datos generados en el reporte. De esta manera se reporta información relacionada al registro histórico del uso de la herramienta DLP, lo cual aporta como valor la capacidad para el operador de presentar resultados a las áreas gerenciales.

Gracias al desarrollo que hice, se pueden obtener graficas de las métricas de variables de operación del uso de la herramienta DLP, así como facilitar las funciones de configuración, brindando la posibilidad de automatización en desarrollos particulares de cada cliente.

# CAPÍTULO 1. Descripción de la empresa.

La empresa para la cual presto servicios, es considerada una consultoría de tecnología enfocada brindar soluciones de seguridad informática a diversos clientes. Su modelo de negocio se basa en tener fuertes relaciones con actores clave en la industria, como lo son canales de venta, clientes y proveedores de soluciones.

En este sentido ofrece soluciones de terceros a clientes mediante la utilización de los canales de venta con quienes se tienen acuerdos comerciales.

La propuesta que la empresa ofrece a sus clientes consiste en añadir valor al producto por medio de soporte, adecuación, la oferta de servicios administrados de operación de herramientas y tercerización de actividades de ciberseguridad, facilitando así las operaciones de ciberseguridad de cada cliente.

La compañía fue fundada en junio de 2018 con la finalidad de aportar valor agregado a diversas soluciones que se comercializan en la región del norte de América Latina y el Caribe. Entre las diversas soluciones que están en el catálogo de productos se encuentran aquellas enfocadas en la gestión de la ciberseguridad en el entorno de las organizaciones, filtrado de contenido y protección contra perdida de información.

Tiene varias sedes enfocadas en distintos aspectos de su operación. La oficina principal para la región del Norte de América Latina está en Ciudad de México.

## 1.1 Descripción organizacional.

La compañía pese a ser reciente, carga consigo una experiencia de emprendimientos previos que finalmente convergieron en acuerdos comerciales que dieron justificación a la creación de la nueva organización. En un principio se diseñó para tener las áreas funcionales de la Figura 1.

De las áreas que componen a la compañía se tienen:

- Área comercial, que tiene responsabilidades enfocadas al establecimiento de las relaciones comerciales, la planeación comercial, seguimiento a clientes y estrategia comercial y competitiva.
- Área técnica, que tiene responsabilidades enfocadas a diseño, desarrollo, implementación, atención de requerimientos y casos de uso y puesta en marcha de proyectos.
- Área legal, que se encarga de responsabilidades enfocadas al ámbito legal y se encuentra tercerizada.
- Área de RR.HH. que gestiona las relaciones obrero-patronales.

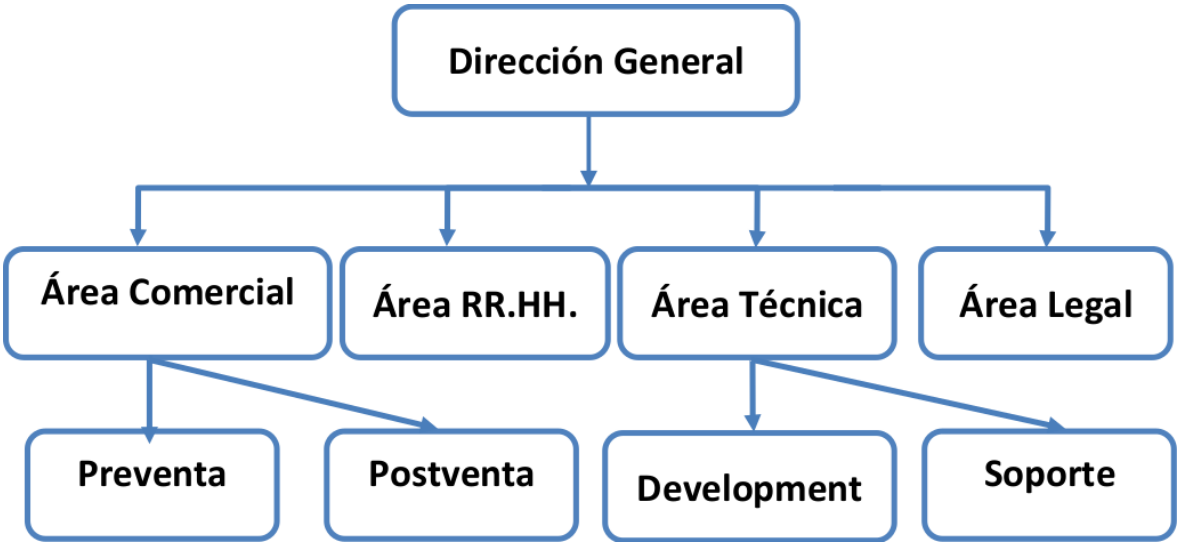


Figura 1. Mapa Organizacional.

**1.2 Descripción del puesto de trabajo.**

Mi entrada a la compañía corresponde a las necesidades de la misma para aportar valor a sus clientes, justificando además su modelo de negocio con respecto a los fabricantes de los productos que comercializa.

Entre estas responsabilidades destacan las siguientes:

- Colaboré con el área de comercial con la implementación de nuevos clientes.
- Llevé a cabo la implementación de soluciones de virtualización para la oferta de servicios en la nube.
- Llevé control y ejecuté visitas presenciales trimestrales con los clientes.
- Desarrollé herramientas que apoyan en la mejora de la experiencia de los clientes.
- Actualicé regularmente componentes funcionales de la página web.
- Generé contenido para captación de tráfico hacia la página de internet.
- Atendí los requerimientos de los casos de uso planteados por los clientes.
- Llevé control de la plataforma de soporte técnico.
- Proporcioné atención técnica postventa a problemáticas derivadas del uso de los productos y servicios a los clientes.
- Desarrollé arquitectura de soluciones para cubrir casos de uso específicos relacionados con seguridad.

Las responsabilidades que ejecuté que tienen relación con este proyecto son las siguientes:

- Detecté las necesidades del mercado para la mejora del producto DLP.
- Realicé el diseño de soluciones enfocadas a resolver la problemática detectada.
- Llevé a cabo el desarrollo de la solución.
- Participé en la implementación de la solución.
- Llevé a cabo el ajuste y mantenimiento de soluciones.

### 1.3 Descripción del Producto DLP.

El producto de protección de perdida de información (DLP) que se encuentra en el catálogo de productos es una solución basada en el modelo SaaS, que permite brindar seguridad de la información.

Por medio de este producto los datos son protegidos en caso de filtraciones accidentales o intencionales por parte del operador de dicha información o un tercero malicioso. Las organizaciones tienen el control de quien accede a sus datos y pueden revocar el acceso a los mismos, sin importar si los datos se encuentren en algún dispositivo ajeno a la organización.

Para poder realizar las labores asignadas es imprescindible el profundo conocimiento de la herramienta en cuestión que forma parte del catálogo de productos, a este respecto se identifican del producto las siguientes características:

- El producto DLP utiliza un modelo cliente-servidor por lo cual cuenta con una agente que es instalado en los ordenadores objetivo.

- El agente cifra los datos de los archivos según un algoritmo configurable por el administrador entre una lista de algoritmos conocidos y categoriza la información en grupos para así jerarquizar los permisos de acceso con base en grupos lógicos configurables. También coloca etiquetas en encabezados de los archivos y al final de la secuencia de datos que representan al archivo en memoria secundaria para distinguir esta categorización. Por último, mantiene comunicación con un servidor dedicado para validar si el usuario en cuestión tiene permitido hacer uso de alguna pieza de información.

- El agente identifica intentos del usuario de copiar o reproducir la información protegida. Solo se autoriza guardar el nuevo archivo con la información que fue copiada sin cifrar introduciendo un código de autorización que el usuario solicita al departamento de tecnología, En caso de no tener el código, el archivo se guarda cifrado.

- El producto DLP cuenta con una interfaz web online para realizar la configuración. Esta interfaz permite la configuración del producto mediante distintos criterios que definen a grupos de ordenadores, usuarios y casos a los que se les aplican políticas de uso de la

información. Estas políticas definen los privilegios de uso que los usuarios finales[1] tendrán sobre conjuntos de datos, de forma tal que se controla el acceso, los niveles de privilegio y se previene la pérdida de la información. La interfaz web cuenta con una opción que sirve para generar "fichas de acceso" a los usuarios administradores de la plataforma, mediante autenticación Oauth2.0[2]. Es en esta sección del portal de administración en que se proporciona autorización por parte del usuario administrativo que gestiona la herramienta para completar el proceso OAuth2.0 y así se obtiene la "ficha de acceso".

Es finalmente esta "ficha de acceso" la que se usará en el encabezado Authentication en el "mensaje de *consulta*" a las distintas URLs que conforman la API, consiguiendo acceso a los *recursos* para el usuario administrativo que proporcionó la confirmación en el portal de configuración.

- La interfaz web del portal de administración no cuenta con opciones de visualización de la información en forma gráfica o analítica, en su lugar existe una funcionalidad que genera archivos de bitácora que pueden ser consumibles por herramientas destinadas a la visualización grafica conocidas como SIEMS (Security Information and Event Management).

- El portal de administración contiene además la funcionalidad necesaria para aprobar intentos del usuario de guardar información en un estado sin cifrar.

1 Puede confundirse el término "usuario final" usado en este contexto con el de "usuario", mientras que el primero es aquel que utiliza las máquinas de una organización para desempeñar sus funciones de trabajo cotidianas, el segundo es aquel administrador de la plataforma que necesita credenciales de acceso para acceder al portal de administración para completar el proceso OAuth2.0 y así obtener la "ficha de acceso"

2 Se presenta con mayor detalle este mecanismo de autenticación en la subsección 2.15.

- Así mismo, el portal de administración, permite configurar la integración con otro producto del mismo fabricante diseñado para proveer seguridad mediante el establecimiento de configuraciones locales de seguridad según ciertos criterios. En lo sucesivo se hará referencia a este último producto como "gestor de configuraciones del sistema".

- Ambos productos de la compañía pueden ser complementarios, permitiendo la adición de las funcionalidades del DLP al producto "gestor de configuraciones del sistema".

- Alternativamente la conexión con la herramienta de visualización provee la posibilidad de comercializar el complemento DLP como un producto independiente del producto "gestor de configuraciones del sistema", facilitando a los clientes que cuentan con herramientas de visualización de información (SIEM) el uso del módulo DLP, sin la necesidad de adquirir el producto "gestor de configuraciones del sistema".

- La herramienta DLP cuenta con una API tipo REST3[3] que permite interactuar con ella por medio de *consultas* HTTP, esto significa que es posible la recuperación de documentos (información) relacionados al uso de la herramienta DLP, a la vez que nos proporciona los medios para establecer parámetros de funcionamiento. La Figura 2 muestra un diagrama de bloques en el que se observa la integración del producto DLP, el desarrollo aquí creado y los agentes de la solución DLP instalados en los equipos de los clientes.

[3] Se revisará con mayor detalle en la sección 2.17.

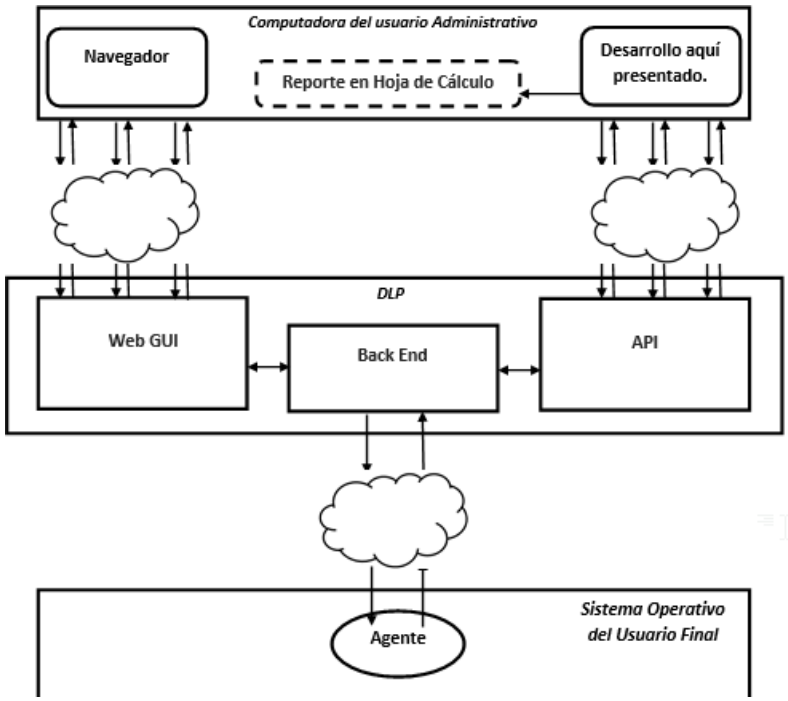

Figura 2. Integración de la solución aquí presentada en el producto DLP.

## CAPITULO 2. Marco Teórico.

La presente explicación, lejos de ser extensiva, tiene como propósito dar un sustento teórico que permita entender el funcionamiento subyacente al desarrollo que esta documentado en este trabajo.

En ese sentido se parte de exponer las bases mínimas suficientes que proporcionen entendimiento del protocolo HTTP y sus características, que lo hacen ideal para el intercambio de información contenida en documentos (*recursos*) entre clientes y servidores.

Se continúa exponiendo el proceso por el cual un cliente logra obtener y alojar *recursos* en un servidor, aportando una explicación de los actores involucrados en el proceso de comunicación, así como del *recurso* mismo.

Se muestran además los mecanismos de autenticación destinados a validar la identidad y consecuente autorización asociada al usuario que desea realizar la consulta de un *recurso*, tomando especial interés en el mecanismo de seguridad OAuth 2.0.

Se presentan los elementos constitutivos del protocolo HTTP, y su utilidad para sustentar el funcionamiento de una API REST como mecanismo de distribución de información.

Las secciones posteriores presentan un acercamiento hacia los mecanismos de detección de patrones enfocados a la prevención de perdida de información que son utilizados en el mercado por los fabricantes de herramientas denominadas Data Loss Prevention.

