Universitat Oberta de Catalunya

uoc.edu

Servicio de firma de documentos almacenados en la nube

David Cuenca García
Máster universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones

Juan Carlos Fernández Jara
Víctor García Font

30 de mayo de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Servicio de firma de documentos almacenados en la nube*
Nombre del autor:	*David Cuenca García*
Nombre del consultor/a:	*Juan Carlos Fernández Jara*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	05/2020
Titulación:	*Máster universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*Ad hoc*
Idioma del trabajo:	*Español*
Palabras clave	*CloudDocSign*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

En el mundo tecnológico actual donde los servicios son cada vez más descentralizados cobra vital importancia la gestión de la seguridad. Por ello, es cada vez más necesaria la existencia de aplicaciones que permitan unir los conceptos de movilidad y seguridad adaptados al mundo digital. La existencia del reglamento eIDAS (reglamento UE Nº 910/2014) abre el camino hacia esta transición en la seguridad digital. Entre otras cosas, plantea los fundamentos para una firma electrónica que será reconocida en todos los estados miembro de la UE.

Por ello, en este trabajo se define el desarrollo de una aplicación que implemente las pautas y fases necesarias para la realización de la firma electrónica cualificada en un entorno web. Se hará uso de un servicio de firma electrónica cualificada que dote de la funcionalidad para la firma adaptada al reglamento eIDAS. También hará uso de servicios de almacenamiento en la nube, como Google Drive y Dropbox, para la obtención y almacenamiento de los documentos objeto de firma. Esto dotará a la aplicación de un nivel de flexibilidad, utilidad y privacidad idóneo.

Así pues, se conseguirá que todos los servicios sean integrados para conseguir un producto que permita llevar la firma electrónica a los servicios de almacenamiento en la nube de la mayoría de usuarios.

Abstract (in English, 250 words or less):

In today's technological world where services are increasingly decentralized, security management becomes vitally important. For this reason, are necessary applications that unite the concepts of mobility and security adapted to the digital world. The existence of the eIDAS regulation (EU Regulation Nº 910/2014) opens the way towards this transition in digital security. Among other things, it lays the foundations for an electronic signature that will be recognized in all the member states of the EU.

Therefore, this work defines the development of an application that implements the guidelines and phases necessary to carry out the qualified electronic signature in a web environment. A qualified electronic signature service will be used that provides the functionality for the signature adapted to the eIDAS regulation. It will also make use of cloud storage services, such as Google Drive and Dropbox, to obtain and store the documents to be signed. This will provide to the application with an ideal level of flexibility, utility and privacy.

Thus, all the services will be integrated to achieve a product that allows the electronic signature to be carried to the cloud storage services of the majority of users.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente existe un auge importante del uso de sistemas digitales para la firma de documentos que están empezando a desbancar los métodos de firma tradicionales. Este hecho está provocando que se inicien nuevos desafíos en el ámbito de la informática para poder proveer de las mismas garantías que los sistemas tradicionales.

Otro gran reto reside en la globalización que el mundo digital está llevando a cabo haciendo necesario que cualquier sistema o protocolo que se desee implantar deba de ser consensuado y aceptado por la mayor cantidad posible de organismos y empresas internacionales. Más concretamente, en el apartado de la firma digital, existía una gran diversidad de interpretaciones que hacía complicado definir un mecanismo conjunto para la verificación y aceptación de una firma digital. Debemos tener presente que la firma digital debe proveer distintos servicios de seguridad como integridad, autenticidad y no repudio.

En este sentido, existe el Reglamento (UE) nº 910/2014 [1], conocido como eIDAS (Identificación electrónica y servicios de confianza) que define una serie de normas creadas por los países miembros de la Unión Europea para dotar a la firma electrónica de mecanismos y protocolos que permitan darle validez jurídica, siendo equivalente a una firma manuscrita. Este reglamento permite establecer una base muy importante para el uso de la firma electrónica frente a la convencional, allanando el camino para su uso por parte de todos los ciudadanos europeos en el ámbito personal y empresarial.

Por su parte, este reglamento introduce el concepto de firma cualificada que es aquella realizada con un certificado de firma electrónica cualificado que debe ser expedido por una Autoridad de Certificación validada. Estas bases serán las que permitirán crear un marco seguro y de confianza para poder aportar veracidad a la firma realizada en un entorno digital. A su vez, el reglamento también introduce dos tipos de escenarios para realizar la firma, local y centralizada. El primero consiste en una firma digital realizada desde el propio equipo del firmante sin hacer uso de componentes externos. Por su parte, el escenario de firma centralizada tiene un enfoque radicalmente opuesto, permitiendo que la firma digital sea efectuada desde cualquier lugar, haciendo uso de certificados electrónicos almacenados en un servidor seguro de Internet.

Sin embargo, para poder llevar a cabo la firma electrónica definida en la normativa eIDAS se hace necesario el uso de Dispositivos Cualificados de Firma (QSCD, Qualified Signature Creation Device), que no son más que software o hardware específico para esta finalidad. Son definidos en la normativa eIDAS en la que se indican una serie de requisitos que deberán satisfacer estos dispositivos para que los procesos de creación de firmas digitales sean seguros y bajo un marco de confianza establecido. Como ejemplos de estos dispositivos cualificados podemos encontrar diversos certificados electrónicos, tokens USB o incluso el DNI electrónico (DNIe) de España.

En concreto, centrándonos en el DNIe, y cualquiera que utilice un mecanismo similar, podemos encontrar una limitación que impide que su uso y extensión no sea el deseado. Para poder utilizarlo y realizar una firma digital debemos hacer uso de un dispositivo hardware (lector de tarjetas) que sea compatible con el DNIe y permita acceder a su información criptográfica. Esto crea una limitación importante ya que para

poder realizar una firma digital tenemos total dependencia tanto del DNIe como del lector hardware. Estas limitaciones pueden ser evitadas utilizando un esquema de firma centralizado, ya que no es necesario disponer de ningún dispositivo hardware para realizar la firma, sólo se debe disponer de un acceso a Internet y las credenciales necesarias para acceder al certificado de firma electrónica.

Por lo tanto, para la realidad actual del mundo digital en el que estamos inmersos cobra mucho más sentido el uso de un esquema centralizado basado en el reglamento eIDAS que permita realizar firmas digitales bajo cualquier circunstancia. Además, aporta otras bondades como alta disponibilidad, flexibilidad, escalabilidad, sencillez, etc. A su vez, de la mano con estas características, apreciamos que en la actualidad se está implantando el uso del almacenamiento en la nube (cloud) en la vida cotidiana para almacenar información, tanto por empresas como por particulares. Estos detalles crean un marco ideal para el desarrollo de una solución de firma digital basada en la normativa eIDAS que siga un esquema basado en la nube. Si ya se disponen de los documentos almacenados en la nube, parece ser ideal conseguir unir el concepto de firma digital en un escenario cloud para conseguir lo mejor de ambos aspectos.

Así pues, utilizando las plataformas de almacenamiento en la nube junto a la normativa eIDAS, surge la idea de crear un producto software bajo una plataforma web sencillo e intuitivo que permita crear firmas digitales directamente en la nube sin necesidad de disponer, en ningún momento, de elementos hardware que compliquen o limiten su creación. Para ello, en el desarrollo de este trabajo haremos uso de un servicio de confianza cualificado que nos permitirá realizar la validación y creación de firmas digitales, junto con el material criptográfico necesario, actuando como PKI, para llevar a cabo este proceso. En concreto, haremos uso del producto TrustedX de Entrust Datacard, implementando la normativa eIDAS junto con un flujo de autorización OAuth2 para autorizar el acceso a los diferentes servicios proporcionados por el software. A su vez, se hará uso del Identity Provider de Google [2] (IdP) para autenticar a los usuarios de forma adecuada en el acceso a la plataforma web, así como de los servicios de almacenamiento Google Drive y Dropbox para obtener los documentos a firmar, todo debidamente autenticado y autorizado mediante OAuth2 [3].

Por último, nos centraremos en la firma digital de documentos en formato PDF (Portable Document Format), ya que es el estándar más extendido para documentos digitales, siendo muy versátil y compatible. En este sentido, el estándar de firma digital a emplear será PAdES [4] ya que es el estándar compatible con la firma electrónica cualificada que define la normativa eIDAS.

1.2 Objetivos del Trabajo

Nuestro trabajo, entre otros, consta de un objetivo claro, implementar un mecanismo de firma digital sencillo e intuitivo que cumpla con la normativa eIDAS. Para ello, debemos realizar la integración de diversos sistemas digitales basados en la nube que interactuando entre ellos compongan un sistema basado en el concepto SaaS (Software como servicio).

El producto software creado por este trabajo debe ser capaz de realizar firmas digitales para documentos PDF sin la necesidad de emplear elementos hardware adicionales (como lectores de DNIe). Es muy importante que permita generar firmas digitales cualificadas sin requerir protocolos de uso complejos, pero si usando material criptográfico complejo y seguro. Una de las premisas de este trabajo es que el concepto de firma digital sea fácil e intuitivo para permitir su expansión en la sociedad a colectivos que no dispongan de amplios conocimientos en informática y en el concepto de firma digital.

Al igual que una firma manuscrita realizada en un papel dispone de toda validez legal y no requiere que una persona tenga amplios conocimientos en este aspecto, el objetivo de este trabajo es conseguir que sea igual de sencillo y simple obtener una firma digital que disponga de la misma validez legal.

Para ello, creemos que hacer uso del Proveedor de Identidad de Google para realizar la autenticación en la plataforma web es la opción más ideal dada la gran expansión de Google en el mundo actual. Prácticamente todo el mundo dispone, si no de un teléfono con software Android (Google como propietario), de una cuenta de correo electrónico de esta plataforma. Así pues, sólo con este requerimiento cualquier persona podría hacer uso de la misma sin añadir mayor complejidad a nuestro producto software ni requerir de un registro previo. Este último paso podría generar reticencias y limitaciones de expansión de nuestro producto puesto que no dispone de la misma reputación que Google.

Por otro lado, basándonos en el mismo motivo haremos uso de los servicios de almacenamiento cloud Google Drive y Dropbox al ser unos de los más extendidos y reputados en la actualidad. De esta forma, evitamos que nuestro producto software tenga que almacenar documentos personales evitando posibles futuros problemas de privacidad. A su vez, aporta una visión útil y sencilla al usuario ya que simplemente tendrá que seleccionar los documentos de su plataforma de almacenamiento, sin tener que añadirlos a nuestro producto software para realizar la operación de firma.

Todos los mecanismos de autenticación y autorización empleados en nuestro producto software están basados en OAuth2 ya que nos proporciona mecanismos simples, concisos y personalizados para cada servicio con los que se integrará nuestro producto. Todos los procesos de intercambio de información irán basados en un token de autorización previamente concedido por cada uno de los servicios.

Por último, para añadir más privacidad y seguridad a nuestra aplicación, haremos uso de comunicaciones cifradas mediante SSL, tanto para la conexión entre los clientes de nuestro producto software, como para la comunicación entre los distintos servicios de los que haremos uso.

1.3 Enfoque y método seguido

Para la realización de este trabajo hemos optado por seguir una estrategia basada en fases y a su vez, algunas de estas fases pueden estar divididas en distintas etapas. Por ejemplo, la integración de nuestro producto con el servicio Google Drive se considerará como una etapa en una de las fases. Se ha decidido desarrollar un producto nuevo que integrará distintos servicios ya existentes en la actualidad. Nuestra aplicación aglutinará estos servicios para proporcionar una funcionalidad novedosa y funcional.

Las fases son incrementales, siendo necesario completar una fase anterior para poder avanzar. Sin embargo, las etapas que pueden existir en una fase se podrán abordar en paralelo. Las distintas fases en las que hemos dividido el desarrollo de nuestro producto son:

Fase 1: Análisis y diseño

> En esta fase debemos realizar un análisis de los propósitos que necesitamos que cumpla nuestra aplicación. En adición a esto, se debe realizar una toma de

todos los requisitos que se deben satisfacer, así como conocer las precondiciones y postcondiciones que el software deberá cumplir.

A su vez, se realizará el diseño de la arquitectura de la aplicación y del entorno de despliegue. Tenemos que debatir y analizar si vamos a realizar un despliegue en un servidor web, en contenedores Docker, etc. También se deberá valorar si es necesario el uso de base de datos, cortafuegos, etc. En cuanto al diseño de la aplicación, tendremos que determinar cuantos formularios tendremos que crear para cumplir los requisitos marcados, analizar los casos de uso y los posibles flujos que el usuario podrá realizar en la aplicación.

Fase 2: Planificación

En esta fase tendremos que partir de todo el contenido analizado en la anterior para realizar una estimación y planificación del desarrollo del producto.

Fase 3: Desarrollo del producto

Esta fase es una de las más importantes del proyecto. Tendremos que implementar el código necesario para satisfacer los requisitos y los casos de uso. Se deberán realizar todas las integraciones con los distintos servicios que utilizaremos (Drive, Dropbox, Proveedor de Identidad de Google y TrustedX). Cada una de estas integraciones se considerarán etapas distintas que se podrán realizar de forma paralela. Es decir, no es necesario haber finalizado la integración con el Proveedor de Identidad para haber comenzado a realizar la integración con el servicio de Dropbox. En posteriores puntos de este documento se expondrán los detalles tanto a nivel técnico como de planificación que permiten esta planificación.

A su vez, también iremos realizando de forma modular distintas pruebas. Por ejemplo, cuando hayamos completado la integración con un servicio podremos realizar una batería de pruebas para verificar si el funcionamiento es el esperado.

Fase 4: creación del protocolo de puesta en producción

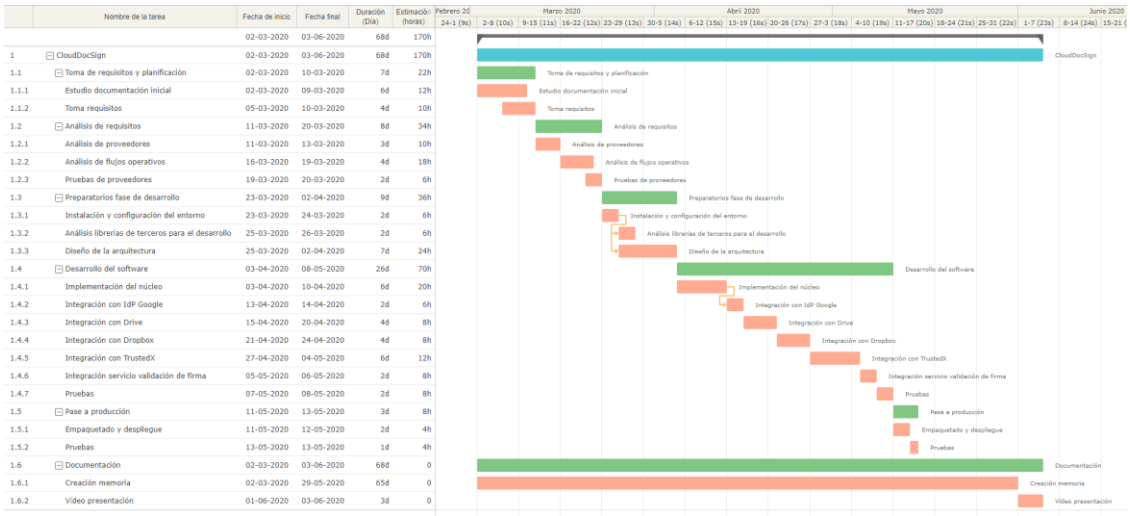
Se elaborará una guía que permitirá realizar el despliegue desde cero del producto desarrollado.

Fase 5: documentación

Se creará la documentación relativa al proyecto, así como la memoria relativa al proyecto.

1.4 Planificación del Trabajo

En el siguiente diagrama de Gantt se puede ver una planificación de la evolución del proyecto.



