Servicio de firmado digital de facturas electrónicas con control de acceso basado en MFA

Fernando González Hernández

Máster Universitario en Ciberseguridad y Privacidad

Seguridad Empresarial

Tutor: Juan Carlos Fernández Jara

Profesor responsable: Víctor García Font

06/06/2023

Universitat Oberta de Catalunya

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Servicio de firmado digital de facturas electrónicas con control de acceso basado en MFA*
Nombre del autor:	*Fernando González Hernández*
Nombre del consultor/a:	*Juan Carlos Fernández Jara*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	*06/2023*
Titulación o programa:	*Máster Universitario en Ciberseguridad y Privacidad*
Área del Trabajo Final:	*Seguridad Empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave	*Firma digital, OpenID Connect, certificado digital*

Resumen del Trabajo

La identificación electrónica y la transformación digital se han vuelto cada vez más importantes en los últimos años debido al aumento del uso de servicios en línea y de transacciones electrónicas. En este contexto, la UE posee un marco regulador sobre la identificación electrónica llamada eIDAS (Electronic Identification, Authentication and Trust Services) en el que también se incluyen las normas y requisitos para la firma digital en la UE, que son necesarias para llevar a cabo la transformación digital y el uso de los servicios y transacciones en línea de forma segura. Uno de los servicios derivado de la transformación digital es el uso de las facturas electrónicas en sustitución de las facturas tradicionales en papel. La UE ya establece en la Directiva 2010/45/UE los requisitos específicos para las facturas electrónicas, incluyendo el uso de las firmas electrónicas avanzadas o cualificadas para garantizar la autenticidad e integridad de una factura.

En el presente Trabajo de Fin de Máster se pretende implementar una solución personalizada para firmar digitalmente facturas electrónicas considerando la importancia de la seguridad del proceso. Además, se pretende realizar un estudio sobre el control de accesos, las bases de la identificación electrónica, las firmas digitales y las facturas electrónicas.

Abstract

Electronic identification and digital transformation have become more and more important in recent years due to the increased use of online services and electronic transactions. In this context, the EU has a regulatory framework on

electronic identification called eIDAS (Electronic Identification, Authentication and Trust Services) which also includes the rules and requirements for digital signatures in the EU, which are necessary to carry out digital transformation and the safe use of online services and transactions. One of the services derived from the digital transformation is the use of electronic invoices in order to replace traditional paper invoices. The EU establishes in the Directive 2010/45/UE the specific requirements for electronic invoices, including the use of advanced or qualified electronic signatures to guarantee the authenticity and integrity of a invoice.

In this Master's Thesis, it is intended to implement a personalized solution to digitally sign electronic invoices considering the importance of the security of this process. In addition, it is intended to carry out a study on the fundamentals of access control, electronic identification, digital signatures and electronic invoices.

Índice

1. Introducción

1.1. Contexto y justificación del Trabajo

La identidad electrónica (eID) es un proceso que utiliza tecnología digital para autenticar y verificar la identidad de una persona o entidad [1]. La eID cada vez es más utilizada en la vida cotidiana para realizar accesos a servicios en línea, transacciones bancarias, firmar contratos electrónicos, etc. La eID puede ser emitida por diversas entidades como gobiernos, instituciones financieras y otras organizaciones que requieren que la verificación del usuario se haga de forma segura.

El uso de la eID reduce la necesidad de que el usuario proporcione de forma repetida su información personal, por lo que se reduce a su vez el riesgo de fraude y de suplantación de la identidad. Es decir, la eID conduce a una mejora de la seguridad. Además, la identificación electrónica también puede aumentar la privacidad al reducir la cantidad de datos personales que se comparten con los servicios en línea. Sin embargo, la eID también contiene riesgos relativos a la seguridad y la privacidad, especialmente si la información personal es almacenada o transmitida de manera insegura.

La UE establece un marco regulador sobre la eID, denominado eIDAS (Electronic Identification, Authentication and Trust Services) [2]. En este marco regulador se determina un conjunto de normas y estándares con el objetivo de que las transacciones electrónicas que se realicen entre ciudadanos, empresas y administraciones públicas de cualquier Estado de la UE se realicen de forma segura y fiable. El reglamento eIDAS [2] regula aspectos como la firma electrónica avanzada, firma electrónica cualificada, certificado cualificado de firma electrónica y servicios de confianza.