## 2.1 HTTP.

El protocolo HTTP es un protocolo sin estado diseñado para la transferencia de documentos (*recursos*) entre servidores web y diversos clientes. Para efectos de la comunicación un cliente establece comunicación con un servidor que aloja información con la intención de obtener esos *recursos*. Se exponen a continuación los elementos y actores involucrados en el proceso. [1]

## 2.2. El "agente de usuario".

Para lograr el propósito de solicitar un recurso, un "agente de usuario" debe ser usado con el objetivo de efectuar las *consultas* de interés para el usuario final. El "agente de usuario" procede entonces a enviar la *consulta* a un servidor, y se encarga de gestionar la *respuesta* que este último devuelve. [2]

Existen múltiples tipos de "agentes de usuario", entre los que destacan los siguientes:

1. Navegadores de Internet (web browsers) – habilitan a un usuario *consultar* y mostrar *recursos* en una interfaz gráfica constituida por una ventana que permite navegar a través del documento. Toman la información del usuario acerca del correspondiente identificador (URL) que referencia a un *recurso* que se desea *solicitar* para emitir un "mensaje de *consulta"* al servidor que tiene el *recurso*. [3]

2. Agentes de línea de comandos (command line agents) – Consisten en piezas de software de línea de comandos que por medio de *banderas* permiten establecer los parámetros de la *consulta*: el método, los encabezados, y el cuerpo del mensaje. Con esta información el agente por línea de comandos generara la *consulta* que es enviada al servidor. El resultado consiste en procesar y mostrar la *respuesta* del servidor completa, incluyendo encabezados y códigos de error. Estos agentes de línea de comandos muchas veces son integrados en scripts que tienen como finalidad solicitar recursos y extraer información de los mismos. Esto es particularmente útil para propósitos como el monitoreo de precios, gestión de cambios en documentos, indexación de documentos, diagnósticos de seguridad, adquisición de noticias, actualización periódica de firmware, etc.

3. Existen desarrollos de software que de forma automatizada generan *consultas* HTTP, con el objetivo de obtener *recursos* de diversos formatos (imágenes, texto, documentos, etc.), periódicamente. Estos desarrollos pueden ser implementados en distintos lenguajes de programación y scripting. Algunas de las funciones del desarrollo que presento pertenece a esta categoría. [4]

### 2.3. El servidor web.

En términos generales el servidor web constituye una pieza de software que funciona en una máquina (homónima a la pieza de software) que debe gestionar el acceso a los *recursos* de los que dispone, para proceder a entregar una representación de los *recursos* a aquellos usuarios autorizados, a quienes están destinados.

Existen distintos tipos de piezas de software que cumplen con la función de servidor. Entre ellas destacan:

- Software de propósito general que permiten almacenar documentos para ser distribuidos.
- Piezas de aplicación (conectan con una aplicación que genera contenido dinámico, como un sistema de cámaras, o un sistema de *consulta* geológica).
- Servidores embebidos en chips, en los que existen portales cautivos por defecto que sirven para la configuración de aparatos, por ejemplo: cámaras, antenas, repetidores, routers, etc. [5]

Entre las funciones principales de los servidores web, se encuentran:

- Establecer conexiones destinadas al intercambio de información. [6]
- Permanecer a la escucha de nuevas *solicitudes*. [7]
- Procesar las *solicitudes*. [8]
- Acceder a los *recursos* alojados en almacenamiento secundario. [9]
- Construir las *respuestas*. [10]
- Enviar las *respuestas*. [11]
- Registrar cada evento relacionado con transacciones. [12]

### 2.4. **Los recursos**.

Los *recursos* pueden ser variados, estos pueden ser desde un archivo estático (de cualquier índole) en el sistema de archivos del servidor, hasta contenido dinámico generado bajo demanda por procesos que son ejecutados por el servidor. Un ejemplo de esto es un sistema de cámaras web que puede ser *consultado* por medio del protocolo HTTP. [13]

Cada uno de estos *recursos* es etiquetado según su tipo, en los llamados MIME types, el servidor web es el encargado de hacer este etiquetado de los objetos. El etiquetado consta de un objeto principal y un objeto secundario separados por una diagonal (/), por ejemplo: text/html, image/jpg, etc. [14]

## 2.5. **Identificación de recursos**.

Para que un servidor pueda devolver un *recurso*, primero tiene que identificarlo, para este propósito el cliente proporciona una URI (Uniform Resource Identifier) que sirve como identificador del *recurso*, indicando su ubicación (el servidor donde el *recurso* está alojado) y la ruta relativa en donde el cliente espera encontrar el *recurso*. [15] Adicionalmente podría indicar información adicional como, por ejemplo, el puerto donde el servidor se encuentra a la escucha de la *solicitud*. [16]

Los URLs son los tipos más comunes de URIs, identifican la ubicación de un *recurso*. Si el *recurso* es eliminado, entonces la referencia deja de ser válida. Estas URLs permiten indicar al cliente donde debe ser buscado el *recurso*.

Generalmente estos URLs se componen de diversas partes [17], siendo las más comunes:

**http://www.unam.mx:80/ubicacion/miarchivo.html?llave1=valor1&llave2=valor2#LugarEnDocumento**

Dónde:

1. El indicador del esquema o protocolo (en el caso mostrado es HTTP), seguido de dos puntos y dos diagonales ascendentes.
2. El nombre de dominio seguido de dos puntos.
3. Opcionalmente el puerto donde el servidor está a la escucha.
4. La ubicación hacia el documento.
5. El documento o *recurso.*
6. Opcionalmente se pueden proveer una serie de parámetros llave-valor que permiten el envío de información a través de la *consulta* HTTP en la URL.
7. Opcionalmente se puede incluir una etiqueta a alguna parte del documento.

Ejemplos de URLs son los siguientes:

- https://www.unam.mx/robots.txt
- https://www.unam.mx/sites/all/themes/unam/logo.png

### 2.6. Comunicación HTTP.

La Figura 3 ilustra el proceso de comunicación que se establece con HTTP, donde, para poder obtener un *recurso* de un servidor web es necesario iniciar una serie de pasos definidos por el protocolo[4]:

- Primero el cliente inicia una *solicitud* por medio de un "mensaje de *solicitud"* HTTP usando un identificador que define la ubicación del *recurso* que se desea *consultar*, hacia el servidor, especificando un verbo (método) y la versión del protocolo.
- A continuación, el servidor, busca el recurso solicitado y en caso de tenerlo *responde* usando un "mensaje de *respuesta*" HTTP, indicando la versión del protocolo, el MIME Type del *recurso* devuelto, su tamaño, algunos encabezados y proporcionando el *recurso* en sí [18].

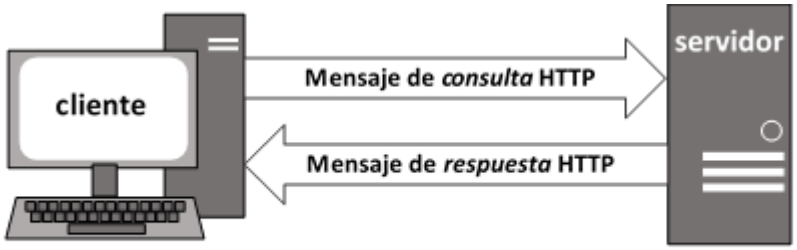


Figura 3. Mensajes de solicitud y respuesta definidos por el protocolo HTTP sin autenticación.

4 Se describe el funcionamiento de este mismo proceso para un recurso que requiere autenticación en la sección 2.13.

## 2.7. Mensajes HTTP.

HTTP establece dos tipos de mensajes: "mensajes de *solicitud"* y "mensajes de r*espuesta"*. Ambos tipos de mensajes HTTP son bastante similares entre sí y comparten algunos atributos en común. En estos mensajes se encuentran elementos en común y otros particulares a cada uno, sin embargo, en ambos casos es posible separarlos en tres porciones. La Figura 4 ilustra el contenido de los mensajes usados en el proceso. [19]

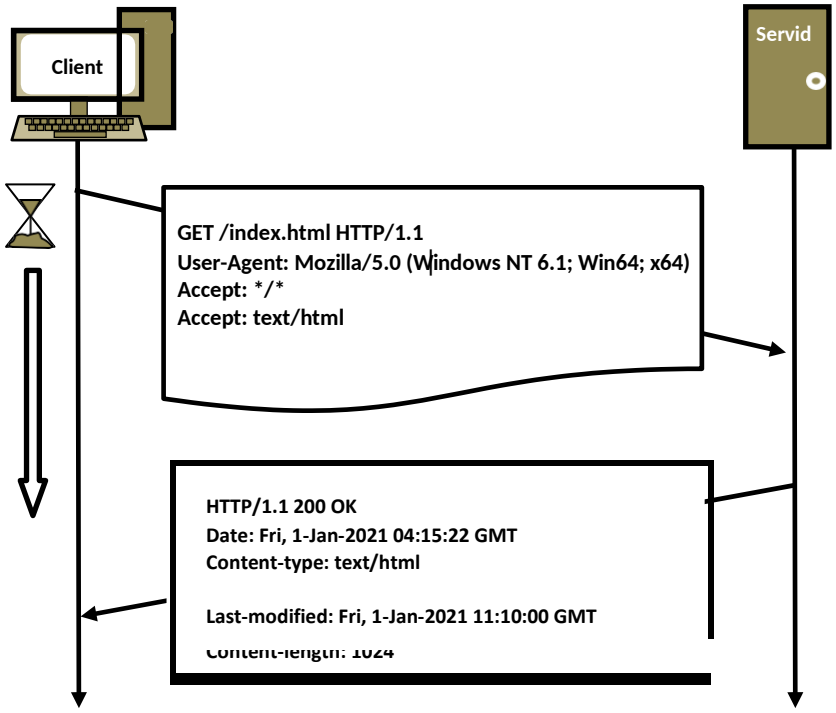


Figura 4. Mensajes de *consulta* y *respuesta* sin autenticación.

En la primera porción de ambos se puede encontrar una línea inicial que tiene información acerca del protocolo que se está utilizando. Sin embargo, en el caso del "mensaje de *solicitud"* es necesario aportar información como el método y la ubicación del *recurso*. En el caso del "mensaje de *respuesta"*, este emite un código informativo indicando el estado de la *solicitud.* [20]

A continuación, aparece una sección, donde el cliente y el servidor intercambiarán información acerca de la comunicación, conocida como el área de encabezados o headers. Mediante el uso de encabezados el cliente y el servidor inician un intercambio de mensajes con el objetivo de negociar algunos parámetros de la comunicación, así como para intercambiar mensajes informativos. [21]

Finalmente se encuentra una sección conocida como “cuerpo del mensaje”, donde el cliente puede enviar información relevante al servidor y por su parte el servidor utiliza esta sección para devolver una representación del *recurso solicitado*. [22]

### 2.8. Los encabezados.

Los encabezados permiten el intercambio de metadatos que sirven indicadores del estado de la conexión y permiten a cliente y servidor cambiar los parámetros acordados para la interacción entre ambos. De la misma manera permiten al cliente el envío de información al servidor como credenciales de acceso. Existen por tanto encabezados de propósito general que son encontrados tanto en los “mensajes de *solicitud”*, como en los “mensajes de *respuesta”*. Así mismo existen otros que son particulares a cada uno de estos tipos de mensaje, encontrando de esta manera algunos en el “mensaje de *solicitud”* exclusivamente y otros en el "mensaje de *respuesta*" exclusivamente. [23]

Combinando métodos y encabezados es posible determinar lo que el cliente requiere y lo que el servidor es capaz de proporcionar, además de permitir que se lleven a cabo de forma trasparente ciertas operaciones rutinarias para el mantenimiento de la comunicación HTTP.

Otro aspecto importante a notar es que algunos encabezados no están definidos en la especificación para HTTP/1.0 e incluso algunos en la especificación para HTTP/1.1. Existen por tanto encabezados que extienden estas especificaciones y que son usados generalmente para propósitos específicos, como selección de opciones de idioma, información especial, establecimiento y seguimiento de sesiones. [24]

Algunos de los encabezados más usuales [25] son los siguientes:

| Encabezado | Tipo | Descripción | Ejemplo |
|---|---|---|---|
| Accept | Cliente | Usado por el cliente para indicar al servidor acerca de sus capacidades al servidor, relacionadas a tipos de medios. | Accept:text/html,image/jpeg, |
| Accept-Charset | Cliente | Usado por el cliente para indicar al servidor acerca del tipo de conjunto de caracteres que el cliente soporta para interpretar la *respuesta*. | Accept-Charset: iso-8859-5, UTF-8 |
| Accept-Encoding | Cliente | Indica al servidor que tipo de codificaciones que son soportados en la *respuesta*. | Accept-Encoding: compress, gzip |
| Accept-Language | Cliente | Indica al servidor acerca de las preferencias de idioma relacionadas con el *recurso* que se *solicita*. | Accept-Language: es |
| Allow | Servidor | Es usado por el servidor para informar al cliente acerca de los métodos válidos para ese servidor. | Allow: GET, HEAD, POST |
| Authorization | Cliente | Usado por el cliente para autenticarse y obtener acceso a los *recursos* dentro de un realm, proporcionando las credenciales o fichas correspondientes. | Authorization: Basic DJkuMTEcowcipTIhVUAyp2Sg MD |
| Cache-Control | General | Incluye directivas que son destinadas a los intermediarios en la cadena de envió del mensaje. | Cache-Control : no-store |
| Connection | General | Permite especificar opciones deseadas para la conexión | Connection: close |
| Content- Encoding | Servidor | Indica al cliente acerca de codificaciones adicionales que fueron aplicadas al cuerpo del mensaje. | Content-Encoding: gzip |
| Content-Language | Servidor | Indica el idioma del público al que está destinado el *recurso*. | Content-Language: es, en |
| Content-Length | Servidor | Indica el tamaño del cuerpo del mensaje | Content-Length: 2565 |
| Content-Location | Servidor | Informa al cliente acerca de ubicaciones alternativas para un *recurso* en particular. | Content-Location : <URL> |
| Content-MD5 | Servidor | Informa al cliente el digest del cuerpo del mensaje con el objetivo de realizar una revisión de integridad delmismo. | Content-MD5:<DigestMD5> |
| Content-Range | Servidor | Permite indicar al cliente en caso de un mensaje parcial donde puede incorporar la nueva información recibida a mensajes anteriores para conformar el cuerpo del mensaje completo. | Content-Range: bytes 0-499/4444 |

