

Plataforma de Ejecución de Flujos de Firma en la Nube.

Gabriel Moreno Pérez

Máster Universitario de Seguridad de las Tecnologías de la Información y Comunicaciones.
Sistemas de Autenticación y Autorización.

Juan Carlos Fenández Jara
Victor García Font

Fecha Entrega



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Plataforma de Ejecución de Flujos de Firma en la Nube</i>
Nombre del autor:	<i>Gabriel Moreno Pérez</i>
Nombre del consultor/a:	<i>Jose Carlos Fernández Jara</i>
Nombre del PRA:	<i>Víctor García Font</i>
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	Máster Universitario de Seguridad de las Tecnologías de la Información y Comunicaciones.
Área del Trabajo Final:	Sistemas de Autenticación y Autorización.
Idioma del trabajo:	<i>Español</i>
Palabras clave	<i>Firma digital, archivos en la nube, eIDAS</i>

Resumen.

Con la reciente entrada en vigor de la reglamentación comunitaria conocida como eIDAS (Electronic Identification and Signature) se pretende reforzar los medios de identificación electrónica y los servicios de confianza. Este reglamento, de obligado cumplimiento por parte de los Estados Miembros, afecta a la firma electrónica, sellos y marcas de tiempo electrónicas o servicios de certificados para autenticación de sitios webs. Se pretende unificar criterios y eliminar barreras entre los distintos Estados de forma que cualquier ciudadano comunitario pueda realizar cualquier trámite administrativo dentro de las fronteras comunitarias.

Aparece la problemática a la hora de utilizar complejos sistemas de firma digital entre los usuarios, es habitual que el uso de estos requiere de conocimientos técnicos relativamente avanzados y complejos. Con eIDAS se abre la oportunidad de desarrollar nuevos sistemas más fáciles de utilizar que sigan cumpliendo con los requisitos de seguridad y confianza de una firma digital cualificada.

Con este Trabajo de Fin de Máster se ha implementado un sistema de firmas de documentos fácil de utilizar y que se adapte a la reglamentación del eIDAS,

a través de este sistema los usuarios podrán compartir sus documentos para ser firmado entre ellos, se aprovechará los actuales sistemas de almacenamiento de documentos en la nube como Google Drive para poder compartir los documentos, y como prestador de servicios de confianza se empleará la solución aportada por Entrust Datacard con su producto TrustedX, una plataforma de servicios webs para la gestión de procesos de firma electrónica avanzada y cualificada. Previo al desarrollo se ha realizado un estudio de las diferentes tecnologías que emplean los sistemas implicados, como el uso del estándar OAuth 2.0 para integración con estos, y la norma PAdES que especifica el proceso de firma de documentos en formato PDF.

Abstract.

With the recent entry into force of the community regulations known as eIDAS (Electronic Identification and Signature), the aim is to strengthen the means of electronic identification and trust services. This regulation, which must be complied with by the Member States, affects electronic signatures, electronic stamps and time stamps or certificate services for the authentication of websites. The aim is to unify criteria and eliminate barriers between the different States so that any community citizen can carry out any administrative process within the community borders.

The problem arises when it comes to using complex digital signature systems among users, it is common that the use of these requires relatively advanced and complex technical knowledge. With eIDAS, the opportunity to develop new, easier-to-use systems that continue to meet the security and trust requirements of a qualified digital signature is opened.

With this Master's Final Project, an easy-to-use document signing system has been implemented that adapts to the eIDAS regulations, through this system users can share their documents to be signed among them, the current ones will be used cloud document storage systems such as Google Drive to share documents, and as a trusted service provider will use the solution provided by Entrust Datacard with its TrustedX product, a web services platform for the management of electronic signature processes advanced and qualified. Prior to the development, a study was made of the different technologies used by the

systems involved, such as the use of the OAuth 2.0 standard for integration with them, and the PAdES standard that specifies the process of signing documents in PDF format.

Índice

1. Introducción	1
Contexto y justificación.....	1
Objetivos del Trabajo.	3
Enfoque y método seguido.....	4
Planificación del Trabajo.	4
Estado del arte.	5
Desarrollo obtenido.	6
2. El estándar abierto Open Authorization 2.0	7
Roles de OAuth2.0.	7
Autorizaciones otorgadas.	8
Flujo del protocolo.	9
3. Open Authorization 2.0 en API de Google	10
API de Google Drive.....	12
Petición de token de acceso.....	14
Listado de archivos de usuario.....	18
Link de descarga de archivo.....	19
Bajada, subida y compartición de archivos.	20
4. API de TrustedX eDIAS Platform	21
Autenticación de usuarios.	22
Petición de token de código de autorización.	23
Registro de entidad de firma.	26
Listado de entidades de firma.	27
Firma de documentos PDFs.....	28
Borrado de entidades de firma.	30
5. Aplicación Portafirmas	30
Casos de Uso de Portafirmas.....	31
Diseño del sistema.	35
Modelo de Base de Datos.	38
Flujos de Firma.....	39
Clases de acceso a los servicios de Google Drive y TrustedX.....	42
Proceso de firma de archivo PDF mediante PAdES.	45
4. Conclusiones	49
5. Glosario	51
6. Bibliografía	52
7. Anexos	53
Despliegue de la aplicación.....	¡Error! Marcador no definido.
Manual de uso Portafirmas.....	¡Error! Marcador no definido.

1. Introducción

Contexto y justificación.

Con la aparición del Reglamento (UE) N° 910/2014 del Parlamento Europeo y del Consejo, del 23 de julio de 2014, conocido como eIDAS (Electronic Identification and Signature) [1], relativo a la identificación electrónica y a los servicios de confianza en transacciones electrónicas, y que deroga la Directiva 1999/93/CE [2], se pretende reforzar los medios de identificación electrónica y los servicios de confianza. Al ser un reglamento y no una directiva es de aplicación directa en el ámbito de la Unión Europea que entró en vigor el 1 de julio de 2016.

Este reglamento afecta a aspectos tan importantes como la firma electrónica, sellos y marcas de tiempo electrónicas o servicios de certificados para autenticación de sitios webs. Uno de los objetivos es eliminar barreras entre Estados miembros para posibilitar la identificación de los distintos ciudadanos pertenecientes a esos países y puedan realizar tramitaciones telemáticas en los distintos países de la Unión Europea, aparecería pues un documento de identificación electrónica (eIDS) para poder realizar trámites telemáticos entre los diferentes gobiernos, tales como servicios sanitarios o trámites administrativos entre empresas siendo estos trámites más ágiles y eficientes.

La anterior Directiva 199/93/CE regulaba los servicios de firma electrónica en la UE, reconocía la validez para un determinado tipo de firmas haciéndolas equivalentes a una firma manuscrita, no obstante cada país miembro interpretaba la directiva a su manera siendo esto un obstáculo para validar y reconocer de forma común tales mecanismos de firma.

Aparece con este reglamento el concepto de firma electrónica cualificada, definida como *una firma electrónica avanzada que se crea mediante un dispositivo cualificado de creación de firmas electrónicas y que se basa en un certificado de firma electrónica*. Este tipo de firma ofrece un alto nivel de seguridad, no obstante su uso puede llegar a ser complejo ante los usuarios ya que es necesario disponer de un certificado cualificado de firma electrónica, como por ejemplo el DNle, y de un dispositivo de firma cualificado que debe cumplir una serie de requisitos. Usualmente el empleo de este tipo de firmas es

para la realización de trámites en administraciones públicas, y debido a su complejidad de manejo por un usuario con pocos o nulos conocimientos técnicos no lo hacen una opción recomendable para empresas o personas que necesiten realizar firmas electrónicas con relativa frecuencia.

Uno de los objetivos del reglamento eIDAS es regular la identificación electrónica de los diferentes Estados miembros para la autenticación en los servicios públicos que estos ofrecen independientemente de su país de origen. Cada Estado miembro deberá adquirir un nivel de confianza en sus sistemas de identificación electrónica y que exista un reconocimiento mutuo entre ellos, para esto los medios de seguridad empleados en la identificación deben cumplir un nivel igual o superior exigido al servicio en línea que se trate, y a su vez este nivel ha de ser sustancial o alto.

Con todo esto aparece la problemática del uso de medios técnicos de un elevado nivel de confianza y seguridad ya que poseen unos requisitos derivados de las especificaciones de los sistemas de identificación electrónica, por ejemplo el uso de tarjetas inteligentes como el DNle, que pueden ser engorrosos para los titulares debido a su complejidad técnica. Con el reglamento eIDAS se abre la posibilidad que un Estado miembro tenga la libertad de definir otros tipos de servicios de confianza junto a aquellos que ya existen, estos han de ser reconocidos a nivel nacional como un servicio de confianza cualificado adoptándose un planteamiento abierto a innovaciones siempre y cuando estos nuevos servicios cumplan los requisitos estipulados. Aparecerán nuevos servicios de confianza cualificados y prestador cualificado de servicios de confianza que garanticen un alto nivel de seguridad y sean más fáciles de emplear por un usuario final, un ejemplo es el uso de éstos servicios por personas discapacitadas, así estarán en igualdad que el resto de los usuarios.

Como innovación el Reglamento eIDAS contempla la creación de firmas electrónicas a distancia gestionado por un prestador de servicios de confianza en nombre del firmante, y estas firmas han de tener la misma validez jurídica que las firmas electrónicas creadas en un entorno completamente gestionado por el firmante, como por ejemplo el DNle, los prestadores de este servicio deberán aplicar procedimientos de seguridad específicos y utilizar sistemas fiables para garantizar que el entorno de firma electrónica sea fiable y esté bajo

el control exclusivo del firmante. Se podrán generar firmas electrónicas cualificadas a distancia basadas en la nube donde se aplicarán los requisitos contemplados en el Reglamento eIDAS a los prestadores de servicios de confianza cualificados.

El objetivo este Trabajo de Fin de Máster es la creación una plataforma que permita firma en la nube de documentos por múltiples usuarios cumpliendo lo dictado por el eIDAS. Para ello será necesario el empleo de un sistema de almacenamiento de documentos en formato PDF en la nube y accesibles por los usuarios en cuestión, existen soluciones comerciales como las proporcionadas por Google Drive o DropBox con sistemas avanzados de autenticación para el acceso a estas plataformas como lo es el estándar OAuth 2.0 (Open Authorization) [3], mediante una API la plataforma de firmas se podrá consumir de estos sistemas de almacenamiento los documentos que posteriormente serán firmados por un prestador de servicios de confianza también en la nube. Como prestador de servicios de confianza se empleará la solución aportada por Entrust Datacard con su producto TrustedX [4], una plataforma de servicios webs para la gestión de procesos de firma electrónica avanzada y cualificada.

Objetivos del Trabajo.

El principal objetivo es la implementación de una plataforma que permita flujos de firma múltiples sobre documentos en la nube cumpliendo con reglamento eIDAS. Todo el proceso debe realizarse de forma remota sin la necesidad de tener los documentos ni el sistema de firmas en local. Los distintos usuarios involucrados en la firma de un documento no tendrán que tener instalados en local ningún dispositivo de manejo complejo para poder realizar la firma facilitándoles la tarea y posibilitando que la firma pueda ser realizada en múltiples plataformas, ya sea un ordenador personal o un dispositivo móvil. Para ello será necesario resolver las siguientes cuestiones:

- Acceso a repositorios en la nube de documentos mediante el sistema de autenticación OAuth 2.0, estos documentos serán accesibles por los usuarios implicados en la firma múltiple. Como se ha comentado anteriormente, existen soluciones que nos permiten esta operativa como

Google Drive, estos sistemas proporcionan APIs [5] para realizar este tipo de autenticación y poder acceder a los recursos almacenados en ellos.

- Realización remota de la **firma avanzada** por un prestador de servicios de confianza en la nube, en este caso se realizará la integración con TrustedX de Entrust Datacard [6], este producto ofrece mediante servicios webs la posibilidad de firmar de forma remota un documento, TrustedX soporta distintos estándares de firma avanzada y cualificada, cumpliendo así el reglamento eIDAS para que la firma sea reconocida por cualquier Estado miembro.

Enfoque y método seguido.

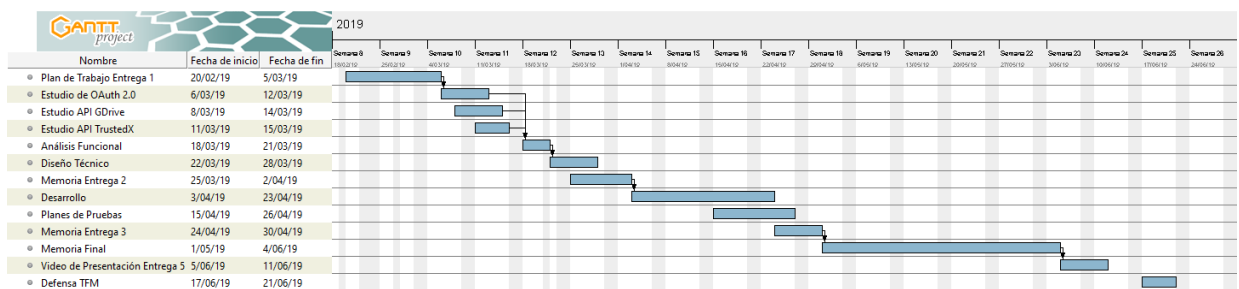
- Estudio del Reglamento (UE) N° 910/2015 del Parlamento Europeo y del Consejo, del 23 de julio de 2014, conocido como eIDAS (Electronic Identification and Signature) con el propósito de ser compatible la plataforma de firmas con este reglamento.
- Estudio del estándar OAuth 2.0 (Open Authorization) para su uso como método de autenticación entre los distintos sistemas implicados.
- Estudio de las APIs de integración con Google Drive para su posterior uso en la plataforma de firmas ya que esta accederá a estos sistemas para recuperar los documentos que serán procesados por los flujos de firma.
- Estudio de la plataforma de servicios webs para la gestión de procesos de firma electrónica TrustedX y la integración de la plataforma de firmas con este sistema y poder realizar una firma avanzada en la nube.

Planificación del Trabajo.

A continuación detallamos las tareas a realizar durante la realización de este TFM:

- Plan de Trabajo Entrega 1.
- Estudio de OAuth 2.0.
- Estudio API GDrive.
- Estudio API TrustedX.
- Análisis Funcional.

- Diseño Técnico.
- Memoria Entrega.
- Desarrollo.
- Planes de Pruebas.
- Memoria Entrega 3.
- Memoria Final.
- Video de Presentación Entrega 5.
- Defensa TFM.



Estado del arte.

TrustedX de Entrust Datacard es una solución certificada temporalmente de firma cualificada en la nube, en espera de obtener la certificación definitiva basada en el Reglamento 910/2014 eIDAS. Este sistema almacena en servidores criptográficos las claves privadas de los usuarios protegidas por PIN/passwords conocidos por los propios usuarios.

En el mercado se pueden encontrar otras soluciones que posibilitan realizar firmas de documentos en la nube también basados en los requerimientos del eIDAS. Algunos de estos sistemas están certificados como prestadores de servicios para administraciones públicas y emiten certificados de firma tanto avanzadas como cualificadas cumpliendo los requisitos establecidos en el artículo 43 de la Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público [7], para la firma electrónica del personal al servicio de la Administración Pública. No obstante todo lo relativo a generación de firmas cualificadas que estas soluciones pueden ofrecer no es posible realizarlas en la nube, así que es necesario tener un dispositivo seguro de creación de firmas

por parte del firmante en local con la complejidad que esto puede acarrear como se ha mencionado con anterioridad.