La firma electrónica (o firma digital) es un mecanismo criptográfico que permite a los usuarios firmar documentos electrónicos de forma segura y confiable [3]. Si la firma electrónica se encuentra bajo la regulación de eIDAS [2], entonces tendrá el mismo valor legal que la firma manuscrita en papel. Esto implica que a través de la firma electrónica se puede eliminar la necesidad de imprimir y enviar de forma física documentos en papel para que sean firmados. Debido a la relevancia legal de las firmas digitales, su creación y su almacenamiento seguro son asuntos a considerar muy sensibles. El proceso de creación de la firma electrónica estará estrechamente vinculado al proceso de identificación del dueño de la firma por parte de la entidad certificadora. El almacenamiento seguro deberá mantener la confidencialidad de la firma digital para que solamente el dueño de la firma pueda usarla, es decir, el almacenamiento de la firma deberá disponer de algún mecanismo de seguridad que permita identificar al dueño de una firma y así permitir su uso solo a éste y restringir cualquier otro tipo de acceso ilegítimo. De este modo, el proceso de identificación de una persona o entidad es de vital importancia para salvaguardar la creación y el uso legítimo de las firmas electrónicas.

El proceso de transformación digital impulsa el uso de la firma digital para diferentes servicios, entre ellos, el de la facturación electrónica. La factura

electrónica permite a las empresas realizar el envío y recepción de facturas por vía telemática en lugar de tener que imprimir, firmar y enviar físicamente documentación. Respecto a la firma digital, se trata de un mecanismo fundamental para dar garantía sobre la autenticidad e integridad de las facturas electrónicas. Además, la factura electrónica y la firma digital pueden integrarse con otros sistemas empresariales, lo que puede mejorar la eficiencia de los procesos comerciales, reducir costos operativos y reducir los errores en la facturación y los procesos de pago.

Actualmente existen diferentes soluciones para abordar el firmado de facturas electrónicas que principalmente podrían abordarse dentro de los siguientes grupos:

- Software de facturación electrónica. Este tipo de software permite que las empresas puedan crear, enviar y firmar facturas electrónicas.
- Plataformas de facturación electrónica. Este tipo de plataformas proveen a las empresas de soluciones integrales para crear, enviar, recibir y gestionar facturas electrónicas.

El uso de un software comercial o de una plataforma de servicios puede ser una opción efectiva para muchas empresas o usuarios. Sin embargo, cabe destacar que el uso de éstos tiene algunos riesgos, como son los siguientes:

- Costes: el coste del uso de software comercial o de una plataforma de servicios puede llevar a un elevado gasto por parte de la empresa contratante. Esto puede ser un problema, principalmente para las pequeñas empresas o empresas en crecimiento.
- Dependencia con terceros: si una empresa usa un software comercial o una plataforma de servicios, implica que dependerá de un tercero para el proceso de firma de las facturas electrónicas. Si el tercero tiene problemas en el servicio, estos problemas también impactarán a la empresa contratante, lo que podría repercutir en retraso de pagos o de la gestión de facturas.
- Limitación de personalización: los softwares comerciales y las plataformas de servicios suelen permitir cierto grado de personalización para adaptarse a las necesidades de sus clientes. Sin embargo, puede haber limitaciones que repercutan en la integración con algunas empresas que posean requisitos muy específicos o incluso únicos.
- Seguridad y privacidad: si una empresa usa un software comercial o una plataforma de servicios, implica que deberá confiar en la seguridad y la privacidad de un tercero. Si el tercero sufre algún tipo de vulnerabilidad, el riesgo asociado también podría impactar a la empresa contratante. En esa situación, incluso podría exponerse información confidencial de la empresa o de sus clientes.

Algunas empresas no pueden asumir los riesgos de contratar un software comercial o una plataforma de servicios para realizar el firmado de facturas electrónicas. Por ese motivo, existen algunas alternativas disponibles, como pueden ser las siguientes:

- Crear una solución personalizada: permitiría tener un mayor control y personalización del proceso de firma.