	Nombre de la tarea	Fecha de inicio	Fecha final	Duración (Día)	Estimación (horas)
		02-03-2020	03-06-2020	68d	170h
1	CloudDocSign	02-03-2020	03-06-2020	68d	170h
1.1	Toma de requisitos y planificación	02-03-2020	10-03-2020	7d	22h
1.1.1	Estudio documentación inicial	02-03-2020	09-03-2020	6d	12h
1.1.2	Toma requisitos	05-03-2020	10-03-2020	4d	10h
1.2	Análisis de requisitos	11-03-2020	20-03-2020	8d	34h
1.2.1	Análisis de proveedores	11-03-2020	13-03-2020	3d	10h
1.2.2	Análisis de flujos operativos	16-03-2020	19-03-2020	4d	18h
1.2.3	Pruebas de proveedores	19-03-2020	20-03-2020	2d	6h
1.3	Preparatorios fase de desarrollo	23-03-2020	02-04-2020	9d	36h
1.3.1	Instalación y configuración del entorno	23-03-2020	24-03-2020	2d	6h
1.3.2	Análisis librerías de terceros para el desarrollo	25-03-2020	26-03-2020	2d	6h
1.3.3	Diseño de la arquitectura	25-03-2020	02-04-2020	7d	24h
1.4	Desarrollo del software	03-04-2020	08-05-2020	26d	70h
1.4.1	Implementación del núcleo	03-04-2020	10-04-2020	6d	20h
1.4.2	Integración con IdP Google	13-04-2020	14-04-2020	2d	6h
1.4.3	Integración con Drive	15-04-2020	20-04-2020	4d	8h
1.4.4	Integración con Dropbox	21-04-2020	24-04-2020	4d	8h
1.4.5	Integración con TrustedX	27-04-2020	04-05-2020	6d	12h
1.4.6	Integración servicio validación de firma	05-05-2020	06-05-2020	2d	8h
1.4.7	Pruebas	07-05-2020	08-05-2020	2d	8h
1.5	Pase a producción	11-05-2020	13-05-2020	3d	8h
1.5.1	Empaquetado y despliegue	11-05-2020	12-05-2020	2d	4h
1.5.2	Pruebas	13-05-2020	13-05-2020	1d	4h
1.6	Documentación	02-03-2020	03-06-2020	68d	0
1.6.1	Creación memoria	02-03-2020	29-05-2020	65d	0
1.6.2	Vídeo presentación	01-06-2020	03-06-2020	3d	0

Figura 1: diagrama Gantt de la aplicación

1.5 Breve sumario de productos obtenidos

En el momento en el que se finalice este trabajo esperamos obtener un producto software disponible a través de internet que permitirá realizar la firma de documentos. Dicha firma se corresponderá con una firma digital cualificada que tendrá la misma validez que una firma manuscrita.

Para ello, el usuario necesitará autenticarse en las diversas plataformas con las que el producto se integrará para conseguir el propósito final, así como disponer de unas credenciales de firma que permitan al usuario realizarla con todas las garantías.

Nuestro producto conseguirá todos estos hitos gracias a la realización de todas las distintas fases indicadas en la planificación del mismo. Para ello, haremos uso de distintos servicios web como Drive, Dropbox y TrustedX para obtener el producto con todas y cada una de las características marcadas.

1.6 Breve descripción de los otros capítulos de la memoria

Arquitectura
En este apartado se detallan los distintos servicios que van a ser utilizados en la aplicación, así como los flujos más importantes entre estos servicios y CloudDocSign. Se presenta toda la información que permita tener una visión global de los escenarios que se deben implementar en la aplicación, así como tecnologías y particularidades de cada servicio.

Diseño e implementación de la solución
En este apartado se presentan las partes la planificación relacionadas con el desarrollo del producto. Se divide en dos apartados principales.

> *Diseño*
> Se definen las entidades, hitos y los aspectos de diseño empleados para definir el núcleo de la arquitectura que serán seguidos durante la implementación de la aplicación.
>
> *Implementación*
> En este lugar se llevan a cabo las tareas relacionadas con la implementación de la aplicación siguiendo todos los aspectos previamente definidos.

Flujo funcional
Se presentan imágenes que muestran los pasos seguidos en la aplicación para cumplir con el flujo establecido. Se podrá ver un ejemplo de cómo utilizar la aplicación para el fin previsto.

Conclusiones
Se presenta un breve resumen de lo acontecido durante el proyecto, así como posibles vías de mejora futuras.

Glosario
Se presenta la definición de los acrónimos y términos técnicos expuestos en el documento.

Bibliografía
En este apartado se mencionan las fuentes y la documentación relevantes utilizadas en este proyecto.

2. Arquitectura

2.1 Descripción general

En este apartado vamos a explicar detalladamente la arquitectura de la aplicación que vamos a desarrollar en este trabajo, así como cada uno de los módulos que la van a integrar y las interacciones entre distintos servicios que se van a producir. Es muy importante entender que rol va a ejercer nuestra aplicación y de que manera se va a integrar con los distintos servicios de los que se hará uso. Como una primera idea podemos distinguir entre proveedores de identidad, proveedores de almacenamiento y proveedores eIDAS. Estas tres entidades, respectivamente, son las que van a permitir autenticar a los usuarios para el acceso al portal, acceder a los documentos que se firmarán y, por último, acceder a los servicios que permitirán obtener el material criptográfico de confianza para realizar la firma. A continuación, se muestra un diagrama conceptual de la arquitectura con más detalle.

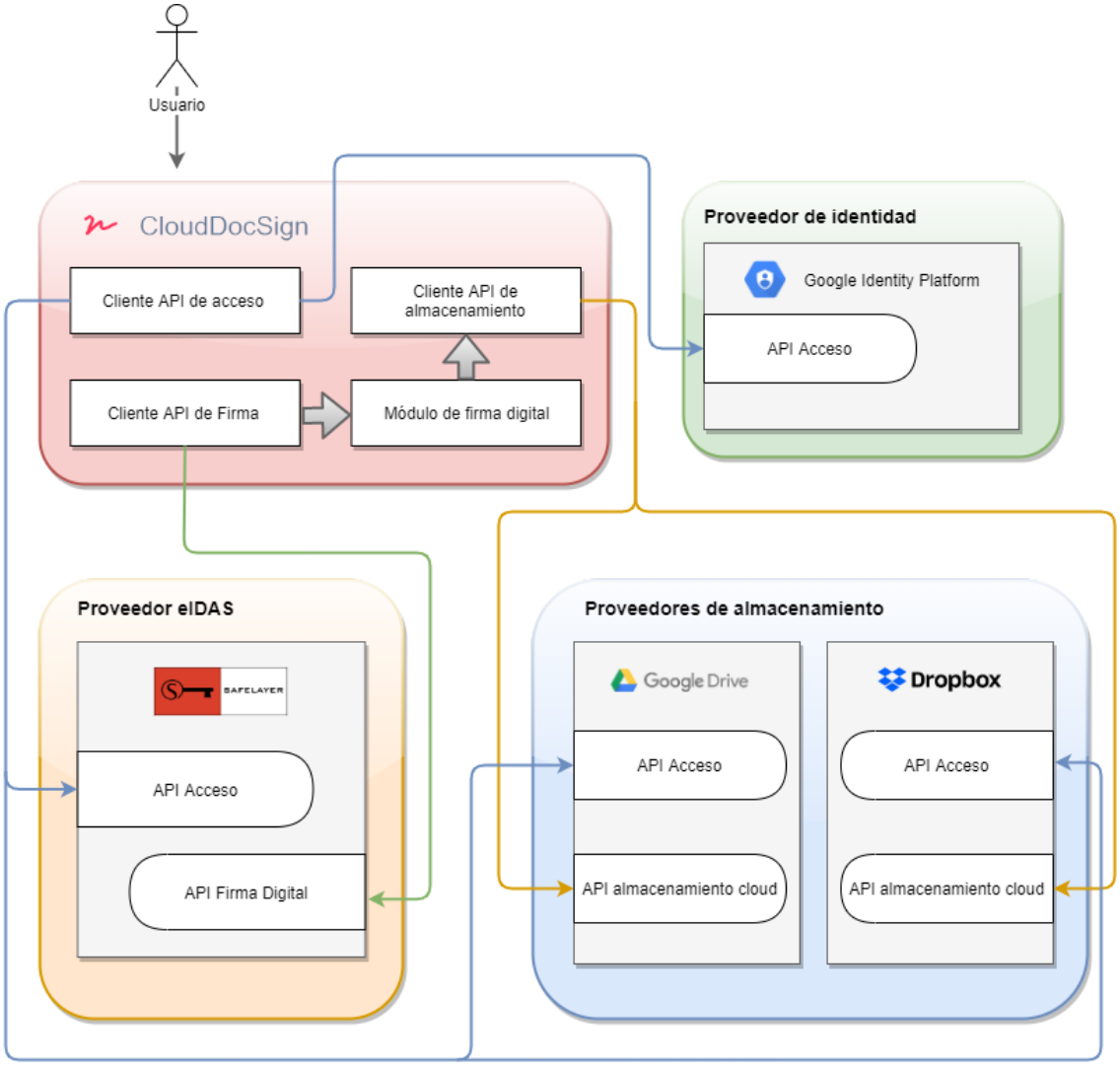


Figura 2: arquitectura general

Como podemos apreciar en el diagrama, tenemos un componente que se corresponde con la aplicación CloudDocSign que vamos a crear. Dicha aplicación dispone de una serie de módulos que se encargarán de interactuar con cada una de los servicios

necesarios para crear el documento firmado final. Por ejemplo, se puede ver como interactúa con un Proveedor de Identidad para llevar a cabo una autenticación que permita acceder a la aplicación. También dispone de un módulo que permitirá interactuar con los distintos servicios de almacenamiento en la nube que permitirán obtener los documentos de los usuarios.

Por último, vemos como dispone de un módulo de firma digital que consumirá datos proporcionados por otro módulo que interactuará con el servicio eIDAS para, posteriormente, interactuar con el modulo de almacenamiento y guardar el documento firmado. Es muy importante este punto, ya que como se explicará más adelante, la firma PAdES del documento nunca será generada en un servicio externo y se realizará en la propia aplicación CloudDocSign. Esto nos proporcionará una serie de ventajas que veremos en detalle posteriormente, de ahí la importancia de resaltar este módulo, ya que es una de las partes más relevantes del desarrollo.

A grandes rasgos, esta es la arquitectura que vamos a implantar para el desarrollo de la aplicación. Más en detalle, podemos avanzar que para el proveedor de identidad utilizaremos Google Identity Platform, para los proveedores de almacenamiento usaremos Google Drive y Dropbox, así como el uso del servicio TrustedX de Entrust Datacard para acceder a los servicios eIDAS que permitirán acceder y generar la información segura necesaria para la firma cualificada.

En los siguientes apartados, explicaremos en detalle los flujos generales que se seguirán en la aplicación, así como una explicación más concreta de los protocolos, flujos concretos y tecnologías de cada una de las interacciones que realizará CloudDocSign con los servicios necesarios.

2.2 Flujos

En este apartado vamos a explicar el flujo completo de interacción que se seguirá entre los distintos servicios para realizar una firma cualificada de un documento. La idea es plasmar teóricamente los pasos que se deberán seguir en la aplicación para realizar el proceso de firma cualificada de un documento.

Para ello, la idea es mostrar unos diagramas que permitan ver de una forma sencilla y rápida todas las interacciones que se van a realizar. En la imagen que podremos ver a continuación, se aprecia claramente el proceso de firma de un documento obtenido del proveedor de almacenamiento Google Drive.

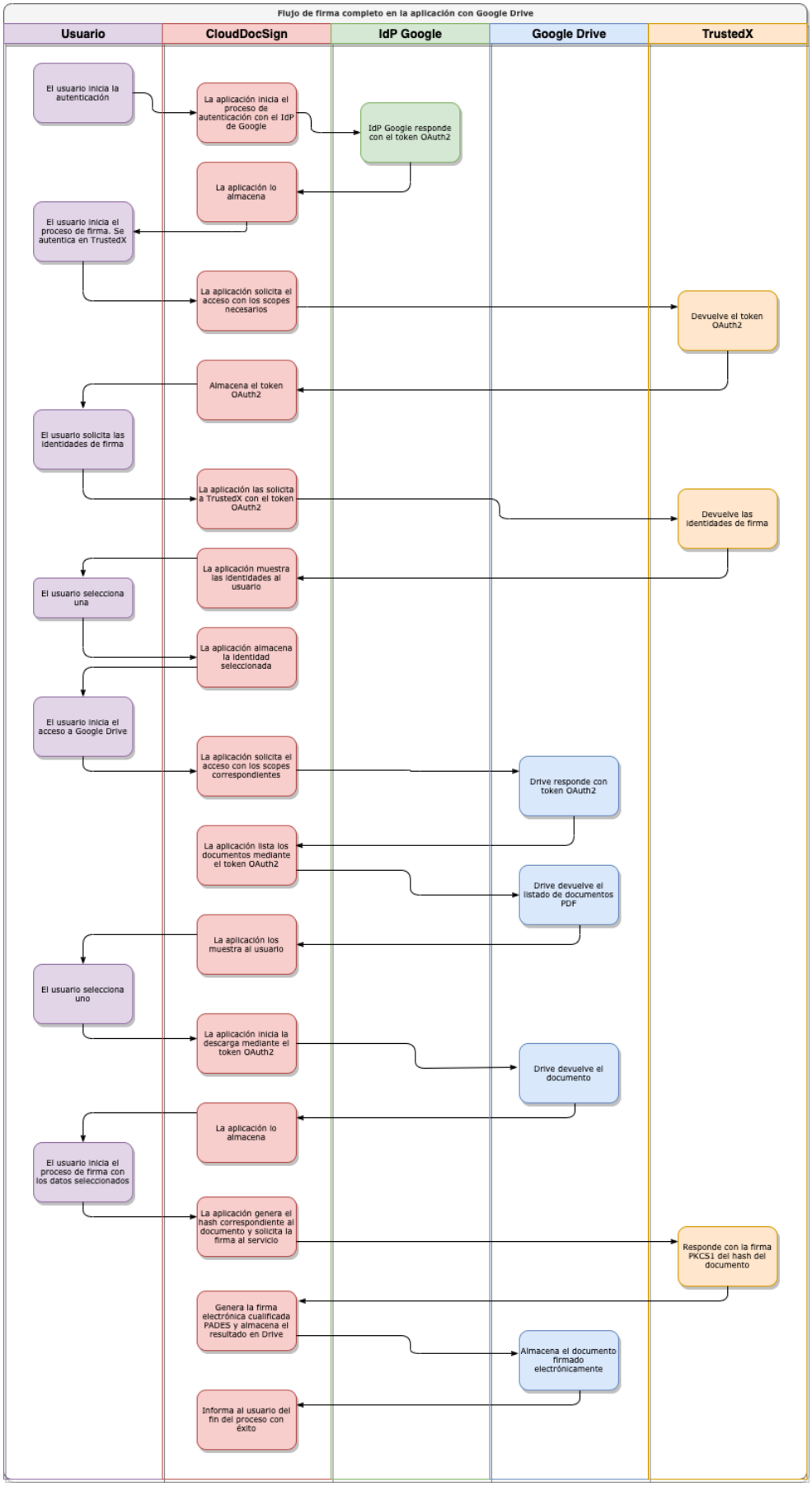


Figura 3: flujo de firma completo con Drive

Como se aprecia, el primer paso es que el usuario desee iniciar sesión. Para ello, nuestra aplicación CloudDocSign interactuará con el proveedor de identidad de Google para obtener un token de acceso OAuth2 mediante el cual se podrán obtener datos del usuario autenticado para generar un perfil de usuario en la aplicación. Poseer este token es sinónimo de tener el acceso permitido a la aplicación.

En el siguiente paso, el usuario debe realizar un proceso de autenticación con el servicio TrustedX de Entrust Datacard para validar su acceso y poder acceder a los servicios eIDAS proporcionados. De igual modo se obtendrá un token OAuth2, siendo necesario para acceder a las identidades de firma (las que se emplearán para realizar la firma PAdES del documento). Cuando haya obtenido el token OAuth2, se solicitarán al servicio las identidades de firma registradas para el usuario y se deberá elegir una de ellas.

Una vez hecho, el usuario solicite acceso a Google Drive. Para ello, la aplicación realizará una integración con la API REST de Google Drive (lo veremos en detalle en posteriores apartados) para realizar un proceso de autenticación y autorización. Si ha resultado exitoso, se obtendrá un nuevo token OAuth2 que permitirá acceder a la información necesaria de Google Drive para obtener los documentos. Seguidamente, el usuario seleccionará un documento para proceder con su firma.

Cuando el usuario haya completado estas dos acciones y tenga la identidad de firma y el documento, CloudDocSign ya tendrá disponible la información necesaria para proceder con la firma del documento. Este proceso será realizado por el módulo de firma que vimos en el diagrama de arquitectura general. Para ello, se debe generar un nuevo token OAuth2 que representa la **autorización para la activación de la clave de firma** y, una vez disponible, solicitar al servicio TrustedX la firma del hash del documento. Los detalles técnicos sobre este proceso se mostrarán en posteriores apartados.

Cuando este proceso sea finalizado y se disponga de la firma PAdES, la aplicación añadirá el fichero firmado digitalmente al mismo proveedor de almacenamiento del que se obtuvo (en este caso Google Drive) para que el usuario disponga del mismo. Remarcar que en el momento de acceder a cada proveedor de identidad o eIDAS, la aplicación comprobará si ya existe una sesión abierta con dicho proveedor para no repetir el proceso de autenticación y autorización.

Así pues, este es el flujo general que se debe seguir en la aplicación para realizar todo el escenario completo. Es importante destacar que por usabilidad se ha decidido que el flujo operativo requiera seleccionar primero la identidad de firma y posteriormente el documento.

De la misma forma, para el proveedor de almacenamiento Dropbox, el flujo a seguir es similar debido a que la API REST de autenticación y acceso a los documentos son similares en tecnología. Por lo tanto, el flujo a seguir será similar, cambiando solamente el servicio al que accedemos, que en este caso es Dropbox. A continuación, se puede ver el diagrama de flujo completo para este proveedor de almacenamiento.