| | | | |
|---|---|---|---|
| Content-Type | Servidor | Indica al cliente acerca del tipo de medio del *recurso* solicitado. | Content-Type: text/html |
| Date | General | Específica el tiempo exacto en el que el mensaje fue generado. | Date: Mon, 6 Nov 2000 20:59:59 GMT |
| ETag | Servidor | Permite establecer etiquetas informativas. | ETag: "abcd" |
| Expect | Cliente | Informa al servidor que ciertos comportamientos serán *solicitados* por el cliente. | Expect: 100-continue |
| Expires | | Establece un tiempo de expiración para el mensaje. | Expires: Mon, 6 Nov 2000 20:59:59 GMT |
| From | Cliente | Especifica una dirección de correo electrónico asociada al usuario que gestiona al cliente. Su función es la de recibir "retroalimentación" por parte de los administradores de sistemas cuando un script que genera *consultas* está resultando problemático. | From: user@dominio.com |
| Host | Cliente | Hace referencia al servidor al cual está destinado el mensaje. | Host: www.unam.mx |
| If-Match | Cliente | Usualmente utilizado en conjunción con caches permite especificar etiquetas que si son reconocidas por el cache detonaran el envío de información acotada. | If-Match: "abc" "123" "xyz" |
| If-Modified-Since | Cliente | Al igual que el encabezado anterior permite obtener el *recurso* solo si cumple con una condición de fecha. | If-Modified-Since: Mon, 6 Nov 2000 20:59:59 GMT |
| If-None-Match / If-Unmodified-Since | Cliente | Un cliente puede usar estos encabezados por ejemplo en conjunción con el método PUT para conseguir el efecto de crear un *recurso* si es que este no existía. | If-None-Match: "abc" "123" "xyz" |
| Last-Modified | Servidor | Informa al cliente sobre el momento en que fue modificado el *recurso* por última vez. | Last-Modified: Mon, 6 Nov 2000 20:59:59 GMT |
| Location | Servidor | Sirve para redirigir al cliente a una nueva ubicación donde el *recurso* podrá ser encontrado. | Location: <URL> |
| Max-Forwards | General | Establece un número máximo de intermediarios activos (proxies y gateways) que recibirán el mensaje. | Max-Forwards: 5 |
| Pragma | General | Permite él envió de directivas de implementación especifica. | Pragma: no-cache |

| | | | |
|---|---|---|---|
| Proxy-Authenticate | Proxy | Similar a www-Authenticate, pero usado para solicitar autenticación por parte de un proxy. | Proxy-Authenticate: Basic realm="finanzas" |
| Proxy-Authorization | Cliente | Análogo a Authorization, pero utilizado para *responder* el challenge de un proxy. | Proxy-Authorization: Basic DJkuMTEcowcipTIhVUAyp2Sg MD |
| Referer | Cliente | Especifica la dirección del *recurso* de donde el cliente tomo la URL que ahora *consulta*. | Referer: www.unam.mx |
| Retry-After | Servidor | Indica cuando estará disponible un *recurso* nuevamente. | Retry-After: Mon, 6 Nov2000 20:59:59 GMT |
| Upgrade | Cliente | Permite al cliente solicitar al servidor un cambio en los protocolos usados. El servidor es libre de decidir si acepta los cambios. | Upgrade: HTTP/2.0 |
| User-Agent | Cliente | Informa al servidor acerca del "agente de usuario" que está siendo utilizado para formular los *"mensajes de consulta"*. | User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.157 Safari/537.36 Brave/74 |
| WWW-Authenticate | Servidor | Usado por el servidor para indicar que el cliente debe autenticarse. | WWW-Authenticate Basic realm="finanzas" |

### 2.9. Métodos.

Para definir lo que el cliente espera al iniciar la interacción con respecto a un *recurso* del servidor se cuenta con diversos métodos que permiten especificar la acción que el cliente pretende ejecutar.[26] Los más comunes son los siguientes:

| Método | Acción | Semántica |
|---|---|---|
| GET | Obtener un *recurso* en particular del servidor web. | "Dame una representación de este documento". |
| PUT | Guardar datos del cliente a un *recurso* del servidor web. | "Coloca este recurso en el servidor". |
| DELETE | Solicita la eliminación un *recurso* del servidor web en una ubicación determinada. El servidor *responde* con un código que indica éxito en la acción solicitada por el cliente, sin embargo, no existe una garantía del borrado del documento, ya que esta acción generalmente esta supervisada por algún otro proceso, haciendo el cumplimiento de la consulta no fiable para el cliente. | "Borra este recurso en el servidor". |

| | | |
|---|---|---|
| POST | Envía información por parte del cliente a un servidor. Esta funcionalidad lo hace ideal para el envío de información capturada por medio de formularios. Esto puede funcionar por ejemplo para actualizar registros en una base de datos. Debido a lo anterior, el servidor no necesariamente *responde* con una URL que referencia a un nuevo recurso en el encabezado Location. | ”Recibe esta información” |
| HEAD | Recupera exclusivamente encabezados de los mensajes HTTP relacionados con un *recurso* en particular, omitiendo el cuerpo del mensaje (documento). | Similar a GET, pero recupera una porción del “mensaje de *consulta”*. |
| TRACE | A lo largo del recorrido del paquete desde el origen, hasta el destino es común que este atraviese diversos puntos medios como proxies, gateways, o firewalls. Cada uno de estos intermediarios tiene la capacidad de modificar el “mensaje de *solicitud”*. Mediante la utilización del método TRACE, se puede *consultar* el estado final del *“mensaje de consulta”* en algún punto determinado, para fines de diagnóstico y estadística. Este mensaje no incluye el cuerpo del mensaje original. El punto medio *responde* con la totalidad del mensaje en el cuerpo del mensaje en la *respuesta*. | “Obtener representación en ese punto intermedio” |
| OPTIONS | Permite *consultar* al servidor acerca de los métodos que soporta un *recurso*, sin implicar la transferencia del *recurso* en cuestión. | “Obtener opciones” |
| PATCH | Este método no está definido en la especificación HTTP. Como método de extensión su propósito radica en interactuar con Web API’s. | N/A |

No todos los servidores implementan todos los métodos.

### 2.10. Códigos de *respuesta*.

Una vez que ha sido recibida una *consulta*, el servidor *responderá* utilizado una serie de códigos diseñados para informar al usuario el estado de la *consulta*. [27]. Los más comunes son los siguientes:

| Código | Significado |
|---|---|
| 1xx | Informacional. |
| 2xx | El documento es regresado correctamente. |
| 3xx | Se redirige la *solicitud* a otro servidor o URL, esto ocurre cuando un *recurso* ha sido movido. Se refiere la nueva ubicación en el "mensaje de *respuesta*" por medio del encabezado destinado para tal fin. |

| | |
|---|---|
| 4xx | No se ha localizado el *recurso* solicitado por el cliente o a ocurrido un error derivado de la forma en que se realizó la *consulta*. |
| 5xx | Existe un problema con el servidor. |

Si bien la semántica relacionada al primer digito a los códigos de error aporta bastante información acerca del estado de la *respuesta*, se debe mencionar que en cada una de las categorías de error existen diversos tipos de mensajes.

De esta forma un código de éxito (2xx) en el servidor puede deberse a que el recurso fue devuelto correctamente (200) o a que el recurso fue creado en el servidor correctamente (201) debido a un "mensaje de *solicitud"* que es usado con el verbo PUT. Otros códigos de tipo 2xx hacen referencia a que el estatus es exitoso.

Por otro lado, un código de redirección (3xx) puede informar que el recurso fue movido permanentemente (301), o tal vez temporalmente (307), entre otros.

En el caso de tener un código de error del lado del cliente (4xx), este puede deberse a que la *solicitud* se realizó usando un método no permitido por el servidor (405), a que el servidor no entiende (o no permite) el tipo de método que es enviado (415), o a que se requiere algún tipo de autenticación para acceder al mismo (401, 407), entre otros.

Respecto al código de error del lado del servidor (5xx), este puede referirse a una interrupción del servicio de forma momentánea (503), o a un error interno del servidor (501), entre otros.

El protocolo permite la ampliación de estos códigos de error siempre y cuando la aplicación cliente entienda la semántica involucrada en el primer digito del código devuelto por el servidor.

### 2.11. Cuerpo del mensaje.

Se trata de la sección donde el *recurso* es devuelto, dependiendo de las opciones establecidas en el *"*mensaje *de consulta".* Por ejemplo, este *recurso* podrá ser codificado para efectos de la transferencia según lo acordado entre el cliente y el servidor, mediante el uso del encabezado Transfer-Encoding. El tipo de archivo estará determinado por el encabezado Entity-Type del "mensaje de *respuesta*" donde el servidor envía el *recurso*. [28]

## 2.12 Ejemplo de comunicación.

Como se ha dicho antes, la utilización de los códigos de respuesta, junto con los encabezados, sirve como mecanismo para comunicar efectivamente lo que el cliente requiere y lo que el servidor es capaz de proporcionar. A continuación, dos ejemplos que muestran como la conjunción de encabezados y códigos de respuesta permiten al cliente y al servidor comunicarse efectivamente:

Un ejemplo, que se ilustra en la Figura 5, muestra una *solicitud* donde el *recurso* ha sido movido del servidor. En este caso la correspondiente *respuesta* del servidor, contendrá el código de redirección 301 (Permanently Moved), conteniendo además el encabezado Location, referenciando la nueva ubicación donde el *recurso* podrá encontrarse, La *respuesta* correspondiente se ve como se muestra a continuación:

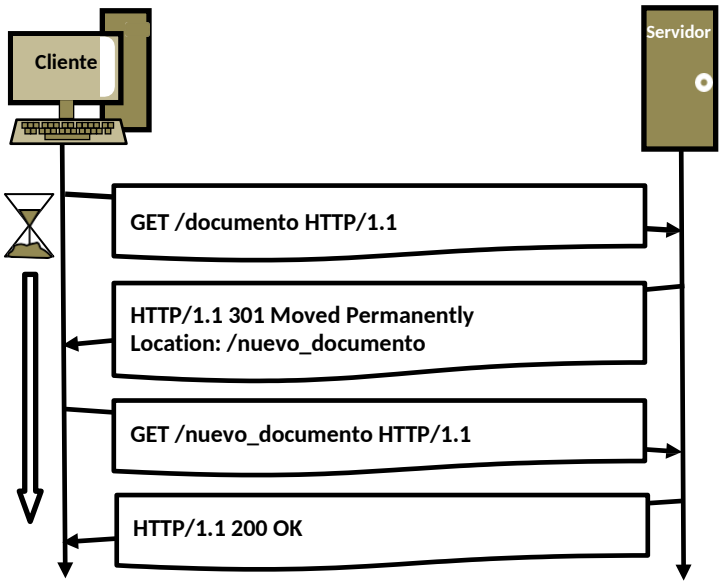


Figura 5. Mensajes de *consulta* y *respuesta* en una comunicación con redirección.

## 2.13 Recursos protegidos.

En algunos casos el *recurso* que el cliente desea consultar está protegido para preservar confidencialidad del mismo, es por ello que el proceso a seguir para obtener el *recurso*, aunque similar debe contemplar la autenticación por parte del cliente. La Figura 6 ilustra el proceso de comunicación de un *recurso* protegido.

En este caso, después de una *solicitud* inicial ordinaria, el servidor *responde* usando un código 401, junto con el Encabezado WWW-Authenticate, con el cual indica al usuario que debe autenticarse para obtener el *recurso*, así como las opciones de autenticación disponibles. Este mensaje es conocido como el mensaje de desafío (challenge message).

Es entonces que el cliente procede a hacer él envió de la información relativa a la autenticación mediante el uso del encabezado Authentication, especificando el tipo de autenticación elegida y la información que valida que el usuario tiene acceso al *recurso*.

Si el proceso es exitoso entonces el servidor *responde* con un código de éxito 200 y una representación del *recurso* solicitado en el cuerpo del mensaje.

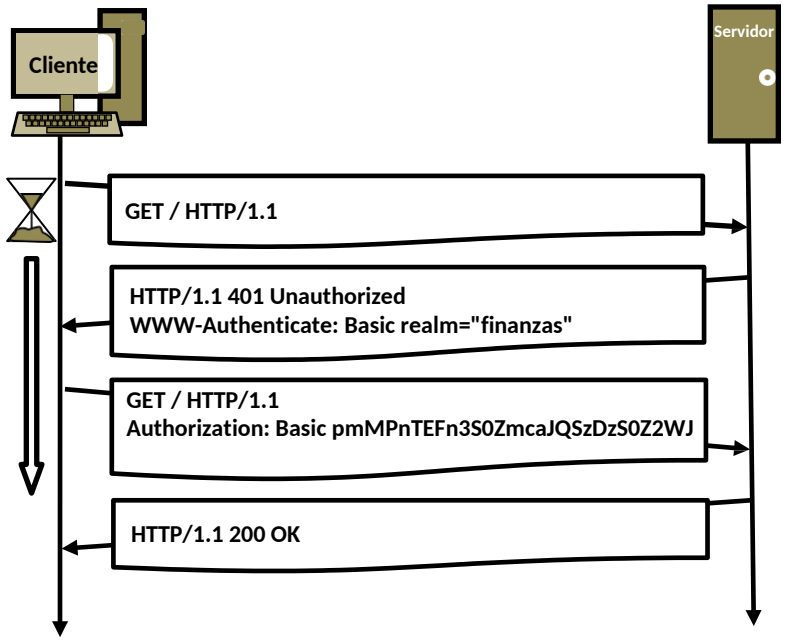


Figura 6. Proceso de comunicación HTTP donde el servidor solicita autenticación (En este caso autenticación básica) al cliente.