- Firmar las facturas manualmente: imprimiendo las facturas en papel, firmándolas a mano y posteriormente escaneándolas y enviándolas por algún medio electrónico.

El presente Trabajo de Fin de Máster (en adelante TFM) pretende implementar una solución personalizada para firmar digitalmente facturas electrónicas considerando la importancia de la seguridad del proceso. Además, se pretende realizar un estudio sobre los fundamentos del control de accesos, las bases de la identificación electrónica, las firmas digitales y las facturas electrónicas, así como del marco normativo correspondiente.

1.2. Objetivos del Trabajo

El TFM tiene los siguientes objetivos:

- Estudiar los conceptos fundamentales relacionados con el control de accesos, la identificación electrónica y el firmado de facturas electrónicas.
- Analizar las normativas y regulaciones aplicables a la firma electrónica en las facturas electrónicas, como el Reglamento eIDAS, la Ley de Firma Electrónica, la Directiva de Facturación Electrónica, etc.
- Describir el procedimiento de firmado de facturas electrónicas
- Revisión de soluciones comerciales y herramientas de software disponibles para el firmado de facturas electrónicas.
- Diseñar, desarrollar e implementar un servicio de firmado digital de facturas electrónicas con control de acceso basado en autenticación multi-factor.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

El TFM trata sobre aspectos técnicos de digitalización y seguridad. Esto implica que en lo relativo a la sostenibilidad, como toda digitalización, permite la disminución de consumo de papel a costa del aumento del consumo de material informático. Asimismo, el trabajo permite acercar a todas las personas a la posibilidad de realizar facturación electrónica de forma telemática, lo que posibilita la reducción de la barrera de las distancias en algunas zonas. Además, dado que la naturaleza del trabajo es técnica, éste no atiende a razones de género o raza.

La motivación de la realización de este trabajo es la adquisición y consolidación de conocimientos relativos a la seguridad informática en el ámbito de la seguridad empresarial. Se pretende demostrar capacidad en el aprendizaje de nuevos conocimientos, así como utilizarlos para implementar una solución personalizada para firmar digitalmente facturas electrónicas.

1.4. Enfoque y método seguido

El método seguido para la realización del TFM consiste en la realización de los siguientes grupos de tareas de forma secuencial:

1. Revisión de conceptos relacionados con la firma digital, la identidad electrónica, las facturas electrónicas, procedimientos de firmado.
2. Revisión de normativa y regulación sobre la gestión de identidades y el firmado de facturas electrónicas.
3. Revisión de soluciones ya existentes para la realización de firmado de facturas electrónicas y para la gestión de identidades digitales.
4. Diseño, desarrollo e implementación de un servicio de firmado digital de facturas electrónicas con control de acceso basado en autenticación multi-factor:
    a) Determinación del alcance de la solución a implementar
    b) Análisis funcional: identificación de los componentes, casos de uso, selección del stack tecnológico y diseño de la arquitectura final.
    c) Implementación de la solución: sistema de control de identidad, generación de claves de firma, servicio de firmado de facturas, etc.
    d) Pruebas del sistema desarrollado
5. Determinación de conclusiones del trabajo

1.5. Planificación del Trabajo

Cod.	Nombre	Tipo	Fecha de inicio	Fecha de fin
1	**Planificación**	**Grupo**	**2023-03-06**	**2023-06-13**
1.1	Introducción del TFM	Tarea	2023-03-06	2023-06-13
1.2	Planificación del TFM	Tarea	2023-03-06	2023-06-13
1.3	*PEC 1: Entrega del plan de trabajo*	*Hito*	*2023-06-13*	*2023-06-13*
2	**Marco teórico conceptual y normativo**	**Grupo**	**2023-03-14**	**2023-04-21**
2.1	Revisión bibliográfica	Subgrupo	2023-03-14	2023-04-14
2.1.1	Estudio de conceptos fundamentales	Tarea	2023-03-14	2023-03-31
2.1.2	Revisión de normativas y regulación	Tarea	2023-03-14	2023-03-31
2.1.3	Identificación de los conceptos y redacción del marco teórico	Tarea	2023-04-03	2023-04-14
2.2	Revisión de soluciones ya existentes	Subgrupo	2023-04-17	2023-04-21
2.2.1	Revisión de soluciones ya existentes para el firmado de facturas electrónicas	Tarea	2023-04-17	2023-04-21