Como ejemplos de sistemas podemos encontrarnos Signaturit y esFIRMA [8][9]. En el caso de Signaturit se emplea firma avanzada para la firma de múltiples documentos en la nube, el usuario puede firmar en su dispositivo móvil mediante un trazo con su dedo obteniendo una traza biométrica del grafo generado, a esta información se le añaden otros datos como la geolocalización y las direcciones origen y destino, con todo esto y usando una autoridad de sellado temporal se garantiza la integridad de la firma, este tipo de firma tiene validez jurídica, no obstante no sería válida para realizar trámites con la administración pública ya que sería necesario el empleo de firma cualificada. Por otro lado la solución esFIRMA de forma similar a TrustedX almacena las claves privadas de los usuarios en servidores criptográficos protegidos por PIN/passwords, sin embargo no se encuentra evaluada basándose en la reglamentación eIDAS si no por la CEN/TS 419241:2014 “Security Requirements for Trustworthy Systems supporting Server Signing” [10]

Desarrollo obtenido.

El desarrollo obtenido es la aplicación web Portafirmas que permite flujos de firma avanzada múltiple sobre documentos en la nube cumpliendo con reglamento eIDAS. Al respecto de la posibilidad de la firma cualificada el sistema sería similar, en ese caso el proceso de firma sería completamente delegada en TrustedX pero se ha implementado parte de la firma en la aplicación Portafirmas en beneficio del estudio de cómo se firma un documento PDF siguiendo el estándar PAdES. Todo el proceso se realiza de forma remota sin la necesidad de tener los documentos en sistemas de firmas en local. Los distintos usuarios involucrados en la firma de un documento no necesitarán tener instalados en local ningún dispositivo de manejo complejo para poder realizar la firma facilitándoles la tarea y posibilitando que la firma pueda ser realizada en múltiples plataformas, ya sea un ordenador personal o un dispositivo móvil.

2. El estándar abierto Open Authorization 2.0

Open Authorization 2.0 (OAuth) es un protocolo estándar abierto para permitir autorizaciones de accesos a sitios webs. Mediante este sistema un usuario registrado y validado en un sitio web determinado podrá obtener autorización en otros sitios webs sin la necesidad de realizar una segunda validación, tenemos pues un proveedor de servicio de validación y un consumidor de ese servicio. El usuario no necesita introducir sus credenciales en la web consumidora, esa información se encuentra protegida en la web proveedora. Para esto OAuth 2.0 proporciona una API estándar que puede ser empleada tanto en aplicaciones de escritorio, web y dispositivos móviles. El cliente obtiene del servidor de autorización un token de acceso que es una cadena que contiene atributos de acceso, tiempo de vida y valores que representan un ámbito específico, con este token los clientes accederán a los recursos protegidos del propietario que se encuentran alojados en servidores de recursos. Este sistema es utilizado por compañías como Google, Github, Microsoft y Facebook.

Roles de OAuth2.0.

OAuth 2.0 contempla cuatro tipos de roles:

- Cliente o aplicación de terceras partes, es la aplicación que accede a la cuenta del usuario una vez autorizado por éste.
- Servidor de recursos, es la API del servidor empleada para acceder a la información protegida de las cuentas de usuario.
- Servidor de autorización, verifica la identidad del usuario, en caso de éxito genera tokens de autenticación para el acceso a otras aplicaciones.
- Usuario o propietario del recurso, es el usuario que permite a un cliente o aplicación el acceso a su cuenta, y este acceso dependerá del alcance de la autorización otorgada.

El servidor de autorización puede ser el mismo que el servidor de recursos, a su vez pueden existir múltiples servidores de recursos que aceptan tokens emitidos por un único servidor de autorización.

Autorizaciones otorgadas.

Una autorización otorgada es una credencial que representa la autorización del propietario del recurso, mediante esta el cliente obtendrá un token de acceso.

Existen cuatro tipos de autorizaciones:

- Código de autorización, es obtenida usando una autorización del servidor, el cliente no la obtiene de forma directa del propietario del recurso. El servidor de autorización deberá autenticar al propietario del recurso, en esta situación el propietario del recurso no comparte sus credenciales de autenticación con el cliente.
- Implícita, es una forma simplificada de la anterior, es usada por clientes implementados en navegadores que emplean lenguajes de scripting tales como JavaScript. En este caso el cliente recibe un token de acceso directamente como resultado de una autorización del propietario del recurso, el servidor de autorizaciones no autentica al cliente. Es un método más óptimo que el anterior pero puede llegar a ser menos seguro. Es empleado en aplicaciones móviles o webs.
- Credenciales de contraseña del propietario del recurso, en este caso es necesario disponer de las credenciales del propietario del recurso, por ejemplo, su usuario y su clave. Se emplearía directamente para lograr el otorgamiento de autorización y obtener el token de acceso. Para esto deberá existir un alto acuerdo de confianza entre el propietario del recurso y el cliente, serán pues aplicación es confiables como aquellas que tienen altos privilegios o pertenecen al sistema.
- Credenciales del cliente, se emplea cuando el ámbito del otorgamiento de autorización está limitado a la protección de recursos bajo control del cliente o aquellos que previamente fueron ordenados por el servidor de autorización. Es empleada en el acceso de las API de aplicación.

Como se ha visto un token de acceso es una credencial para acceder a un recurso protegido, es una autorización expedida a un cliente representado por

una cadena que contiene información del ámbito y duración del acceso, provee una capa de abstracción que sustituye a distintos sistemas de autenticación, como el empleo de usuarios y claves. Existen también los tokens de refresco o actualización que son empleados para obtener tokens de acceso, estos son emitidos por el servidor de autorización al cliente cuando su token de acceso ha caducado o deja de ser válido.

Flujo del protocolo.

A continuación se muestra el flujo de interacción entre los distintos roles de OAuth2.0 de forma general.



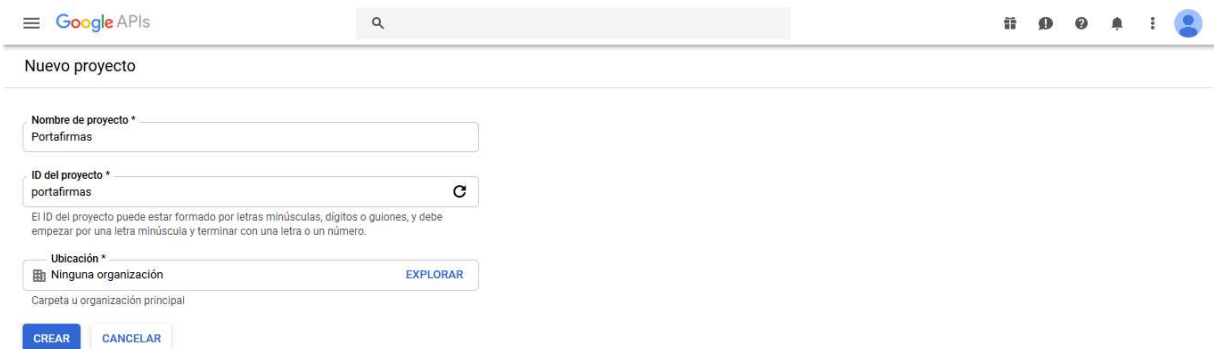
Los pasos que componen el anterior flujo son los siguientes:

1. Petición de autorización por la aplicación cliente al propietario del recurso, lo puede hacer de forma directa o indirecta a través del servidor de autorización.
2. El propietario del recurso otorga autorización al cliente mediante una credencial, esta credencial será de uno de los cuatro tipos de autorizaciones descritas y esto dependerá del método utilizado por el cliente para pedir la autorización y los tipos permitidos por el servidor de autorización.
3. Petición de token por la aplicación cliente al autenticarse contra el servidor de autorización y presentar la concesión de autorización.
4. Al servidor de autorización autentica a la aplicación cliente y procede a validar la concesión del cliente, si es válida le emite un token de acceso.

5. El cliente presenta el token de acceso al servidor de recursos para solicitarle el recurso protegido requerido.
6. El servidor de recursos valida el token recibido de la aplicación cliente y si es correcto le envía el recurso protegido.

3. Open Authorization 2.0 en API de Google.

Mediante esta API Google nos ofrece la posibilidad de integrarnos con sus servicios a través de mecanismos de autenticación y autorización. En este caso la integración será para la aplicación web Portafirmas, una aplicación que accederá a los recursos de un usuario registrado en Google y tenga archivos almacenados en Google Drive, inicialmente es necesario que la aplicación sea registrada en la consola de desarrolladores de Google. Hay que disponer de un usuario registrado en este sistema para poder acceder a esta consola. Una vez dentro procedemos a la creación de un nuevo proyecto, habrá que asignarle un nombre y un identificador del proyecto:



Nuevo proyecto

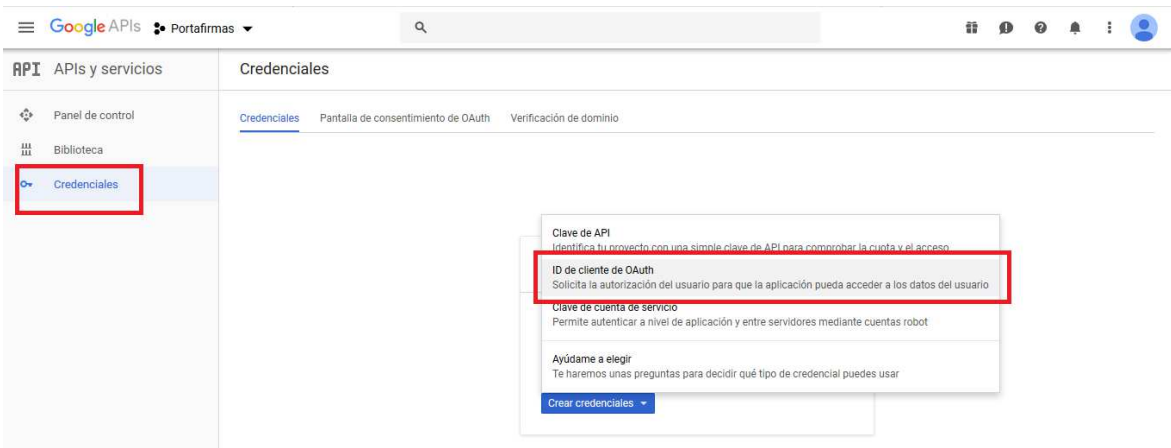
Nombre de proyecto *
Portafirmas

ID del proyecto *
portafirmas

Ubicación *
Ninguna organización

CREAR CANCELAR

A continuación habilitamos la API que nos interesa, en nuestro caso será la de Google Drive, seguidamente será necesaria la creación de credenciales, en este caso emplearemos la opción de ID de cliente OAuth:



API APIs y servicios

Credenciales

Panel de control

Biblioteca

Credenciales

Clave de API
Identifica tu proyecto con una simple clave de API para comprobar la cuenta y el acceso.

ID de cliente de OAuth
Solicita la autorización del usuario para que la aplicación pueda acceder a los datos del usuario

Clave de cuenta de servicio
Permite autenticar a nivel de aplicación y entre servidores mediante cuentas robot

Ayúdame a elegir
Te haremos unas preguntas para decidir qué tipo de credencial puedes usar

Crear credenciales

Seguidamente se solicitará el tipo, en este caso aplicación web, y se introducirá la URL de la dicha aplicación web. A continuación se completará la configuración de la pantalla de consentimiento de OAuth, ahí indicamos el nombre de la aplicación, un logo, correo electrónico de asistencia, el nombre de dominio al que pertenece y enlace desde ese dominio hacia la aplicación, y enlaces hacia la política de seguridad y a las condiciones. Es necesario completar la verificación del dominio, previamente hay que verificar la propiedad del dominio, introducimos la URL del dominio registrado para crear la pantalla de consentimiento, posteriormente obtendremos la siguiente pantalla donde habrá que descargarse un archivo en formato html que habrá de desplegarse en el directorio raíz de la aplicación web cliente.

Google

Centro para webmasters

Eres un propietario verificado de [https://\[redacted\].com/Portafirmas/](https://[redacted].com/Portafirmas/). La verificación se ha realizado mediante estos métodos:

- Archivo HTML [Detalles](#)
 - google7[redacted].html

[Verificar con otro método](#)

Intentos de verificación

Fecha:	Correo electrónico	Resultado	Método
Hace 5 minutos 24/03/19 11:10:51 UTC	[redacted]@gmail.com	La verificación se ha realizado correctamente.	Archivo HTML

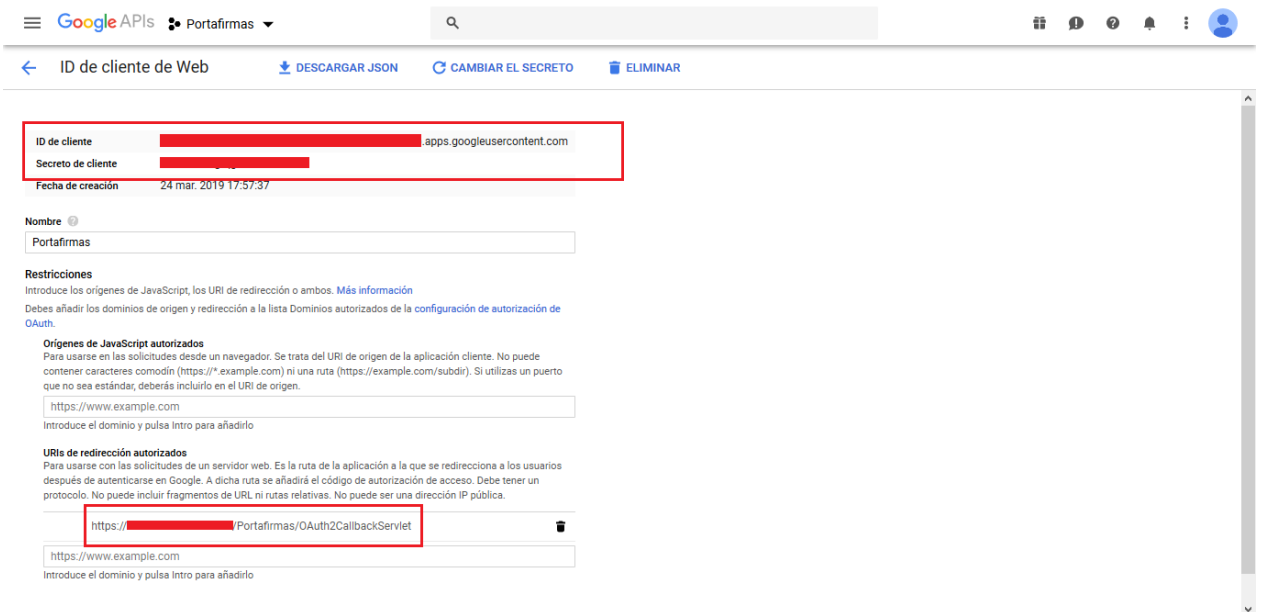
Propietarios verificados

Dirección de correo electrónico	Detalles	Acción
[redacted]@gmail.com - usted	Detalles de verificación	Anular verificación