## 2.14. Mecanismos de Autenticación.

El protocolo HTTP descrito en el RFC 2616 no contiene mecanismos de seguridad natos, sin embargo, permite la implementación de diversos mecanismos opcionales por lo que muchas extensiones al protocolo han sido diseñadas, consiguiendo cada una su implementación (RFC) particular. [29]

Algunas de las formas de implementar autenticación por medio de HTTP son las siguientes:

| Mecanismo | RFC | Comentario | Ventajas | Desventajas |
|---|---|---|---|---|
| Basic | RFC 7617 | En este tipo de autenticación el usuario y la contraseña son procesados por un algoritmo que los hace ilegibles. | Implementación Simple. | Es considerado como uno mecanismo no seguro, aunque preferible a no asegurar el acceso a los *recursos*. Un atacante malicioso puede obtener la información de usuario y contraseña sin mayor esfuerzo a partir del digest que se envía en el encabezado Authorization del *"mensaje de consulta"*. |
| Digest | RFC 7616 | Envía un digest de distintos datos junto con la contraseña del usuario. Opcionalmente puede verificar la integridad del cuerpo del mensaje. | Se encarga de que la contraseña nunca viaje en texto plano a través de la red. | Utiliza mayor cantidad de *recursos* de procesamiento y *recursos* de red. |
| HOBA | RFC7486 | El usuario puede autenticarse por medio de su llave pública. | No involucra contraseñas en lo absoluto. Puede ser usado con JavaScript sobre HTML. | Un par de llaves son generadas por cada sitio contra el cual se desea autenticar. |
| Negotiate | RFC4559 | Genera fichas (tickets) de servicio que posibilitan a un cliente el acceso a los *recursos* de diversos servicios. | Impide que las contraseñas viajen por la red en forma de texto claro, a la vez que proporciona los medios para el acceso a *recursos* sin que los servidores que proveen estos *recursos* conozcan la contraseña del usuario. | Relativa complejidad en su implementación. |
| OAuth2.0 | RFC6750 | Particularmente útil para poder controlar los *recursos* de un usuario en una aplicación. | Fácil configuración. La seguridad de acceso viene dada por el "servidor de autenticación" y requiere aprobación del usuario. Las contraseñas nunca son enviadas a través de la red. | La "ficha de acceso" es emitida "al portador" por lo que un atacante en control de la "ficha de acceso" puede hacerse pasar por el usuario. |

Para lograr este propósito HTTP divide diversos conjuntos de archivos dentro de realms[5]. De esta forma el servidor puede controlar el acceso a ciertos *recursos* basándose en la identidad de los usuarios. [30]

Para el objetivo que persigue este trabajo, me centraré exclusivamente en la autenticación con el mecanismo Oauth2.0.

### 2.15. Autenticación usando OAuth 2.0.

OAuth2.0 es un mecanismo de autenticación abierto y extensible que permite que un Cliente pueda acceder y utilizar los recursos protegidos de un usuario final (con la autorización de este usuario) ante un aplicativo, para tal propósito valida su identidad ante un "servidor de autenticación" que finalmente le provee el acceso a los recursos del "servidor de recursos". [31]

Este mecanismo es útil porque permite controlar por medio de una API operaciones normales de un Usuario en un aplicativo. Ejemplos de este tipo de comportamiento por APIs se pueden encontrar en las APIs de Facebook y de Twitter que permiten a un usuario tomar la funcionalidad de estas plataformas de forma tal que pueda ser utilizada en conjunción con otras herramientas.

Un caso de uso interesante derivado del ejemplo anterior es el de unir tecnologías de distintos fabricantes y con distintas vocaciones, tal es el caso de hornos de cocina equipados con tecnología IoT, cuya funcionalidad puede ser unida a las funcionalidades de Twitter, de forma tal que al estar lista la preparación de un alimento se publique un tweet indicando el evento.

Otro ejemplo que hace más evidente lo anterior es la integración de un foco inteligente con una bocina inteligente. En principio de cuentas tecnologías desarrolladas por fabricantes diferentes pueden encontrar integración para trabajar coordinadamente y lograr metas comunes.

5 No confundir con el concepto de realms aquí usado con el de "security realms" que se emplea en la tecnología Active Directory o LDAP.

Habiendo dicho lo anterior conviene repasar un poco como se lleva a cabo la autenticación por el mecanismo OAuth2.0.

OAuth 2.0 define a los siguientes actores que intervienen en el proceso de autenticación [32]:

1. Dueño del recurso (Owner).
2. Cliente (Client)[6].
3. "Servidor de autenticación" (Autenticación Server).
4. "Servidor de recursos" (Resource Server).

Los tipos de concesiones (Type of Grants), son aquellos mecanismos por los cuales el usuario que es dueño del recurso proporciona la autorización al cliente para poder acceder al recurso. [33]. Los más comunes son los siguientes:

1. Authorization Code Grant.
2. Password Grant.
3. Client Credential Grant.
4. Implicit Grant.

Para los fines que busca este trabajo se explica exclusivamente el tipo Authorization Code Grant, debido a que este es de relevancia para el proyecto desarrollado. La figura 7 ilustra el proceso de autenticación.

6 En el contexto de la autenticación OAuth 2.0 no se debe confundir “Cliente” con el “cliente” descrito en la especificación HTTP, puesto a que, si bien ambos pueden estar integrados en el mismo desarrollo, el término “Cliente” de Oauth 2.0 hace referencia al aplicativo que desea hacer uso de los recursos de un usuario en el "servidor de recursos" (Al margen del lugar donde se ejecuta. Ej. servidor, equipo de escritorio, aplicación de teléfono móvil inteligente, etc.), mientras que en el contexto de HTTP el “cliente” hace referencia al “agente de usuario” que es utilizado por un usuario para realizar *consultas* de interés al servidor.

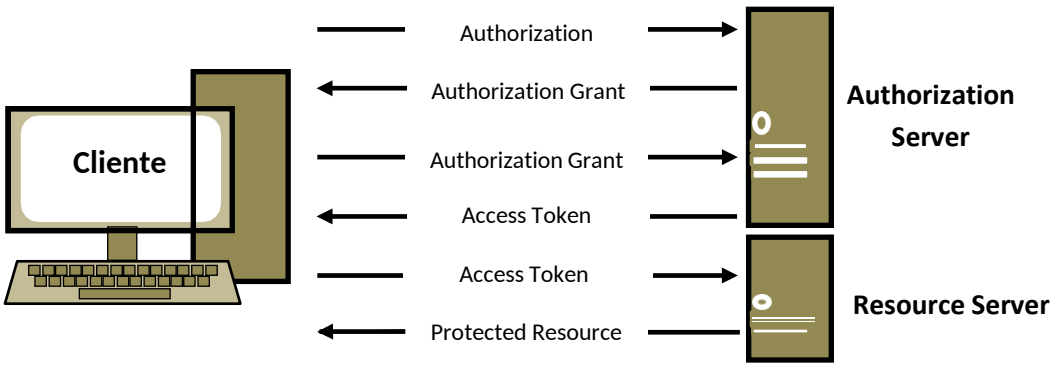


Figura 7. Mensajes de *consulta* y *respuesta* usando el mecanismo OAuth 2.0.

En este tipo de acceso el "servidor de autenticación" actúa como intermediario entre el usuario y el Cliente.

Cuando un Cliente desea obtener los recursos que se encuentran en el "servidor de recursos", el Cliente debe hacer una consulta al "servidor de autenticación", denominada "solicitud de autenticación" (Authorization Request) proporcionando ciertos parámetros en la URL que el "servidor de autenticación" utiliza para identificar al Cliente [34]. Esta URL se forma con los siguientes paramentos:

```
https://ServidorAutenticación.com/Autenticación/?Response_Type=code&<Client_id>&<URL>&<scope>
```

Donde [35]:

| Parámetro | Significado |
|---|---|
| Resṕonse_Type | Siempre debe ser code para el tipo Authorization Code Grant. |
| Client_id | El "servidor de autenticación" proporciona al cliente Client_id representando la información de registro del cliente. |
| URL | Se trata de la URL de retorno que es enviada por el "servidor de autenticación" al Agente de usuario con el objetivo de redirigirlo hacia la aplicación Cliente. |
| Scope | Se trata del alcance que tendrá el Cliente. |

El "servidor de autenticación" verifica con el usuario y si este concede el acceso a sus recursos entonces el "servidor de autenticación" redirige al “agente de usuario” hacia el Cliente (aplicativo que desea hacer la *consulta*) mediante la URL proporcionada por el “agente de usuario” al "servidor de autenticación". En esta redirección, el "servidor de autenticación" adjunta en la URL un código de autenticación y ciertos datos proporcionados por el Cliente al "servidor de autenticación". Esta respuesta se conoce como la “respuesta de autorización” (Authorization Response). [36]

Mediante la utilización de este Código de Autenticación el Cliente puede hacer una nueva *consulta*, esta vez de tipo POST, conocida como “solicitud de ficha de acceso” (Access Token Request) al "servidor de autenticación". [37]. La *consulta* luce como la siguiente:

```
POST /token HTTP/1.1

Host: api.unam.com
Authorization: Basic pmMPnTEFn3S0ZmcaJQSzDzS0Z2WJ
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&code=XuqcqTGjEVVDgDX6BcXgNF&redirect_uri=https%3A%2F%2Faplicativo%2Eclien
te%2Ecom%2Fcb
```

En esta *consulta* el Cliente proporciona:

- grant_type - Tipo de mecanismo usado por el usuario para proporcionar concesión de sus recursos.
- Code - Código de Autorización obtenido del "servidor de autenticación".
- redirect_uri - URL de redirección proporcionada en mensajes anteriores.
- client_id - Identificador del Cliente.

Si el proceso ha sido exitoso entonces el “servidor de autenticación” nos responderá con un mensaje conocido como “respuesta de Access Token” (Access Token Response) con un código de éxito y con un mensaje en el que incluirá la “fichas de acceso” y opcionalmente una “ficha de renovación” (refresh token).[38]

La *respuesta* luce como la siguiente:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
  ,
    "access_token":"2LbgaSMSRwe1mPfvpZJcNN",
    "token_type":"bearer",
    "expires_in":3600,
    "refresh_token":"gTmi3WBxS0KT5Dk2GyXJVN",
  -
```

La "ficha de acceso" es la forma en que el "servidor de autenticación" nos provee del medio para obtener acceso al "servidor de recursos".

A partir de este momento por medio de la utilización de la "ficha de acceso" es posible operar a nombre del usuario final con los recursos de este en el "servidor de recursos" utilizando la API como interfaz para lograr este propósito por un tiempo limitado, En cada *consulta* se adjuntará la "ficha de acceso" en el encabezado Authentication, especificando la directiva "bearer" y con esto los recursos del usuario en ese aplicativo estarán a disposición del Cliente. [39]

La "ficha de acceso" puede ser enviada de diversas formas al "servidor de recursos":

- En el encabezado Authentication del *"*mensaje *de consulta".* [40]
- En el cuerpo del mensaje por medio del parámetro access_token en una *consulta* POST. [41]
- En la URL del *"*mensaje *de consulta"* usando el método GET mediante el parámetro access_token (se recomienda ampliamente evitar la utilización de este estilo de enviar la "ficha de acceso" ya que es propenso a ataques de evesdroping y en muchos casos se hace uso del registro de eventos en bitácoras especificando la URL del recurso solicitado). [42]

El proceso de *consulta* del recurso en el "servidor de recursos" se verá como sigue:

1. El Cliente envía una consulta al servidor intentando acceder a un recurso protegido.

2. El servidor indica que se requieren los permisos necesarios y que el Cliente debe autenticarse usando una “ficha de acceso”. Opcionalmente indica un realm en el cual la autenticación será válida, y el alcance (scope) que sirve para proporcionar etiquetas definidas semánticamente por el servidor para proporcionar al Cliente un alcance donde las acciones de la autenticación serán significativas [43]. La respuesta luce como la siguiente:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="finanzas"
```

3. El Cliente proporcionara la “ficha de acceso” en una nueva consulta al "servidor de recursos". La nueva consulta luce como la siguiente:

```
GET /resource/1 HTTP/1.1
Host: www.unam.com
Authorization: Bearer rK_6.G1k-8.3OvR
```

4. Finalmente, si la “ficha de acceso” es válido el "servidor de recursos" otorga acceso al Cliente sobre los recursos del usuario.

## 2.16 Data Loss Prevention (DLP).

Cuando hablamos de prevención de perdida de datos, podemos referirnos tanto al conjunto de técnicas que están destinadas a proveer de prevención de exposición de los datos en un ambiente, como a cada uno de los productos diseñados para este fin.

Debido a que este tipo de tecnología carece de madurez, existe una falta de consenso acerca de qué implicaciones tiene un producto DLP e incluso respecto a cuál debe ser el mejor término a utilizar (Aunque DLP parece ser el más común de todos). No obstante, lo anterior podemos aproximarnos a una definición como la siguiente:

*Productos que, basados en políticas, permiten reconocer, clasificar, monitorear, y proteger información en cualquiera de las fases de su ciclo de vida (Uso, reposo, transporte), por medio de técnicas de análisis profundo.* [44]

A este sentido se reconoce que diversos fabricantes se enfocan en diversos aspectos de la protección de la información, no necesariamente abarcando todos los puntos críticos donde la información puede ser comprometida. Algunos, por ejemplo, se especializan en el bloqueo y gestión de dispositivos USB, otros en la protección de datos en el trasporte durante la transmisión en medios no seguros, unos más se especializan en la protección de datos mediante filtrado y sanitización de mensajes de correo electrónico.

Se procederá a hacer una categorización de los mecanismos de prevención de exposición con base en los mecanismos que utilizan para el reconocimiento de la información, esto es: las técnicas de detección de datos que cada uno implementa.

**Detección basada en contexto.**

Este tipo de técnicas involucran un análisis acerca de cómo es que la información está siendo utilizada en sus diferentes fases del ciclo de vida (uso, reposo, transporte). Esto significa que existe un análisis de la fuente de la información (del usuario final que la generó en primer lugar y con qué propósito), el destino de los datos, tamaño, participantes en el uso de la misma, encabezados, metadatos, tiempo de uso, formato, etc. [45]

**Detección basada en contenido.**

Este tipo de análisis, si bien resulta más completo que el análisis contextual (y no es mutuamente excluyente a este), tiene un nivel de complejidad mayor en el análisis de los datos y depende de una utilización de recursos considerablemente alta. Esto puede resultar en una tarea extraordinariamente intensiva puesto que la información puede estar embedida en varios niveles

de encubrimiento. Herramientas modernas implementan diversos tipos de técnicas para el análisis de contenido, sin embargo, la mayoría de ellas tienen funcionalidad encapsulada que se encuentra basada en heurísticas. [46]. Entre las técnicas más populares se encuentran las siguientes:

- **Detección basada en Expresiones Regulares**. Esta puede detectar patrones comunes en que la información puede estar representada (por ejemplo, formatos de números de tarjeta de crédito, direcciones de correo electrónico, formato de claves asociadas a usuarios, etc.).

- **Detección basada en firmas (integridad de los datos)**. Usando esta técnica se verifica la exactitud de un documento, permitiendo así aplicar reglas sobre este.