2.2.2	Revisión de soluciones ya existentes para la gestión de identidades	Tarea	2023-04-17	2023-04-21
2.3	*PEC 2: Entrega del marco teórico conceptual y normativo*	*Hito*	*2023-04-21*	*2023-04-21*
3	**Implementación de caso práctico**	**Grupo**	**2023-04-24**	**2023-05-26**
3.1	Determinación del alcance	Tarea	2023-04-24	2023-04-25
3.2	Análisis funcional	Tarea	2023-04-25	2023-04-28
3.3	Implementación de la solución	Tarea	2023-05-01	2023-05-19
3.4	Pruebas	Tarea	2023-05-22	2023-05-26
3.5	*PEC 3: Entrega de la implementación de la solución*	*Hito*	*2023-05-26*	*2023-05-26*
4	**Presentación y defensa**	**Grupo**	**2023-05-29**	**2023-06-16**
4.1	Conclusiones y trabajos futuros	Tarea	2023-05-29	2023-06-02
4.2	Elaboración de la memoria final	Tarea	2023-06-02	2023-06-09
4.3	Elaboración de presentación de la memoria	Tarea	2023-06-12	2023-06-16
4.4	*PEC 4: presentación de la memoria y defensa*	*Hito*	*2023-06-16*	*2023-06-16*

Tabla 1.1. Planificación del TFM

1.6. Breve sumario de productos obtenidos

El TFM se divide en varios entregables que en su conjunto formarán parte del resultado final del trabajo:

- PEC 1: Entrega del plan de trabajo y de la introducción del TFM.
- PEC 2: Entrega del marco teórico conceptual y normativo.
- PEC 3: Implementación de caso práctico.
- PEC 4: Entrega de la memoria final del TFM y realización de una defensa.

1.7. Breve descripción de los otros capítulos de la memoria

La memoria se compone de los siguientes capítulos:

1. Introducción. Se trata de contextualizar el TFM, sus objetivos, impacto en sostenibilidad, metodología seguida y su planificación.
2. Marco teórico conceptual y normativo. Se analizan los conceptos fundamentales, la normativa y las soluciones ya existentes sobre el firmado de facturas electrónicas y la identificación de usuarios.
3. Implementación de caso práctico. Se lleva a cabo un caso práctico de la implementación de un servicio de firmado de factura electrónica con control de acceso basado en autenticación multi-factor.
4. Conclusiones y trabajos futuros. Se analizan las conclusiones del trabajo y se exponen los posibles trabajos futuros derivados de éste.

2. Marco teórico conceptual y normativo

2.1. Introducción

El presente capítulo analiza los conceptos relacionados con la firma digital, la identidad electrónica, las facturas electrónicas y el procedimiento de firmado. Además, se hace una revisión sobre su normativa y regulación correspondiente. Finalmente, se revisan algunas soluciones ya existentes para la realización de firmado de facturas electrónicas y para la gestión de identidades electrónicas.

2.2. Control de identidades

2.2.1. Introducción

El control de identidad es la práctica utilizada para proteger la seguridad y la privacidad de las personas, ya que limita el acceso a información y a recursos sensibles para que solo aquellos que tengan derecho a acceder a ellos puedan hacerlo. Para realizar esta distinción entre quienes sí pueden tener derecho al acceso y quienes no pueden tenerlo es necesario poder diferenciar a los individuos entre sí. Esta diferenciación se realiza a través de unos rasgos característicos propios del individuo que hacen que sea único frente a los demás, es decir, un individuo se diferencia de otro a través de su identidad.