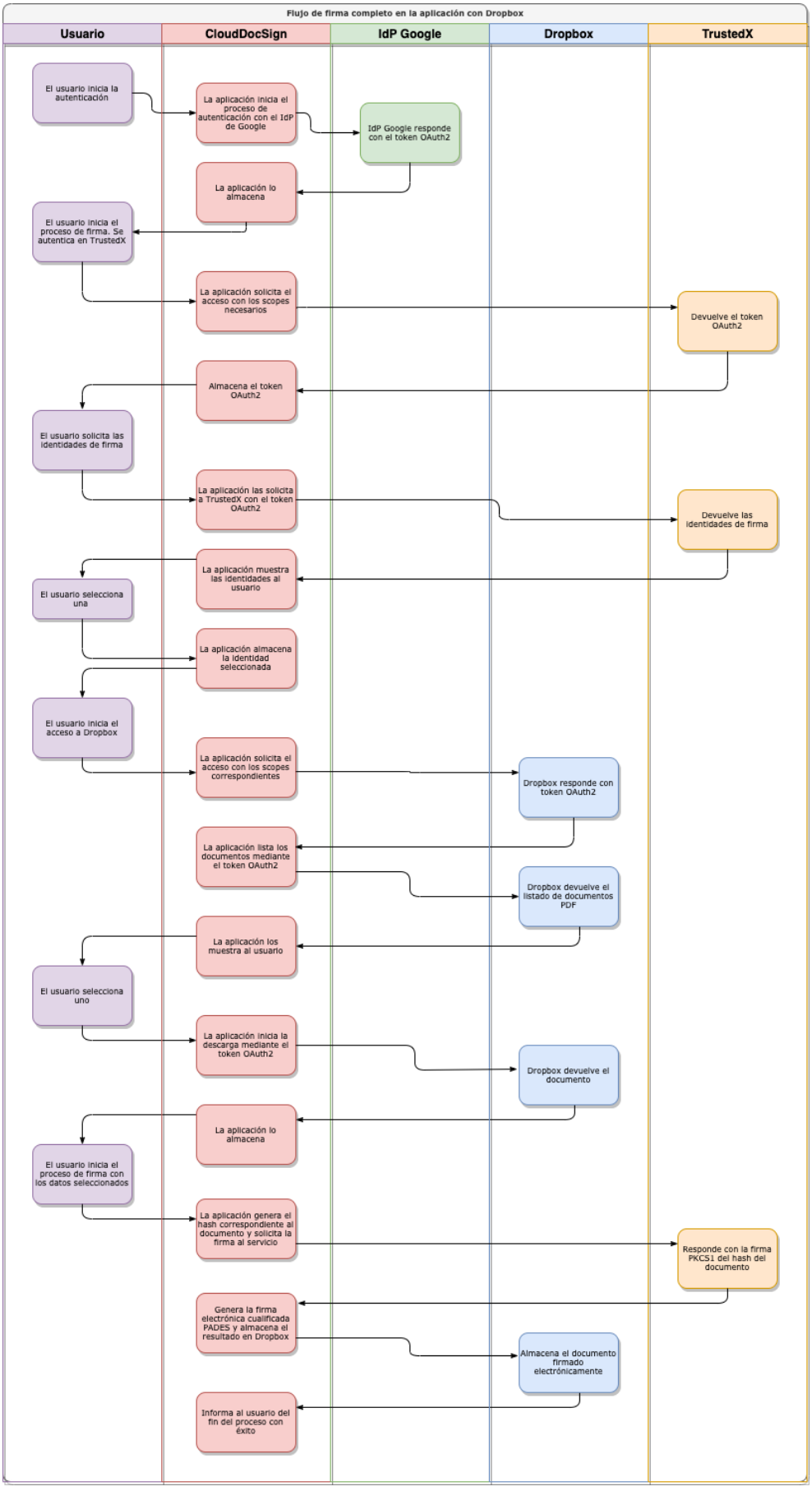


Figura 4: flujo de firma completo con Dropbox

Por último, indicar que hay diversos flujos más específicos que han sido simplificados en este diagrama (por ejemplo, la obtención de un token OAuth2). Serán expuestos con más detalle en apartados posteriores, donde se comente con exactitud los protocolos y tecnologías que se emplearán para la integración de los diversos proveedores.

2.3 Proveedores de Identidad

Los proveedores de identidad son unos servicios existentes en la red que tienen la autoridad de emitir y procesar información relacionada con la identidad de los usuarios de un sistema. Básicamente, son servicios encargados de gestionar las cuentas de usuario y de corroborar que los accesos a los sistemas solicitados se realizan a los usuarios legítimos. En nuestro caso, lo utilizaremos para obtener información de la identidad de los usuarios que deseen acceder a la aplicación y poder generar un perfil de los mismos.

En la actualidad son servicios altamente imprescindibles en el ámbito informático, ya que son esenciales en escenarios con un mínimo de seguridad, tales como Single Sign-On o escenarios triple AAA (Autenticación, Autorización y Contabilidad).

Existen diversos protocolos a utilizar en estos servicios, siendo los más extendidos SAML, OAuth y OpenID. En nuestro caso, todos los proveedores de identidad utilizan OAuth con una base de OpenID.

2.3.1 Protocolos

A continuación, se van a exponer los detalles de los protocolos de autenticación y autorización que se van a utilizar en este proyecto.

2.3.1.1 OpenID

OpenID es un estándar utilizado para la identificación digital descentralizado. Mediante este protocolo un usuario puede identificarse en una web sin necesidad de registrarse para obtener acceso. Si un sitio web tiene soporte de OpenID, el usuario sólo debe mostrar su identificador para que el sitio web pueda validarlo y verificar su perfil mediante el Proveedor de Identidad (IdP).
Es importante remarcar que OpenID no especifica el mecanismo mediante el cual realizar la autenticación, sólo es un marco de actuación. La seguridad en una autenticación mediante este protocolo dependerá de la confianza que tenga el cliente OpenID con el proveedor de Identidad. En este sentido, por ejemplo, OAuth2, como veremos ahora, depende de OpenID para completar el proceso de autenticación.

2.3.1.2 OAuth

Open Authorization (OAuth) es un estándar libre que permite realizar un intercambio de información de autorización para sitios web o aplicaciones. Permite una autorización segura a través de una API simple que puede ser utilizada para aplicaciones de diversa índole.

A modo básico, permite que un usuario de un sitio A (proveedor del servicio) comparta cierta información de su perfil con un sitio B (consumidor). Permite interactuar con información protegida, realizar un intercambio seguro, mientras las credenciales de acceso a la cuenta permanecen seguras en el proveedor de servicio. Este protocolo es

ampliamente usado por empresas como Google, Facebook, etc. Para permitir a sus usuarios el acceso a aplicaciones de terceros.

OAuth 2.0 no es compatible con OAuth 1.0., ya que esta nueva versión proporciona flujos de autorización específicos para aplicaciones web, aplicaciones de escritorio y teléfonos móviles.

2.3.2 Integraciones implementadas

En este apartado vamos a explicar todos los protocolos e integraciones realizados para llevar a cabo la autenticación y autorización en la aplicación CloudDocSign. Se van a dividir en dos grupos. El primero será autenticación y se corresponderá con la autenticación para acceder a la aplicación. El segundo grupo, autorización, va a tratar las diferentes integraciones que se llevarán a cabo en la aplicación desarrollada para interactuar con los proveedores de almacenamiento y eIDAS. De esta forma se podrá conseguir la autorización necesaria que permita acceder a los datos y operaciones esenciales.

Como detalle técnico mencionar que todos los servicios que se van a describir en estos subapartados se basan en API REST, mediante la cual podremos interactuar para enviar o solicitar información.

2.3.2.1 Acceso a la aplicación

En primer lugar, la autenticación para acceder al portal la vamos a realizar con Google Identity Platform. Es un proveedor de identidad (IdP) proporcionado por Google que nos permitirá autenticar y conocer detalles del usuario que desea acceder a nuestra aplicación. Es muy útil emplear esta solución ya que nos va a permitir la elaboración de un perfil del usuario (nombre, email, foto de perfil, etc.) sin tener la necesidad de almacenar nosotros mismos esa información. Además, todas las credenciales de autenticación van a residir en Google, por lo tanto, se delega en este servicio la gestión de identidad.

Optamos por la opción de Google debido a que actualmente es la más extendida. Prácticamente la totalidad de los usuarios potenciales van a disponer de un perfil en este sistema, por lo que el acceso a nuestra aplicación va a ser total.
Es importante remarcar, que este servicio es factible gracias a los protocolos que ya explicamos anteriormente en el apartado 2.3.1 Protocolos de este documento, como OAuth2. Por lo tanto, nuestra aplicación se va a integrar con la API REST OAuth2 a través de Google Identity Platform.

Por otro lado, este servicio permite que se le especifiquen unos ámbitos (scopes) de acceso para los cuales nos proporcionará el token que permitirán acceder a los mismos. Más a groso modo explicado, si no especificamos a la hora de iniciar el proceso OAuth2 que queremos acceder al ámbito *userinfo.email*, si la autenticación es correcta, no nos autorizará el acceso al email del usuario.

Por ese motivo y, para crear un perfil adecuado en nuestra aplicación, optamos por utilizar los dos ámbitos que permite esta API.

- https://www.googleapis.com/auth/userinfo.email: acceder al email del usuario.
- https://www.googleapis.com/auth/userinfo.profile: acceder a la información personal que el usuario ha hecho pública (como nombre, foto de perfil, etc).

Sin embargo, en nuestra aplicación creemos que los únicos datos que son necesarios para crear un perfil óptimo son:

Nombre	**Descripción**
Sub	Identificador del usuario en Google
Given_name	Nombre de pila
Family_name	Apellidos
Email	Dirección de correo electrónico
Picture	URL de la foto de perfil en Google

Para finalizar este servicio, simplemente falta por mostrar el flujo que se llevará a cabo para completar el proceso. Con un sencillo diagrama vamos a poder ver los intercambios realizados entre nuestra aplicación y el IdP para obtener el acceso y la información solicitada.

Figura 5: Fujo de autenticación contra IdP Google

En la imagen apreciamos como se realizan cuatro intercambios entre nuestra aplicación y el IdP de Google. En el primero, se realiza una petición GET en la que solicitamos el inicio de la autenticación. Para ello, se deberán enviar una serie de datos como los ámbitos ya mencionados o la URL a la que queremos que nos redireccione el IdP cuando finalice el proceso de autenticación.

La respuesta que vamos a recibir de esta primera petición es una URL a la que redirigir al usuario para que continúe con el proceso de autenticación en Google. Como hemos comentado, en estos escenarios con OpenID y OAuth2 se delega el proceso de autenticación en un servicio con el que establecemos una relación de confianza.

Cuando este proceso ha terminado, se ejecuta el paso 2 del diagrama. Obtenemos el OAuth2 code (Authorization Code Grant) mediante el que podremos solicitar nuestro token OAuth2 de acceso junto con la información que queremos recuperar del perfil. Así pues, los pasos 3 y 4 del diagrama se corresponden con la petición al IdP de Google del token OAuth2.

Es importante destacar que poseer dicho token permite demostrar ante el servicio IdP (o para los que el token haya sido validado) que se disponen de los permisos necesarios para acceder a la información indicada en los ámbitos.

2.3.2.2 Autorización en proveedores

En este apartado, vamos a explicar como se lleva a cabo la autorización para acceder a los datos y contenidos personales de cada uno de los proveedores de los que

haremos uso. Cada uno de ellos dispone de un mecanismo que permitirá autenticar y autorizar el acceso a los diversos recursos disponibles.

Por otro lado, todas las peticiones realizadas a las API REST para obtener el token OAuth2 y el proceso de autenticación disponen de un token CSRF para evitar el temido ataque Cross-Site Request Forgery. De esta forma se aporta más seguridad al flujo que se establecerá. Por otro lado, las peticiones realizadas al IdP de cada proveedor deben ir autenticadas en origen, es decir, se debe indicar una especie de identificador y contraseña (*client_id* y *secret_key*) que permitirá al IdP verificar la identidad de la aplicación que está realizando la petición. En apartados posteriores se detallará el proceso seguido para obtener estos valores que permitan realizar los intercambios.

Por último, la aplicación CloudDocSign gestionará los distintos tokens temporales OAuth2 que se obtengan en los diferentes flujos que se realicen para acceder a los servicios. Todos ellos quedarán almacenados a nivel de sesión en el servidor, evitando que puedan ser transmitidos y robados por un tercero.

2.3.2.2.1 Google Drive

En este caso, el mecanismo para obtener la autorización al acceso de documentos almacenados en Drive es similar a la ya comentada, puesto que para acceder a esta API se utiliza el servicio Google Identity Platform que explicamos anteriormente.

La única diferencia la podemos apreciar en los ámbitos que le indiquemos a la hora de iniciar el flujo de autenticación y autorización. En este caso, debemos añadir los ámbitos relativos al acceso de los documentos de Google Drive. También añadiremos otros dos ámbitos pertenecientes a la API Google People para acceder a los datos del perfil de la cuenta que se ha iniciado en Google Drive. Básicamente, podemos tener el caso en el cual la cuenta de Google utilizada para acceder a la aplicación sea distinta de la cuenta de Google Drive donde estén almacenados los documentos a firmar.

Dichos ámbitos son:

- https://www.googleapis.com/auth/drive: permite ver y gestionar ficheros de Drive.
- https://www.googleapis.com/auth/drive.file: permite ver y gestionar ficheros y carpetas creados directamente en Drive.
- https://www.googleapis.com/auth/userinfo.profile: acceso a la información básica del perfil.
- https://www.googleapis.com/auth/userinfo.email: acceso a la dirección de email del perfil.

Así pues, con esta información y siguiendo el siguiente flujo, conseguiremos obtener un token OAuth2 que permitirá acceder a todos los ficheros de Drive para listarlos, descargarlos o subirlos al servicio. A su vez, nos permitirá obtener información básica del perfil del usuario que se ha autenticado para mostrarlo en la aplicación. Es importante destacar que todos los permisos que se están solicitando durante el flujo de autenticación y autorización deben ser concedidos por el usuario.

A continuación, se muestra el flujo seguido por la aplicación CloudDocSign para obtener dicho token.

Figura 6: flujo OAuth2 contra IdP Google Drive

Sin extendernos demasiado, vemos como el flujo es igual que el seguido durante la autenticación con el IdP de Google para permitir el acceso a la aplicación. El esquema seguido para todos los proveedores y servicios será el mismo.

2.3.2.2.2 Dropbox

Dropbox también dispone de un IdP para autenticar y autorizar el acceso mediante su API REST a los documentos almacenados. En este caso, se disponen de una serie de peticiones que se pueden realizar para obtener el token OAuth2 mediante el flujo de autenticación ya explicado. Por lo tanto, el flujo seguido para este proceso es similar a los ya comentados.

La única característica que encontramos con el resto de proveedores es que Dropbox no utiliza los ámbitos ya comentados con anterioridad. La autorización y el flujo OAuth2 no está basado en estos parámetros, si bien, ofrece un mecanismo para detallar los permisos y acciones que se permitirán realizar.

La API de Dropbox, durante la creación de la aplicación, permite indicar (lo detallaremos más adelante) el nivel de acceso y permisos que se dispondrá a través del token OAuth2 obtenido. Ofrece distintos tipos como AppFolder (sólo hay acceso a todos los documentos de una carpeta específica), Full Dropbox (acceso a todo el contenido), etc.

A continuación, se muestra el flujo OAuth2 seguido por CloudDocSign.

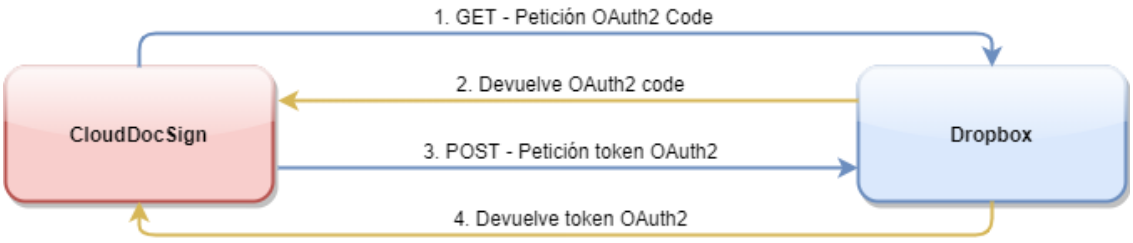


Figura 7: flujo OAuth2 contra IdP Dropbox

2.3.2.2.3 Autenticación en proveedor de firma digital

En este apartado vamos a explicar cómo nos podremos autenticar en el proveedor de firma digital. En este caso, emplearemos el servicio TrustedX, el cual dispone de una API REST securizada que soporta el protocolo OAuth2 para obtener el

correspondiente token. Dicho proveedor eIDAS implementa Authorization Code Grant (RFC 6749) para obtener el token de acceso OAuth2 que protegerá las peticiones.