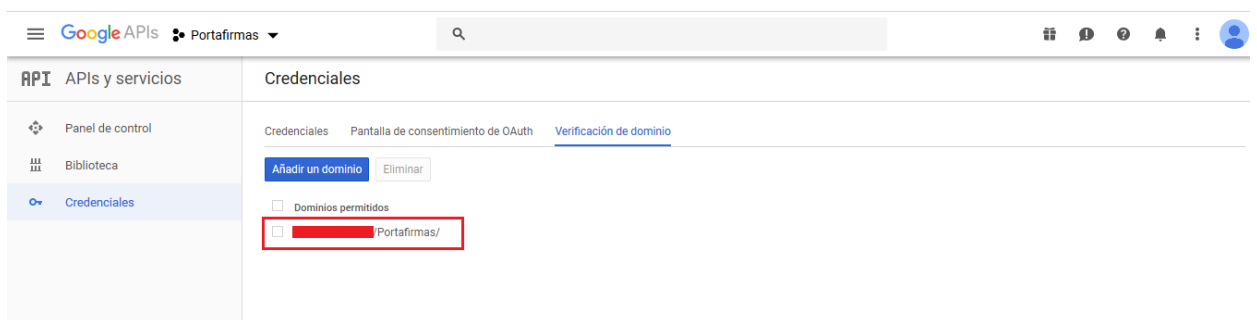
[Añadir un propietario](#)

© 2019 Google LLC - Centro para webmasters - Condiciones del servicio - Política de privacidad - Ayuda de Search Console

Será necesario añadir una URL de redireccionamiento, conocida como función de callback, que será empleada por Google para completar la autenticación y la autorización de forma que se generará un identificador de cliente y su secreto. Editando la credencial se llegará a la siguiente pantalla donde se introducirá la URL de redireccionamiento necesaria para recibir en ésta la información generada por Google y poder operar:

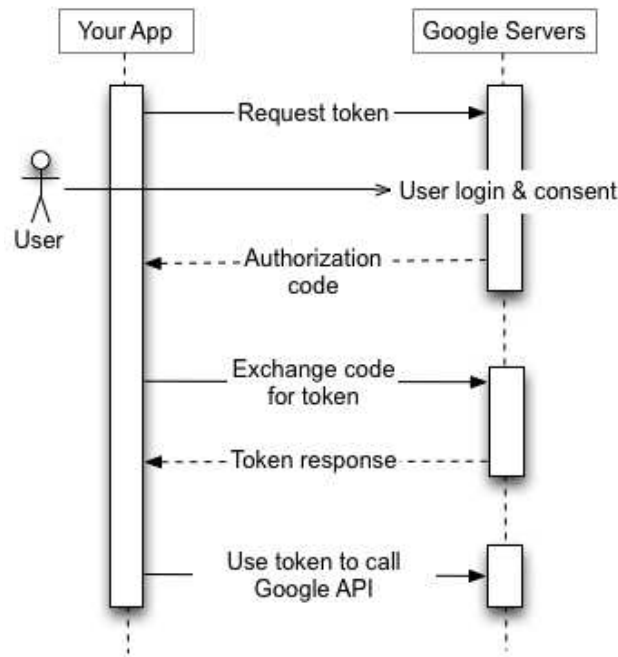


Por último, en verificación del dominio, introducimos la URL de nuestra aplicación:



API de Google Drive.

Mediante esta API se pueden acceder a los recursos disponibles en Google Drive. Previamente es necesario obtener un código de autorización y un token de acceso para poder acceder a los archivos almacenados por el usuario tal y como se procede en los protocolos basados en OAuth2.0. La siguiente imagen muestra los mensajes entre la aplicación y los servidores de Google para realizar la autenticación:



Fuente Google

Google Drive proporciona diferentes API clientes que encapsulan las diferentes llamadas para ser empleadas en función de la tecnología de desarrollo del cliente, como podría ser en Java. No obstante bajo estos clientes podemos ver que emplean llamadas a servicios REST, que no dejan de ser un intercambio de parámetros a través de URLs destinadas para lograr esta autenticación. En este caso nos centraremos en el uso de los servicios REST.

El primer paso es solicitar un token, para ello previamente será necesario obtener un código de autorización, el cliente deberá enviar sus credenciales a Google de forma que si la validación es exitosa éste devolverá el token de acceso. Es necesario por parte del cliente proporcionar a dónde ha de enviar el token los servidores de Google, para ello durante la solicitud del código de acceso se le envía anexo la URL destino, que no deja de ser la URL de callback que fue configurada durante la creación de la credencial de la aplicación web.

También es necesario proporcionar a Google el alcance o scopes, de esta forma indicamos a qué tipo de API y qué funcionalidad de ella vamos a emplear, estos scopes vienen representados por URLs, a continuación mostramos los empleados para Google Drive:

SCOPE	USO
https://www.googleapis.com/auth/drive	Acceso completo a todos los recursos menos a la carpeta de los datos de aplicación, debería usarse sólo en caso necesario.
https://www.googleapis.com/auth/drive.readonly	Acceso de solo lectura a metadatos y contenido de archivos.
https://www.googleapis.com/auth/drive.appfolder	Permite acceso a la carpeta de datos de aplicación.
https://www.googleapis.com/auth/drive.file	Acceso a los archivos creados o abiertos por la aplicación.
https://www.googleapis.com/auth/drive.install	Empleado para permitir al usuario aprobar la instalación de una aplicación.
https://www.googleapis.com/auth/drive.metadata	Permite acceso de escritura y lectura de metadatos de archivo, pero no permite ningún acceso de lectura, descarga, escritura o carga de archivo.
https://www.googleapis.com/auth/drive.metadata.readonly	Permite el acceso de solo lectura de metadatos de archivo, pero no permite ningún acceso para leer o descargar contenido del archivo.
https://www.googleapis.com/auth/drive.scripts	Permite acceso a los archivos de scripts de la aplicación.
https://www.googleapis.com/auth/drive.apps.readonly	Permite acceso de solo lectura las aplicaciones instaladas.

A continuación vamos a ver con un ejemplo cómo se realizaría una petición de token de acceso para obtener un link de descarga de un archivo almacenado en Google Drive para la aplicación Portafirmas.

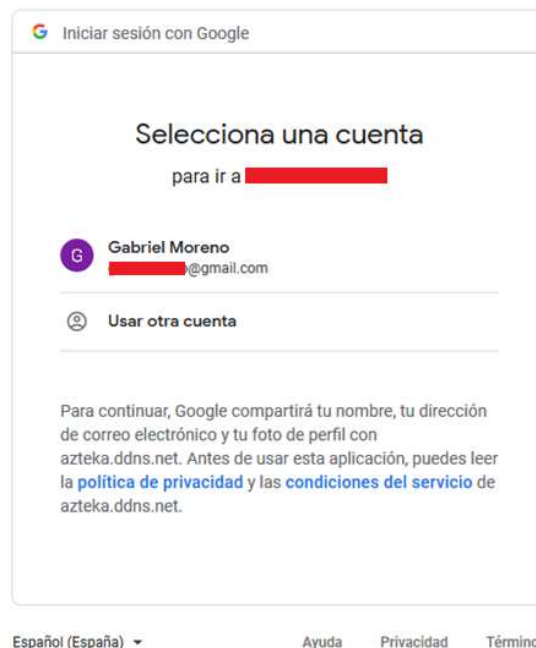
Petición de token de acceso.

Es necesario realizar una primera llamada a la siguiente URL con los parámetros adecuados para recibir el código de autorización necesario para recuperar el token de acceso:

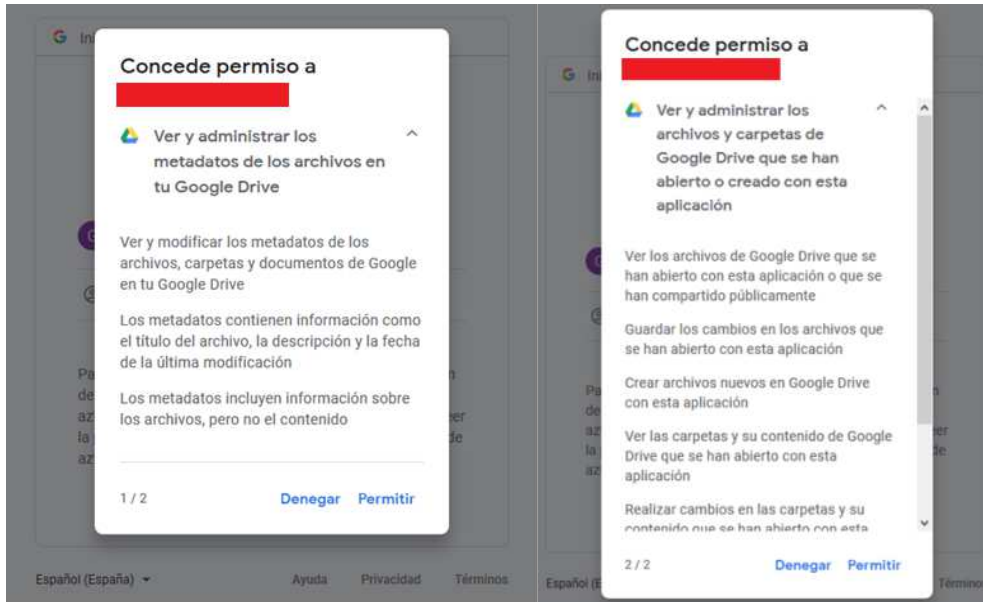
https://accounts.google.com/o/oauth2/auth	
Parámetro	Valor
client_id	Identificador del cliente recuperado al registrar la aplicación web, indicado en la consola de desarrolladores de Google.
redirect_uri	URL de callback indicada en la configuración de la consola de desarrolladores de Google, en esta URL se recibirá el código

	de autorización, en este caso: https://<dominio>/Portafirmas/OAuth2CallbackServlet
response_type	code , indica que la respuesta sea el código de autorización.
scope	Alcance que indican a qué funcionalidad se quiere acceder, en este caso: https://www.googleapis.com/auth/drive.file https://www.googleapis.com/auth/drive.metadata https://www.googleapis.com/auth/drive.readonly

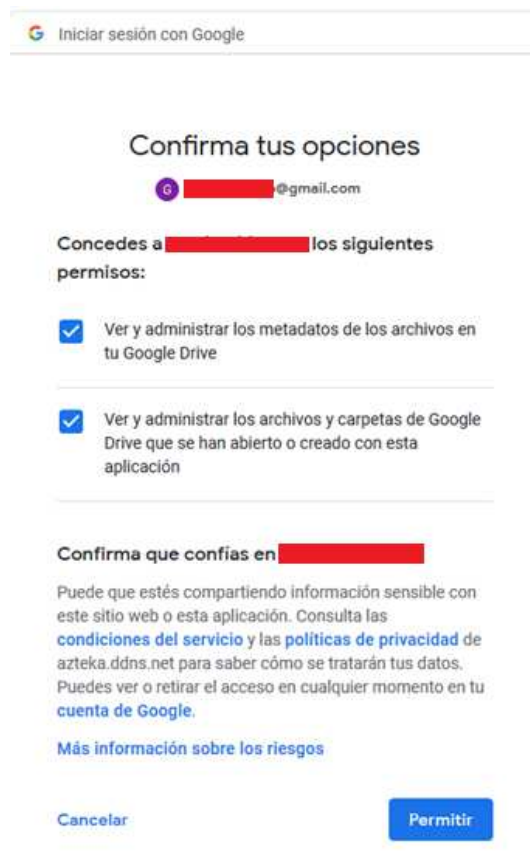
Una vez lanzada la URL desde la aplicación, aparecerán una serie de pantallas generadas por Google donde el usuario introducirá sus credenciales de validación y concederá su autorización a la aplicación Portafirmas. Inicialmente nos pedirá la cuenta de usuario indicando el dominio donde se encuentra desplegada la aplicación.



A continuación se solicita al usuario permisos sobre el dominio relacionados con el alcance o scopes pasados por parámetros en la URL.



Y por último se solicita confirmación final del usuario:



En respuesta a todo esto desde Google se invocará a la URL de callback enviando el código de acceso en el parámetro code de la siguiente forma:

https://<dominio>/Portafirmas/OAuth2CallbackServlet?code=4/HQFoo44PA9lqerqYj_fcWMUql6XEnwlqrHOwC62eC6VsglyRMch5L1gK9Pn3McQkvXefHcoGIP8UZ7sFUAArs

Una vez recuperado el código de autorización se realizará la llamada a la URL de petición por post del token de acceso:

https://accounts.google.com/o/oauth2/token	
Parámetro	Valor
Code	Código de autorización recibido, en este ejemplo será el valor 4/HQFoo44PA9lqerqYj_fcWMUql6XEnwlqrHOwC62eC6VsglyRMch5L1gK9Pn3McQkvXefHcoGIP8UZ7sFUAArs
client_id	Identificador del cliente recuperado al registrar la aplicación web, indicado en la consola de desarrolladores de Google.
client_secret	Secreto del cliente recuperado al registrar la aplicación web, indicado en la consola de desarrolladores de Google.
redirect_uri	URL de callback indicada en la configuración de la consola de desarrolladores de Google, en esta URL se recibirá el código de autorización, en este caso: https://<dominio>/Portafirmas/OAuth2CallbackServlet
grant_type	authorization_code , valor que es definido en la especificación OAuth2.0

En respuesta exitosa a la anterior llamada recibiremos el token de acceso que vendrá en forma de objeto JSON como el siguiente:

```
{
  "access_token": "ya29.GlzcBhyFfzJrwn3OqxHjM6ovDJfJHxBbPZsIdlRShdUbcTeWC-
jDnWCqofg5YipWRCSWEm8jzBIZpoQCdWhmY6BmHpSsaBxOqtqk_JbdIjsP2EBZBpPAMgzdV10MXA",
  "expires_in": 3600,
  "scope": "https://www.googleapis.com/auth/drive.file https://www.googleapis.com/auth/drive.readonly
https://www.googleapis.com/auth/drive.metadata",
  "token_type": "Bearer",
  "id_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6ImE0MzEzZTdmZDFlOWUyYTRkZWQzYjI5MmQyYTMmNGU1MTk1NzQzMDgiLCJ0eXAiOiJKV1Qi
fQ.eyJpc3MiOiJhY2NvdW50cy5nb29nbGUuY29tIiwiaXpwIjoimzM0MTU1Mjk1MjE1LXJvMDJwOGRuNGk2bzlnYjlkbg9iNGo5aG
hiYm5uOXBwLmFwcHMuz29vZ2xldXNlcmNvbnRlbnQuY29tIiwiaXVkiOiJoimzM0MTU1Mjk1MjE1LXJvMDJwOGRuNGk2bzlnYjlkbg9
iNGo5aGhiYm5uOXBwLmFwcHMuz29vZ2xldXNlcmNvbnRlbnQuY29tIiwic3ViIjoimTA0ODM0NzA4NDQ4MzAyMTE4MzQ3IiwiaXN1bW
aWwiOiJnYjYwLmVcmVub0BnbWVpbC5jb20iLCJlbWVpbG92ZXJpZm1lZCI6dHJlZSwiYXRfaGFzaCI6InpEaUNxNEJySTgwdTNXe
```

```

nozTkFwN1EiLCJpYXQiOjEu1NTM5NzExMzYsImV4cCI6MTU1Mzk3NDczNn0.Bp9LGGQEY2jJ-
n176uxU7SDw9fMQCgVYiaZdbgmZOCf-
xv7H0dUuY0dGS0Brziezrv0NmI1nVHTOQNeNZQ1b1ZulR51hMCsNZAb0X7ibcm9SHdy3Wt jJNm-
vBxG0lzaAlaMxRP7JZp7fluSbdo5UNw7JC2m6xe67yBNxz-
Vii6tSLGWHMaF6H4wmAft9FfMR0iKhWYorpr1Vysos4nflyuXp4vwYoQc0cl6afS7bXIIFMrYGmfWsssB69vqdlf5i-
1nWFCDPWovHY_x3WsSzGb14d0IJFT01dTBpp322-811mwP8sxc0iK4G_MHuuJmiKYzH-agNHBWV_h6tQvnGOg"
}

```

Dicho token deberá emplearse en las siguientes peticiones a Google para poder acceder a los recursos necesarios.