- **Detección basada en contenido exacto.** Esta técnica implica preliminarmente tener una base de datos con contenidos exactos que deben ser protegidos (tarjetas de crédito, información de planes de negocio, direcciones de correo electrónico, etc.). De esta manera se puede hacer una búsqueda de los datos en diversos documentos para reconocer en donde los datos se encuentran plasmados.

- **Detección Bayesiana.** Esta técnica puede hacer uso de las anteriores, aumentando el nivel de protección ya que infiere el grado de similitud entre una pieza de información protegida y una nueva pieza de información que contiene características comunes a la primera. Esto permite reconocer contenido parcialmente similar.

- **Detección conceptual.** Usada generalmente cuando no existe contenido estructurado en documentos o bases de datos. Realiza la detección de contenido utilizando similitud en conceptos en que la información está siendo utilizada, de esta manera es posible detectar, lenguaje de odio, contenido sexual, etc. Si bien esta detección es propensa a errores, permite tener una idea general del contenido, y en conjunción con otras técnicas puede realizar la detección con mayor precisión.

## 2.17 REST APIs.

Una API expone un conjunto de datos y funcionalidades para facilitar la interacción entre programas y permitirles el intercambio de información. Resulta entonces, en una interfaz que permite que dos aplicaciones, que en principio no fueron diseñadas para interactuar, colaboren y se integren para producir trabajo en conjunto. La utilización de una API permite conjuntar funcionalidades que de otra forma requerirían un desarrollo particular. [47]

REST (Representational State Transfer) es un tipo de web API que toma las restricciones o limitaciones de uso de las arquitecturas que toma como base y aplica sus propias limitaciones. Como lo indica Roy Thomas Fielding, REST fue diseñado usando la perspectiva de diseño arquitectónico basado en restricciones que son agregadas a partir de la identificación de las necesidades del sistema para darle cohesión al mismo. De esta manera incrementalmente se añaden limitaciones al sistema que le permiten irse aproximando a la solución deseada, creando así un estilo hibrido de las tecnologías que sustentan al sistema. [48]

Es en este sentido de ideas que, partiendo de un sistema sin restricciones o limitaciones entre componentes, se agregan conceptualmente las limitaciones de otros modelos. Del modelo cliente-servidor, se toma la separación de roles que diferencia las capacidades del usuario para interactuar con un sistema y las capacidades del servidor para alojar datos. Este modelo permite proveer de escalabilidad, portabilidad de la interfaz del usuario entre diversas plataformas. [49]

Ahora bien, una característica fundamental del modelo REST consiste en la comunicación “sin estado”, de forma tal que cada una de las solicitudes hacia el servidor contengan solo la información necesaria para entender la naturaleza de la solicitud. Estas limitaciones aportan confiabilidad. Un efecto secundario es que aumenta el uso de recursos de red, ya que múltiples solicitudes deben ser enviadas para obtener datos. [50]

Aunque REST ignora los detalles de los componentes con los que es implementado y la sintaxis del protocolo, estas restricciones de uso pueden ser tomadas de la arquitectura web que define el protocolo HTTP. Ahora bien, una limitación adicional que establece la arquitectura REST consiste en la "uniformidad de la interfaz", es decir, una forma común de representar los datos. Esto permite que la información sea transferida de una forma estandarizada, que no necesariamente empata con las necesidades particulares de cada aplicación, lo cual, si bien resulta eficiente en términos de la transferencia granular de datos, provoca que las aplicaciones se tengan que adaptar a este tipo de representación. [51]

## CAPÍTULO 3. Identificación del problema y proyecto solución.

Mi experiencia en campo, me permitió reconocer que una de las necesidades primordiales de las áreas técnicas (sobre las cuales recae la tarea de administrar estas soluciones) es la obtención de información consumible por maquinas o por personal humano. Esta demanda de las áreas técnicas proviene de la necesidad de presentar resultados de interpretación fácil y ágil a las áreas gerenciales para así justificar los costes de adquisición de tecnologías en la definición de una arquitectura en particular.

Es a tal efecto que para el personal técnico que opera esta herramienta resulta fundamental la automatización en la generación de reportes.

Por otro lado, cabe destacar que en la industria se carece de personal altamente capacitado para llevar a cabo la totalidad de las funciones que desempeña, lo que se traduce en tiempos de demora e improductividad en los departamentos de tecnología. La carga de trabajo suele ser excesiva para el número de personas destinadas a cubrirla. En muchos casos se depende parcial o totalmente de consultores externos que mantienen las actividades de estos departamentos. Esto representa condiciones iniciales desfavorables, ya que la venta del producto DLP depende en gran medida de la facilidad de implementación del mismo. Se debe procurar entonces, que, incluso ante las adversidades comunes en los departamentos de tecnología, el despliegue y la gestión de la herramienta pueda llevarse a cabo con la menor complejidad.

Si bien es cierto que la herramienta DLP provee la posibilidad de exportar bitácoras del sistema para poder visualizarlas por medio de herramientas externas, esta capacidad requiere que el técnico operador posea los conocimientos en herramientas de visualización (SIEM) necesarios para llevar a cabo esta tarea. Ante la problemática que representa para muchas empresas la carga técnica y el costo de adquisición de herramientas de visualización (SIEM) se produce el efecto de reducir el tamaño del mercado objetivo, puesto que un comprador promedio se ve disuadido de adquirir el producto. Esto significa que el mercado objetivo se ve limitado a aquellos compradores que:

- Ya cuentan con una herramienta de visualización de información (SIEM).
- Tienen personal capacitado en la tarea de operar esta solución.
- Tienen la necesidad de aplicar protección por medio del cifrado a sus máquinas.

Estas características del producto representan para la compañía en la que trabajé una problemática mayor puesto que se cuentan con acuerdos comerciales con el fabricante de la solución para la venta de este producto para un periodo amplio de tiempo.

Sin embargo, las características antes mencionadas hacen que la venta del mismo resulte en algunos casos infructífera.

Referente al fabricante que desarrolla los productos, este desarrolló originalmente un producto "gestor de configuraciones del sistema". El producto DLP constituye una extensión del "gestor de configuraciones del sistema".

Así, un cliente que decidiese pagar un costo adicional podría contar con la conjugación de ambas funcionalidades y de esta manera omitir la compra de otro producto especializado en DLP de

algún otro fabricante que tendría un costo mayor. Ambas herramientas utilizan un agente instalable diferente por lo que complementan su información en la consola principal del producto "gestor de configuraciones del sistema" (portal de administración con interfaz web). Sin embargo, en la práctica siguen operando como productos separados, simplemente configurables desde la consola web del producto "gestor de configuraciones del sistema".

Con fundamento en lo anterior detecté la necesidad de proveer a clientes y socios de negocio que operan servicios administrados de la capacidad de generar reportes derivados de la utilización de la herramienta DLP que forma parte del catálogo de productos.

Esta necesidad frecuentemente solicitada por los equipos técnicos viene dada por la carencia de la herramienta DLP de contar nativamente con una funcionalidad para la presentación ordenada de la información que permita visualizar la configuración aplicada y los resultados de las aplicaciones de las configuraciones.

### 3.1 Factores a contemplar en el diseño de la solución.

Para cubrir la necesidad de proveer al personal del departamento de tecnología de los clientes que administra la herramienta de la capacidad de generar reportes y visualización de configuraciones propuse un sistema de generación de reportes.

Este sistema permite visualizar de forma sintética los datos extraídos en la plataforma por medio de tablas que reflejan información acerca de cómo los documentos son utilizados por los usuarios, así como diversas métricas del funcionamiento de la herramienta. Permite además configurar parámetros que ajusten la funcionalidad a las necesidades particulares de los clientes.

Se tiene identificado que la mayoría de los clientes tienen instalado el sistema operativo Windows por lo que se propone que la solución sea orientada a cubrir la necesidad de esos usuarios administrativos de la solución.

Para los fines planteados se requiere de una plataforma que permita la rápida edición y exportación de los reportes por lo que se prefirió usar software de hojas de cálculo como aplicativo final de trabajo para el administrador de la plataforma. Se requiere que los reportes

puedan ser obtenidos periódicamente pero que también puedan representar una instantánea de lo que ocurre en la plataforma al momento de ejecutar el desarrollo. Esto permite generar un compendio histórico que además posibilite a los administradores para presentar resultados a las áreas gerenciales.

Se requiere flexibilidad del desarrollo para ser escalado, implementado en nuevos desarrollos de forma transparente y a su vez dar la posibilidad a los usuarios administrativos puedan incorporar la solución como elemento constructivo de aplicaciones más complejas que permitan cubrir más específicamente los requerimientos de uso de cada cliente.

Igualmente, a la generación de reportes la herramienta permite la configuración de la plataforma de forma simple. Esto es ventajoso porque de esta forma el usuario administrativo no depende de dos interfaces diferentes (la consola original desarrollada por el fabricante que carece de opciones de visualización, y el desarrollo aquí planteado). Para la gestión del producto DLP, sino por el contrario poder depender exclusivamente del desarrollo para interactuar con la herramienta.

Debido a las premisas anteriores emití como recomendación técnica la utilización de Powershell como lenguaje para el desarrollo del sistema de reportes, debido a las ventajas que se enuncian a continuación:

- Powershell permite hacer *consultas* de tipo HTTP por medio de sus cmdlets.
- Powershell corre sobre el ambiente de desarrollo .NET con lo cual tiene acceso al sistema de objetos de Windows, esto significa que nativamente tiene integración con diversos componentes del ecosistema .NET, entre ellos los objetos de Microsoft Excel.
- Powershell permite la creación de interfaces graficas de usuario para una mayor facilidad con la interacción con la herramienta, aunque este punto se deja para fases posteriores, tuve que contemplarlo para un correcto planeamiento de la escalación, en caso de se decida expandir las funcionalidades de la solución.

- Powershell ofrece flexibilidad y encapsulamiento, derivados de su orientación a objetos. Esto otorga la posibilidad de una correcta escalabilidad del proyecto, así como fácil integración a desarrollos externos.

Cabe destacar que la información que se extrae de la API consiste en documentos en formato JSON, predefinidos en su estructura por la documentación proporcionada por el fabricante del producto DLP. Lo anterior permite que los datos puedan ser procesados para después ser sintetizados en forma de tablas[7].

## 3.2 Metodología de Trabajo.

Partí de la identificación del problema. La identificación representa una fase esencial del ciclo de vida del desarrollo, ya que, la pronta identificación de obstáculos en el desarrollo de la solución permite anticipar acciones y producir un ahorro de tiempo que es mejor aprovechado al iterar sobre el ciclo de producción.

Detecté las siguientes condiciones:

- Poco tiempo neto para el desarrollo.
- Número reducido de personas responsables de producir y validar desarrollo.
- Reuniones periódicas para afinar requerimientos, resolver dudas técnicas acerca de la utilización del producto y recibir apoyo y revisión por parte del equipo técnico del fabricante del producto DLP.

[7] Es importante mencionar esto para evitar la tentación de asumir que al hablar en lo sucesivo de tablas se hace referencia a un sistema de base de datos tradicional, donde por medio de un lenguaje de *consultas* SQL se realizan *consultas* que resultan en la generación de tablas. Este no es el caso, por lo que la extracción de información de estos documentos JSON se realiza por medio de técnicas de parseo.

De esta manera se llegó a la conclusión que el mejor mecanismo de trabajo consistía en la generación de un esquema por el cual se tuvieran ciclos periódicos de iteración en los cuales hubiera cinco fases repetitivas que sirvieron para analizar el avance realizado, detectar tempranamente dificultades, atender la necesidad de replantear estrategias y recibir retroalimentación de los actores interesados.

Una vez que los ciclos iterativos condujeron a la aprobación del desarrollo como producto mínimo viable se pudo llevar a cabo un escape del ciclo que permitirá el despliegue del desarrollo. La Figura 8 ilustra este proceso.

Se planteó que cada iteración tuviera un tiempo de 15 días (1 quincena) debido al acoplamiento con la agenda de los supervisores del desarrollo, coincidiendo con una junta entre el fabricante del producto y el área gerencial de mi compañía. De esta manera inicialmente se plantearon 8 ciclos iterativos en el proceso de desarrollo.

Por motivos de carga de trabajo en otros frentes y debido a las adecuaciones y adiciones efectuadas en el proyecto, se terminó con el desarrollo en un total de 13 iteraciones, sin incluir el tiempo dedicado a la implementación, pruebas y monitoreo inicial del desarrollo con los clientes. Esto significa que la fase de desarrollo duró 26 semanas.

No debe aquí confundirse este tiempo de desarrollo con el tiempo total del proyecto puesto que a esta cantidad de semanas deben incluirse aquellas de pre planeación donde se llevaron a cabo entrevistas con clientes para entender mejor la problemática, proceso en el cual llevé a cabo la investigación necesaria para permitirme conformar una base teórica suficiente para llevar a cabo este proyecto.

Finalmente, no debe dejar de considerarse que una vez listo el desarrollo siguieron fases donde llevé a cabo diversas labores de implementación, monitoreo y seguimiento.


Figura 8. Metodología de trabajo.

Referente al diseño de la solución, se hace pensando en la utilización que se le dará y la resolución de las problemáticas antes discutidas, teniendo en cuenta el público objetivo de tal desarrollo. A este sentido se puede destacar que diseñé el desarrollo que aquí presento para utilizarse en sistemas Windows, así mismo detecte que el desarrollo estará destinado a personal que posee conocimientos mínimos de Powershell.

Respecto a la fase de desarrollo, es en esta en la que produje resultados en la conformación de la solución de forma acelerada con un periodo de tiempo muy reducido, esto atendiendo a criterios relacionados con la intención de la compañía para realizar la pronta implementación para algunos clientes.

Es en esta fase donde realice el código que conformaría la solución y realicé pruebas unitarias de los diversos componentes de código con los que cuenta la solución.

En la fase de Pruebas se obtiene la información a partir de datos precargados en la consola del producto DLP por medio de la configuración de políticas y la creación de grupos de equipos y usuarios a los cuales les asignaban tales políticas.

Es en esta misma fase de pruebas donde se detectaron algunas áreas de oportunidad, tal es el caso que el proyecto original contemplaba la creación de un conjunto reducido de tablas, mientras que el proyecto una vez aplicadas las modificaciones derivadas del proceso de iteración y retroalimentación, terminó teniendo un aumento en el número de tablas a presentar.