De la misma forma, este token temporal debe ser generado con los ámbitos correspondientes para que se permita el acceso a las operaciones que nuestra aplicación tendrá que realizar.

En este caso, los ámbitos que vamos a emplear durante el flujo OAuth2 son:

- urn:safelayer:eidas:sign:identity:register: permite generar identidades de firma a partir de un PKCS12 proporcionado.
- urn:safelayer:eidas:sign:identity:manage: permite gestionar las identidades de firma en el servidor (borrar, listar, etc.).

Por lo tanto, para el flujo normal de la aplicación, obtendremos un token OAuth2 que permitirá gestionar las identidades de firma utilizadas para firmar los documentos. Al igual que en el resto de servicios, quedará almacenado por la aplicación, siendo renovado en el caso de que caduque y se requiera su uso. A continuación, se muestra el flujo llevado a cabo con la API del servicio para obtener el token.

Figura 8: flujo OAuth2 contra IdP TrustedX

Al igual que en los otros servicios, el flujo es exactamente el mismo ya que todos ellos son implementados con OAuth2, con una base de OpenID, y Authorization Code Grant. Es importante remarcar que el token obtenido en este flujo no permite realizar el proceso de firma. Para ese escenario se obtendrá un nuevo token, de un solo uso, con el ámbito específico que permita firmar asociado a una identidad de firma. De esta forma se aporta más seguridad a la aplicación y al proceso de firma como tal, al ser muy sensible. Es necesario asegurar que en cada firma realizada el usuario aporta su consentimiento explícito, de ahí que sea necesario obtener un nuevo token. En apartados posteriores se podrá ver el flujo correspondiente a este escenario.

Un detalle muy importante es que en este escenario demostrativo estamos empleando un proveedor de identidad propio de TrustedX que no está clasificado con el nivel necesario para autorizar y realizar firmas cualificadas. La normativa eIDAS establece unos distintos niveles de seguridad necesarios para tratar con identidades de este tipo. Estos niveles de seguridad, llamados LoA (Level of Assurance) deben ser *high o substantial*, para poder tratar con identidades cualificadas y, por tanto, realizar este tipo de firmas. Sin embargo, **en este escenario de pruebas se está empleando un mecanismo de identificación propio de TrustedX** basado en usuario y contraseña que no dispone de este LoA. Independientemente de esto, no supone ningún problema para nuestra aplicación, ya que **en un entorno real bastaría con modificar el proveedor de identidad empleado en el proveedor de firma digital para poder adquirir el nivel LoA necesario para las identidades cualificadas**.

2.4 Proveedores de almacenamiento

Estos proveedores prestan un servicio de almacenamiento de archivos en la nube. Como ya hemos comentado, todos ellos disponen de una API que soporta el protocolo OAuth2 para obtener el token que proporcione seguridad al resto de operaciones sobre el servicio.

Estos sistemas están orientados a una gestión eficiente del almacenamiento de grandes cantidades de archivos y nos van a permitir operar con sencillez sobre los documentos de los mismos. En este caso, trabajaremos con dos de los más importantes, Google Drive y Dropbox. Ambos disponen de una API REST que permitirá, mediante una serie de operaciones, acceder a los archivos y poder gestionarlos. A continuación, se explicarán en detalle los aspectos técnicos y los diversos flujos para cada una de las operaciones que se deberán implementar para proporcionar el servicio necesario.

2.4.1 Google Drive

Este servicio permite el almacenamiento de cualquier tipo de fichero en la nube sólo con tener una cuenta de Google. Al acceder al servicio con nuestra cuenta tendremos acceso a un disco duro virtual accesible desde Internet en el que se podrá almacenar cualquier tipo de archivo.

Por nuestra parte, dispone de una API REST que permite realizar operaciones con todos los archivos creados. Un primer paso que se debe realizar para utilizar la API es registrar la aplicación CloudDocSign en Drive. Esto es necesario para obtener las credenciales que se mencionaron anteriormente y que permiten identificar a la aplicación que quiere usar la API y permitir o no el servicio que está solicitando.

Básicamente, la API sólo permitirá las solicitudes sobre las aplicaciones de terceros que se hayan registrado previamente. Es algo común y necesario que se realiza en todos los proveedores y que aporta seguridad. Así se evita que cualquier aplicación maliciosa pueda tomar el control sobre los archivos de un usuario permitiendo verificar las aplicaciones de terceros. A su vez, cuando se registre la aplicación debemos habilitar la API Drive, ya que Google tiene muchas por todos los servicios que ofrece.

La API de Drive es muy extensa y potente. Tiene infinidad de operaciones que se pueden utilizar y, en este caso, utilizaremos las operaciones para listar, descargar y subir ficheros. El resto no son de relevancia para este trabajo.

En todas estas operaciones, partimos de que se dispone de un token OAuth2 que nos ha proporcionado previamente la API de Drive a través de su IdP, tal y como se pudo ver en los flujos explicados en apartados anteriores.

En primer lugar, utilizaremos la operación para *listar* los documentos. Esta operación permite aplicar filtros para obtener solamente ciertos tipos de archivo. En nuestro caso, aplicaremos el filtro para obtener sólo los documentos con de tipo *pdf*. A continuación, se puede ver el flujo correspondiente.

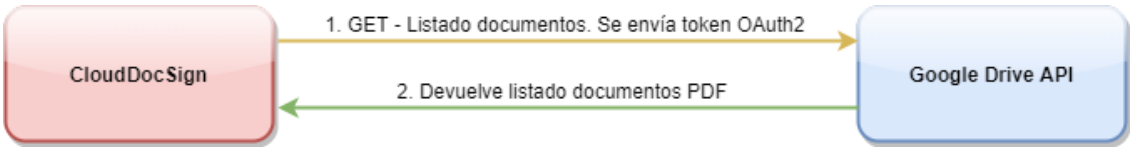


Figura 9: flujo para obtener todos los documentos de Drive

La segunda operación que utilizaríamos se corresponde con la operación *get* de la API que permite descargar un fichero. Esta operación necesita que se le proporcione el *identificador* del archivo en Drive (obtenido mediante la operación listar). A su vez, se añade un parámetro que permite la descarga del archivo en base64 (parámetro *alt=media* en la operación de la API REST).

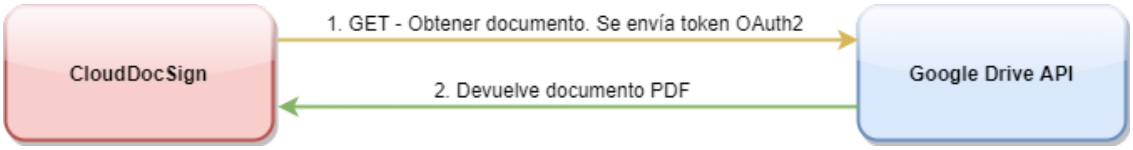


Figura 10: flujo para obtener un documento de Drive

Por último, la operación que utilizaremos para subir el documento a Drive una vez que ha sido firmado es la operación *upload*. Esta operación deberá ser una operación POST, puesto que no vamos a modificar el original existente (en ese caso sería un PUT). Se muestra el flujo de dicha operación en el siguiente diagrama.

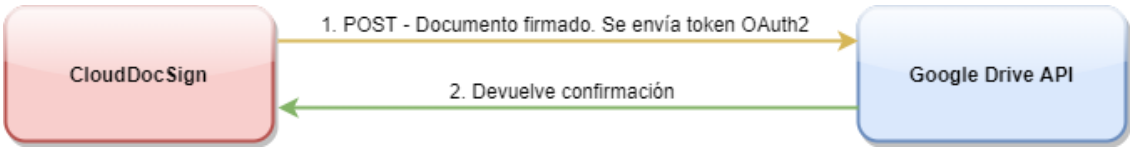


Figura 11: flujo para subir un documento a Drive

Como vemos, todas las operaciones son similares en cuanto al patrón. Se realiza una llamada GET o POST, en función de la operación, que es aceptada o no en función del token OAuth2 que se envíe para garantizar que se disponen de la autorización necesaria.

Por último, comentar que desde la consola de Google Drive para desarrolladores (donde se dio de alta la aplicación CloudDocSign para su uso) se pueden contemplar diversas estadísticas de uso, así como restringir determinadas operaciones para esta aplicación o dar de alta las *url* de redirección que estarán habilitadas para realizar la autenticación OAuth2 contra el IdP del servicio (url callback ya explicada en los apartados de proveedores de identidad).

2.4.1 Dropbox

Este proveedor de almacenamiento cumple las mismas funcionalidades que Google Drive. Permite almacenar todo tipo de documentos en la nube con tan solo crear una cuenta de usuario. Desde el punto de vista técnico también actúa de forma similar a Drive. Es necesario registrar la aplicación CloudDocSign en Dropbox para que se acepten las peticiones hacia la API REST del servicio. Una vez registrada, nos proporcionará dos valores secretos y únicos que identificarán a nuestra aplicación (*key*

y secret). Estos valores se deberán enviar al IdP de la API cuando se inicie el flujo OAuth2. Por otro lado, ya se comentó anteriormente que Dropbox no dispone de ámbitos (scopes) a la hora de solicitar el token OAuth2. En este caso, sólo es posible configurar a nivel de aplicación registrada los permisos sobre los que se tendrá acceso. Por ejemplo, cuando se registre en Dropbox la aplicación CloudDocSign debemos especificar si el nivel de acceso a la cuenta se quiere que sea para todo lo que esté almacenado o sólo a una carpeta concreta. En nuestro caso, lo configuraremos para tener acceso a todo el espacio de almacenamiento del usuario. Aún así, será el usuario quien tendrá que dar el consentimiento final cuando quiera sincronizar su cuenta de Dropbox dentro de nuestra aplicación.

Una vez se ha obtenido el token OAuth2 para acceder a las operaciones y recursos la aplicación hará uso de las operaciones útiles para nuestro desarrollo. En cada una de las peticiones realizadas se deberá indicar este token. Por ejemplo, utilizaremos la operación *list* y *list/continue* de la API para obtener todos los documentos *pdf* que existan en la cuenta de Dropbox. A continuación, se muestra el flujo.

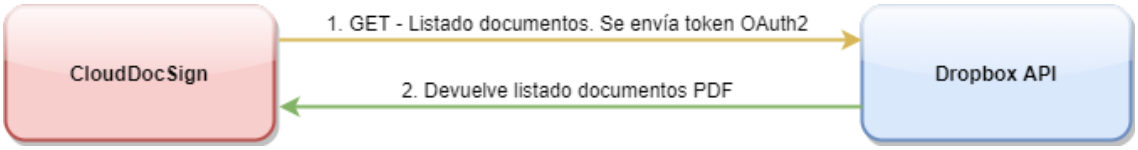


Figura 12: flujo para obtener todos los documentos de Dropbox

A nivel más técnico, indicar que el directorio de archivos se recorre como si fuera un árbol de forma recursiva. Se empezaría por la carpeta raíz y, para cada directorio existente, se vuelve a realizar una petición para listar los documentos. Por lo tanto, este flujo mostrado se repetirá tantas veces como carpetas existan en los archivos del usuario que ha iniciado sesión con Dropbox.

La siguiente operación necesaria se encargará de descargar el documento. Una vez que el usuario ha seleccionado el fichero, la aplicación se dispone a descargarlo con la operación *download*. Se pueden aportar dos valores para descargarlo, la ruta completa dentro del directorio de archivos del usuario o el identificador del archivo en Dropbox.

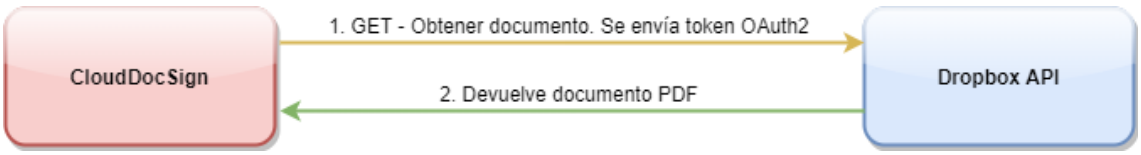


Figura 13: flujo para obtener un documento de Dropbox

La última operación que se utilizará en este servicio se corresponde con la operación *upload*. Nos permitirá subir documentos con un tamaño hasta 150 MB. Se deberán aportar diversos datos como la ruta, si se desea que notifique al usuario en su panel Dropbox o incluso si renombrar el documento o rechazarlo si existe un conflicto. A continuación, los detalles del flujo.

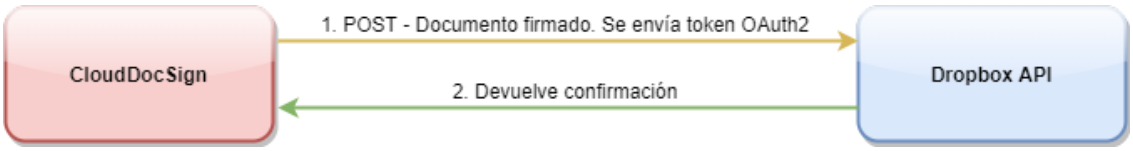


Figura 14: flujo para subir un documento a Dropbox

2.5 Proveedor de servicio eIDAS

2.5.1 TrustedX

En nuestro trabajo vamos a utilizar este proveedor eIDAS que nos va a permitir obtener el material criptográfico necesario para generar firmas cualificadas de documentos. Para este nivel de firma, que es el más seguro, debemos hacernos valer de una entidad que disponga de los dispositivos adecuados para generar la firma de forma segura y, además, que tenga buena reputación. Como bien sabemos, la seguridad también está basada en la buena reputación que se tenga, al igual que sucede con los certificados para SSL en Internet.

Este servicio, de la empresa Entrust Datacard, nos va a proporcionar una API REST que dispone de una infinidad de métodos para que podamos obtener, de diversas formas, la firma de documentos. Hay varias opciones disponibles a la hora de realizar la firma. Una de ellas consiste en enviar el documento a TrustedX para que en sus servidores realice la firma y nos lo devuelva firmado con una firma PAdES. La otra opción, mucho más segura, consiste en enviar simplemente un hash del documento a firmar y, a partir del PKCS1 que nos devuelve TrustedX, generar en CloudDocSign la firma PAdES. Esta segunda opción es mucho más segura y eficiente, pues el documento original del usuario nunca abandonará nuestra aplicación y, por tanto, existe un menor riesgo de que pueda caer en manos de un tercero.

Por este motivo, en nuestra implementación vamos a optar por esta opción, que es mucho más segura e interesante.

La API de TrustedX dispone de una serie de métodos y componentes que se encargarán de proporcionar todo el servicio. Ya se pudo ver que dispone de un IdP que se encargará de gestionar los flujos OAuth2 para obtener el correspondiente token con Authorization Code Grant. A parte de este componente, también emplearemos otro denominado eSigP. Será el encargado de gestionar las identidades de firma empleadas para hacer la firma de los documentos, así como de generar la propia firma del hash y proporcionarla a las aplicaciones correspondientes.

Por lo tanto, en nuestro escenario tendremos que trabajar con las identidades de firma para poder crearlas, listarlas y borrarlas. Más tarde, una vez tengamos estas identidades creadas, podremos usarlas para hacer la firma del hash del documento y poder generar la firma PAdES en nuestra aplicación.

Ahora es momento de explicar los distintos flujos que vamos a realizar con la API de TrustedX. Los flujos con este componente tendrán que llevar el token OAuth2 que obtuvimos al inicio, con los ámbitos (scopes) que explicamos en apartados anteriores.

En primer lugar, disponemos de la operación *listar identidades* que nos devolverá todas las identidades de firma que hayan sido registradas en el perfil de usuario autenticado contra TrustedX. A continuación, el flujo intercambiado.

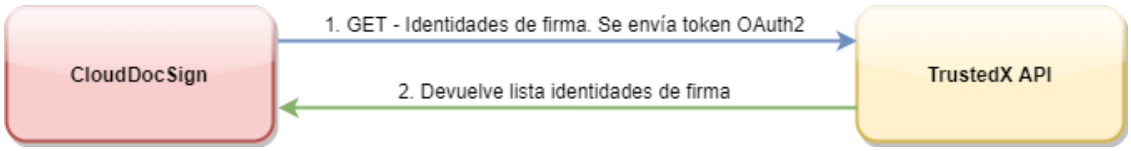


Figura 15: flujo para obtener las identidades de firma

La idea es similar al resto de flujos. La única diferencia a nivel técnico es que esta API trabaja con JSON, por lo tanto, tendremos que enviar y recibir la información con ese formato.

También disponemos de una operación que permitirá *registrar identidades* de firma. En este caso, se tratará de una operación POST, con el mismo token OAuth2 ya obtenido, y los datos necesarios en la petición para registrarla. Entre otras cosas, tendremos que adjuntar el PKCS12 del usuario codificado en Base64. Se muestra el flujo seguido.