Listado de archivos de usuario.

Para obtener el listado de archivos de un usuario autenticado deberá realizar una petición por *get* a la siguiente URL especificando dos parámetros de la cabecera y aquellos parámetros que pueden configurar el tipo de petición, en este ejemplo se listarán aquellos documentos con formato PDF ordenados por orden alfabético, la API REST nos proporciona otros parámetros para configurar la búsqueda.

https://www.googleapis.com/drive/v3/files	
Parámetros GET	
Parámetro	Valor
Q	mimeType='application/pdf', recupera archivos PDF.
orderBy	name, ordenados por nombre.
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Bearer <acces_token>, enviamos el token de acceso recibido en el anterior paso.
Accept	application/json, obtendremos una respuesta en formato JSON

El resultado de la llamada será un listado de objetos en formato JSON que representa los archivos recuperados en la búsqueda. Los siguientes objetos JSON representan dos archivos recuperados tras la llamada:

```

{
  "kind": "drive#file",
  "id": "1vGfXzS8StMdbUUxaqXv7ETqsiBX1liJO",
  "name": "Modulo_5_PID_00239303.pdf",
  "mimeType": "application/pdf"
}

```



```

},
{
  "kind": "drive#file",
    "id": "1tzAoFdtvlpnUkcOilB9KnBO6hrdiEDtI",
    "name": "Modulo_6_PID_00204294.pdf",
    "mimeType": "application/pdf"
}

```

Con el campo id podemos recuperar el archivo en cuestión como veremos en el siguiente paso.

Link de descarga de archivo.

Para ello es necesario disponer del id de archivo a descargar, en el paso anterior hemos visto cómo podemos recuperar un listado de archivos y cómo obtener su id. En este paso vamos a solicitar un link desde donde podemos descargarnos ese archivo y poder visualizarlo. Para ello utilizaremos la siguiente URL llamada por get con sus correspondientes parámetros y cabecera: En respuesta a la anterior llamada al servicio REST obtendremos un JSON con el enlace de descarga tal y como se muestra en el siguiente ejemplo:

<a href="https://www.googleapis.com/drive/v3/files/<ID_FILE>">https://www.googleapis.com/drive/v3/files/<ID_FILE>	
Parámetros GET	
Parámetro	Valor
Fields	webViewLink, solicitud de link de descarga.
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Bearer <acces_token>, enviamos el token de acceso recibido durante la autorización.
Accept	application/json, obtendremos una respuesta en formato JSON

```

{
  "webViewLink":
https://drive.google.com/file/d/1tzAoFdtvlpnUkcOilB9KnBO6hrdiEDtI/view?usp=drivesdk
}

```

Bajada, subida y compartición de archivos.

Tenemos otras operaciones disponibles para la gestión de archivos empleadas por Portafirmas para realizar sus operaciones, que son bajada, subida y compartición del archivo a firmar.

- Bajada de un archivo, se parametriza de la siguiente forma:

<a href="https://www.googleapis.com/drive/v3/files/<ID_FILE>">https://www.googleapis.com/drive/v3/files/<ID_FILE>	
Parámetros GET	
Parámetro	Valor
Alt	Media
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Bearer <acces_token>, enviamos el token de acceso recibido durante la autorización.
Accept	application/json, obtendremos una respuesta en formato JSON

En respuesta se recibirá un array de bytes correspondiente al archivo descargado, en el destino se recompondrá dicho archivo en el espacio de almacenamiento adecuado para su posterior firma.

- Subida de un archivo, será necesario realizar dos llamadas a la API de Google que detallaremos a continuación:

https://www.googleapis.com/drive/v2/files	
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Bearer <acces_token>, enviamos el token de acceso recibido durante la autorización.
Accept	application/json, obtendremos una respuesta en formato JSON.
Post de Objeto JSON	
Parámetro	Valor

Title	Nombre del archivo en el destino.
contentType	Tipo de archivo, en este caso al procesarse PDFs el valor será application/pdf.
description	Descripción del fichero a subir.

En respuesta a la anterior llamada se recibirá un objeto en formato JSON donde recuperaremos un ID de archivo empleado en la segunda llamada donde será efectiva la subida del archivo:

<a href="https://www.googleapis.com/drive/v2/files/<ID_FILE>">https://www.googleapis.com/drive/v2/files/<ID_FILE>	
Parámetros PUT	
Parámetro	Valor
uploadType	media
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Bearer <acces_token>, enviamos el token de acceso recibido durante la autorización.

Esta llamada incluirá el envío del archivo a subir en formato de array de bytes.

4. API de TrustedX eIDAS Platform

Esta plataforma permite la identificación, autenticación y firma electrónica para entornos webs. Ofrece una API Web para que una aplicación web pueda integrarse con esta plataforma, Portafirmas hará uso de esta API para poder realizar firmas avanzadas de aquellos documentos que el usuario tiene almacenado en la nube en plataformas como Google Drive. Esta plataforma de firmas permite la gestión de los atributos de identidad de infraestructura de clave pública PKI y funciones de firma remota según la norma técnica CEN TS 419 241 [11], y actualmente está siendo certificado como Qualified Signature Creation Device (QSCD) para ser utilizado por proveedores de servicio de confianza.

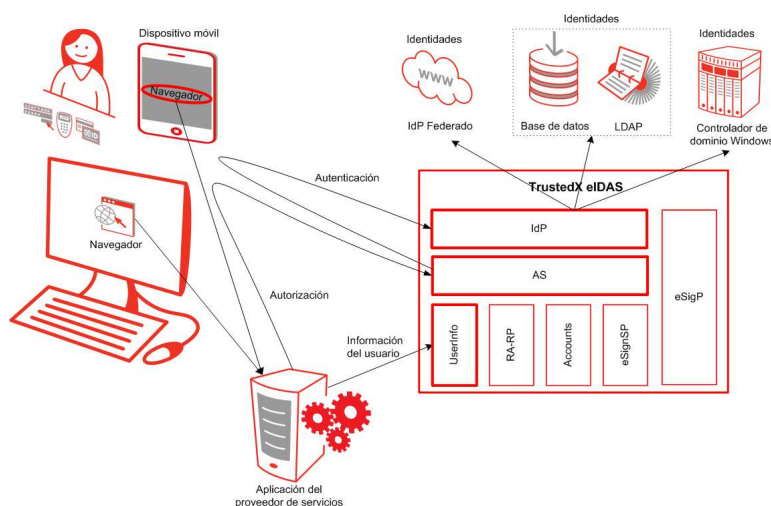
Las principales funcionalidades de la plataforma son:

- Proveedor de identidad (IdP), proporciona validación de identidades de usuarios y gestión del nivel de confianza, incluye mecanismos de autenticación basados en PKI , SMS/Email OPT y Safelayer Mobile.
- Proveedor de firma electrónica (eSigP), gestiona las PKI de los usuarios como atributos de identidad en un repositorio seguro y auditado basado en un Hardware Security Manager (HSM) donde un usuario puede disponer de uno o varios certificados digitales para firmar electrónicamente documentos, uno de los objetivos de la aplicación Portafirmas.
- Estándares de integración, soporte para SAML, OAuth/OpenID, formatos de firma ETSI, PAdES, XAdES, CAdES y RSA PKCS#1 [12].

La integración de Portafirmas con esta API será mediante servicios Web REST que implementan distintos estándares entre ellos el abierto OAuth2.0.

Autenticación de usuarios.

La arquitectura de TrustedX eIDAS para la autenticación está basada en el estándar OAuth2.0 permitiéndose a las aplicaciones clientes como Portafirmas autenticar a usuarios y obtener atributos de identidad:



Viendo la imagen anterior podemos ver los siguientes componentes:

- Proveedor de información de usuario.
- Servidor de Autorización (AS).
- Proveedor de identidad (IdP).

- Distintos servicios externos de validación, como validación de certificados y controladores de dominio.
- Servidores de procesos delegados.
- BBDD de gestión de cuentas de usuario.

En este apartado se centrará en el Servidor de Autorización (AS) el cual controla los accesos a los recursos del servidor principal que proporciona las identidades de firma. Proporciona tokens de acceso a las aplicaciones cliente para que puedan acceder a los recursos web de TrustedX eIDAS, para ello es necesario que se cumpla lo siguiente:

- El propietario del recurso debe autorizar a la aplicación cliente acceder a dicho recurso.
- La propia aplicación cliente es la propietaria del recurso.
- La aplicación cliente dispone de permisos administrativos para acceder al recurso web.

Estos servidores también se encargan de validar los tokens de acceso necesarios para poder autorizar las peticiones de los distintos servicios que proporciona TrustedX eIDAS. A continuación veremos el proceso de cómo obtener un token de acceso mediante las llamadas al servicio REST de la plataforma, dicho servicio estará instalado en un servidor cuyo dominio y puerto de escucha es **uoc.safelayer.com:8082**.

Petición de token de código de autorización.

Este es el primer paso para obtener un token de acceso, inicialmente es necesario obtener un código de autorización. El proceso se inicia con una petición de autorización que no es más que un folio de autorización OAuth2.0 de tipo Authorization Code Grant. Se realizará una serie de intercambios de mensaje entre el usuario y el navegador hasta su obtención.

Es necesario indicar el alcance o scope de la autorización, siendo los más importantes los siguientes:

SCOPE	USO
urn:safelayer:eidassign:identity:register	Permite generar claves en el servidor o importar un PKCS#12 con claves pregeneradas y los certificados asociados ya emitidos.
urn:safelayer:eidassign:identity:manage	Permite administrar las identidades de firmas, se pueden listar, actualizar o borrar identidades de firma de usuario.
urn:safelayer:eidassign:identity:use:server	Permite utilizar una identidad de firma, un token

	de acceso para este alcance deberá renovarse para cada firma realizada.
--	---

La URL del servicio REST y sus parámetros para la solicitud del código de autenticación es la siguiente:

https://uoc.safelayer.com:8082/trustedx-authserver/main/oauth	
Parámetro	Valor
acr_values	urn:safelayer:tw:polices:authentication:flow:basic. Establece condiciones a la autenticación del usuario (niveles mínimos y/o flujos específicos) que debe autorizar el acceso.
redirect_uri	https://<dominio>/Portafirmas/CallbackTrustedXPkcs12Servlet URL de callback donde se recibirá el código de autorización.
client_id	Identificador del cliente, en este caso la cadena portafirmas .
response_type	code , indica que se solicita código de autorización.
Scope	urn:safelayer:eid:sign:identity:register y urn:safelayer:eid:sign:identity:manage
register_group_labels	uoc,student . Este parámetro solo debe utilizarse para obtener la autorización de un usuario para crear identidades de firma en servidor que se activen mediante una contraseña en el HSM

Una vez lanzada la URL desde la aplicación nos mostrará una pantalla donde un usuario registrado procederá a autenticarse contra el IdP:

Seguidamente se mostrará una pantalla de solicitud de permisos relacionadas con el alcance solicitado, en este ejemplo la generación de claves en el servidor y la gestión de estas.



Funciona con TrustedX de Safelayer Secure Communications, S.A.

La respuesta exitosa de este proceso hará que desde la función de callback se reciba el código de autorización **code** necesario para recuperar el token de acceso. Para ello se realizará la siguiente llamada por **post** al servicio REST donde será necesario pasarle una serie de parámetros tanto por la URL como por cabecera. A destacar que por cabecera se pasará el secreto del cliente junto a su identificación formateado en utf-8 de la forma siguiente:

```
SECRETO = base64(url_encode(utf8(client_id)) ':' url_encode(utf8(client_secret)))
```

https://uoc.safelayer.com:8082/trustedx-authserver/oauth/main/token	
Parámetros POST	
Parámetro	Valor
grant_type	authorization_code
Code	Código de autorización recibido en la anterior llamada
redirect_uri	https://<dominio>/Portafirmas/CallbackTrustedXPkcs12Servlet URL de callback donde se recibirá el código de autorización.
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Basic <SECRETO>
Content-Type	application/x-www-form-urlencoded; charset=UTF-8

Tras llamar a esta URL se recibirá en caso de éxito el token de acceso necesario para realizar las operaciones necesarias para los casos de uso de TrustedX eIDAS empleados en la aplicación Portafirmas. Vendrá recogido en un objeto JSON con el siguiente contenido:

```
{  
  "access_token" : {token de acceso},  
  "token_type" : "Bearer",  
  "expires_in" : {tiempo de vida en segundos}  
}
```

Registro de entidad de firma.

Este caso de uso describe cómo se crea una identidad de firma en servidor que corresponde a las claves que se proporcionan en una PKCS#12 en formato de objeto JSON como se muestra en el siguiente ejemplo.

```
{
  "labels" : [ "uoc", "student" ],
  "password" : "jnQqzegfxy3a9a3vEcX",
  "pkcs12" : "RXN0byBlcyB...1biBQS0NTICMx"
}
```

Siendo los atributos del objeto los siguientes:

- labels: Lista de etiquetas asociadas a la identidad de firma que se quiere crear.
- password: Contraseña que protege al PKCS#12 que contiene las claves que corresponden a la identidad de firma que se quiere crear.
- pkcs12: Contiene las claves que corresponden a la identidad de firma que se quiere crear. El PKCS #12 está codificado en DER y base64.

Para ello se realizará la siguiente llamada por **post** al servicio REST donde será necesario pasarle sólo parámetros por cabecera y el objeto JSON de la PKCS#12 en el cuerpo de la llamada.

https://uoc.safelayer.com:8082/trustedx-resources/esigp/v1/sign_identities/server/pki_x509/pkcs12	
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Basic <TOKEN DE ACCESO>
Content-Type	application/json

Tras llamar a esta URL se recibirá un objeto JSON que representará la entidad de firma:

```
{
  "id" : {string},
  "self" : {string},
  "description" : {string},
  "labels" : [ {string} ],
  "type" : {string},
  "device_id" : {string},
  "domain" : {string},
  "access" : [ {
    "user_id" : {string}
  } ],
  "details" : {
    "certificate" : {string},
    "public_key" : {string}
  },
  "links" : {
    <operation_alias> : {
```



```

    "auth" : {
      "oauth2": {
        "scopes": [ {string} ]
      }
    }
  },
  "status" : {
    "value" : {string},
    "reason" : {string}
  }
}

```

El identificador de la identidad de la firma será necesario para poder realizar operaciones de firma donde previamente necesitará código de autorización y token de acceso asociados a este recurso.

Listado de entidades de firma.