Es en la fase de presentación de resultados donde se auditaba el avance efectuado en el proyecto, se planteaban problemas afrontados en el desarrollo del mismo, en ocasiones se planteaban dudas al fabricante del producto y se detectaban áreas de oportunidad con respecto a funcionalidades que se ofrecerían a través del proyecto. Así mismo se resolvieron problemáticas que surgieron durante el desarrollo derivadas de insuficiencia en los permisos de usuario que tenía mi rol de usuario en la plataforma.

Se decidió agregar la funcionalidad que ya se encuentra en la consola web de administración de la solución DLP. Esto es la capacidad por parte del administrador de la solución de aprobar las solicitudes de los usuarios finales hacen para guardar archivos en estado no cifrado y la capacidad de configurar ciertas funciones de la solución.

Estas características del desarrollo no estaban incluidas en la propuesta inicial, sin embargo, se agregaron para evitar que el usuario administrador de la solución dependa de dos plataformas diferentes para interactuar con el mismo producto.

Cabe destacar que, para este proyecto en particular, si bien se establecieron rangos de fechas para la entrega y revisión, su cumplimiento no fue requerido de forma estricta, ya que los supervisores del proyecto desempeñaron otras labores propias de su área de competencia de forma simultánea mientras que el desarrollo del proyecto tenía lugar. Este desarrollo en particular se puede categorizar como un desarrollo importante pero no urgente, como sí lo fueron otras labores del deber diario de mi puesto de trabajo.

Aun cuando el personal de los canales socios que comercializan el producto, así como el área gerencial de mi compañía entendían la importancia del producto, se llegó a la conclusión de que era inviable tener un recurso humano dedicado exclusivamente al desarrollo del mismo, por lo

cual llegaron a la conclusión de aprobar el desarrollo de la solución "Tras bambalinas" en tiempos muertos de la operación normal de las tareas que forman parte de mi esfera de competencias.

Al término de los ciclos se planteó si se hacía una ampliación en el periodo de tiempo destinado al desarrollo de la solución aquí presentada o si esta estaba lista para pasar a la fase de implementación.

### 3.3 Propuesta de solución.

Debido a las características de la problemática y a los elementos a disposición con los que contaba llegué a la conclusión de usar la funcionalidad de la API REST del producto DLP, para así extraer datos por medio de consultas HTTPS. De esta forma el desarrollo consultaría información que se encuentra guardada en un servidor dedicado para agrupar la información y generar reportes. Esto permitiría explotar a mayor profundidad la funcionalidad del producto DLP, subsanando algunas carencias en la presentación de resultados del funcionamiento de la misma.

Respecto a las *consultas* HTTPS, estas responden a la necesidad planteada en un caso de uso presentado en juntas tempranas de la fase de retroalimentación. Este planteamiento pide obtener información específicamente de los usuarios, las aplicaciones usadas por estos, y las reglas aplicables a los grupos de usuarios, de forma tal que se abarca cada una de las funcionalidades requeridas mediante un diseño de código modular escalable.

Para poder realizar las consultas a la API REST es necesario contar un identificador del usuario que se genera en la plataforma web del producto DLP. Este identificador está ligado a los permisos de rol asignado al usuario administrativo que lo genera. Junto con el identificador se genera una contraseña que está asociada al mismo.

Usando este identificador y la contraseña generada para el mismo en la interfaz web es posible realizar una *consulta* inicial a un punto de entrada (URL), haciendo una *consulta* inicial que devuelve una "ficha de acceso" que se utilizara en las futuras *consultas* HTTP a forma de autenticación. Esta "ficha de acceso" expira por diseño del sistema 10 minutos después de la

generación del mismo. La ficha se envía en el encabezado “Authorization: Bearer <token>” del "mensaje de *respuesta*" por parte del servidor.

En cada *consulta* se generan estructuras con objetos representados en formato JSON, donde cada objeto es un elemento devuelto en la *consulta* HTTP. Los arreglos objetos son entonces enviados a las funciones de procesamiento del presente desarrollo, donde los atributos de cada uno de ellos son analizados para generar una agrupación de datos que permite procesar la información y generar los renglones de cada una de las tablas que a su vez conformarán el reporte final. La Figura 9 ilustra el diagrama a bloques de la solución, más detallado que el mostrado en la Figura 2.

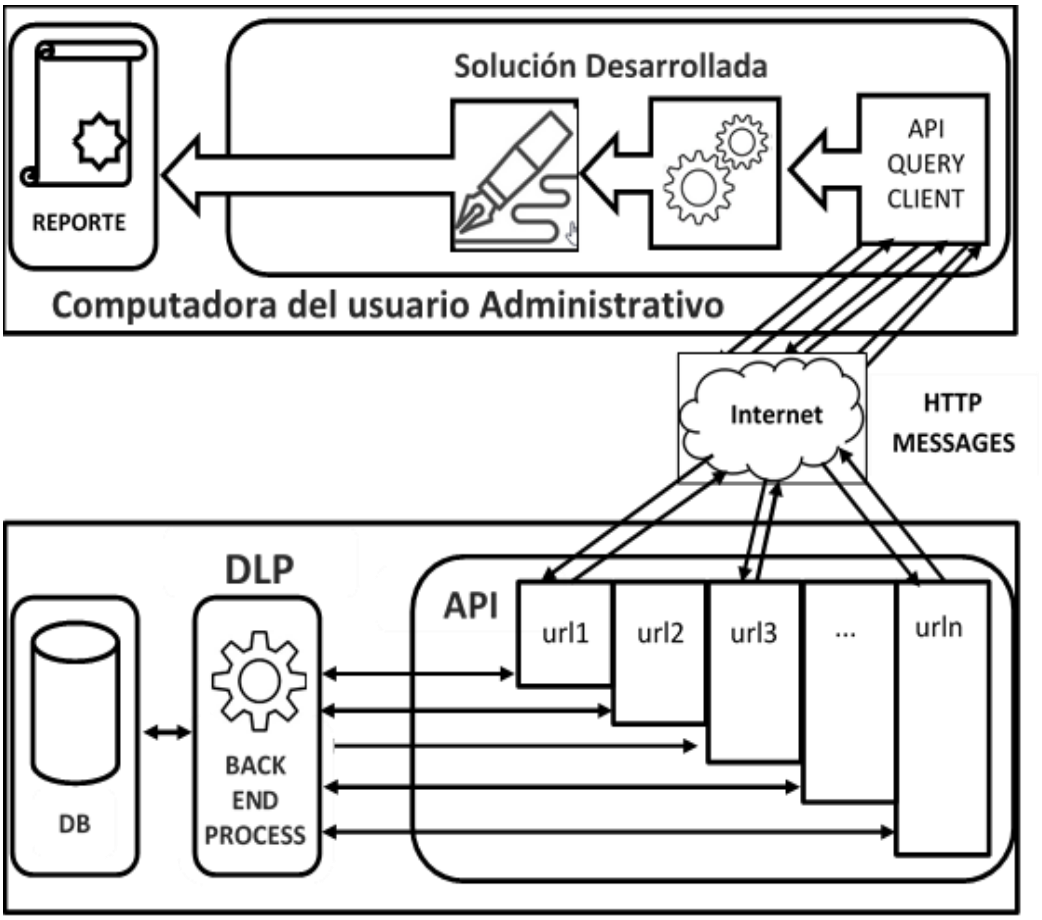


Figura 9. Partes operativas de la solución.

De esta forma el proceso es el siguiente:

1. Se solicita la "ficha de acceso" para usarlo en futuras *consultas*.
2. Se realizan las *consultas* a diversos puntos de entrada (URLs) de la API usando en cada una de ellas la "ficha de acceso" como método de autenticación.
3. Se crean objetos de PowerShell para albergar los resultados de las *consultas* y estos son albergados en arreglos para su posterior tratamiento.
4. Las estructuras con objetos son procesadas y se obtiene una agrupación de datos en forma de tablas.
5. Se envía la agrupación de datos, junto con las gráficas correspondientes al reporte final en un documento de hoja de cálculo.

### 3.4 Partes funcionales de la solución.

El sistema de *consultas* se puede dividir en tres partes funcionales, que son mostradas en la Figura 10:

- Aquella que se encarga de recopilar la información del servidor dedicado mediante la generación de "mensajes de *consulta*" HTTP dirigidos a la API REST del producto DLP.
- Aquella que se enfoca la categorización, procesamiento de la misma para obtener relaciones significativas.
- Aquella que se encarga de la generación del reporte según las especificaciones y con los cambios derivados del proceso de pruebas y revisión.

Cada una de estas partes funcionales está compuesta por diversas funciones que trabajan en conjunto para producir el resultado final.

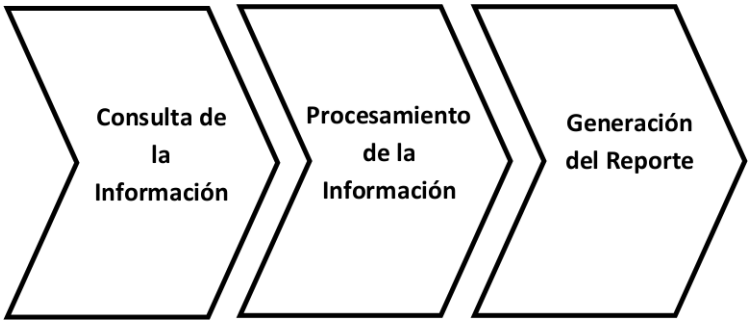


Figura 10. Partes funcionales de la solución.

### 3.5 Consulta de la información.

Respecto a la parte funcional que se encarga de recopilar información, esta consta de múltiples funciones, cada una haciendo una *consulta* en particular a diversos puntos de entrada (URLs) de la API, según las acciones definidas por el desarrollo aquí presentado. Es esta parte funcional que recupera los recursos JSON generados por el servidor, derivados del uso y funcionamiento del producto DLP. Estos documentos son recuperados y almacenados en memoria en forma de objetos.

### 3.6 Procesamiento de la información.

Por otro lado, se tiene la parte funcional dedicada al procesamiento de la información, esta área se toman diversos objetos que se encuentran en memoria generados por las *consultas* HTTPS realizadas en PowerShell a la API REST del producto DLP. Se procesa la información conjuntando datos que servirán para la generación de tablas que permitirán la agrupación de la información. Esto puede lograrse mediante la manipulación de la información para extraer los datos contenidos en atributos de los objetos. Cabe destacar que en este punto del proceso ya no se manipula documentos JSON obtenidos en la fase anterior, si no que estos tienen una representación en forma de objetos en PowerShell.

La información que se obtiene está definida por el fabricante puesto que la documentación del producto permite saber el tipo de datos y el orden en que estos son recuperados. De entre la información que es posible ser recuperada se encuentran:

- Datos asociados a los usuarios: nombre, departamento (finanzas, RR.HH., ventas, etc.), nivel de confianza, atributos especiales, así como otros.
- De los grupos de usuarios se tiene: nombre, usuarios asignados, política de seguridad de uso, política de seguridad de extracción, excepciones, etc.
- De los grupos de información se tiene: categorización de seguridad (critico, muy alto, alto, destacable, medio, preferente, bajo, y categorías definibles por el administrador), permisos asociados por default, etc.
- De la información puede *consultar*: registro de cambios, marcas de tiempo, registro de geolocalización del acceso de los datos, registro de solicitudes de aprobación de extracción no cifrada, etc.
- De los dispositivos, se puede consultar: un identificador, nombre de host, lista de puertos USB, etc.
- De los dispositivos externos, se puede consultar: marcas de tiempo, un identificador, fabricante, puertos USB usados, cantidad de datos transferidos, archivos transferidos.
- De los administradores se puede consultar: nombre, perfil, permisos, etc.
- De los registros de auditoria se puede consultar: actividad relacionada a administradores en la plataforma, registro de generación y modificación de llaves para utilizar la API, etc.
- De las políticas se puede *consultar*: periodo de gracia, permisos asociados a los datos, cuotas de modificación, dispositivos permitidos, umbral de similitud, políticas de expresiones regulares, política de bloqueo basado en URL, opciones de limpieza, etc.

- De los registros de uso se puede *consultar*: usuarios que utilizaron la información, marcas de tiempo, porcentaje de cambios, funciones de extractado (md5, sha256), etc.

Estos datos se obtienen a través de distintas *consultas* HTTP a diversas URLs que forman parte de la API y que permiten obtener esta información.

Aunque las *consultas* por sí mismas aportan información valiosa del uso de los documentos y por tanto del funcionamiento de la herramienta, se carece de la capacidad de hacer *consultas* complejas para establecer hechos que trasciendan en un entendimiento correlacionado de los datos anteriores. Esto significa que los usuarios de la API deben hacer estas asociaciones de forma manual, cotejando diversos datos otorgados por la plataforma.

Ante la ineficiencia que esto representa para los usuarios administrativos contar con esta información careciendo de un sistema de gestión de información que les permita la representación gráfica de datos, se pierde la capacidad automatizada de obtener correlaciones significativas en forma de tablas que respondan a diversas preguntas como las siguientes:

- ¿Cuál fue el grado de participación que cada usuario tuvo en la conformación de un documento?
- ¿Cuáles documentos fueron sujetos a cambios durante el mes que se reporta?
- ¿Qué administradores autorizaron liberar documentos cifrados en el departamento de mercadeo en el mes de febrero?

### 3.7 Generación del reporte.

Por último, se tiene la parte funcional dedicada a la generación del reporte que mediante el framework .Net genera objetos que representan hojas de cálculo a partir de los datos procesados en el punto anterior. Se produce una gestión separada de cada hoja de cálculo de Excel, que a su vez permite conformar la agrupación de información, y la correspondiente selección de datos para ejecutar el proceso de graficación de los mismos.

Para esta primera etapa se tiene contemplado que el desarrollo pueda ser ejecutado por medio de la línea de comandos de Powershell, de forma tal que la sola tenencia del desarrollo, y las claves de funcionamiento correspondientes permitan a los operadores de la solución DLP obtener los reportes generados por la solución que aquí se presenta.

Procediendo de esta forma se obtiene la ventaja adicional de que por medio de herramientas nativas del sistema operativo los operadores de la solución aquí presentada, pueden obtener los reportes no solo bajo demanda, según sus necesidades lo dicten, sino que además la solución puede ser integrada en desarrollos de automatización producidos por los administradores, que automaticen bajo demanda la adquisición y el procesamiento de los reportes.