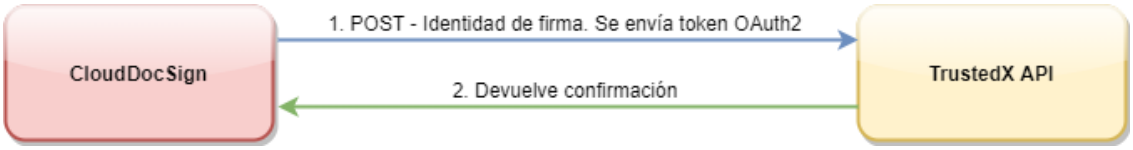


Figura 16: flujo para añadir una identidad de firma

Por último, relativo a las identidades de firma, tenemos la operación que nos permitirá *borrar identidades* de firma del perfil del usuario. Tan sólo tendremos que enviar un método DEL hacia el servidor de TrustedX con el identificador de la firma en el servicio (obtenido al listar las identidades o registrar una nueva). De nuevo, tenemos que autorizarla con el token ya existente.

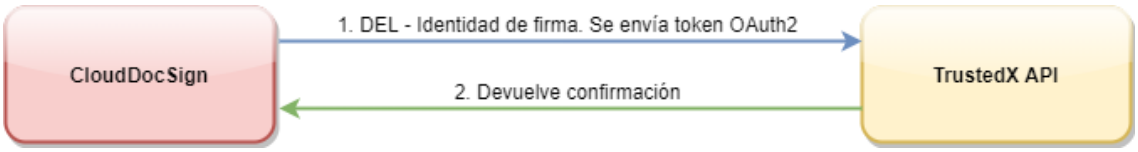


Figura 17: flujo para eliminar una identidad de firma

Por último, relativo a la integración con el proveedor eIDAS, tenemos que crear la firma en el servidor con los dispositivos seguros. Para ello, haciendo uso de una de las identidades de firma ya registradas, generaremos el PKCS1 con el hash del documento, firmado con la clave privada de la identidad de firma seleccionada. Sin embargo, hay un punto muy importante con esta operación. Debido a la relevancia e importancia que tiene firmar un documento, no se puede utilizar el token OAuth2 que ya tenemos. Para obtener la firma del hash en formato PKCS1 (cifrar el contenido con la clave privada asociada a la identidad de firma) tendremos que obtener un nuevo token mediante Authorization Code Grant con un nuevo ámbito de TrustedX.

- urn:safelayer:eidas:sign:identity:use:server: permite generar el PKCS1 con la identidad de firma.

Obtener este nuevo token requiere que se vuelva a realizar un flujo de autorización hacia TrustedX en el que el usuario tendrá que dar su consentimiento sobre la firma para que CloudDocSign pueda recibir el token. Dicho token será de un solo uso, siendo desechado cuando la firma se haya completado con éxito y el proceso haya finalizado. Una petición a la API para firmar el hash tendrá que tener, obligatoriamente, un nuevo flujo OAuth2 para obtener el token.

La necesidad de establecer este flujo y obtener un nuevo token OAuth2 reside en que **se debe obtener una autorización explícita para la activación de la clave de firma**, tal y como se indica en la normativa eIDAS. En ella se puede ver como se precisa que, para cada firma a realizar en un documento, es obligatorio que los usuarios autoricen explícitamente el uso de su clave para cada firma.

Una vez dicho esto, pasamos a mostrar el flujo completo del proceso.

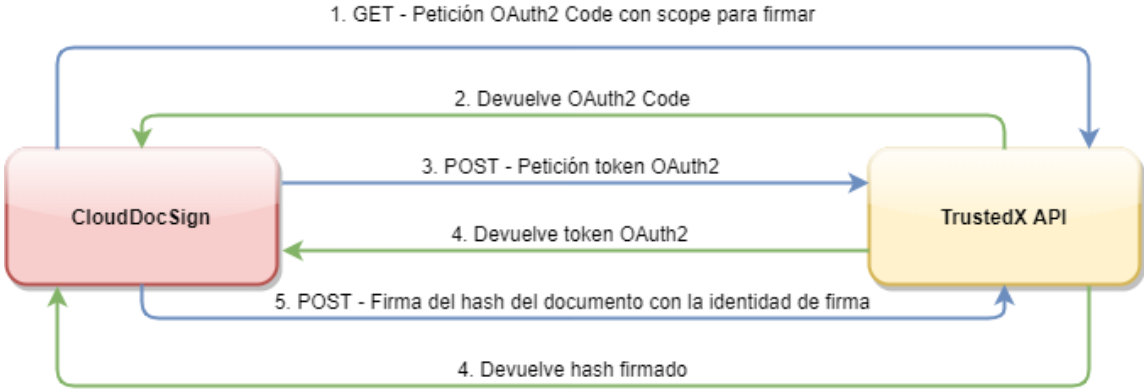


Figura 18: flujo para realizar la firma del hash de un documento

Podemos apreciar como en primer lugar obtenemos el token con el ámbito para autorizar la firma. Una vez obtenido, se realiza un POST al servidor pasando tanto la identidad de firma a emplear como el hash del documento. Si todo va con éxito, se recibirá el hash firmado, es decir, el PKCS1 para construir en la aplicación la firma PAdES y generar el documento *pdf* firmado.

En línea con lo comentado anteriormente en el apartado 2.3.2.2.3 Autenticación en proveedor de firma digital, **en el demostrador de este trabajo estamos empleando un proveedor de identidad de TrustedX que no cumple con el LoA necesario para tratar con identidades cualificadas**. En este sentido, la firma que realicemos no puede ser cualificada y, por tanto, el certificado empleado para la firma tampoco lo es. Por este motivo, el certificado raíz del certificado empleado en la firma debe ser añadido a los diccionarios de confianza de los equipos para poder validar la firma correctamente.

Por otro lado, de la misma forma en un entorno real la autorización para emplear la identidad cualificada al firmar se debería realizar mediante una clave única para cada usuario generada por el HSM, así se cumpliría lo que la normativa eIDAS exige. En este escenario demostrativo esto no ocurre así y **simplemente se exige autorizar el permiso de firma del usuario sin introducir clave alguna durante la firma**. La clave empleada por cada usuario para la firma se genera en el momento de **registrar una identidad de firma a partir de un PKCS12 subido al servicio TrustedX**.

En resumen, la firma generada en este demostrador no puede ser cualificada al emplear una clave privada obtenida de un PKCS12, ya que no se puede demostrar

donde se generó o quién pudo acceder a ella. El modo correcto sería emplear una identidad de firma *cloud* creada a partir de unas claves generadas en un HSM, ya que sólo así se puede garantizar la identidad.

Sin embargo, volvemos a recalcar que **estas particularidades son inherentes al escenario demostrativo y que en ningún caso van a requerir un cambio en la aplicación desarrollada**. No es necesario modificarla porque para obtener una firma cualificada sólo deben modificarse los requisitos de autenticación y autorización en el momento de autorizar la firma por parte del servicio TrustedX y esto es independiente de nuestra aplicación. Es decir, sólo habría que emplear un servicio TrustedX con un LoA que permita obtener identidades de firma cualificadas. La autorización y firma cualificada siempre se realiza en el proveedor de firma digital, nuestra aplicación simplemente se encargará de obtener los datos firmados y ensamblar el documento PDF final.

2.5.2 Servicio de validación de firma

En este trabajo también hemos implementado una funcionalidad que permite la validación y verificación de firmas digitales contenidas en documentos PDF. Se trata de un servicio DSV (Digital Signature Verification) que nos devolverá la información de las firmas PAdES contenidas en un documento, así como el certificado que la realizó, etc.

Este servicio está basado en SOAP, por lo que no es un servicio similar a los demás. En este caso, no disponemos de autenticación mediante OAuth2, si no que la autenticación se basa en WS-Security (WSSE) que no es más que un protocolo de comunicaciones que suministra un medio para tener seguridad en servicios web. En concreto se hace uso del elemento *wsse:UsernameToken* introducido en la cabecera *wsse:Security* del mensaje SOAP para proporcionar un nombre de usuario y contraseña que permita autenticar al usuario y añadir seguridad al servicio.

Por lo tanto, la autenticación simplemente se basará en añadir esta cabecera al mensaje SOAP que enviemos al servicio web. A su vez, el canal de comunicación entre la aplicación y el servicio de validación de firma estará protegido por una conexión HTTPS. En este sentido, los datos enviados tendrán confidencialidad.

Como hemos comentado, es un servicio web SOAP, por lo que se debe enviar un mensaje XML que cumpla con la especificación definida en el servicio web mediante un esquema XML. Este esquema definirá los elementos que tendrá que tener un objeto XML enviado, así como las ocurrencias de los mismos, etc. Para nosotros, simplemente va a ser invocar a una operación enviando el documento PDF codificado en Base 64. Una vez procesado, recibiremos un mensaje SOAP con el resultado de la validación de firma. Si hubiera algún error en la comunicación y/o formato del mensaje recibiríamos un mensaje SOAP Fault indicando el motivo del error. A continuación, se muestra el flujo seguido por nuestra aplicación para consumir este servicio.

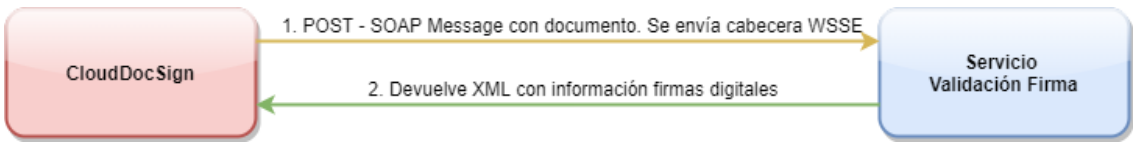


Figura 19: flujo para obtener las firmas de un documento

En concreto, vamos a hacer uso de la operación *Verify* para obtener la validación de las firmas encontradas en un documento. En nuestro caso, la aplicación va a ser capaz de detectar los siguientes casos:

- Que el documento PDF no esté bien formado.
- Que la firma del documento no sea válida
- Que el documento haya sido modificado después de ser firmado. No cumple la integridad del mismo.
- Que el documento sea válido en la revisión firmada, pero contenga modificaciones posteriores no firmadas.
- Que el documento esté correctamente firmado y la firma sea válida.

En cualquiera de estos casos nuestra aplicación notificará al usuario con un mensaje y la información más relevante de la firma contenida en el documento.

Por lo tanto, nuestra aplicación CloudDocSign es capaz de definir un flujo completo de portafirmas y actúa en todo momento como un engranaje que es capaz de unir distintos sistemas para crear firmas digitales en documentos PDF. Podemos obtener los documentos de servicios de almacenamiento cloud, firmarlos mediante un servicio cualificado de firma digital y, posteriormente, validar la firma que hemos realizado para ver si es correcto y valido. Además, en todo momento hemos preservado y mantenido la seguridad y privacidad del contenido de los documentos con los que tratamos.

2.6 Firma electrónica y estándar PAdES

El principal hito de este trabajo consiste en realizar una firma digital cualificada haciendo uso de distintos servicios. Nuestra aplicación será la encargada de orquestar todas estas integraciones para conseguir obtener una firma segura y útil, siendo así proveedores de un servicio cada vez más necesario en el mundo digital en el que nos encontramos.

Por este motivo, vamos a explicar en este apartado que es una firma digital, que tipos existen y motivar la elección de la firma digital cualificada. Posteriormente, explicaremos el estándar PAdES (ETSI TS 102 778) que será el empleado para generar la firma digital en los documentos *pdf*. Este estándar define una estructura y unos datos que se deben crear para generar la firma digital.

Los tipos de firma electrónica reglados por ETSI (European Telecommunications Standars Institute) que existen en la actualidad son:

- Simple: fácil de utilizar, permite identificar al firmante, pero no aporta seguridad.
- Avanzada: más segura que la anterior. Identifica al firmante y aporta integridad de los datos firmados. Válida como evidencia jurídica.
- Reconocida o cualificada: es la más segura de todas. Aporta lo mismo que la avanzada, pero debe ser amparada por un certificado reconocido y realizada por un dispositivo seguro de creación de firma. Este dispositivo debe cumplir el Reglamento EU 910/2014.

Por este motivo, una vez explicado lo que requiere una firma electrónica cualificada podemos entender la necesidad de utilizar un proveedor externo que cumpla con la normativa eIDAS. Estas firmas deben ser generada por autoridades de confianza certificadas, de ahí la necesidad de delegar este paso.

Así pues, TrustedX nos aporta este servicio de firma generado por dispositivos electrónicos seguros y cualificados. Particularmente, se denominan HSM (Hardware

Security Module) y son dispositivos criptográficos basados en hardware que generan, almacenan y protegen claves criptográficas.

Por otro lado, el estándar PAdES define la especificación para firmas digitales del formato PDF 1.7 (ISO 32000-1) con la finalidad de poder incluir firmas electrónicas avanzadas en los documentos PDF. Una de las principales ventajas de este estándar es que se implementa mediante software que está ampliamente disponible no siendo necesario disponer de un software específico para validar estas firmas.

Además, define una serie de extensiones adicionales que sirven para mantener la validez de las firmas durante largos periodos de tiempo y que cumplen con todos los requisitos de la Directiva europea sobre un marco comunitario para firmas electrónicas (Directiva 1999/93 / CE).

A continuación, se explican brevemente las extensiones PAdES.

- PAdES-CMS: Define una firma CMS/PKCS#7 basada en el estándar ISO 32000-1. Este es el perfil soportado por la mayor parte de las soluciones de software que generan firmas en PDF.
- PAdES-BES: Este perfil define una firma PDF avanzada basada en CAdES-BES que además incorpora la opción de incluir en la firma un sello de tiempo.
- PAdES EPES: Este perfil especifica una firma PDF avanzada basada en CAdES-EPES. Es similar a PAdES-BES añadiendo un identificador de política de firma y, opcionalmente, una referencia al tipo de compromiso adquirido.
- PAdES-LTV: Es el formato de firma longeva. Este perfil permite prorrogar por tiempo indefinido la validez de las firmas en formato PDF. Garantiza la validación de la firma a largo plazo.

Por este motivo, utilizamos el proveedor eIDAS para obtener el valor de la firma calculada de forma segura y poder generar la firma PAdES correspondiente acorde a las especificaciones de la extensión del estándar.

3. Diseño e implementación de la solución

3.1 Diseño

En este punto de la memoria vamos a explicar el diseño que vamos a llevar a cabo para implementar los flujos vistos en apartados anteriores. Se van a enumerar las distintas entidades que van a formar parte de nuestra solución software, así como los hitos que tenemos que alcanzar y cubrir para que los flujos queden correctamente implementados. Seguidamente se explicará el diseño software que vamos a implementar para la tecnología elegida y que vistas van a formar parte de nuestra aplicación. Por último, se llevará a cabo una explicación más técnica de las tecnologías y aspectos de implementación específicos que consideramos más importantes.

3.1.1 Entidades

En el marco del diseño de esta aplicación encontramos 6 entidades que estarán involucradas durante el flujo de firma ya definido. A continuación, podremos verlos.

- Usuario: representa a los usuarios comunes que utilizarán la aplicación.
- CloudDocSign: es el portal desarrollado que proporcionará el servicio.
- Proveedor eIDAS (TrustedX): proveedor eIDAS que nos aportará los mecanismos para poder obtener el hash del documento firmado PKCS1 y la validación de las firmas de un documento.
- Proveedor de identidad de Google: representa la conexión con Google Identity para autenticar el acceso de los usuarios a CloudDocSign.
- Proveedor de almacenamiento Drive: proveedor de almacenamiento en la nube para obtener los documentos PDF.
- Proveedor de almacenamiento Dropbox: proveedor de almacenamiento en la nube para obtener los documentos PDF.

3.1.2 Hitos

El hito más importante que podemos encontrar en esta aplicación es conseguir obtener un documento de un proveedor en la nube y poder firmarlo. A continuación, se muestran los hitos más relevantes que debemos conseguir en este trabajo.

- Permitir la descarga, firma y subida de un documento PDF al proveedor de almacenamiento seleccionado con previo consentimiento por parte del usuario.
- Permitir el acceso a CloudDocSign mediante una cuenta de Google del usuario.
- Permitir la visualización y descarga de los documentos de los proveedores de almacenamiento, previo consentimiento del usuario, para realizar la operación de firma y validación de firma.
- Permitir la autenticación del usuario en una cuenta eIDAS, así como gestionar sus identidades de firma, previa autorización por parte del usuario.
- Permitir el acceso a los documentos de los proveedores de almacenamiento para llevar a cabo la validación de la firma de los mismos.

3.1.3 Diseño arquitectura

En este punto vamos a explicar como se han diseñado las vistas de la aplicación, así como la estructura del menú y los distintos patrones de diseño empleados durante el desarrollo.

La aplicación está basada en un núcleo que sigue el patrón Modelo Vista-Controlador (MVC). En la figura 19 se puede ver un diagrama del concepto.