Este caso de uso describe cómo se recupera un listado de entidades de firma. Únicamente será necesario proporcionar un token de acceso por cabecera y que haya sido generado con alcance o scope de tipo **urn:safelayer:eidas:sign:identity:manage**.

https://uoc.safelayer.com:8082/trustedx-resources/esigp/v1/sign_identities/server/pki_x509/pkcs12	
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Basic <TOKEN DE ACCESO>

La respuesta será un objeto JSON con un listado de uno o más entidades de firma con el siguiente aspecto:

```

{
  "sign_identities" : [ {
    "id" : {string},
    "self" : {string},
    "description" : {string},
    "labels" : [ {string} ],
    "type" : {string},
    "device_id" : {string},
    "domain" : {string},
    "access" : [ {"user_id" : {string}} ],
    "links" : {
      <operation_alias> : {
        "auth" : {
          "oauth2": {"scopes": [ {string} ]}
        }
      }
    },
    "status" : {
      "value" : {string},
      "reason" : {string}
    }
  }
]
}

```

Firma de documentos PDFs.

Se realizará una firma avanzada en formato PKCS#1 [13] sobre un resumen de datos hash obtenido del archivo PDF, estos datos están determinados por el estándar de firmas PaDES aplicado basado en la norma PDF 32000-1:2008 [14]. El servicio de firma electrónica (eSigP) será el responsable del proceso de firma sobre los datos resumen. La firma realizada sobre el documento será la avanzada, no obstante es posible obtener la firma cualificada utilizando mecanismos de integración similares que facilita TrustedX mediante llamadas a servicios REST ya que este sistema emplea servidores HSM para almacenar PKIs de usuarios. TrustedX solicitará al usuario autorización explícita sobre el recurso de firma, será necesario la obtención de un código de autorización en primer lugar, y luego un token para acceder a la entidad de firma identificada mediante su id.

La URL de petición de código de autorización estará parametrizada acorde con el tipo de operación, a destacar que el alcance será distinto al utilizado para obtener el token de acceso inicial, también será necesario proporcionar el identificador de la firma de identidad a utilizar ya que la solicitud de autorización de hace de forma explícita sobre este recurso.

https://uoc.safelayer.com:8082/trustedx-authserver/main/oauth	
Parámetro	Valor
acr_values	urn:safelayer:twspolicies:authentication:flow:basic . Establece condiciones a la autenticación del usuario (niveles mínimos y/o flujos específicos) que debe autorizar el acceso.
redirect_uri	https://<dominio>/Portafirmas/CallbackTrustedXPkcs12Servlet URL de callback donde se recibirá el código de autorización.
client_id	Identificador del cliente, en este caso la cadena portafirmas .
response_type	code , indica que se solicita código de autorización.
Scope	urn:safelayer:eidassign:identity:use:server
sign_identity_id	Identificador de la firma sobre la que se solicita de forma explícita la autorización

Seguidamente se mostrará una pantalla de solicitud de permisos relacionadas con el alcance solicitado, en este caso, usar la identidad de firma del servidor:



Una vez obtenido el código de autorización se procederá a solicitar un token de acceso de forma similar a como se ha procedido anteriormente, y será enviado por **post** a la URL del servicio REST junto con el siguiente objeto JSON:

```
{
  "digest_value" : {string},
  "signature_algorithm" : {string},
  "sign_identity_id" : {string}
}
```

Siendo los atributos del objeto los siguientes:

digest_value, HASH de los datos a firmar codificados en Base64.

signature_algorithm, algoritmo con el que se tiene que crear la firma digital (rsa-sha1, rsa-sha256, rsa-sha380, rsa-sha512).

sign_identity_id, Identificador de la firma con la que se va a realizar la firma y pertenece al usuario que va a realizar la operación.

https://uoc.safelayer.com:8082/trustedx-resources/esigp/v1/signatures/server/raw	
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	bearer <TOKEN DE ACCESO>
Content-Type	aplication/json

Como respuesta exitosa a una firma se obtendrá el valor binario de la firma resultante sobre la hash. Dicho valor se incorporará al documento original a firmar obteniendo así la firma avanzada de este según el estándar PAdES.

Borrado de entidades de firma.

Este caso de uso describe cómo se elimina una entidad de de firma. Será necesario proporcionar un token de acceso por cabecera y que haya sido generado con alcance o scope de tipo **urn:safelayer:eidas:sign:identity:manage**, además del identificador de la identidad de firma que se quiere eliminar, y será una petición tipo **delete**.

https://uoc.safelayer.com:8082/trustedx-resources/esigp/v1/sign_identities	
Parámetros	
Parámetro	Valor
sign_identity_id	Identificador de la identidad de firma
Parámetros de cabecera (header)	
Parámetro	Valor
Authorization	Basic <TOKEN DE ACCESO>

Si el borrado ha sido exitoso se recibirá un objeto JSON que representará la identidad de firma que ha sido eliminada.

5. Aplicación Portafirmas

Portafirmas está implementada como una aplicación web utilizando tecnología Java Enterprise Edition (Java EE) [15] con una base de datos relacional contenida en un gestor de base de datos MySql [16]. La ventaja del empleo de Java EE es la independencia de la plataforma donde se ejecuta el sistema y la facilidad del acceso distribuido a ésta, con un simple navegador web se accede, la actualización y el mantenimiento es fácil, solo se tiene que actuar en el servidor donde se encuentre ubicada la aplicación y la base de datos.

Para el desarrollo del sistema se ha empleado el entorno NetBeans 8.2 [17], este entorno facilita la integración con el servidor de aplicaciones y el gestor de base de datos necesarios para la ejecución del sistema durante el proceso de implementación. Para el despliegue de la aplicación se ha empleado Apache Tomcat 9.0.12 [18], no obstante cualquier servidor de aplicaciones Java EE puede contener el sistema. El gestor de base de datos es MySql Server 8.0.13, para acceder a este mediante Java se emplea drivers JDBC que proporciona la funcionalidad necesaria.

Casos de Uso de Portafirmas.

La aplicación web Portafirmas contemplará sólo un tipo de actor, el usuario de la aplicación que hará uso de la funcionalidad que ésta le proporciona. A continuación enumeraremos los casos de uso que el sistema ofrece.

Caso de Uso	Autenticación en Google Drive
Contexto	El usuario se autenticará para acceder a sus recursos que tendrá almacenados en Google Drive, para ello proporcionará su usuario y su clave.
Actores	Usuario de la aplicación
Precondiciones	El usuario debe tener una cuenta en Google.
Garantías mínimas	El usuario no está registrado o los datos proporcionados de validación contra Google no son válidos
Garantías de éxito	Obtención de token de acceso para operar en Google Drive.
Escenario de éxito principal	<ol style="list-style-type: none">1. Google solicita credenciales de validación al usuario.2. El sistema comprueba la validez de sus credenciales.3. Google solicita permisos de scopes al usuario para el servidor de Portafirmas.4. Se muestra listado del contenido de archivos que el usuario que tiene almacenados en Google Drive.

Caso de Uso	Búsquedas de archivos PDF Google Drive
Contexto	El usuario busca y recupera listados de sus archivos almacenados en Google Drive, dispondrá filtros para hacer la búsqueda más selectiva.
Actores	Usuario de la aplicación
Precondiciones	El usuario deberá estar autenticado y disponer de un token de acceso válido para acceder a la API de Google Drive.
Garantías mínimas	Listado vacío en caso de no existir documentos que coincidan con la búsqueda.
Garantías de éxito	Listado de documentos que coinciden con la búsqueda.
Escenario de éxito principal	<p>El usuario se encuentra en la pantalla de listado de archivos.</p> <p>El sistema muestra filtros para realizar una búsqueda más selectiva.</p> <p>El usuario selecciona el filtro que necesite.</p> <p>El sistema recuperará listado de archivos acorde con el filtro seleccionado.</p>

Caso de Uso	Registrar Entidad de Firma en TrustedX eIDAS
Contexto	El usuario creará una nueva entidad de firma con una PKCS#12.
Actores	Usuario de la aplicación.
Precondiciones	El usuario debe estar autenticado previamente en el IdP y disponer de una PKCS#12 válida para poder realizar el registro.
Garantías mínimas	Mensaje de no haber podido crear la entidad de firmas y el motivo.
Garantías de éxito	Nueva entidad de firmas.
Escenario de éxito principal	<ol style="list-style-type: none"> 1. Si el usuario no está autenticado en el sistema, el IdP le solicita sus credenciales. 2. El servidor de autorización solicita permisos de scopes al usuario para el servidor de Portafirmas. 3. El sistema crea la nueva entidad de firma y muestra por pantalla que la creación ha sido exitosa.

Caso de Uso	Eliminar Entidad de Firma en TrustedX eIDAS
Contexto	El usuario eliminará una entidad de firma propiedad de éste.
Actores	Usuario de la aplicación.
Precondiciones	El usuario debe estar autenticado previamente en el IdP y existir la entidad de firma a eliminar.
Garantías mínimas	Mensaje de no haber podido eliminar la entidad de firmas y el motivo.
Garantías de éxito	Entidad de firma eliminada.
Escenario de éxito principal	<ol style="list-style-type: none"> 1. Si el usuario no está autenticado en el sistema, el IdP le solicita sus credenciales. 2. El usuario recibe un listado de sus entidades de firma. 3. El usuario selecciona la entidad de firma a eliminar y solicita su eliminación. 4. El sistema informa que la eliminación fue exitosa.

Caso de Uso	Listar Entidades de Firma disponibles en TrustedX eIDAS
Contexto	El usuario recuperará un listado de sus entidades de firma
Actores	Usuario de la aplicación
Precondiciones	El usuario debe estar autenticado previamente en el IdP.
Garantías mínimas	Mensaje indicando que el usuario no tiene entidades de firma registradas.
Garantías de éxito	Listado de sus entidades de firma.

Escenario de éxito principal	<ol style="list-style-type: none"> 1. Si el usuario no está autenticado en el sistema, el IdP le solicita sus credenciales. 2. El sistema muestra listado de entidades de firma registradas.
------------------------------	--

Caso de Uso	Creación de Flujo de firmas de documento PDF.
Contexto	El usuario firma un documento y lo comparte con otros usuarios para sean firmados por ellos creando un flujo de firmas.
Actores	Usuarios de la aplicación.
Precondiciones	El usuario debe estar autenticado previamente en Google Drive para acceder al documento PDF a firmar.
Garantías mínimas	Mensaje indicando que el usuario no ha podido realizar la acción con éxito y su motivo.
Garantías de éxito	Documento PDF firmado y compartido creando un flujo de firmas.
Escenario de éxito principal	<ol style="list-style-type: none"> 1. Si el usuario no está autenticado en el sistema, el IdP le solicita sus credenciales. 2. El usuario busca y selecciona el documento para firmar desde Google Drive. 3. El usuario selecciona otros usuarios con los que compartirá el documento para que estos a su vez lo firmen. 4. El usuario seleccionará la entidad de firma a utilizar. 5. El usuario manda a firmar el documento. 6. TrustedX solicitará autorización al usuario para el uso de la entidad de firma. 7. El sistema genera la firma en el documento y lo compartirá con los usuarios seleccionados y estos serán notificados.

Caso de Uso	Firma de documento PDF perteneciente a un flujo de firmas.
Contexto	El usuario firma un documento perteneciente a un flujo de firmas generado previamente por otro usuario.
Actores	Usuarios de la aplicación.
Precondiciones	El usuario debe estar autenticado en Google Drive y tendrá una notificación informándole que tienen pendiente una firma de documento.
Garantías mínimas	Mensaje indicando que el usuario no ha podido realizar la acción con éxito y su motivo.
Garantías de éxito	Documento PDF firmado.
Escenario de éxito principal	<ol style="list-style-type: none"> 1. Si el usuario no está autenticado en el sistema, el IdP le

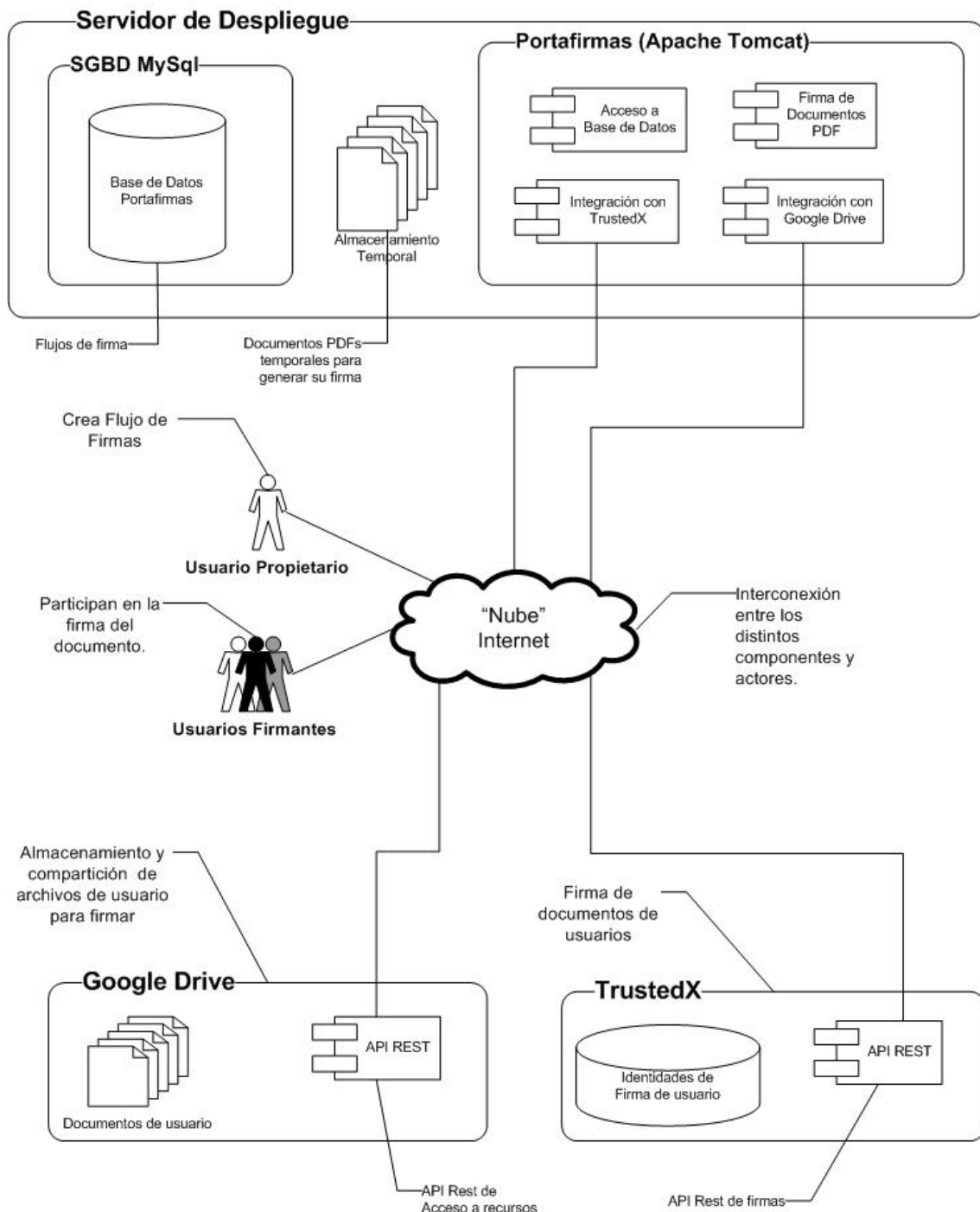
principal	<p>solicita sus credenciales.</p> <ol style="list-style-type: none"> 2. El usuario seleccionará la entidad de firma a utilizar. 3. El usuario manda a firmar el documento compartido. 4. TrustedX solicitará autorización al usuario para el uso de la entidad de firma. 5. El sistema genera la firma en el documento y notificará al usuario que generó el flujo de firmas que ya fue firmado por este.
-----------	---

Caso de Uso	Estado de flujo de firmas de documento compartido.
Contexto	El usuario podrá comprobar el estado de un flujo de firmas sobre un documento compartido, verá si los destinatarios han firmado o no y podrán comprobar el estado del flujo.
Actores	Usuarios de la aplicación.
Precondiciones	El usuario debe estar autenticado previamente en Google Drive.
Garantías mínimas	Mensaje indicando que no se ha podido consultar el estado del flujo de firmas y su motivo.
Garantías de éxito	Estado del flujo de firmas.
Escenario de éxito principal	<ol style="list-style-type: none"> 1. El usuario listará los flujos de firma que ha generado. 2. El usuario seleccionará el flujo y comprobará el estado actual de este.