No obstante, lo anterior en una junta de retroalimentación se planteó la necesidad de tener contemplado para una segunda etapa donde se proveerá de interfaz gráfica al desarrollo para ser configurado de forma gráfica y de esta manera que la interacción del usuario se pueda llevar a cabo de forma simplificada. Esta segunda etapa excede los límites de este trabajo. Para fines de este desarrollo la utilización por medio de línea de comandos cumple con el objetivo planteado.

### 3.8 Proceso de utilización.

Primeramente, el usuario ejecuta la solución proporcionando algunos parámetros destinados a configurar el rango de fechas, el tipo de reporte y el nombre del archivo de reporte que se obtendrá al finalizar la ejecución de la solución.

Con esto obtiene un reporte generado para describir los usuarios de la plataforma y las características de los mismos. La Figura 11 muestra la ejecución de un reporte para un periodo definido. La Figura 12 muestra el resultado de la ejecución de la Figura 11.

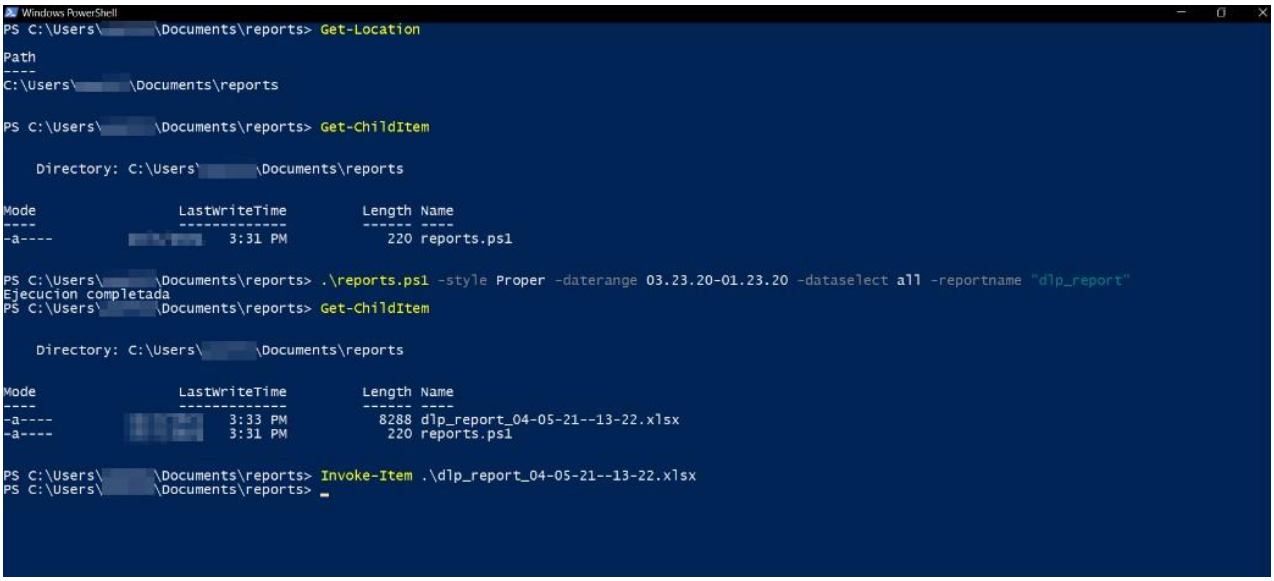


Figura 11. Ejecutando la solución.


Figura 12. Uno de los reportes generado.

Así mismo obtiene en una hoja a parte métricas del funcionamiento de la herramienta y reportes gráficos que ayudan visualizar las métricas derivadas del uso del producto DLP. La Figura 13 muestra el reporte de métricas de funcionamiento.

El sistema de reporte permite establecer parámetros que cambian el aspecto final del reporte según las necesidades del mismo.

Una vez concluida la generación del reporte con las características que al usuario convienen, el usuario administrativo puede continuar con su flujo normal de trabajo con el reporte al poder editarlo en un aplicativo de hoja de cálculo. De esta forma el usuario administrativo también pude exportar los datos a otros formatos que pueden ser usados para presentar a las áreas gerenciales o para conformar una instantánea del estado del producto.

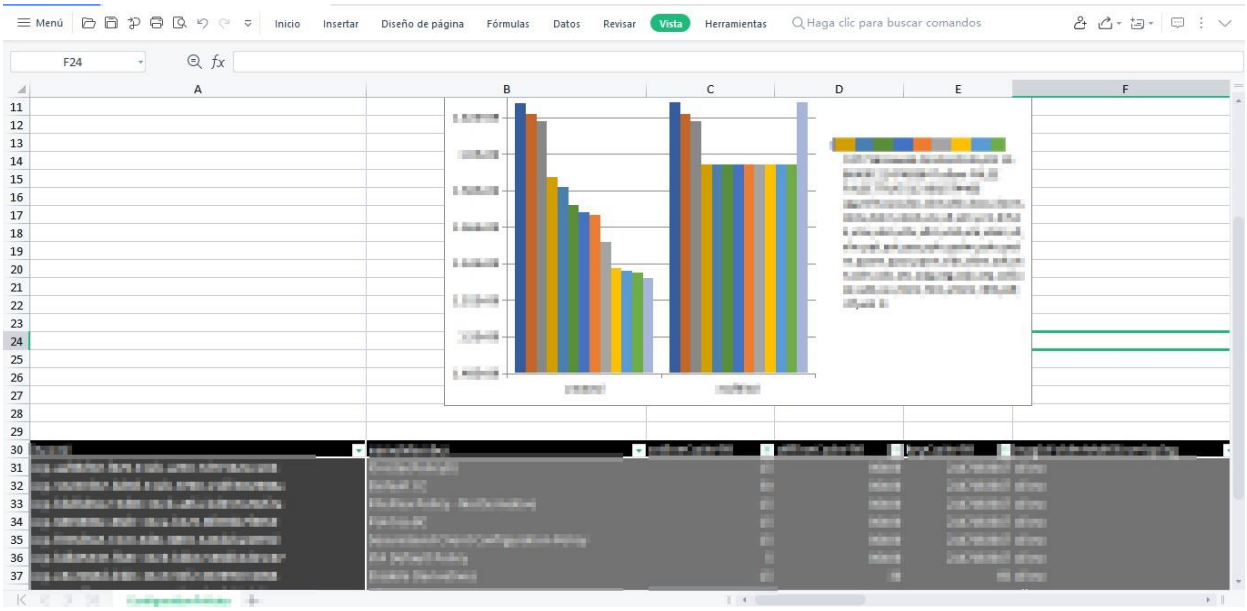


Figura 13. Otro reporte generado.

## CAPITULO 4. Resultados y Conclusiones.

Este proyecto como uno de los primeros desarrollos de mi vida profesional resulta muy gratificante en diversos aspectos. Primeramente, por el reto técnico de entregar a tiempo un proyecto de estas características con un alto nivel de exigencia en el desarrollo del mismo. Existe además el reto interpersonal por el cual los requerimientos y la obtención de resultados se torna en uno de los temas principales y más difíciles de ejecutar.

De entre los principales resultados en el poco tiempo transcurrido desde la implementación, se tienen los que se tratan a continuación.

### 4.1 Resultados asociados a la venta del producto.

Referente a las ventas del producto, los cierres de venta tuvieron un aumento significativo del 21%, después de realizar implementación es a prospectos que buscan herramientas que solucionen la problemática para la cual el producto DLP fue diseñado. Esto debido a que muchos clientes deciden llevar a cabo el proceso de compra por medio de la emisión de RFP's (request for proposals) que son documentos donde exponen una rúbrica de puntos que los productos que están dispuestos a adquirir deben tener.

Gracias a este desarrollo es posible cumplir con los requerimientos asociados al reporteo de la solución en estos RFP's, permitiendo que la labor de venta no sea descartada desde la etapa temprana de respuesta a los requerimientos.

Se lograron concretar al menos 3 ventas que estaban en peligro debido a (entre otras cosas) al cumplimiento técnico de requerimientos relacionados al reporteo planteados por los clientes en sus RFP's (request for proposals).

Es temprano aún para contarlo, pero se prevé que exista un porcentaje de renovación mayor al obtenido en años anteriores, dicho de otra forma, se prevé una disminución en la fuga de cartera.

## 4.2 Resultados asociados a la operación del área de soporte.

En aquellos clientes que ya cuentan con el desarrollo y que auto gestionan sus herramientas de seguridad:

Estos clientes no solo mejoraron su flujo de trabajo al generar reportes que anteriormente tenían que elaborar de forma manual, sino que además esto les permite optimizar su productividad ya que ahora se ahorran tiempo que pueden dedicar a otras actividades.

Se redujeron las solicitudes de servicio relacionadas a problemáticas de la herramienta, si bien muchas de las solicitudes abordaron aspectos no directamente relacionados con el reporteo de la herramienta. La mayoría provenían de la problemática del usuario de no poder obtener datos concretos en la misma.

De esta forma un usuario podía por ejemplo, solicitar ayuda referente a problemas con la aplicación de ciertas reglas, sin tener elementos para verificar que los cambios hubieran sido correctamente aplicados, gracias al sistema de reportes se puede capturar esta información y el usuario puede consultar el estado del sistema. De esta manera evita hacer uso del servicio de soporte técnico.

La Figura 14 ilustra la cantidad de solicitudes de servicio recibidas los días hábiles posteriores a la implementación. En ella se puede observar un mes posterior a la puesta en marcha del desarrollo. Se observa una variación en la cantidad de solicitudes recibidas en cada día. Esta variación es atribuible a que no todos los clientes pudieron agendar con nosotros el mismo día para llevar a cabo la transferencia del desarrollo y la consecutiva explicación en el uso de la misma.

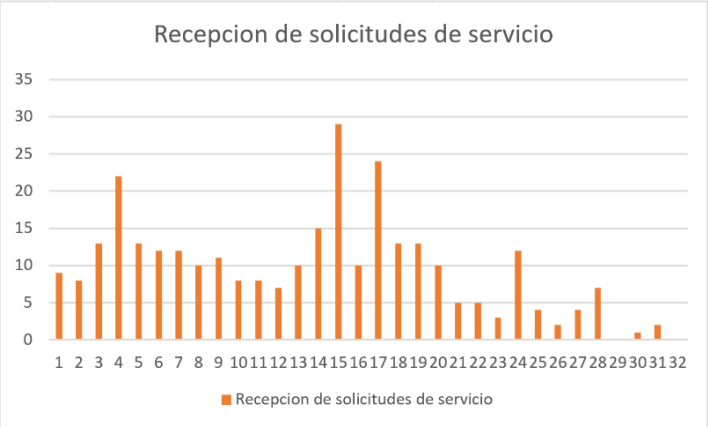


Figura 14. Monitoreo de solicitudes de servicio recibidas en los días posteriores a la implementación de la solución.

### 4.3 Resultados asociados a los operadores técnicos del producto DLP.

En aquellos operadores que proporcionan servicios administrados:

Para los usuarios administradores, se otorga la posibilidad de agendar los reportes para proveerlos a las áreas gerenciales a las que proporcionan servicio. Se tiene además la ventaja de que los administradores pueden integrar esta solución en sus propios desarrollos. Esto les ofrece la ventaja de conjuntar las funcionalidades de este desarrollo con otras herramientas de su elección.

## 4.4 Resultados en el impacto de marca de la compañía.

Referente al valor de marca de la compañía en la que laboré, esta se posiciona como un referente en la industria, consiguiendo una revalorización de su reputación en un mercado competido. Se puede destacar lo siguiente:

1. Se reafirma la posición como distribuidor de valor agregado de la compañía donde laboré. Se obtiene así la simpatía de los fabricantes para preferirnos como medio de distribución de soluciones.
2. Se obtiene la preferencia de los clientes, que ahora entienden la necesidad de contar con un intermediario que provea de valor en el diseño a medida de soluciones. Esto proporciona a la marca de una ventaja competitiva respecto a los demás distribuidores en el mercado.
3. El presente proyecto por características de diseño puede ser ampliado, ya que fue concebido teniendo en mente ser integrado en un posterior desarrollo que conjunte distintas funcionalidades de distintas herramientas en el catálogo de productos, de forma tal que se permita la gestión centralizada y automatizada de las labores de los equipos de seguridad en un único desarrollo integrado.

## 4.5 Habilidades y conocimientos aplicados.

En el desarrollo de este proyecto se vieron manifiestas diversas habilidades obtenidas a lo largo de mi formación profesional en la Facultad de Ingeniería. Estas habilidades fueron indispensables para permitirme conducir un proyecto como este. Entre las más destacadas se encuentran las siguientes:

Habilidades propias de la programación aprendidas en la materia de **Computación para Ingenieros**, gracias a los cuales apliqué la lógica y la estructura como herramientas primordiales en el diseño de la arquitectura de la solución. Así mismo es en esta materia donde aprendí el uso de pruebas unitarias que ayudaron a la verificación del funcionamiento de diversos componentes de software antes de ser integrados en la versión final.

Conocimiento y manejo del paradigma orientado a objetos adquirido en la materia de **Programación Avanzada y Métodos Numéricos**, en el desarrollo aquí planteado, estas habilidades fueron utilizadas en el mismo código que compone la solución.

Aquellos saberes propios de la materia de **Algoritmos y Estructuras de Datos** fueron esenciales para el manejo de estructuras de arreglos, ampliamente usadas en el desarrollo de la solución.

De la materia de **Lenguajes Formales y Autómatas** se obtuvieron conocimientos relacionados a gramáticas regulares en la implementación de expresiones regulares que sirvieron para extraer algunas piezas de información que se encontraron embebidas en diversos formatos.

Las materias de **Ingeniería de Software** y **Administración de Proyectos de Software** me permitieron poder planear, coordinar y llevar a cabo la gestión del proyecto de forma tal que este desarrollo se llevó a cabo de manera estructurada, respetando los objetivos iniciales.

La asignatura de **Redes de Datos** me otorgó los conocimientos mínimos necesarios para el entendimiento del modelo cliente-servidor, así como el entendimiento de los protocolos subyacentes en el funcionamiento del desarrollo.

Respecto a las habilidades blandas que fueron de gran ayuda en el proyecto se destacan aquellas aprendidas en las materias de **Ética Profesional** y **Cultura y Comunicación**.