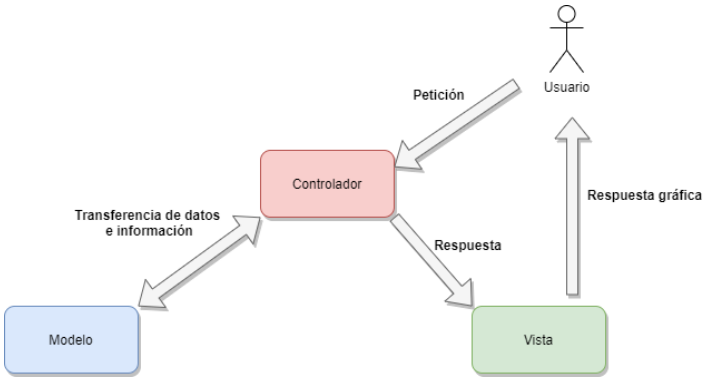


Figura 20: diagrama MVC general

Se aprecia que el usuario interactúa con el controlador que será el que se encargue de interactuar con las entidades necesarias (proveedores de almacenamiento, proveedor eIDAS, etc.). Una vez se obtiene toda la información, se genera la vista y se muestra al usuario.

Además de este patrón de diseño muy común en el desarrollo de aplicaciones, se han integrado otros patrones que nos van a permitir dotar a la aplicación de un marco de desarrollo común para los distintos proveedores, así como dotar de un prototipo de operaciones mínimas que se deben satisfacer. Hacemos uso del patrón de método plantilla que será empleado para las entidades que representan las cuentas de los usuarios en los proveedores de almacenamiento. Básicamente se necesita disponer de un marco común para poder proporcionar a la vista la información básica. De esta forma, se crea una clase abstracta que proporcionará la información básica que se debe proporcionar para que cada integración relativa a los proveedores de almacenamiento herede de ella y completen la información.

Por otro lado, se implementa un patrón adaptador que es la pieza fundamental para un correcto desempeño de los flujos. Se han creado dos interfaces, una para cada tipo de proveedor (almacenamiento y eIDAS). Estas interfaces definen los métodos mínimos y básicos que deben ser implementados por los proveedores para poder dar soporte al flujo definido para la aplicación. Así pues, estas interfaces definen métodos para listar los ficheros, descargarlos, firmarlos o crear identidades de firma, entre otras. De esta manera nos aseguramos que cualquier proveedor que se integre con la aplicación va a cumplir los requerimientos mínimos para un correcto desempeño de la misma.

En cuanto a la integración con el proveedor de identidad de Google, se ha llevado a cabo de forma totalmente integrada con el núcleo de la misma. De esta forma conseguimos que, de una forma muy sencilla, que el acceso a cualquier vista queda perfectamente protegido. Todo se gestiona mediante el framework Spring Security que nos permite de una forma sencilla integrar y gestionar este aspecto de una forma óptima. Por ejemplo, podremos definir que vistas serán protegidas y cuales no (página de login, entre otras). También gestiona el tiempo de validez del token OAuth2 obtenido en el proceso de autenticación con el IdP de Google, facilitando la obtención de uno nuevo en caso de ser necesario.

Dentro del núcleo de la arquitectura se han definido una serie de APIs que serán las encargadas de implementar todas las operaciones necesarias para integrarse con los distintos servicios REST y SOAP ya indicados anteriormente. Estas clases son propias de cada proveedor (Dropbox, Google y TrustedX) que se encargan única y exclusivamente de implementar la funcionalidad necesaria para enviar y recibir respuestas. En el apartado 3.2 Implementación se podrán ver detalles más específicos acerca de estas APIs.

Un aspecto muy importante es que hemos necesitado definir que información se necesita disponer de forma persistente a nivel de sesión. Es decir, información que debe estar disponible durante el tiempo que la sesión del usuario sea válida en CloudDocSign. Para ello, hemos creado un *bean* válido para el tiempo de vida de la sesión en el que almacenaremos toda esta información. Entre otros, se almacenarán los distintos tokens OAuth2 de acceso a los proveedores, la identidad de firma elegida para realizar la firma, la identidad del usuario autenticado en CloudDocSign o en los proveedores, entre otros datos. Por otro lado, también dispondremos de otros *bean* válidos para el tiempo de vida de una vista, en el que almacenaremos otra información menos vital pero que es necesaria para un correcto funcionamiento de los diversos controladores. De la misma forma, nuestra aplicación tendrá tantos controladores como vistas disponga para realizar todas las operaciones solicitadas por el usuario.

Una vez expuestos estos detalles de diseño de la arquitectura, pasamos a ver la estructura que se ha decidido diseñar para el menú y las distintas vistas. La forma y secciones en las que vamos a presentar la información al usuario se estructura de la siguiente forma.

- Vista para realizar login: redirigirá al usuario al IdP de Google para autenticarse.
- Vista principal con el menú: se disponen de las siguientes opciones.
    - TrustedX. Este apartado se encargará de realizar toda la integración con el proveedor eIDAS. Se gestionará la autenticación como la gestión de identidades de firma y el inicio de la misma.
    - Documentos. Este apartado mostrará en la parte principal de la vista dos botones para iniciar el proceso con Google Drive o Dropbox. Una vez autenticado en un servicio se mostrará una lista paginada con los documentos PDF disponibles. Por cada documento se mostrará un botón para descargar, firmar o validar la firma del documento.
    - Cerrar sesión. Finalizará la sesión en CloudDocSign borrando toda la información de sesión disponible.

A grandes rasgos, esas son las vistas con las que se podrá realizar todo el flujo. En el apartado 4 de este documento se podrá ver un detalle más específico con ilustraciones.

3.2 Implementación

En este apartado vamos a detallar aspectos más técnicos de la implementación. Se detallarán tecnologías y componentes utilizados para llevar a cabo el desarrollo del flujo en la aplicación.

Se trata de una aplicación que utiliza Maven como gestor de dependencias y que ha sido implementada en el lenguaje de programación Java haciendo uso del framework Spring Boot que nos permite dotarla de una arquitectura MVC. Para la gestión de la seguridad relativa a la aplicación web hemos integrado Spring Security. Es otro framework que nos permite gestionar de forma sencilla la protección de las distintas

vistas de nuestra aplicación web. Por otro lado, se ha utilizado el IDE Eclipse que nos va a permitir desarrollar el proyecto con las tecnologías indicadas desde las primeras etapas hasta las pruebas finales realizadas sobre la aplicación.

A continuación, se detallan los componentes más relevantes que se han creado la aplicación.

3.2.1 Controladores

Son una parte fundamental de la arquitectura. Se encargan de realizar las acciones necesarias para satisfacer las peticiones de los usuarios. En nuestro caso, se encargarán de orquestar la integración con los distintos proveedores de nuestro escenario. En particular, se van a encargar de realizar las peticiones a las distintas APIs que explicaremos más adelante para obtener la información que permita completar el flujo de firma y el de validación.

Los controladores más relevantes son los siguientes:

- Index: es el encargado de lanzar la autenticación del usuario en la plataforma.
- ServicioAutenticación: este controlador es empleado para gestionar los parámetros de autenticación a través de SecurityConfig para SpringSecurity, así como manejar los eventos producidos al producirse una autenticación correcta y un cierre de sesión correcto. Para ello se emplean las clases AuthenticationSuccessHandler y LogoutSuccessHandler.
- DriveBean: gestiona las interacciones de los usuarios para tratar las peticiones al proveedor de almacenamiento Google Drive. A su vez, dispone de un controlador para recibir las respuestas redirigidas tras la autenticación contra el servicio.
- DropboxBean: gestiona las interacciones de los usuarios para tratar las peticiones a Dropbox (listar documentos, descargar, etc). También dispone de un controlador para recibir las respuestas redirigidas tras la autenticación contra el servicio.
- TruxtedXBean: este controlador implementa toda la funcionalidad para interaccionar con la API de TrustedX que explicaremos más adelante. Se encarga de gestionar las interacciones de los usuarios para iniciar los procesos de firma o registrar identidades de firma. También dispone de un controlador para recibir las respuestas redirigidas tras la autenticación contra el servicio.

3.2.2 Modelos

Se corresponden con una serie de APIs que implementan la funcionalidad de cada entidad con la que se integra CloudDocSign. Han sido creadas con el propósito de encapsular la funcionalidad específica de cada proveedor. En este contexto particular, no representan una entidad como tal si no la gestión para la integración con los distintos proveedores mediante REST o SOAP. Como ya se comentó en apartados anteriores, las APIs deben implementar las interfaces creadas por el patrón adaptador. De esta forma, cada servicio que quiera integrarse en la aplicación tendrá que proporcionar unos servicios básicos que se satisfacen mediante estas interfaces. IDocumentProvider es la que proporciona los métodos para los proveedores de almacenamiento. La funcionalidad que debe proporcionar es obtener un listado de documentos, descargar un documento, subir un documento y obtener información de la cuenta del usuario. Por otro lado, IEidasProvider es la encargada de proporcionar la funcionalidad para los proveedores eIDAS. La funcionalidad que debe proporcionar es

borrar, registrar u obtener las identidades de firma y, firmar un documento. Esto nos permitirá desligar las características que pueda tener cada proveedor para acoplar más fácilmente otros proveedores en un futuro.

3.2.2.1 Google Drive

Esta API implementa la comunicación con la API de Google Drive. Este acceso se ha realizado con peticiones REST mediante la librería Google-Api-Services-Drive para Java. Implementa la funcionalidad para satisfacer el flujo OAuth2 de autenticación visto en el apartado 2.3.2.2.1 Google Drive que permite autenticar y autorizar el acceso a los servicios y documentos de la plataforma. De la misma forma obtiene la información del perfil autorizado en Drive, así como satisfacer los requisitos de funcionalidad definidos por IDocumentProvider.

3.2.2.2 Dropbox

Esta API es muy similar a la anterior, sólo que se conecta con un proveedor de almacenamiento distinto. El acceso vuelve a ser a través de peticiones REST haciendo uso de la librería Dropbox-core-sdk para Java. Implementa los flujos de autenticación OAuth2 definidos en el apartado 2.3.2.2.2 Dropbox, así como los flujos de interacción con el servicio definidos en el apartado 2.4.1 Dropbox. Implementa IDocumentProvider.

3.2.2.3 TrustedX

Esta API contiene la funcionalidad principal del propósito del trabajo. Es la encargada de orquestar las acciones necesarias para generar la firma PAdES en el documento. Como comentamos en el apartado 2.5.1 TrustedX, la firma se va a componer en la propia aplicación CloudDocSign. Se hará uso del servicio de firma de TrustedX para obtener el PKCS1 a partir del hash del documento que será enviado al servicio. En este apartado vamos a explicar con más detalle como se ha implementado toda esta funcionalidad básica e imprescindible.

Esta API implementa la interfaz IEidasProvider para proveer de la funcionalidad básica que es necesaria para este propósito. Sin embargo, para realizar la comunicación con el servicio TrustedX no hemos podido emplear librerías como en los proveedores de almacenamiento. En este caso, hemos tenido que proveer de unas funciones que nos permitan realizar peticiones REST bajo las especificaciones del servicio de TrustedX. Por ejemplo, hemos tenido que permitir que una petición REST sea capaz de enviar distintos tipos de autenticación, así como distintos tipos de contenido (Content-Type de tipo de tipo JSON o form-urlencoded). Por lo tanto, esta API ha sido la más trabajada y diseñada a medida. Sin embargo, implementar el patrón adaptador nos permite que sea cual sea la forma de comunicación con el servicio en cuestión, tenemos la certeza de que vamos a tener la funcionalidad requerida.

Entre los métodos que proporciona encontramos los básicos para realizar la autenticación/autorización en el servicio (diferenciaremos el proceso de autenticación con la plataforma del proceso de obtención de un token OAuth2 para firma). También podremos obtener las identidades de firma, como registrarlas o borrarlas. Para estas funcionalidades, se permite la subida de un fichero PKCS12 a la aplicación, que tendrá que disponer también de la contraseña para desbloquearlo.

A continuación, se presentan dos subapartados explicando con más detalle como se realiza el proceso de composición y generación de la firma en CloudDocSign, así como la validación de las firmas de un documento.
ServerSignature: haces uso de *iText* para firmar, en este paso generas el PKCS1 con la llamada a TrustedX

3.2.2.3.1 Proceso de firma

El proceso de firma se inicia con la selección de una identidad de firma. Una vez se ha escogido, se debe seleccionar un documento para continuar. Cuando se dispone de toda la información, desde la aplicación se pulsa el botón para iniciar el proceso. En ese punto, el usuario es redirigido a la plataforma TrustedX para obtener un token específico de firma. Como se comentó en el apartado 2.5 Proveedor de servicio eIDAS, la firma de un documento debe tener la autorización explícita del usuario, deber dar su consentimiento para cada nueva firma.

Llegados a ese punto, se comienza desde CloudDocSign con la generación de la firma PAdES. Para este proceso hacemos uso de la librería *iText* de Java que nos permite tratar con documentos PDF, así como crear este tipo de firmas en un documento. Pero para poder crear la firma, necesitamos obtener el certificado correspondiente del firmante. Esto lo podemos conseguir a través del servicio eIDAS implementado. TrustedX nos proporciona un método para obtener todos los detalles de la identidad de firma en cuestión y, por eso, antes de iniciar la firma obtendremos los detalles de dicha identidad del servicio TrustedX. Ahí podemos encontrar, entre otra información, el certificado digital del firmante.

Una vez se dispone del certificado, entra en juego la librería *iText* para la generación de la firma PAdES. Se crea la firma incluyéndole el certificado de usuario, una ubicación y razón, así como indicando el nivel de certificación del documento. En este caso, optamos por el nivel *NOT_CERTIFIED*, el cual indica que el documento contiene una firma no certificada, también conocida como firma de destinatario. En este caso, se pueden incluir una o más firmas de destinatario para su aprobación, pero no se permite la modificación del documento para completar campos de un formulario (para permitirse se debería cambiar el nivel de certificación a otro superior).

Una vez creado el contenedor para la firma, se incluye en el documento indicando su posición y se invoca a la generación de una firma PAdES Detached en el documento PDF seleccionado. Es importante en este punto, indicar a la librería iText el proveedor del contenido del hash firmado, ya que vamos a emplear una firma de tipo detached con un firmante externo. El que sea un firmante externo quiere decir que el contenido de la firma va a ser creado en un servidor, en este caso el servicio eIDAS de TrustedX que nos proporcionará el hash firmado (objeto PKCS1).

Por otro lado, que la firma sea detached de forma estricta significa que la firma y el documento firmado van por separado, es decir, son dos documentos distintos. Sin embargo, en un documento PDF no es del todo correcta esta afirmación, ya que una firma detached en PDF indica que la firma igualmente va contenida en el mismo archivo, pero los datos firmados van dentro del propio contenedor de firma asociado al archivo pdf.

Como se ha indicado, el contenido firmado va a ser obtenido a través del servicio TrustedX. Aquí entra en juego una entidad auxiliar del a API de TrustedX. Hablamos de *ServerSignature*, que consiste en una clase Java que implementa un servicio de firma externo de *iText* para obtener la firma PKCS1. En nuestro caso, esta clase

implementa el método *sign* que será en encargado de realizar la petición POST de la API REST de TrustedX para enviar el hash del documento original y obtener dicho hash firmado con la identidad de firma seleccionada. La firma será realizada sobre un hash SHA-256 y encriptada con el algoritmo asimétrico RSA. Esto nos devolverá un array de bytes que será empleado internamente por *iText* para generar la firma PAdES detached ya indicada.

Cuando el proceso finaliza, si ha sido exitoso y no se ha devuelvo ninguna excepción, se obtendrá el documento PDF que será subido al proveedor de almacenamiento correspondiente al documento original. En este punto, el usuario ya dispondrá en ese servicio de su documento original pero ahora firmado por el servicio eIDAS. Como vemos, en ningún momento el contenido del documento original del usuario ha sido enviado a un servicio externo y la generación de la firma ha sido realizada íntegramente en la aplicación CloudDocSign.

3.2.2.3.2 Validación de firma

Para la validación de las firmas de un documento PDF emplearemos de nuevo uno de los servicios de TrustedX. En este caso, se trata de un servicio SOAP mediante el cual intercambiaremos el documento PDF y nos devolverá la información correspondiente para las firmas encontradas, si las hay. No tiene porque ser un documento firmado con CloudDocSign, cualquier que contenga firmas se podrá comprobar. La validación, en contra del método que empleamos en la firma, si requiere que se envíe el documento hacia un servicio externo. Si bien, la privacidad podría verse comprometida, se establecerá una conexión SSL entre CloudDocSign y el servicio de validación de firma. La integridad del documento en todo momento queda protegida mediante la firma digital que contenga, en caso de disponer de ella.

Para implementar esta funcionalidad hemos creado la clase DSVService que se encargará de implementar la comunicación con el servicio de TrustedX, así como realizar la petición SOAP componiendo el mensaje correspondiente. La operación que se invocará es *verify* y en caso de error devolverá una excepción (SOAP Fault) indicando el motivo del error. El servicio SOAP está autenticado mediante WS-Security, por lo que cumple con unos parámetros de seguridad acordes a la importancia del servicio.