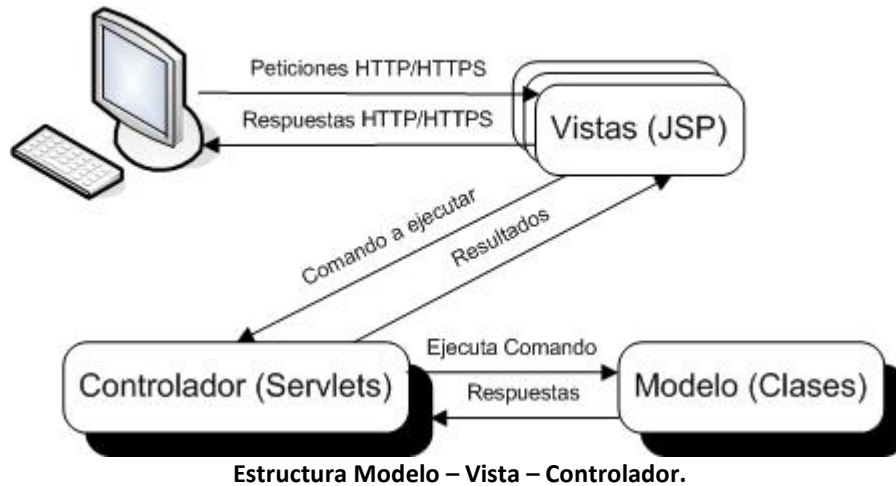
Caso de Uso	Cancelación de flujo de firmas de documento.
Contexto	El usuario podrá cancelar un flujo de firmas que haya creado previamente.
Actores	Usuarios de la aplicación.
Precondiciones	El usuario debe estar autenticado previamente en Google Drive.
Garantías mínimas	Mensaje indicando que no se ha podido cancelar el flujo de firmas y su motivo.
Garantías de éxito	Flujo de firmas cancelado
Escenario de éxito principal	<ol style="list-style-type: none"> 1. El usuario listará los flujos de firma que ha generado. 2. El usuario seleccionará el flujo y lo cancelará. 3. El sistema informará que el flujo ha sido cancelado correctamente. 4. Los usuarios implicados en el flujo de firma serán informados de la cancelación.

Diseño del sistema.

El sistema en general está compuesto por subsistemas interconectados entre sí a través de la nube, Portafirmas es accesible a los usuarios a través de la Web, no obstante cabe la posibilidad de desplegarse Portafirmas dentro de una intranet siempre y cuando sea posible el acceso exterior hacia los servicios de Google Drive y TrustedX



La arquitectura empleada para implementar la aplicación Portafirmas se basa en patrón de diseño de arquitecturas Modelo – Vista – Controlador (MVC) [19], los datos y la lógica de negocio se hace independiente del interfaz de usuario. Empleando esta estructura se facilita la reutilización de código, la división en módulos independientes y funcionales, y la escalabilidad, de forma que hace más fácil la ampliación y el mantenimiento del sistema.

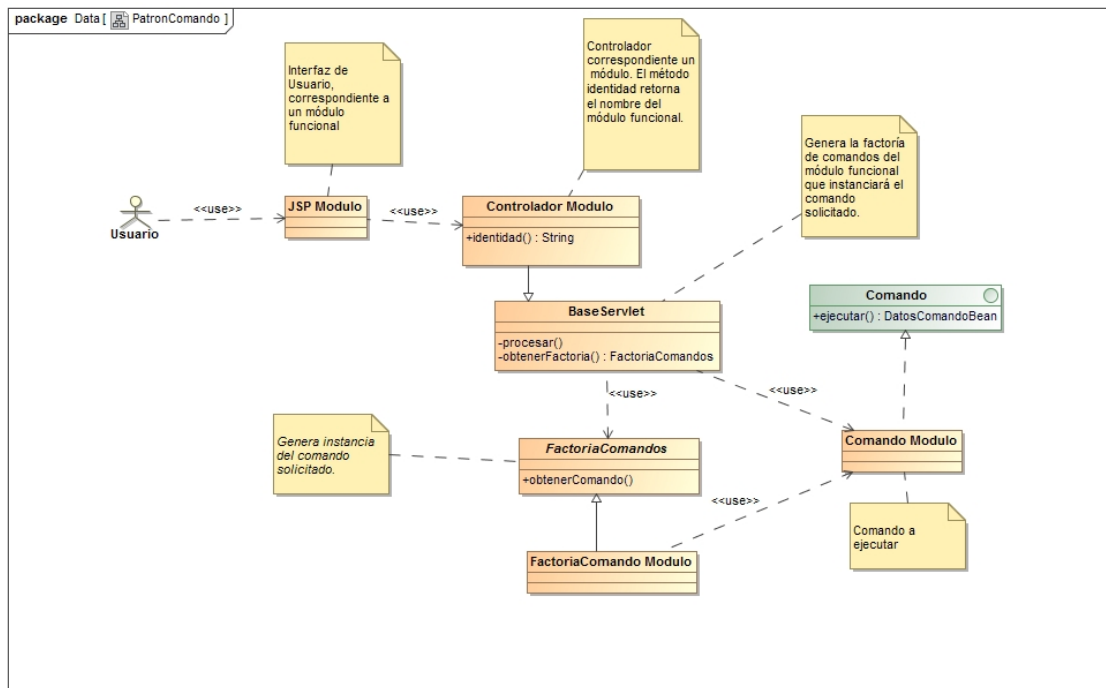


Para llevar a cabo este diseño, se emplea la arquitectura Java Enterprise Edition (Java EE), el sistema es una aplicación contenida en un servidor de aplicaciones que soporte Java EE. Los principales elementos que componen la estructura son los siguientes:

- Las vistas son una serie de JavaServer Pages (JSP) con las que el usuario interactúa con el sistema. Envían y reciben datos de los controladores.
- Los controladores son una serie de servlets que interactúan con las vistas y el modelo para realizar las acciones requeridas por el usuario.
- El modelo está compuesto por una serie de clases que conforman la lógica de negocio, acceden al gestor de base de datos del sistema y al sistema de archivos.

Gracias a esta arquitectura se facilita la división en módulos funcionales independientes y con capacidad de interactuar entre ellos. Cada módulo dispone de sus vistas, sus controladores y sus clases de modelo. Para ello se

emplea un framework con un comportamiento similar al patrón de diseño Comando, la interacción de cada JSP con un servlet controlador correspondiente hace que este último invoque un comando encapsulado en una clase enviándole los parámetros necesarios para la ejecución, y reciba a los resultados y la vista donde a mostrar al usuario. El servlet controlador hará uso de una factoría de clases para instanciar la clase del comando solicitado.



Patrón Comando.

La arquitectura facilita la escalabilidad. La creación de nuevos módulos funcionales implica la creación de las vistas con su correspondiente controlador de módulo y sus comandos.

Los módulos funcionales del sistema son los siguientes:

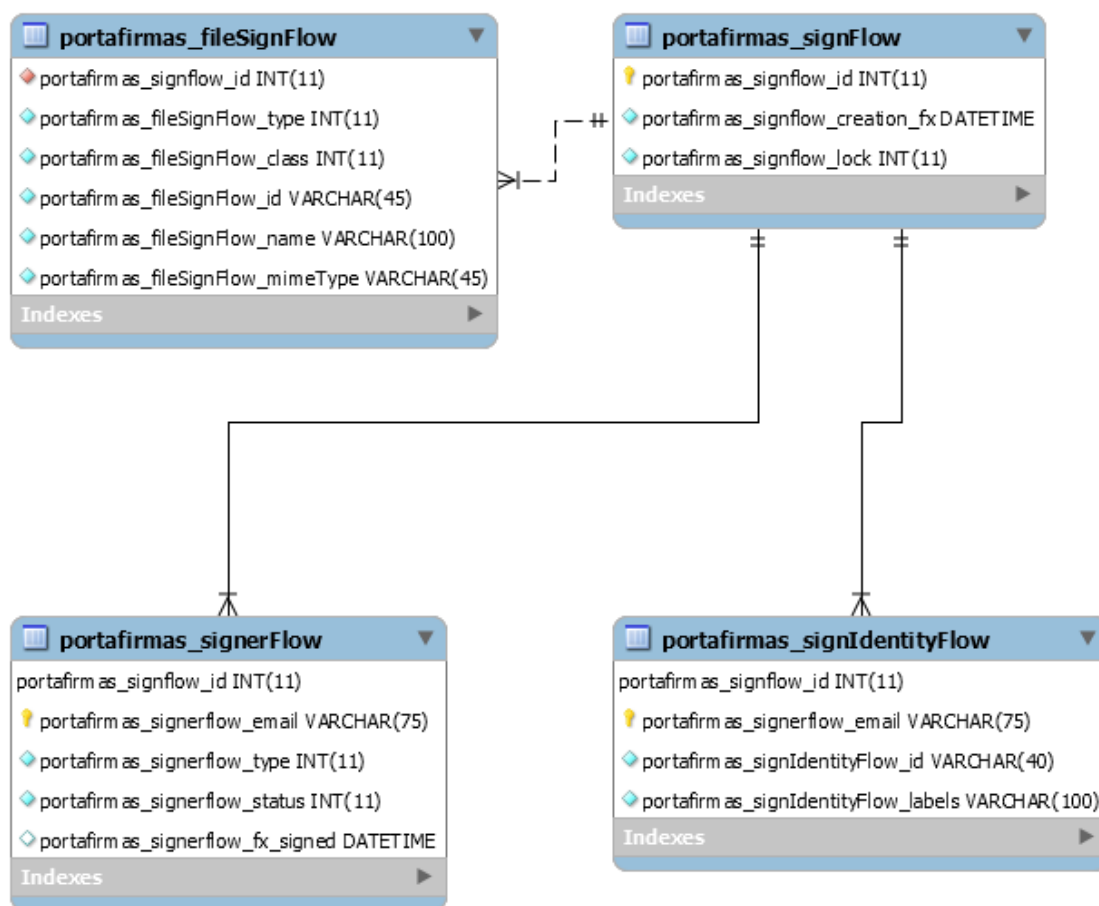
- Documentos, donde se gestiona el acceso a los documentos del usuario almacenados en Google Drive, desde ahí se visualiza sus directorios y documentos en formato PDF y se crean flujos de firma sobre éstos
- TrustedX, donde se realiza una pequeña gestión de las identidades de firma del usuario, pudiendolas listar, eliminar y crear.

La funcionalidad de cada módulo está contenida en paquetes independientes con sus distintos comandos y acciones tales como acceso a base de datos y al sistema de archivo. Toda la infraestructura del patrón comando se encapsula un componente que dará servicio a cada módulo funcional. A su vez se dispone

de componentes con clases para dar soporte a los módulos funcionales, llamadas a los servicios REST de Google Drive y TrustedX, acceso a base de datos, y creación de archivos de logs.

Modelo de Base de Datos.

Como se ha comentado anteriormente, Portafirmas utiliza MySql como gestor de base de datos. Cada módulo funcional tendrá clases de acceso a las tablas que almacenan la información que serán empleadas por las clases comando. La gestión de conexión a la base de datos se realizará a través de un pool de conexión que proporciona el servidor de aplicaciones que contendrá la aplicación web. El modelo Entidad Relación de la base de datos del sistema es el siguiente:



A continuación describiremos la tablas anteriores:

- Tabla **portafirmas_signFlow**: contiene los flujos de firma creados, cada flujo tiene un identificador único, el usuario que lo creó (propietario), la fecha de creación, y un campo de bloqueo utilizado para el acceso exclusivo del archivo a firmar por parte de un firmante.

- Tabla **portafirmas_signerFlow**: contiene los firmantes involucrados en un flujo de firma, si es o no el propietario del flujo, el estado de su firma (pendiente o firmada) y la fecha en que realizó la firma.
- Tabla **portafirmas_signIdentityFlow**: contiene las identidades de firma empleadas por cada firmante para firmar el flujo correspondiente.
- Tabla **portafirmas_fileSignFlow**: contiene los archivos involucrados en la firma, serán siempre dos, el archivo original y el archivo firmado. Lleva asociado un identificador que lo relaciona con Google Drive de forma que pueda interactuarse con éste en la nube.

Flujos de Firma.

Desde el punto de vista de un usuario se distinguen dos tipos de flujos de firma:

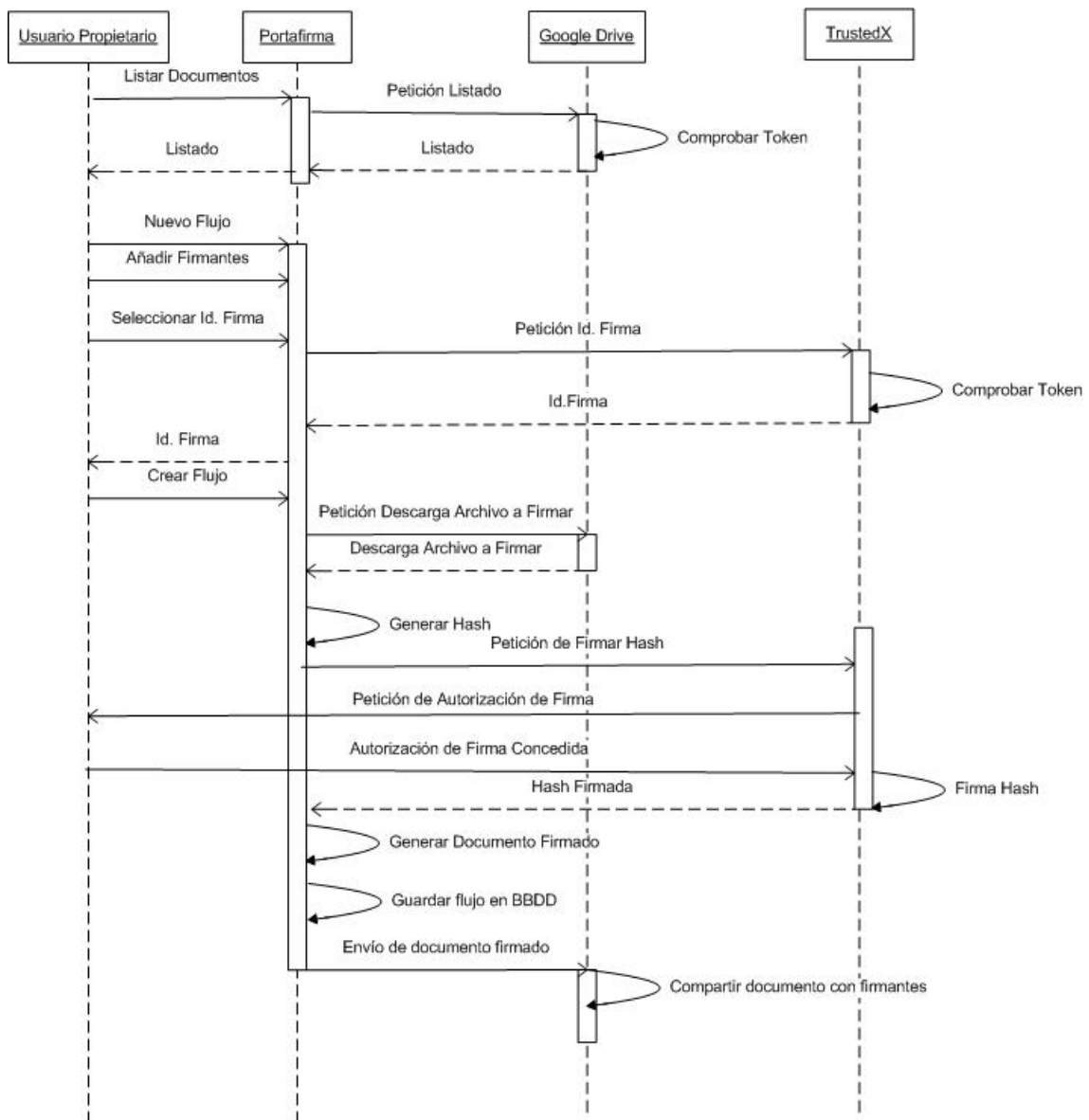
1. **Flujo propietario**, es aquel flujo que un usuario crea, éste será propietario del flujo y será compartido por el resto de los usuarios firmantes.
2. **Flujo de tercero**, es aquel flujo que un usuario participa como firmante no habiéndolo creado éste.