Las formas de control de identidad más habituales se basan en uno o varios de los siguientes factores [4]:

- Algo que el usuario sabe. Si un usuario y solo ese usuario conoce un dato concreto, puede utilizarse ese dato para identificarle. La forma de demostrar que sabe algo de forma exclusiva generalmente es a través del uso de contraseñas. El usuario es el único conocedor de su contraseña, así que cuando introduce su contraseña en un control de acceso podría demostrar que él es quien dice ser.
- Algo que el usuario tiene. Si el usuario y solo ese usuario tiene algo en concreto, puede utilizarse ese algo para identificarse. La forma de demostrar que se tiene algo de forma exclusiva suele ser mediante una tarjeta identificativa o incluso un smartphone. Si el usuario puede demostrar que tiene en su posesión su smartphone, entonces podría demostrar que él es quien dice ser.
- Algo que el usuario es. Si el usuario y solo ese usuario tiene alguna característica física intransferible y única, puede utilizarse para identificarse. La formar de demostrarse suele realizarse a través de biometría por escáner de huellas digitales, escáner de retina, escáner facial, etc.
- Algo que solo el usuario es capaz de hacer. Si el usuario y solo ese usuario tiene alguna capacidad única para realizar algo, puede utilizarse para identificarse. La forma de demostrar eso suele ser mediante la

repetición de un patrón como la firma, la forma de escribir en sí o incluso la manera de andar.

El uso de alguno de los factores descritos anteriormente permitiría identificar a un usuario. Sin embargo, cada uno de los factores podría llegar a ser vulnerable por separado. Es decir, en el caso de la contraseña, un usuario podría haberla expuesto y ese dato podría estar en posesión de un tercero, por lo que ese tercero podría hacerse pasar por el usuario titular de la contraseña. Una forma de minimizar el riesgo de suplantación consiste en combinar estos factores. Es decir, que, durante el proceso de autenticación, un usuario tenga que demostrar que es quien dice ser mediante el uso de una o varias contraseñas, validarse en un escáner facial e incluso pasar una tarjeta de identificación por el escáner. De este modo, el usuario podría garantizar que es quien dice ser y la suplantación sería más difícil de realizar. A la combinación de varios factores de autenticación se le denomina “autenticación multi-factor” (o, en inglés, Multi Factor Authentication o MFA) [5] [6] [7].

Debido a la transformación digital, ha surgido el concepto de “identidad electrónica”, también conocida como eID, por sus siglas en inglés. El eID permite a una persona o entidad demostrar su identidad utilizando un método de autenticación digital, que se basa en el uso de tecnología criptográfica avanzada [1]. Es decir, se trata de una evolución sobre la identificación física tradicional. La eID incluye información personal sobre el titular y también otra información relevante, que es almacenada en un certificado digital. De este modo, el titular de una eID podrá demostrar su identidad a través del certificado.

La eID es altamente relevante por los siguientes motivos:

- Seguridad: la eID es un método seguro y confiable que permite la reducción del riesgo de fraude y delitos cibernéticos.
- Comodidad: la eID permite que los usuarios puedan identificarse y firmar documentos digitales desde cualquier lugar y en cualquier momento.
- Privacidad: la eID permite mayor control para que su titular solo tenga que compartir su información personal mínima necesaria para realizar la operación que necesite.
- Eficiencia: la eID permite la reducción de la carga administrativa ya que los titulares pueden realizar en línea ciertos trámites.

El ciclo de vida de una eID consiste en el conjunto de procesos que permiten la creación, el uso y la eliminación de las identidades electrónicas. Las fases del ciclo de vida de una identidad electrónica son las siguientes [4]:

1. Registro. En esta fase el interesado genera una solicitud a una entidad de emisión de identidades electrónicas aportando una serie de documentación. A su vez, la entidad de emisión de identidades electrónicas realiza una serie de verificaciones para comprobar la identidad del solicitante y generar el certificado digital de identidad electrónica correspondiente.
2. Autenticación. La autenticación es el proceso por el cual se verifica la identidad de un usuario cuando éste ya ha sido previamente registrado en el sistema.

3. Autorización. La autorización es la determinación de los permisos y privilegios concedidos al usuario en función de su identidad, después de haber sido autenticado correctamente.
4. Mantenimiento. En esta fase se realizan tareas relativas a la actualización de información personal, renovación de credenciales y modificación de permisos y privilegios.
5. Desactivación. Cuando la identidad electrónica ya no es necesaria se procede a su desactivación y revocación de privilegios y certificados.