El resultado obtenido es tratado, dando soporte a cinco posibles casos en base al documento proporcionado.

- Que el documento no tenga firmas digitales.
- Que el documento contenga una o varias firmas digitales correctas.
- Que el documento contenga firmas digitales no correctas o bien definidas.
- Que el documento contenga una firma digital válida para una revisión del documento, pero que contenga cambios posteriores no firmados.
- Que el documento contiene una firma digital que no es válida porque el documento o la firma han sido manipulados a posteriori.

Este servicio de validación de firma está disponible desde cualquiera de los dos proveedores de almacenamiento implementados en esta aplicación. Para cada fichero, existe un botón que permite proceder con la validación de firmas del mismo, notificando al usuario del resultado del proceso basándose en los cinco casos explicados.

3.2.3 Entidades lógicas

En este apartado se van a exponer las clases que se han definido para modelar las entidades lógicas que son necesarias en el contexto de nuestra aplicación. Se encargan de modelar objetos que serán necesarios para almacenar determinada información que tendrá que estar disponible a nivel de sesión.

- UsuarioAutenticado: esta entidad representa las credenciales del usuario que está autenticado en CloudDocSign mediante el IdP de Google. Es creado a partir del objeto GoogleUser que obtenemos durante el proceso de autenticación con OAuth2 mediante Spring Security.
- DocumentProviderAccount: esta entidad implementa el patrón plantilla que debe ser utilizado por los proveedores de almacenamiento para cumplir la especificación de IDocumentProvider al obtener la cuenta de usuario del servicio. La implementan las entidades DriveAccount y DropboxAccount creadas en las respectivas APIs de comunicación con el servicio.
- DocumentSign: se crea esta entidad para representar el documento que va a ser firmado. Lo necesitamos para encapsular la información necesaria a transmitir de un proveedor a otro durante el proceso de firma. Se almacenará en sesión y se reinicializará por cada nuevo proceso de firma que se inicie.
- FirmaDocumento: esta entidad representa algo similar a la anterior, sólo que este se utiliza durante el proceso de validación de firma. Contiene la información necesaria a emplear durante la validación de firmas de un documento.
- IDocumentProvider: entidad para implementar el patrón adaptador.
- IEidasProvider: entidad para implementar el patrón adaptador.

3.2.4 Vistas

Las vistas representan el punto de partida para la interacción del usuario con nuestra aplicación. A partir de aquí se producen los eventos que iniciarán las acciones para seguir el flujo de firma o validación. Para implementarlas hemos utilizado Primefaces, que es una biblioteca con multitud de componentes para representar elementos en páginas webs. También se hace uso de jQuery, CSS y FontAwesome para complementar Primefaces. A continuación, se muestran las vistas más importantes.

- Vista login: interfaz que muestra un botón para iniciar sesión en la aplicación.
- Vista TrustedX: en esta interfaz se concentran las acciones para interactuar con el proveedor eIDAS. Se visualizan las identidades de firma, así como añadir una nueva o realizar la firma. Se muestra un cuadro indicativo con la información de los documentos e identidades previas al momento de firmar.
- Vista documentos: en esta pantalla se podrá seleccionar uno de los dos proveedores de almacenamiento disponibles.
    - Vista Google Drive: en esta pantalla se presentan las interacciones para listar, descargar, firmar y validar las firmas en los documentos que el usuario tiene alojados en el servicio.
    - Vista Dropbox: esta pantalla es similar a la de Google Drive. Permite las interacciones entre usuario y aplicación para listar, descargar, firmar y validar firmas de los documentos.
- Vista error: Empleada para mostrar un mensaje de error ante eventos no esperados o un funcionamiento anormal durante el flujo.

3.3 Puesta en producción

Para desplegar la aplicación es necesario generar el fichero que contiene la aplicación empaquetada. Para ello, haciendo uso de Maven, desde el entorno de desarrollo Eclipse se generará el archivo *.war* que contiene la aplicación CloudDocSign lista para su despliegue.

Una vez obtenido este archivo, se podrá proceder a su despliegue en un servidor de aplicaciones WildFly, concretamente la versión 18 de este software. Para ello, debemos ubicar el archivo *.war* generado en la carpeta *deployments* del servidor de aplicaciones y, posteriormente, iniciarlo para que nuestra aplicación esté disponible a través de un navegador web.

4. Flujo funcional

En este apartado vamos a realizar una demostración del flujo completo de firma y de validación de firmas de un documento. Se incluirán imágenes demostrativas.

Para acceder a la aplicación web tenemos que introducir la url en el navegador. Una vez hecho, nos mostrará la pantalla de inicio de sesión. Como ya se ha comentado, la autenticación y autorización se realiza mediante el IdP de Google empleando el protocolo OpenId.

Figura 21: pantalla de inicio de sesión

Una vez pulsado el botón, el usuario es redirigido al servidor de Google para proceder con su identificación. Si es correcta, posteriormente el usuario será redirigido de nuevo a CloudDocSign. Justo en este instante, la aplicación ha recibido un token OAuth2 de autorización con un tiempo de expiración. Si caduca, la sesión será marcada como no válida y es necesario que el usuario se vuelva a identificar. Todo este proceso es gestionado por Spring Security.

Una vez que el usuario ha sido redirigido de nuevo a la aplicación, se muestra la siguiente pantalla de bienvenida.

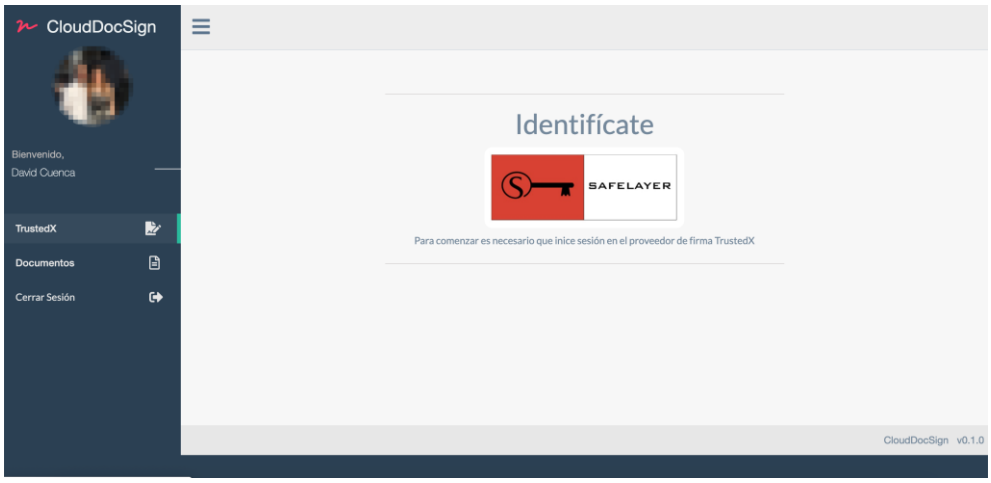


Figura 22: pantalla de acceso a la aplicación

En ella ya se está solicitando que el usuario se autentique en la plataforma TrustedX para iniciar el proceso de firma. En la parte superior izquierda, podemos ver un poco de información acerca de la identidad del usuario que se ha autenticado en el IdP de Google para acceder (vemos su nombre y fotografía de la cuenta de Google). Justo

debajo vemos el menú de la aplicación. Es muy sencillo y sólo dispone de una opción para acceder al proveedor eIDAS y otra para acceder a los proveedores de almacenamiento (Drive y Dropbox). También se muestra un botón para cerrar sesión en la aplicación. Este proceso haría que toda la información a nivel de sesión fuera destruida.

Siguiendo con los pasos definidos en el flujo, tenemos que autenticarnos en la plataforma de TrustedX para obtener las identidades de firma y poder seleccionar una. Se pulsa la imagen en el centro de la pantalla y se redirige al usuario al IdP de TrustedX. De forma análoga al IdP de Google, el usuario tendrá que identificarse ante el servicio para poder obtener el token OAuth2 correspondiente (válido para los *scopes* indicados durante este documento).

Figura 23: inicio de sesión en TrustedX

Si el usuario es autenticado correctamente, lo siguiente que nos muestra el IdP es una pantalla para autorizar los permisos indicados en el momento de iniciar sesión. El usuario tiene que aceptar que se gestionen las identidades de firma asociadas a él.

Figura 24: autorización de permisos en TrustedX

Si el usuario lo acepta, será redirigido a CloudDocSign para continuar el flujo y la aplicación obtendrá el token OAuth2. En este punto, también se solicitarán las identidades de firma para ser mostradas al usuario a la vuelta.

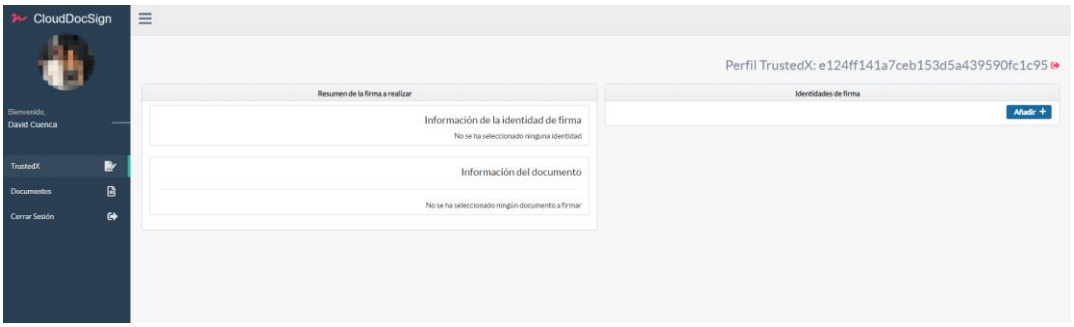


Figura 25: panel TrustedX en CloudDocSign

Como se puede ver, en la parte derecha aparece un panel con la información acerca de las identidades de firma. En este caso, aún no hay ninguna identidad registrada en el servicio TrustedX, por lo que se procede a pulsar el botón añadir. Seguidamente, se muestra una ventana en la que se puede añadir el fichero PKCS12 y cierta información para crear la identidad.

Figura 26: creación identidad de firma en TrustedX

Para este flujo de demostrativo, hemos creado una CA auto firmada con OpenSSL. A su vez, se ha generado un certificado de usuario firmado por esta CA que representa mi identidad de prueba para este supuesto. Con estos certificados, hemos generado el correspondiente fichero PKCS12 mediante comandos OpenSSL. Se debe indicar la contraseña del certificado PKCS12 y seleccionar una de las etiquetas disponibles. Esta información no es más que un método para identificar distintas identidades de firma que se suba, agrupándolo por distintos roles en el entorno en el que se ejecute CloudDocSign. Una vez completado, se envía al servidor de TrustedX para su creación. Si toda la información es correcta, se redirige al usuario a la pantalla anterior.

Figura 27: identidades de firma existentes en TrustedX

Se puede apreciar como ya se muestra la identidad creada. El siguiente paso es seleccionarla para su uso en la firma digital del documento PDF.

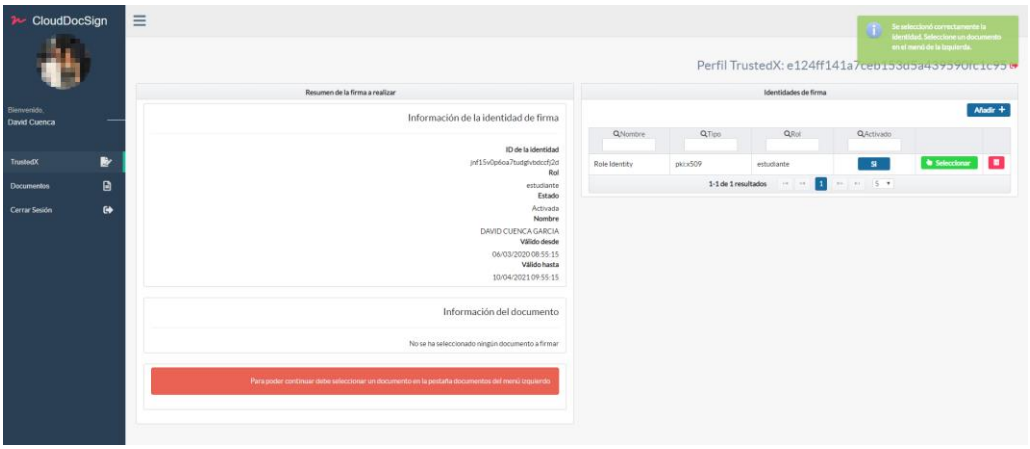


Figura 28: identidad de firma seleccionada

Al seleccionarla, se muestra un mensaje informativo al usuario para que sepa que pasos son los siguientes. También se puede apreciar en el panel izquierdo un pequeño resumen con la información más relevante de la identidad de firma seleccionada. Se puede ver el periodo de validez del certificado asociado, así como el rol y su estado en el servidor TrustedX.
Llegados a este punto, podemos ver como también nos está mostrando la aplicación un mensaje en rojo para que el usuario esté al tanto de cuales son los siguientes pasos a seguir. Ahora, se debe acceder a los proveedores de almacenamiento para seleccionar un documento. Se pulsa en la opción de menú correspondiente y se muestra la siguiente pantalla.

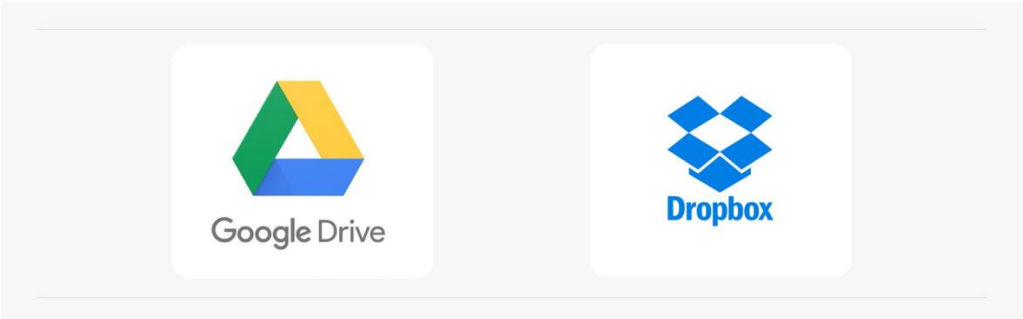


Figura 29: proveedores de almacenamiento de CloudDocSign

Se muestran dos imágenes haciendo referencia a los proveedores implementados. En este caso vamos a acceder con Dropbox. Es indiferente cual se implemente, ya que el mecanismo de autorización es similar (OAuth2) y el diseño de ambas vistas es idéntico (gracias, entre otros factores, a los patrones de diseño implementados).

Al pulsar sobre Dropbox, el usuario es redirigido al IdP de Dropbox para autenticarse.

Se muestra un formulario para autenticarnos. Si el proceso es exitoso, se mostrará otra página en la que se solicitará la autorización por parte del usuario de los permisos solicitados para la aplicación CloudDocSign. Si ya existe una sesión iniciada en el navegador, se mostrará directamente esta página para la autorización.

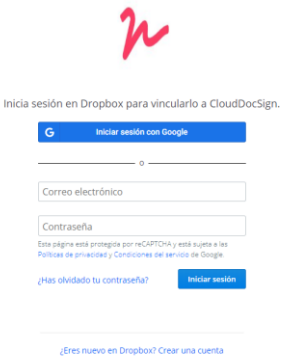


Figura 30: autenticación en Dropbox

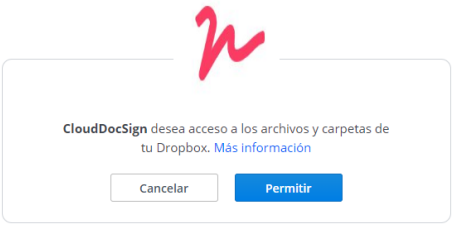


Figura 31: autorización en Dropbox

Una vez que el usuario haya aceptado los permisos, la aplicación obtendrá el token OAuth2 de Dropbox y se procederá a obtener un listado con todos los documentos PDF alojados en la cuenta del usuario para mostrárselos. Resaltar que el proceso en Google Drive es similar, también se solicitará al usuario permiso para autorizar el acceso a los documentos por parte de CloudDocSign. La obtención del token también será similar a los casos anteriores.

Siguiendo con el flujo, el usuario es redirigido a la aplicación y los documentos existentes en su cuenta son mostrados.