Todo flujo de tercero es flujo propietario del usuario que lo creó. A continuación se describirá el proceso de creación de los distintos flujos de firmas:

- **Flujo propietario.**
 - Un usuario validado recupera el listado de sus documentos en Google Drive.
 - Selecciona el documento a firmar y se le mostrará una pantalla donde añadirá el resto de los usuarios firmantes involucrados en el flujo.
 - Seleccionará la identidad de firma a utilizar, si previamente no se ha autenticado contra TrustedX se pide que se realice esta acción obteniendo un token de acceso.
 - Generación de la hash (bytes resumen) del documento y envío de TrustedX para su firma que a su vez pedirá al usuario autorización implícita para que se firme la hash.

- Recepción de la hash firmada, incorporación de esta al documento de a firmar generando así la versión firmada.
- Subida del documento a Google Drive para compartirlo con el resto de los firmantes.
- Registro en base de datos del nuevo flujo. El flujo permanecerá en estado de pendiente de firma hasta que todos los firmantes hayan firmado el flujo.

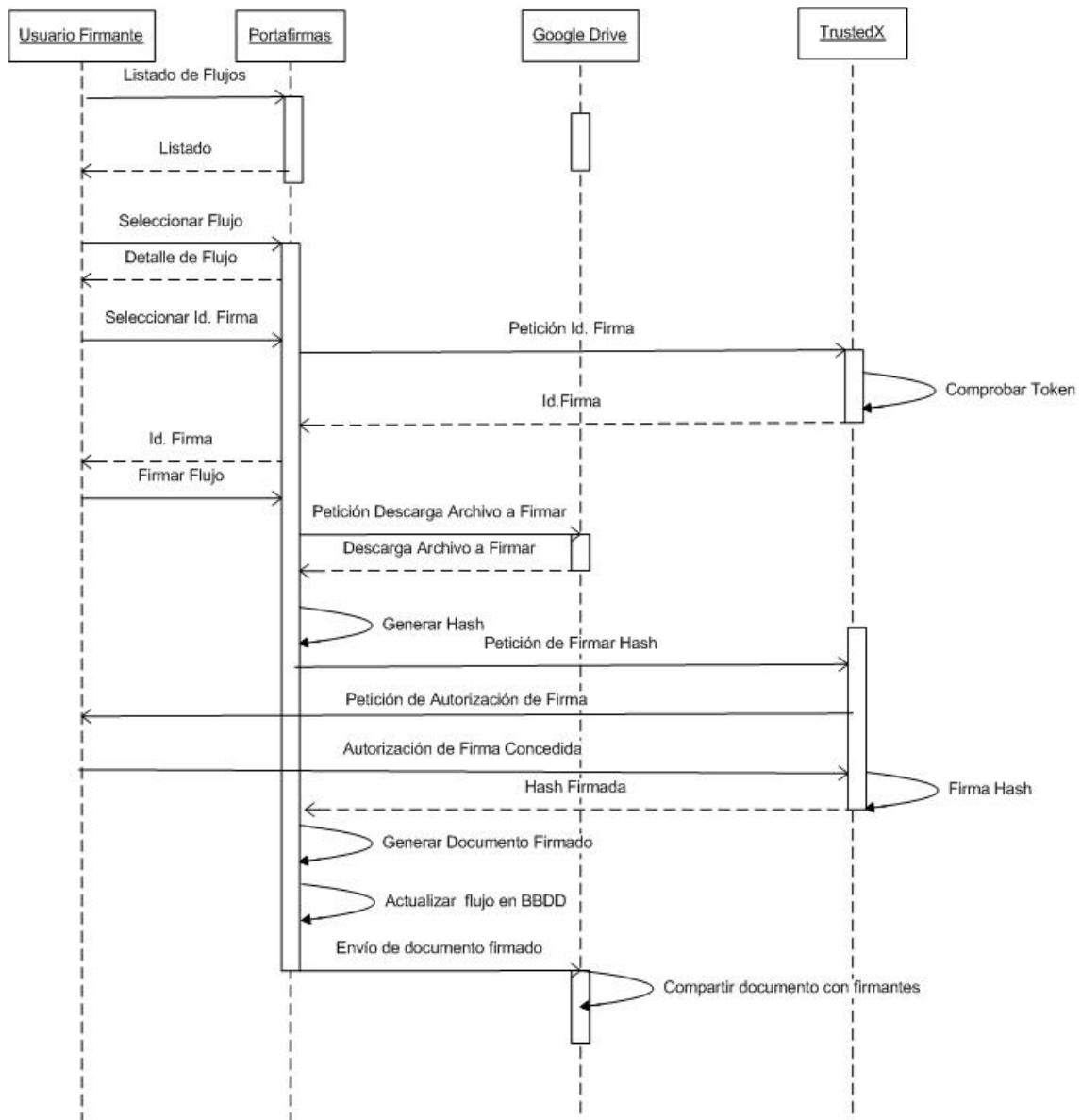
El siguiente diagrama de secuencias muestra la interacción entre el usuario propietario y los distintos procesos anteriormente descritos:



- **Flujo de tercero**

- Un usuario firmante accede al sistema y verá un listado de los flujos de tercero donde está involucrado como firmante.
- Seleccionará el flujo pendiente de su firma y se le mostrará la información de éste.
- Seleccionará la identidad de firma a utilizar, si previamente no se ha autenticado contra TrustedX se pide que se realice esta acción obteniendo un token de acceso.
- Generación de la hash (bytes resumen) del documento y envío de TrustedX para su firma que a su vez pedirá al usuario autorización implícita para que se firme la hash.
- Recepción de la hash firmada, incorporación de esta al documento de a firmar generando así la versión firmada.
- Subida del documento a Google Drive.
- Cambio de estado de firma pasando de pendiente de firmar a firmado. Si todos los firmantes han firmado el flujo este cambiará al estado de firmado.

El siguiente diagrama de secuencias muestra la interacción entre el usuario propietario y los distintos procesos anteriormente descritos:



Clases de acceso a los servicios de Google Drive y TrustedX.

Se ha implementado el acceso a los servicios REST de las APIS de OAuth2.0, tanto de Google Drive como de TrustedX. Para ello se han encapsulado las llamadas necesarias de los servicios descritos en los anteriores apartados en las siguientes clases:

1. Clase **UtilGDrive**, para integración con la API de Google Drive.
2. Clase **UtilTrustedX**, para integración con la API de TrustedX.

Estas clases crean conexiones por HTTPS a través de una la clase auxiliar **HttpMethods** que a su vez emplea la clase **org.apache.http.client** para ejecutar métodos post, get, put y delete:

- org.apache.http.client.methods.HttpDelete.
- org.apache.http.client.methods.HttpGet.
- org.apache.http.client.methods.HttpPost.
- org.apache.http.client.methods.HttpPut.

En las siguientes líneas de código se ejemplifica el uso de uno de los anteriores objetos, en concreto la ejecución de un post:

```
String post(String url, Map<String, String> formParameters, HeaderBean header,
JSONObject oBody)
```

A grandes rasgos el código asociado al método tras su simplificación sería el siguiente:

```
HttpPost request = new HttpPost(url);
// Establecer cabecera
if (header.getAuthorization() != null) {
    request.setHeader("Authorization", header.getAuthorization());
}
if (header.getContentType() != null) {
    request.setHeader("Content-Type", header.getContentType());
}
if (header.getContentLength() != null) {
    request.setHeader("Content-Length", header.getContentLength());
}
if (header.getAccept() != null) {
    request.setHeader("Accept", header.getAccept());
}
// Establecer parámetros
List<NameValuePair> nvps = new ArrayList<NameValuePair>();
for (String key : formParameters.keySet()) {
    nvps.add(new BasicNameValuePair(key, formParameters.get(key)));
}
request.setEntity(new UrlEncodedFormEntity(nvps));
// Establecer datos de envío
StringEntity requestEntity = new StringEntity(
    oBody.toString(),
    contentType.APPLICATION_JSON);

request.setEntity(requestEntity);
// Ejecutar el método
HttpClient httpClient = new DefaultHttpClient();
HttpResponse response = httpClient.execute(request);
HttpEntity entity = response.getEntity();
// Recuperar la respuesta
String sBody = EntityUtils.toString(entity);
```

En las líneas de código anteriores se puede observar el paso de parámetros de por cabecera y por URL, y el envío de información adicional en formato de objeto JSON. En respuesta se recibirá un objeto String que será parseado para interpretar la respuesta, por ejemplo, recibir otro objeto JSON como respuesta,

por ejemplo, un descriptor de archivo en Gogole Drive o información de una identidad de firma en TrustedX.

Las siguientes líneas de código ejemplifica el uso del anterior método post para realizar una petición del token de acceso para TrustedX:

```
/*
    Método que solicita token de acceso a
    partir de un código de autorización.
    Parámetro code: Código de autorización
    Parametro sCallBack: URL de callback
*/
public static String getAccessToken (String sCode, String sCallBack)
    throws Exception {

String sAccesToken = null;

try{
    //comprobar parámetros de entrada
    if (sCode == null || sCode.trim().isEmpty()) {
        throw new Exception ("Código de validación a nulo o vacío.");
    }
    if (sCallBack == null || sCallBack.trim().isEmpty()) {
        throw new Exception ("Callback a nula o vacío.");
    }
    // Generamos los parámetros URL
    LinkedHashMap<String, String> params =
        new LinkedHashMap<String, String>();

    params.put("grant_type", "authorization_code");
    params.put("code", sCode);
    params.put("redirect_uri", URL_SERVER + sCallBack);

    //Generamos el header
    HeaderBean header = new HeaderBean ();

    byte[] encodedBytes = Base64.getEncoder().encode(
        (URLEncoder.encode(TRUSTEDX_CLIENT_ID, "UTF-8") + ":" +
        URLEncoder.encode(TRUSTEDX_CLIENT_SECRET,
        "UTF-8")).getBytes());

    String sBasic = new String (encodedBytes);

    header.setAuthorization("Basic " + sBasic);
    header.setContentType("application/x-www-form-urlencoded");

    // Solicitamos el token
    HttpMethods http = new HttpMethods (HttpMethods.TRUSTEDX);
    String sBody = http.post(
        TRUSTEDX_SERVER + "/trustedx-authserver/oauth/main/token",
        params, header, null);

    JSONObject object = new JSONObject(sBody);
    sAccesToken = (String) object.getString("access_token");

    if (sAccesToken == null || sAccesToken.trim().isEmpty()) {
        throw new Exception ("Token de acceso nulo o vacío");
    }
}
catch (Exception e) {
    throw new Exception (CLASE + ".getAccessToken() "+e.toString ());
}
return sAccesToken;
}
```

Básicamente el resto de los métodos mencionados anteriormente siguen un esquema similar, a grandes rasgos se trata de establecer la cabecera, los parámetros de la URL, y si procede datos adicionales como el archivo a subir, y emplear el método correspondiente que proporciona el objeto cliente disponible en la clase **org.apache.http.client**.

Proceso de firma de archivo PDF mediante PAdES.

El formato de un archivo PDF (Portable File Document) es una estructura compuesta basada en una simplificación del lenguaje PostScript. Contiene diversa información asociada al documento que contiene, como por ejemplo las fuentes que utiliza, enlaces, marcadores, contenido multimedia, etc. Consta de un encabezado (header) que identifica el formato, comenzando por la cadena %PDF y el número de la versión, seguido por una colección de objetos identificados por un número, tablas de referencias cruzadas y la parte final (trailer). Entre estos objetos destacamos el diccionario de firmas identificado por la etiqueta /Contents y empleado para almacenar la firma asociada al documento.

Para realizar la firma sobre un documento PDF se ha seguido el formato PAdES (PDF Advanced Electronic Signature) definido en la ISO 32000-1, norma que especifica cómo realizar firmas digitales de archivos PDFs de forma que se pueda autenticar la identidad del firmante y garantizarla identidad del documento. Los datos de la firma de un documento se incorporan directamente en el propio documento firmado de forma que el contenido del PDF se pueda distribuir como un simple archivo electrónico. La firma puede a su vez tener una representación visual como un campo de un formulario. Para la verificación de la firma es necesario lo siguiente:

- Comprobar la integridad de los datos firmados para tener la certeza de que no se han producido modificaciones.
- Comprobar que el certificado usado para la firma era correcto, es decir, que estaba vigente durante la firma.

El formato PAdES posibilita la incorporación de mecanismos donde se añade información adicional a la firma para garantizar la validez de la firma a largo plazo incluso una vez vencida la validez del certificado empleado. PAdES dispone de diferentes versiones que van incrementando la validez de la firma con plenas garantías jurídicas.

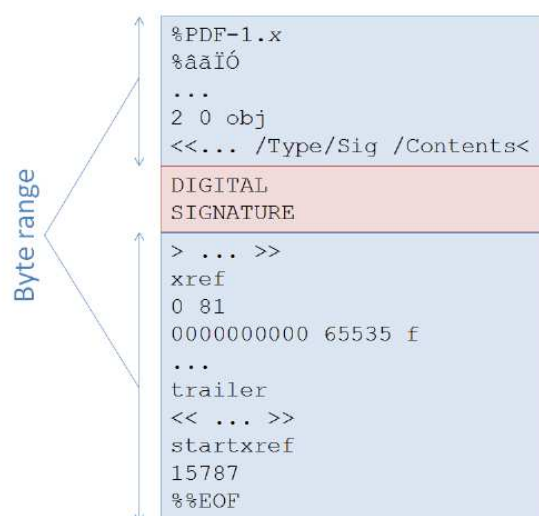
Gracias al sellado de tiempo se puede verificar que un conjunto de datos existió antes de un momento dado y que ninguno de esos datos han sido modificados, es necesario disponer de una Autoridad de Sellado de Tiempo que será una autoridad de confianza dando fe de la veracidad de dichos datos.