2.2.2. Regulación sobre el control de identidades

Debido a la importancia de la identidad electrónica, los diferentes países y regiones han comenzado a crear marcos legales regulatorios sobre el control de identidades electrónicas. El objetivo de estos marcos regulatorios es garantizar la seguridad, la privacidad y la confianza de los procesos de verificación en línea. Los marcos regulatorios más importantes a nivel mundial son los siguientes:

- El Reglamento (UE) 910/2014 [2], también conocido como eIDAS (por su acrónimo en inglés “electronic IDentification, Authentication and trust Services”), es un marco regulatorio de la Unión Europea que tiene como objetivo que cualquier identidad electrónica emitida por un estado miembro de la UE pueda ser reconocida por otro estado miembro de la UE.
- La Ley de Identificación Nacional de 2005 [8], también conocida como la Ley REAL ID, es una ley federal de los Estados Unidos en donde se establecen los requisitos mínimos para la emisión de identificadores estatales y federales que sean aceptables para acceder a instalaciones gubernamentales y para abordar vuelos comerciales.
- La Ley de Protección de la Información Personal y Documentos Electrónicos (PIPEDA) [9] es una ley federal de Canadá en donde se establecen reglas relativas a la recopilación, uso y protección necesaria de información personal en el ámbito de las actividades comerciales.
- La Ley de Identificación y Verificación de Identidad de 2013 [10] es una ley de Australia que establece un marco legal para la identificación electrónica y la verificación de identidad en el país.
- La Ley de Firma Electrónica [11] y la Ley de Certificación Electrónica [12] de Japón, establecen marcos legales para el uso de firmas electrónicas y certificados digitales, así como los requisitos que deben cumplir los proveedores de servicios de certificación electrónica.

El reglamento eIDAS [2] se creó desde la UE para establecer un marco común para la identificación electrónica y los servicios de confianza en toda la UE. Antes de la entrada en vigor de eIDAS [2], cada país miembro tenía sus propias leyes y normativas, por lo que la interoperabilidad entre ellos resultaba difícil. El reglamento consta de tres partes principales:

- Identificación electrónica. El reglamento establece la obligación de los Estados miembros de aceptar y reconocer los identificadores electrónicos emitidos por otros Estados miembros. Además, se establecen los requisitos para la seguridad y fiabilidad de las identificaciones electrónicas.

- Servicios de confianza electrónica. El reglamento establece un marco para los servicios de confianza en la UE, como la firma electrónica, el sello electrónico, el sellado de tiempo y la entrega electrónica certificada. Se establecen normas técnicas y de seguridad para garantizar la integridad, autenticidad y confidencialidad de los datos y de las transacciones electrónicas.
- Disposiciones generales y finales. El reglamento establece una serie de disposiciones relativas a la cooperación entre los Estados miembros, la creación de un comité de regulación de servicios de confianza y la revisión del reglamento.

En España, la regulación de eIDAS [2] está implementada a través de la Ley 6/2020, de 11 de noviembre, reguladora de determinados aspectos de los servicios electrónicos de confianza [13]. En esta Ley se establecen los requisitos y los procedimientos que son necesarios para dar garantía de la validez legal de las firmas electrónicas, sellos electrónicos, servicios de entrega electrónica certificada y otros servicios de confianza en línea. El organismo público que es responsable de controlar la implantación de eIDAS [2] en España y de garantizar la interoperabilidad de los servicios de confianza con respecto al resto de la UE es el Ministerio de Asuntos Económicos y Transformación Digital.

Los sistemas de identificación electrónica que cumplen con los requisitos de eIDAS [2] en España son los siguientes:

- Cl@ve [14]. Es un sistema de identificación electrónica que utilizan todos los servicios públicos españoles en línea.
- DNIe [15]. Es una tarjeta física con un chip emitida por el gobierno de España para la identificación electrónica de los ciudadanos españoles.
- Certificados electrónicos que hayan sido emitidos por proveedores de servicios de confianza acreditados [2]. Estos certificados tienen el mismo valor legal que una firma manuscrita.