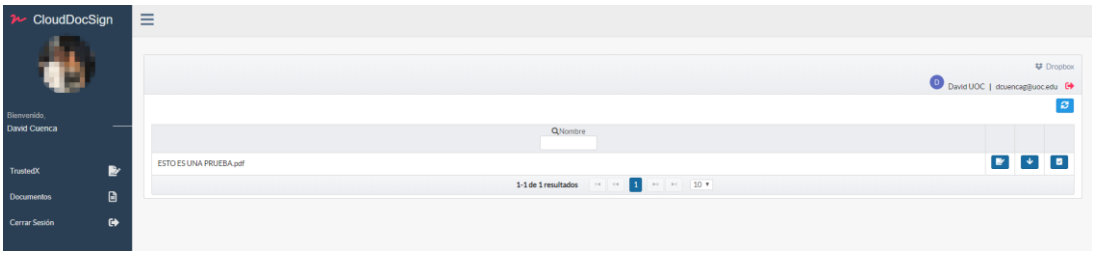


Figura 32: documentos en Dropbox

Se puede ver como a la derecha aparece información acerca de la cuenta conectada en Dropbox, así como un botón que permite cerrar sesión y comenzar de nuevo por si se desea emplear una cuenta distinta en el proveedor. Para realizar la firma, se pulsa el botón que aparece en la izquierda. El botón de que aparece en medio es para realizar la descarga y el de la derecha permite realizar la validación de las firmas. Siguiendo con el flujo, se pulsa el botón firmar.

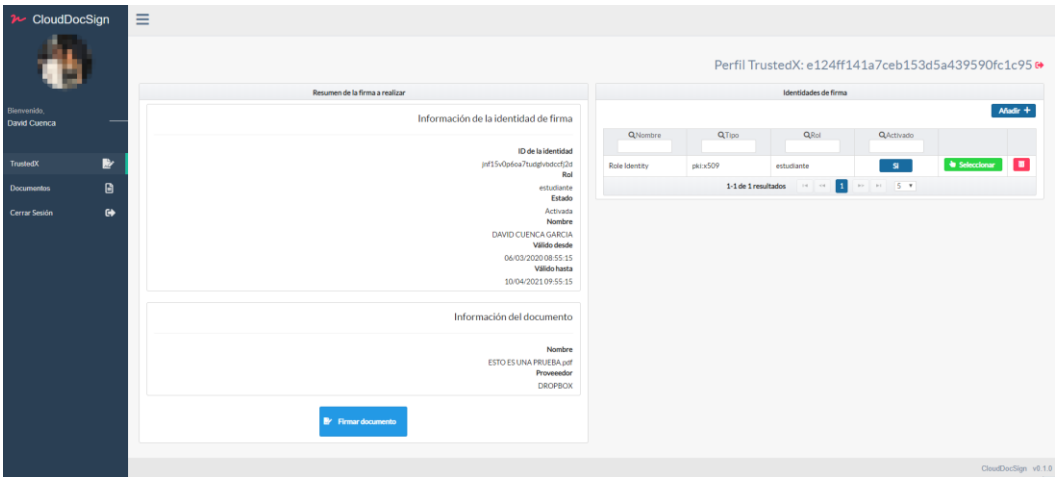


Figura 33: resumen de la firma a realizar

Al pulsar sobre el botón, el usuario es redirigido a la pantalla de TrustedX en la que se muestra el resumen completo de la información de la firma a realizar. Si toda la información es correcta, se muestra el botón que permite comenzar con la firma en TrustedX.

Una vez pulsado, el usuario es enviado al IdP de TrustedX para autorizar el proceso de firma. Recordemos que el reglamento indica que cada proceso de firma requiere de la autorización explícita por parte del usuario para la activación de la clave de firma.

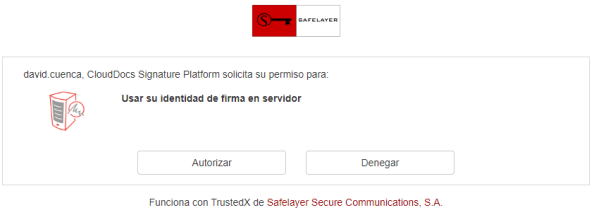


Figura 34: autorización de la firma a realizar

Si se autoriza, se comienza el proceso de firma en CloudDocSign. Como explicamos, se envía el hash del documento al servicio de TrustedX y se genera un documento PDF con la firma. Una vez finalizado, el documento es subido al mismo proveedor del que fue obtenido, añadiéndole la terminación "_FIRMADO". Se muestra el siguiente mensaje al usuario.

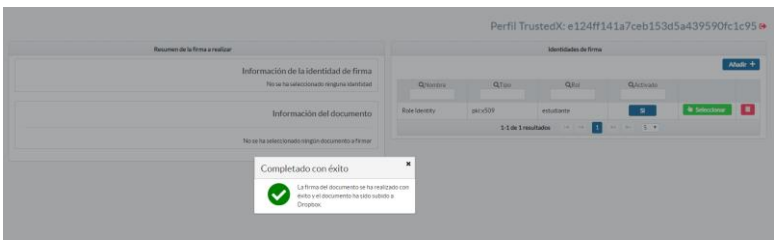


Figura 35: firma realizada con éxito

Así pues, si accedemos a los documentos de Dropbox en la aplicación, vemos como ahora aparece el documento firmado. Seguidamente, lo descargamos para revisar el contenido de la firma y su autenticidad con el software Adobe Acrobat Reader.

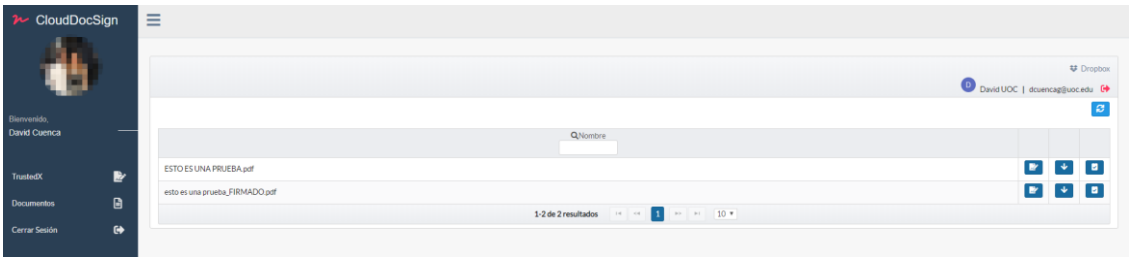


Figura 36: documento firmado en el proveedor de almacenamiento

Una vez abierto con el software, podemos abrir el panel de firma para revisar la información.

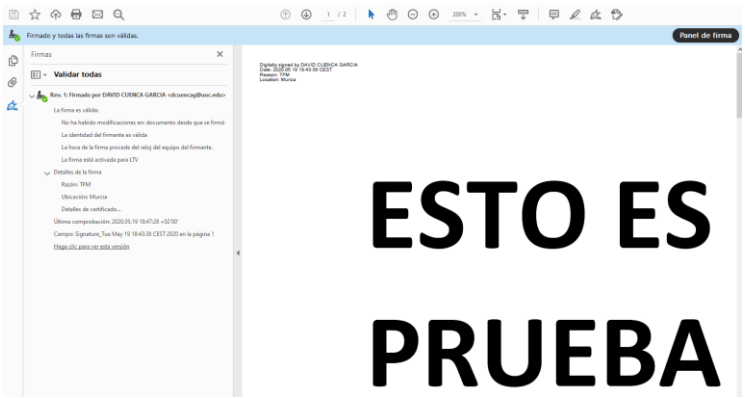


Figura 37: información de la firma en el PDF

Entre otras cosas, se puede ver que contiene sellado de tiempo y que es válida. También se garantiza la integridad del documento y la identidad del firmante.

Por último, queda añadir un ejemplo del uso del sistema de validación de firmas incorporado a la aplicación. Básicamente, para cualquier documento PDF (firmado o no por CloudDocSign) podrá mostrar información acerca de las firmas, su estado y del firmante.

Para comprobarlo, nos dirigimos a uno de los proveedores de almacenamiento. En este caso vamos a entrar en Dropbox, pero el mecanismo es similar para Google Drive. Una vez cargada la lista de documentos, pulsamos en el botón de la derecha, para validar las firmas del documento. Procedemos a pulsar sobre el documento recién firmado.

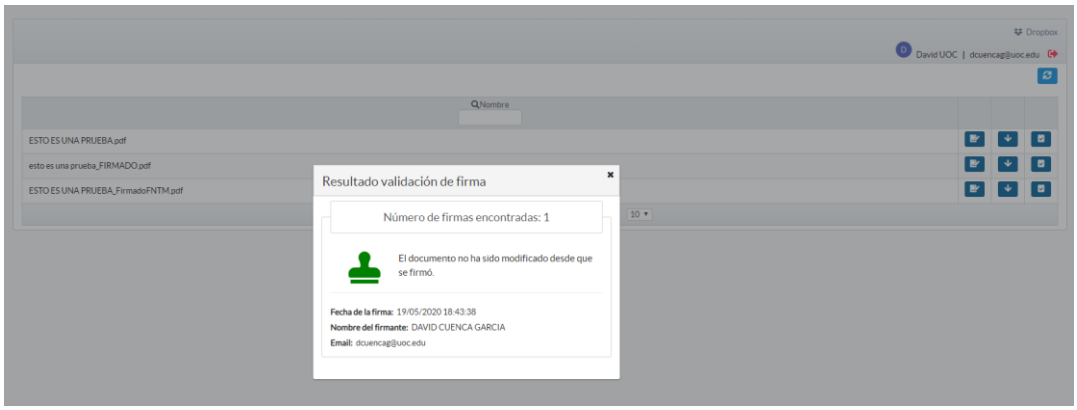


Figura 38: validación de documento con firmas

Podemos ver en la imagen, que el servicio ha detectado correctamente las firmas ubicadas en el documento y que la información es similar a la obtenida con Adobe Reader. Entre otros, podemos ver la identidad del firmante, la fecha de la firma, entre otros.

Por último, para demostrar el funcionamiento con otro documento, pulsamos en el botón para validar las firmas del original que hemos empleado para firmar.

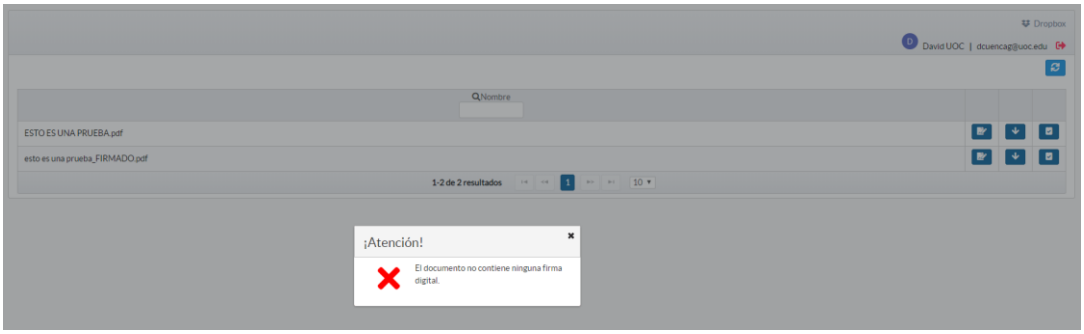


Figura 39: validación de documentos sin firmas

Como vemos, la aplicación muestra que ese documento no dispone de firmas digitales.

5. Conclusiones y vías futuras

Durante la realización de este proyecto queda patente la necesidad de que exista un marco común europeo de la firma electrónica. Es muy importante que este reglamento sea implementado y respaldado por todo el conjunto de países a nivel europeo, pues el futuro relacionado con la firma electrónica y las identidades digitales está cada vez más presente. Para ello, el reglamento eIDAS define las bases para que esta expansión sea una realidad comúnmente aceptada.

La aplicación desarrollada y presentada en este trabajo muestra la utilidad que tiene la firma electrónica adaptada a los entornos de los usuarios en su vida cotidiana. Se ha podido demostrar que es posible unir distintos servicios que permitan cumplir objetivos como implementar un mecanismo de firma electrónica cualificada haciendo uso de proveedores de documentos electrónicos ampliamente extendidos y utilizados. La aplicación y los servicios REST ofrecidos por TrustedX intentan simplificar la firma y gestión de identidades electrónicas para los usuarios no expertos en este campo.

Una de las premisas perseguidas en el diseño de esta aplicación ha sido en todo momento preservar la privacidad de los usuarios, intentando hacer uso de los mínimos datos sensibles, sin almacenarlos, así como diseñando un flujo de firma que no requiera la transmisión de los documentos por la red entre distintos servidores.

Por otro lado, la planificación y metodología que se han seguido han permitido desarrollar la aplicación de una forma ágil y sencilla. Definir los requisitos, así como crear el núcleo y arquitectura de la aplicación en los primeros pasos han permitido que la incorporación de los distintos servicios haya sido más eficiente. También hemos empleado lenguajes de programación versátiles y muy implantados en la actualidad. Por otro lado, las distintas librerías tanto front-end como back-end han permitido dotar al proyecto de una mayor riqueza. Tampoco debemos olvidar los distintos patrones de diseño implantados durante el diseño de la arquitectura que dotan de robustez y de flexibilidad a la aplicación.

En este sentido, a vista futura, cobra gran importancia el uso de estos patrones de diseño. Van a dotar de un marco de desarrollo útil para la incorporación de nuevos proveedores, tanto de almacenamiento como eIDAS. Esta ha sido también una de las motivaciones para su implantación, de forma que incorporar uno no haga que se pierda la esencia de la arquitectura inicial con la que fue creada la aplicación, permitiendo que se cumplan los requisitos de firma necesarios. Por ejemplo, en un futuro, se podrían implementar otros proveedores como iCloud para la obtención de documentos de usuarios o WeTransfer, siendo este último muy interesante al ser volátil y no almacenar de forma permanente los documentos aportando si cabe más privacidad aún. También podría ser interesante trabajar con un proveedor eIDAS en un entorno real, para que la aplicación pueda producir firmas cualificadas.

A su vez, como vías futuras, también se plantea la posibilidad de desplegar la aplicación, o incluso los servicios, en plantillas Docker. Esto dotaría de más modularidad y facilidad para el despliegue de la aplicación en base a las necesidades, por ejemplo, en entornos escalables.

Para finalizar, uno de los hitos más importantes que he conseguido en la realización de este proyecto ha sido la implantación de un producto que emplea el reglamento eIDAS. Sin ninguna duda, crear un producto software que permita la aplicación de la firma electrónica para todo tipo de usuarios es muy interesante.

6. Glosario

UE: Unión Europea.

eIDAS: electronic IDentification, Authentication and trust Services, Reglamento Europeo que trata la identidad y firma electrónica.

OAuth2: Protocolo de autorización.

QSCD: Secure signature creation device, dispositivo seguro de creación de firma utilizado para crear firmas digitales.

USB: Universal Serial Bus. Estándar que define distintos aspectos para la comunicación de dispositivos electrónicos.

DNI: Documento Nacional de Identidad, elemento que acredita la nacionalidad española de un ciudadano.

PKI: Public Key Infraetucture, es una entidad que se encarga de almacenar y gestionar las credenciales de clave pública usadas, entre otras cosas, para firma digital.

IdP: Identity Provider, sistema que se encarga de gestionar la identidad y autenticación de usuarios.

PDF: Portable Document Format, tipo de documento muy extendido para información digital.

PAdES: es un marco de normas y formato definido para la firma digital de documentos PDF.

HSM: Hardware Security Module, equipos hardware destinados a tareas exclusivas de criptografía.

SaaS: Software as a Service, software como servicio.

REST: Protocolo de operaciones sobre HTTP que permiten realizar peticiones tipo GET, POST, PUT y DEL.

SOAP: Simple Object Access Protocol. Protocolo de intercambio de objetos XML para servicios web.

WSSE: Protocolo de comunicaciones que permite la aplicación de seguridad a servicios web.

DSV: Digital Signature Verification. Es un servicio de verificación de firmas electrónicas independiente del prestador, mecanismo y formato de firma que permite validar las firmas contenidas en documentos.

Bean: componente software destinado a almacenar una cantidad de datos para un programa.

CSRF: es un tipo de ataque a un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.

IDE: definido como entorno de desarrollo integrado, consiste en una aplicación informática que engloba todos los servicios necesarios para facilitar al desarrollador la implementación del software.

7. Bibliografía

[1] **Normativa eIDAS**. Parlamento Europeo. [En línea]. Disponible: https://www.boe.es/doue/2014/257/L00073-00114.pdf

[2] **Identity Provider Google**. Google Inc. [En línea]. Disponible: https://developers.google.com/identity/

[3] **Oauth2**. IETF OAuth WG. [En línea]. Disponible: https://oauth.net/

[4] **PAdES**. [En línea]. Disponible. https://www.viafirma.com/faq/es/pades-firma-electronica-avanzada-en-formato-pdf/

[5] **Dropbox Api Documentation**. [En línea]. Disponible. https://www.dropbox.com/developers/documentation/http/documentation

[6] **iText PDF Library Documentation**. [En línea]. Disponible. https://api.itextpdf.com/iText5/java/5.5.11/

[7] **Google Drive Api Documentation**. [En línea]. Disponible. https://developers.google.com/drive/api/v3/about-sdk

[8] **TrustedX eIDAS**. [En línea]. Disponible. https://www.safelayer.com/es/productos/trustedx-eidas-platform

[8] **PDF 32000-1 Documentation**. [En línea]. Disponible. https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/PDF32000_2008.pdf

[8] **Spring Framework Documentation**. [En línea]. Disponible. https://docs.spring.io/spring/docs/current/spring-framework-reference/