Las firmas digitales en ISO 32000-1 admiten tres acciones:

- Agregar una firma digital inmediatamente a un documento.
- Proporcionar un campo de marcador de posición donde irán las firmas en el futuro.
- Verificar la validez de las firmas.

La firma en sí, junto con diversa información opcional, está contenida en una estructura de datos del PDF llamada diccionario de firma. El valor de una firma es un objeto binario codificado, se genera a través de un resumen (digest) sobre un rango de bytes del archivo, siendo el rango el archivo al completo sin incluir la propia firma pero sí el diccionario de firma. Dicho rango se indica mediante la entrada ByteRange del diccionario de firmas. Al restringir la entrada del ByteRange se garantiza que no existan bytes en el PDF que no estén cubiertos por el resumen.

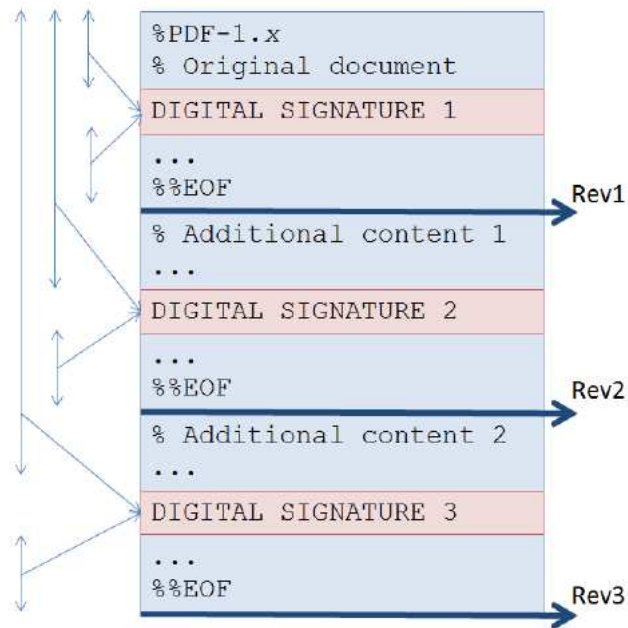
En la siguiente imagen se puede apreciar dónde iría almacenada la firma dentro de la estructura de un archivo PDF, en concreto dentro de la entrada Contents de diccionario de firma:



La firma digital almacenada estará compuesta por la siguiente información:

- Certificado, compuesto a su vez por la clave pública y la identidad del firmante.
- Resumen firmado por la clave privada del firmante.
- Timestamp o sello temporal.

Para el caso de firmas múltiples sobre un documento PDF la estructura interna sería la siguiente:



Portafirmas sigue el anterior esquema para realizar la firma digital sobre un documento PDF. Los pasos a grandes rasgos del proceso de firma son los siguientes:

1. Descarga de documento PDF original desde Goggle Drive hacia sistema de archivos del servidor de Portafirmas.
2. Generación del resumen (digest) del archivo PDF, dicho resumen será el hash a firmar para incorporar en el archivo firmado.
3. Envío de la hash a TrustedX junto a la identidad de firma que será utilizada en la firma.
4. Recepción del hash firmado por TrustedX, para ello la firma será realizada bajo el estándar criptográfico RSA (PKCS#1).
5. Generación del archivo PDF firmado incorporando la hash recibida diccionario de firmas tal y como describe el formato PAdES.
6. Subida del archivo firmado a Google Drive, compartir con el resto de los usuarios si procede.

Los pasos anteriores del 2 al 5 relacionados con la firma en sí son realizados por la clase **Signer** a través del método **sign**. Para ellos se emplea la biblioteca **iText** [20] de código abierto dedicada a la creación y manipulación de archivos PDF. Dicho método recibe los siguientes parámetros:

- String src, path y nombre del archivo origen PDF a firmar.
- String dest, path y nombre del archivo PDF firmado.
- SignIdentityBean signIdentity, objeto identidad de firma de TrustedX.
- String sAccessToken, token de acceso a TrustedX.
- Boolean fAppend, si true se emplea en la concatenación de nuevas firmas a un documento firmado, empleado según sea una firma simple o una firma múltiple.
- Integer nDesplazamiento, empleado para emplazar la firma visible en el documento firmado.

El objeto de firma proporcionado por la librería **iText** es de tipo **PdfSigner**, este objeto precisa que le indiquemos el dispositivo de firma externo que firmará la hash, se trata de un dispositivo externo ya que TrustedX externamente firmará el resumen, un dispositivo externo también podría ser una tarjeta inteligente, como por ejemplo el DNle. Es necesario implementar para ello una clase que implementa la interfaz de tipo **IExternalSignature** que en última instancia generará la firma para que el objeto PdfSigner la incorpore en el archivo firmado. Además se hace uso de un objeto **BouncyCastleProvider** que es proveedor de APIs criptográficas, es también de libre distribución aunque es necesario el uso de distintas librerías adicionales para que lo cumplimente.

```
public static void sign(String src, String dest, SignIdentityBean
singIdentity, String sAccessToken, boolean fAppend, Integer nDesplazamiento)
throws GeneralSecurityException, IOException, Exception {

    FileOutputStream output = null;
    PdfReader reader = null;
    try{
        BouncyCastleProvider provider = new BouncyCastleProvider();
        Security.addProvider(provider);
        reader = new PdfReader(src);
        output = new FileOutputStream(dest);
        PdfSigner signer = new PdfSigner(reader, output, fAppend);
        // Sello de firma a mostrar en el PDF firmado
        PdfSignatureAppearance appearance=signer.getSignatureAppearance()
            .setReuseAppearance(false);
        Rectangle rect = new Rectangle(
            10 + (100 * nDesplazamiento), 400, 100, 100);
        appearance.setPageRect(rect).setPageNumber(1);
        appearance.setRenderingMode(
```

```

        PdfSignatureAppearance.RenderingMode.NAME_AND_DESCRIPTION);
signer.setFieldName(signer.getNewSigFieldName());
// Uso del certificado asociado a la identidad de firma
Certificate[] chain = new Certificate[1];
chain[0] = convertToX509Cert (singIdentity.getCertificate());
// Objeto de dispositivo de firma externa, se inicializa
// con el objeto de identidad de firma y el token de acceso a
// TrustedX ya que se delega la firma en TrustedX empleando su
// llamada correspondiente.
IExternalSignature pks = new MyExternalSigner(singIdentity,
        sAccesToken);
ProviderDigest digest = new ProviderDigest(provider.getName());
// Ejecución de la firma
signer.signDetached(digest, pks, chain, null, null, null, 0,
        PdfSigner.CryptoStandard.CADES);
}
catch (Exception e) {
    throw new Exception(CLASE + ".sing () " + e.toString());
}
finally {
    if (output != null){
        output.close();
    }
    if (reader != null){
        reader.close();
    }
}
}
}
}

```

Por último se mostramos el código asociado al dispositivo de firmas externo implementado en la clase **MyExternaSigner** en e método **sign**:

```

public byte[] sign(byte[] arg0) throws GeneralSecurityException {
    // Generación del hash resumen (digest) a firmar
    MessageDigest digest = MessageDigest.getInstance("SHA-256");
    byte[] encodedhash = digest.digest(arg0);
    // Codificar el hash resumen a base 64 necesario para TrustedX
    String sHash = Base64.encodeBytes(encodedhash);
    byte[] bodyBinary; // Valor de retorno de hash encriptado

    try {
        // Llamada a la API Rest de TrustedX para firmar la hash
        bodyBinary = UtilTrustedX.signHash(SIGN_IDENTITY.getId(),
                sHash, ACCESS_TOKEN);
    }
    catch (Exception ex) {
        throw new GeneralSecurityException ("MyExternalSigner.sign: + " +
                ex.toString());
    }
    // Retorno de la hash firmada
    return bodyBinary;
}
}

```

4. Conclusiones

Cáda vez es más frecuente el uso de las firma digital para autenticar la validez y el origen de un documento, especialmente en las administraciones públicas donde sus empleados y empresas que tienen algún tipo de relación contractual, con éstas tales como proveedores, se les exige emplear este tipo de firmas en

lugar de la manuscrita. Además está el hecho de la existencia de regulación de este tipo de firmas a nivel de Estados Miembros a través del reglamento eIDAS. Dada la relativa complejidad para realizar una firma digital, la aparición de plataformas que faciliten la realización de éstas y la compartición de archivos firmados será de gran ayuda para cualquier tipo de usuarios. Se podrá evitar el uso de complejos sistemas para aquellos usuarios no familiarizados con la firma digital, se abre muchas oportunidades para dar solución a estos requerimientos.

Durante el desarrollo de este TFM se ha visto cómo el estándar OAuth2.0 facilita la integración con los distintos sistemas que lo implementan, siendo muy seguros aunque algo complejos de entender si no se está familiarizado con las APIS de tipo REST. Google Drive ofrece la posibilidad de integrarse directamente con librerías desarrolladas para los distintos entornos de desarrollo, en este caso existe la posibilidad de emplear una SDK para Java, no obstante la escasa documentación de uso y los ejemplos poco claros se ha decidido emplear las llamadas directas a las URLs que ofrecen las diferentes funcionalidades para operar sobre la plataforma. En contraposición TrustedX no dispone de SDKs para integrarse con los habituales entornos de desarrollo, sin embargo la documentación clarifica mucho el uso de las llamadas HTTP para poder interactuar con el sistema.

La parte más complicada en el desarrollo de este TFM fué entender el estándar OAuth2.0 sus conceptos, sin embargo su aplicación los casos prácticos de Google Drive y TrustedX facilitaron su comprensión.

Inicialmente este TFM podría haber incluido la integración con otro sistema de almacenamiento en la nube como es DropBox pero la planificación se hubiera visto comprometida por el tiempo empleado en su estudio centrandose más en en la API de Google Drive y las posibilidades que nos ofrece.

Portafirmas se ha implementado de forma que sea escalable para incorporar módulos de acceso a otros sistemas de almacenamiento como el mencionado DropBox, ya que este sistema, al cumplir con el estándar OAuth2.0, es de fácil integración con Portafirmas.

En futuros desarrollos se podría seguir con esta línea, incorporar nuevos sistemas de almacenamiento en la nube e incluso hacer que Portafirmas en sí sea un sistema de almacenamiento en la nube ya que se trata de una

aplicación web desplegada en un servidor de aplicaciones con capacidad de almacenar los documentos PDFs de los usuarios del sistema.

5. Glosario

eIDAS: es un nuevo Reglamento Europeo que regula la identificación electrónica y establece las pautas para los servicios de confianza relativos a las transacciones electrónicas.

Firma electrónica: La firma electrónica es un concepto jurídico, equivalente electrónico al de la firma manuscrita, donde una persona acepta el contenido de un mensaje electrónico a través de cualquier medio electrónico válido.

Firma electrónica avanzada: La firma electrónica avanzada es un conjunto de datos que se adjuntan a un mensaje electrónico, cuyo propósito es identificar al firmante como autor de ésta de manera única, como si se tratara de una firma manuscrita.

Firma electrónica cualificada: La firma electrónica cualificada es una firma electrónica avanzada pero su creación se realiza mediante un dispositivo cualificado de firmas electrónicas que cumple con unos requisitos especiales enumerados en el reglamento eIDAS.

HSM: Hardware Security Manager es un dispositivo criptográfico basado en hardware que genera, almacena y protege claves criptográficas y suele aportar aceleración hardware para operaciones criptográficas.

IdP: Proveedor de identidad (IdP por sus siglas en inglés), también conocido como Proveedor de confirmación de identidad, es una entidad con la autoridad de emitir información relacionada con la identidad de un sistema.

iText: iText es una biblioteca Open Source para crear y manipular archivos PDF, RTF, y HTML en Java.

Java Enterprise Edition: Plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

JSON: Formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje.

JSP: Tecnología de software a crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

MVC: Modelo-vista-controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

URL: Localizador uniforme de recursos (URL), denominado coloquialmente como una dirección web, es una referencia a un recurso Web que especifica su ubicación en una red de equipos y un mecanismo para recuperarla.

OAuth 2.0: Open Authorization es un estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas.

PAdES: PDF Advanced Electronic Signatures es un conjunto de restricciones y extensiones a PDF e ISO 32000-1 lo que es adecuado para la firma electrónica avanzada.

PDF: Portable Document Format, es un formato de almacenamiento para documentos digitales independiente de plataformas de software o hardware.

PKCS: PKCS (Public-Key Cryptography Standards) grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA en California.

Servicio REST: REST (Representational State Transfer): arquitectura que, haciendo uso del protocolo HTTP, proporciona una API que utiliza cada uno de sus métodos (GET, POST, PUT, DELETE, etcétera) para poder realizar diferentes operaciones entre la aplicación que ofrece el servicio web y el cliente.

6. Bibliografía

[1] Reglamento (UE) Nº 910/2014 del Parlamento Europeo y del Consejo, 23 de julio de 2014
<https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32014R0910&from=ES>

[2] Directiva 1999/93/CE
<https://www.boe.es/doue/2000/013/L00012-00020.pdf>

[3] Estándar Industrial Open Authorization 2.0 (OAuth2.0)
<https://oauth.net/2/>

[4] TrustedX eIDAS Platform
<https://www.safelayer.com/es/productos/trustedx-eidas-platform>

[5] API Rest de Google Drive
<https://developers.google.com/drive/api/v3/about-sdk>

[6] API Rest de TrustedX
https://uoc.safelayer.com:7080/es/html/TX_EIDAS_INTEGRATION/html/93986398.html

[7] Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público
<https://www.boe.es/buscar/doc.php?id=BOE-A-2015-10566>

- [8] Signaturit
<https://www.signaturit.com/es>
- [9] esFIRMA
<https://www.esfirma.com/index-ES.html>
- [10] CEN/TS 419241:2014 “Security Requirements for Trustworthy Systems supporting Server Signing”
<http://store.uni.com/catalogo/index.php/cen-ts-419241-2014.html>
- [11] Norma CEN TS 419 241 en firmas en la nube
https://docbox.etsi.org/workshop/2013/201303_SIGNATURES_IN_CLOUD/3b-CEN-Server-Signing.pdf
- [12] Firma electrónica PAdES (ETSI 102 778): firma de documentos PDF con TrustedX
<https://www.safelayer.com/es/recursos/67-articulos/firma-electronica/459-pades-etsi-102-778-firma-de-documentos-pdf-con-trustedx>
- [13] Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography
<https://tools.ietf.org/html/rfc3447>
- [14] Norma PDF 32000-1:2008
https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/PDF32000_2008.pdf
- [15] Java Enterprise Edition
<https://www.oracle.com/technetwork/java/javase/overview/index.html>
- [16] Gestor de base de datos MySQL
<https://www.mysql.com/>
- [17] Entorno de desarrollo NetBeans
<https://netbeans.org/>
- [18] Servidor de aplicaciones Apache Tomcat
<https://tomcat.apache.org/download-90.cgi>
- [19] Modelo Vista Controlador
<https://es.wikipedia.org/wiki/Modelo%20%80%93vista%20%80%93controlador>
- [20] Biblioteca Open Source iText
<https://itextpdf.com/en>

7. Anexos

Se adjunta el documento TFM_GabrielMorenoPerez_Anexo.pdf con instrucciones de despliegue y uso de Portafirmas.