2.2.3. Tecnologías para la gestión de identidades

La gestión de identidades es el proceso que permite gestionar la identidad de los usuarios de una organización o conjunto de organizaciones. Un sistema de gestión de identidades deberá crear, mantener y eliminar cuentas de usuario, así como también determinar cuáles son los accesos permitidos de cada usuario. Mediante una correcta implementación de un sistema de gestión de identidades, se podrá prevenir el acceso no autorizado a los diferentes recursos de una organización.

En el caso de los sistemas distribuidos, los principales mecanismos para la autenticación y gestión de identidades son la federación de identidad y Single Sign On (SSO). El concepto de funcionamiento del proceso de autenticación en la federación de identidades y en Single Sign On es similar, ya que en ambos casos el usuario o entidad realiza la autenticación una sola vez para acceder a los diferentes recursos [16].

Existen diferentes flujos de trabajo en la federación de identidad y Single Sign On, pero uno típico es el siguiente:

1. El usuario se autentica en el proveedor de identidad correspondiente (IdP).
2. El IdP genera un token de seguridad que contiene información relativa a la autenticación y la autorización.
3. El usuario intenta acceder a un proveedor de servicios (SP) enviándole una petición con el token de seguridad generado por el IdP.
4. El SP comprueba con el IdP la validez del token. Si la validez es correcta, el SP proporcionará al usuario el acceso correspondiente a los servicios solicitados.

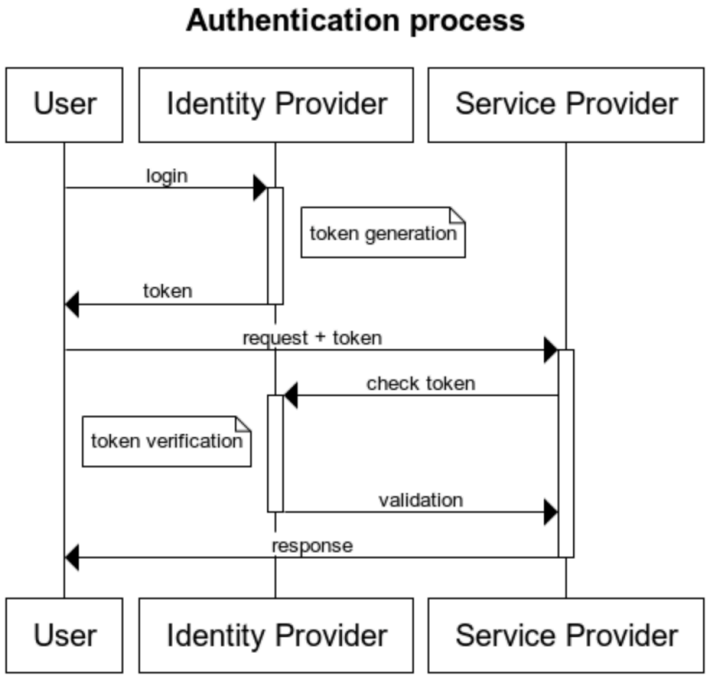

Figura 2.1: Proceso de autenticación en la federación de identidades y en Single Sign On.

La principal diferencia entre la federación de identidad y SSO consiste en el alcance. La federación de identidades permite que los usuarios de diferentes organizaciones puedan acceder a recursos compartidos utilizando sus credenciales de identificación locales sin necesidad de crear cuentas de usuario adicionales. Por otro lado, el Single Sign On permite un alcance local en el que los usuarios de una misma organización accedan a recursos de ésta a través de un solo proceso de autenticación.

Algunos de los protocolos de implementación de federación de identidad y SSO más comunes son: SAML, OAuth 2.0 y OpenID Connect (OIDC).

2.2.3.1. SAML

SAML (Security Assertion Markup Language) es un protocolo convertido en un estándar de código abierto que permite la implementación de federación de identidades y SSO. SAML se basa en el envío de documentos XML para el intercambio de datos de autenticación y autorización. En SAML, un usuario

inicia una sesión con un proveedor de identidad y posteriormente puede acceder a diferentes servidores de recursos a través de ese único inicio de sesión.