## 4.6 Conclusiones.

Es en el campo de la aplicación de conocimiento relacionado a la resolución de problemas, se aprecia la vocación que los desarrollos tecnológicos proporcionan al objetivo de ahorrar de recursos, como lo son el tiempo, los esfuerzos, así como su utilidad en el cumplimiento de metas. Es en este sentido que adquiere importancia el profundo conocimiento del problema a resolver para una adecuada planificación que explore soluciones que faciliten las labores dentro de las organizaciones.

Resulta importante la detección de áreas de oportunidad donde por medio de tecnología se pueda proveer de los mecanismos para la mejora de procesos, el diseño de arquitectura y la adecuación de la oferta de soluciones a las necesidades del cliente y del proveedor de servicios.

Referente a la experiencia adquirida en este proyecto se reconoce que la planeación de tiempos para casa fase del desarrollo, así como la correcta planeación de la carga de trabajo que cada una de estas fases puede requerir, resulta tan importante, que en esta primera aproximación al problema no debe escatimarse en esfuerzos. Es por tanto una de las fases clave en todo proyecto, ya que el ajuste y adecuación del código en etapas posteriores representa un obstáculo que irrumpe en el flujo de trabajo del desarrollo, ralentizando los cronogramas y demandando esfuerzos adicionales por parte de los involucrados. Esto es más notorio en entornos no ideales o controlados, puesto que tales condiciones demandan una porción de atención considerable que con frecuencia no se tiene en consideración en etapas tempranas de la planeación.

En diversas fases de rediseño se presenta que un caso de uso debe estar ligado a un grado de abstracción tal que, este pueda ser dividido a sus partes fundamentales. Es decir, un caso de uso generalmente podría ser dividido en subcasos de forma recursiva hasta que se llegue a un punto donde todos los subcasos resulten en unidades autónomas que interactúen para resolver el caso de uso principal del cual tienen origen.

Se puede considerar todo un éxito la creación de este sistema, no solo por haber cumplido el objetivo planteado, sino porque establece un precedente en el actuar de la compañía con respecto al desarrollo de soluciones. Este precedente logra generar cambio en la cultura de trabajo y en la visualización de prioridades que ayudarán a definir una correcta canalización de los esfuerzos en la generación de código propio que permita atender la amplia cartera de clientes que demandan de soluciones diseñadas a medida.

## Anexo 1 - Glosario.

Ficha de acceso (Access Token)

Una pieza de información que, en un proceso de comunicación, es usado por una de las partes, para autenticarse frente otra.

ASCII

Se trata de un conjunto de caracteres diseñado para representar 128 caracteres, incluyendo así los caracteres latinos, números, símbolos, y otros.

Bajo demanda

Modelo por el cual los recursos son solicitados hasta que el usuario los requiere, este los puede solicitar en el momento en que considere oportuno.

Banderas

Se trata de modificadores del comportamiento de un comando que se ejecuta en una línea de comandos. Estas banderas son también conocidas como flags.

Checksum

Mecanismo de digestión de una secuencia de octetos que permite verificar si existen errores en la transmisión de esa secuencia de octetos.

CMDLETS

Se trata de comandos que en realidad resultan ser clases del framework .NET CORE que ejecutan ciertas operaciones especificas asociadas a cada uno de ellos.

Cliente-Servidor

Este modelo define la comunicación entre uno o varios clientes que hacen utilización de servicios almacenados en un servidor que proporciona estos servicios a los diferentes clientes.

Cloud Computing

A grandes rasgos este concepto hace referencia la proveeduría de infraestructura de servicios a diversos grados de abstracción que permite al usuario de forma remota acceder a los servicios ofertados por una organización de forma transparente y distribuida. Así mismo proporciona a las organizaciones la capacidad de digitalizar su operación o extraerla de los perímetros tradicionales, dando la posibilidad de escalar arquitecturas a precios razonables.

Codificación

Se trata de procedimiento que convierte caracteres de un lenguaje natural en caracteres o símbolos de otro sistema de representación, estableciendo una correlación entre ambos símbolos.

Directivas

Se trata de elementos que permiten adjuntar información útil en un encabezado del protocolo HTTP.

Evesdroping

Consiste en la aplicación de técnicas que utiliza un tercero no autorizado para escuchar en secreto comunicaciones de otros sin su consentimiento.

Framework

Se trata de un marco de trabajo que agrupa llamadas a sistema, *consultas*, funciones y demás elementos que proveen funcionalidades que se agrupan en una metodología en particular de trabajo.

Gartner

Gartner es una empresa que mediante investigación provee consultoría acerca de soluciones en el mercado. Es conocida por loa emisión de los cuadrantes de mágicos de Gartner donde emite su calificación de las soluciones tomando como base dos ejes: Visión de una organización o producto y capacidad de ejecutar esa visión. De esta manera genera cuatro cuadrantes donde categoriza a

las soluciones, siendo los cuadrantes los siguientes: lideres (I), visionarios (II), jugadores de nicho (III) y retadores (IV).

Gateways

Se trata de un dispositivo que interconecta redes, permitiendo traducir la información de un protocolo usado en una de las redes al protocolo usado en la otra.

Hoja de cálculo

Tecnología que sirve para llevar un control de datos ordenados, con el objetivo de poder obtener relaciones significativas y poder dar un tratamiento a los mismos.

HTTP

Se trata de un protocolo diseñado para la recuperación de información, así como el envío de la misma hacia un servidor para proveer servicios a un cliente. Este protocolo tiene la particularidad que no guarda el estado de las *consultas*, es decir, cada *consulta* que se realiza utilizando este protocolo funciona de forma atómica y no relacionada con las demás *consultas*. Para resolver el problema de asociar múltiples *consultas* a un solo usuario o conjunto de acciones correlacionadas se utilizan diversos métodos, entre ellos: La utilización de cookies, mecanismos de autenticación, etc.

IP

Se trata de un protocolo de capa de red que lleva a cabo la entrega de paquetes en medios no fiables.

JSON

Formato basado en texto que permite representar datos de forma estructurada.

Línea de comandos

Interfaz basada en texto por la cual un usuario puede interactuar con un sistema operativo y con diversas piezas de software diseñadas para operar en esta interfaz.

Mime Types

Indica la naturaleza de un conjunto de datos. Esto significa su formato.

.NET CORE

Se trata de un framework open source que integra en un solo marco de desarrollo diversas aplicaciones para fines diversos, como lo son el desarrollo web, aplicaciones de línea de comando, generación de interfaces, programación de librerías, etc.

Open Source

Se trata de aquellas piezas de software que otorgan al usuario la posibilidad de obtener del código con fines de visualizar, modificar y distribuir el código a voluntad. Estos derechos se ven manifestados en la licencia de tipo open source, la cual permite ver o modificar el código.

Portal cautivo

Se trata de portales web que viven en el entorno local y que ayudan a un usuario sin amplios conocimientos realizar configuraciones básicas en productos de hardware.

Powershell

Se trata de una interfaz de línea de comandos asociada a un lenguaje homónimo de scripting que opera como código open source, su paradigma es orientado a objetos, y que al ser construido sobre el framework .NET CORE, permite la ejecución, automatización, calendarización y gestión de tareas, contando con una integración nativa con las tecnologías gestionadas en sistemas Windows.

Proxy

Se trata de un intermediario en un proceso de comunicación que intercepta, analiza y posiblemente modifica la información que atraviesa por él.

Realm

Se trata de una etiqueta que define un espacio de protección que tiene un esquema de autenticación particular. Esto significa que los elementos dentro de un mismo agrupamiento lógico de protección pueden compartir credenciales para lograr ser *consultados.*

SAAS

Se refiere un grado de abstracción de servicios en la nube, que hace transparente para el usuario o la organización que contrata este servicio la configuración, gestión y ajuste de infraestructura física subyacente.

SIEM

Sistema de Gestión de información y eventos de seguridad (SIEM), se trata de un sistema que de forma centralizada recibe información de distintas fuentes con el objetivo de proceder a la categorización de datos con el objetivo de crear dashboards gráficos que representen a la información en tiempo real. Este mecanismo permite a posteriori o mediante los sistemas de visualización detectar anomalías por el contraste con el comportamiento normal de un evento o dato de interés.

Spoofing

Consiste en usurpar la identidad de un tercero. De esta manera el atacante se hace pasar por el usuario.

Start up

Se trata de una empresa de nueva creación enfocada en lograr un crecimiento acelerado.

SQL

Se trata de un lenguaje estructurado de *consultas* que gestiona información en una base de datos.

Token

Se trata de un elemento representativo que resulta de caracteres que pueden tener embebida información significativa.

XML

Formato basado en texto que representa datos de forma estructurada.

**Fuentes Consultadas.**

[1]. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, p. 11. Sec. 1.4, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[2]. Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc.". Sec. 1.8.5. Agents, p. 22.

*3+Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc.".Sec. 1.2. Web Clients and Servers, p. 7.

[4] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc.". Capítulo 9. Web Robots, p. 202.

*5+Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc.". ,Sec. 5.1.1. Web Server Implementations, p. 105.

*6+ Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc."., Sec. 5.4. Accepting Client Connections, p. 110.

[7] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc."., Sec. 5.5. Receiving Request Messages, p. 112.

*8+ Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc."., Sec. 5.6. Processing Requests, p. 115.

*9+Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. , Sec. 5.7. Mapping and Accessing Resources, p. 115.

[10] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. , Sec. 5.8. Building Responses, p. 120.

[11] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. , Sec. 5.9. Sending Responses, p. 122.

[12] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. , Sec. 5.10. Logging, p. 123.

[13] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. Sec. 5.7.3. Dynamic Content Resource Mapping, p. 118.

[14] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, pp. 18-23. Sec. 5.1.1, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[15] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, pp. 2-3. Sec. 1.2, DOI 10.17487/RFC2396, August 1998, <https://www.rfc- editor.org/info/rfc2396>.

[16] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, pp. 12-13. Sec. 3.2.2, DOI 10.17487/RFC2396, August 1998, <https://www.rfc-editor.org/info/rfc2396>.

[17] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. Sec. 3.2.2. Dynamic Content Resource Mapping, p. 18.

[18] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. , Sec. 1.4. Transactions, p. 11.

[19] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, p. 30. Sec. 4.2, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[20]. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, p. 30. Sec. 4.1, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[21]. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, pp. 37-38. Sec. 5.1, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[22]. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, pp. 31-32. Sec. 4.3, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[23] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, pp. 37-38. Sec. 5.3, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[24] Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. “O’Reilly Media, Inc.”. , Sec. 3.5. Headers, p. 67.

[25] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, pp. 99-150. Sec. 11, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[26] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, p. 35. Sec. 5.1.1, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[27] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, p. 38. Sec. 6.1.1, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[28] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, p. 42. Sec. 7.2, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[29] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, p. 42. Sec. 70, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

[30] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, pp. 2-4. Sec. 1.2, DOI 10.17487/RFC2617, June 1999, <https://www.rfc-editor.org/info/rfc2617>.

[31] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, pp. 2-3. Sec. 1.3, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[32] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, p. 5. Sec. 1.1, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[33] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, p. 22. Sec. 4, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[34] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, p. 6. Sec. 1.2, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[35] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, p. 25. Sec. 4.1.1, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[36] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, p. 23. Sec. 4.1, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[37] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, pp. 28-29. Sec. 4.1.3, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[38] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, pp. 29-30. Sec. 4.1.4, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[39] Boyd, R. (2012). Getting started with oauth 2.0: Programming clients for secure web API authorization and authentication. “O’Reilly Media, Inc.”, Sec. 3.5.3. p. 65.

[40] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, p. 4. Sec. 2.1, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[41] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, pp. 4-5. Sec. 2.2, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[42] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, p. 5. Sec. 2.3, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[43] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, pp. 3-4. Sec. 1.3, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[44] SANS Institute, Securosis, L.L.C. (2007). Understanding and Selecting a Data Loss Prevention Solution [White paper]. https://securosis.com/assets/library/publications/DLP-Whitepaper.pdf. p. 5.

[45] SANS Institute, Securosis, L.L.C. (2007). Understanding and Selecting a Data Loss Prevention Solution [White paper]. https://securosis.com/assets/library/publications/DLP-Whitepaper.pdf. p. 6.

[46] SANS Institute, Securosis, L.L.C. (2007). Understanding and Selecting a Data Loss Prevention Solution [White paper]. https://securosis.com/assets/library/publications/DLP-Whitepaper.pdf. p. 7.

*47+ Masse, M. (2011). REST API Design Rulebook. “O’Reilly Media, Inc.”. Cap. 1. p. 5.

[48] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm). Sec 5.1. p. 76.

[49] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm). Sec 5.1.1. p. 78.

[50] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm). Sec 5.1.3. p. 78.

[51] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm). Sec 5.1.5. p. 81.

**Lecturas Adicionales**


- Andrew S. Tanenbaum & David J. Wetherall. (2011). Computer Networks (5ta Edición). Pearson.
- Andrew S. Tanenbaum & Albert S. Woodhull. (2006). Operating Systems: Design and Implementation (3ra Edición). Prentice Hall.
- Instituto Nacional de Ciberseguridad (2017). Cloud Computing, Una guía de aproximación para el empresario (1a Edición). Publicación online.
- Lee Holmes (2008). Windows PowerShell Cookbook (1a Edición). O´reilly.
- James Higginbotham (2021), Principles of Web API Design: Delivering Value with APIs and Microservices (1a Edición). Addison-Wesley Professional.
- Amir Shevat, Saurabh Sahni (2018). Designing Web APIs: Building APIs That Developers Love (1a Edición). O´reilly.
- Richer, J., & Sanso, A. (2017b). *OAuth 2 in Action*. Manning.
- Spasovski, M. (2013). *Oauth 2.0 Identity and Access Management Patterns*. Packt Pub Limited.
- Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: The Definitive Guide. "O'Reilly Media, Inc.".


**Páginas web**


- https://www.rfc-editor.org/rfc/rfc2616 (Consultado en junio 28, 2021)
- Https://www.rfc-editor.org/rfc/rfc6750 (Consultado en junio 28, 2021)
- https://www.rfc-editor.org/rfc/rfc7617 (Consultado en junio 28, 2021)
- https://httpwg.org/specs/rfc9110.html (Consultado en junio 28, 2021)
- https://www.rfc-editor.org/rfc/rfc6749 (Consultado en junio 28, 2021)
- https://www.rfc-editor.org/rfc/rfc2617 (Consultado en junio 28, 2021)
- https://www.rfc-editor.org/rfc/rfc9110.html (Consultado en junio 28, 2021)