El protocolo SAML está especificado en la RFC 7522 “Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants” de IETF [17].

2.2.3.2. OAuth 2.0

OAuth2 es un protocolo abierto estándar para la implementación de SSO que intercambia datos de autorizaciones entre aplicaciones sin tener que exponer la contraseña del usuario [18]. Con OAuth 2.0 se permite que un usuario pueda compartir sus datos de un servicio con otro servicio. Por ejemplo, si un usuario abre una App desde su smartphone y hace login mediante OAuth 2.0 con su cuenta de una red social, el usuario puede permitir compartir sus datos personales y su lista de contactos de la red social con la App que va a utilizar. El objetivo de este protocolo es permitir la compartición de recursos entre aplicaciones. El protocolo OAuth 2.0 sirve para funciones de autorización, no para autenticación. El protocolo OAuth 2.0 está especificado en la RFC 6749: The OAuth 2.0 Authorization Framework [19].

Las cuatro entidades que intervienen en OAuth 2.0 son las siguientes:

- Authorization server: el responsable de gestionar las peticiones de autorización.
- Protected Resource: la API que contiene un recurso protegido de un usuario.
- ClientApp: la aplicación que desea acceder al recurso protegido en nombre del usuario. La aplicación debe estar registrada en el servidor de autorización.
- Resource Owner: el usuario que es dueño del recurso. El usuario no es dueño de la API sino del dato al que se desea acceder.

Existen varios flujos de implementación de OAuth 2.0 [20] y la elección de un flujo u otro dependerá del tipo de aplicación cliente que se tenga. A continuación, se enumeran algunos de los flujos principales de OAuth 2.0:

- Authorization Code Flow. El flujo de autorización de código (Authorization Code Flow) es un flujo de autorización de OAuth 2.0 que se utiliza para obtener un token de acceso en nombre de un usuario. Este flujo es adecuado para las aplicaciones que requieren acceso a recursos protegidos en nombre del usuario, como los datos personales de un usuario o los servicios de terceros. Se trata del flujo más utilizado, más completo y que se considera más seguro. Se utiliza para las aplicaciones web con servidor.
- Client Credentials Flow. El flujo de credenciales cliente es un flujo que permite que la aplicación acceda a su propia cuenta de servicio dentro del servidor de autorización. Es decir, la aplicación solicita al servidor de autorización un token para poder usar ese token para enviar peticiones al servidor de autorización y poder acceder y modificar sus datos de su

propia cuenta. En este flujo el usuario (Resource Owner) y la aplicación (ClientApp) serían la misma entidad. Asimismo, el servidor de autorización (Authorization server) y servidor de recursos (Protected Resource) también serían la misma entidad.

- Implicit Flow. El Implicit Flow es un flujo de autorización utilizado en OAuth 2.0 para obtener tokens de acceso en una aplicación web sin servidor. Se trata de un flujo más simple de implementar que "Authorization Code Flow" pero también más inseguro que éste ya que el token de acceso se envía directamente al cliente, lo que lo hace más susceptible a posibles ataques de tipo CSRF. Actualmente, este flujo se considera en desuso.
- ACG with PKCE. El flujo de autorización de código con prueba de clave para intercambio de código (Authorization Code with Proof Key for Code Exchange, ACG with PKCE) es una variante del flujo de autorización de código que proporciona una mayor seguridad al proteger contra ataques de intercepción de código. Este flujo se utiliza para aplicaciones nativas que no pueden mantener un secreto de cliente de forma segura, como las aplicaciones móviles. "ACG with PKCE" es el flujo que deja en desuso al flujo "Implicit Flow".

2.2.3.3. OpenID Connect

OpenID Connect (OIDC) es un protocolo abierto estándar que utiliza API REST y token Json para la autenticación de usuarios [21]. El usuario puede identificarse directamente en un proveedor de servicios a través de una URL, luego utilizar el token de respuesta para enviar peticiones a proveedores de servicios y los proveedores de servicios comprobarían la identidad del usuario mediante la validación del token con el proveedor de identidad.

El protocolo OIDC realmente es una extensión de OAuth 2.0 [22] [23]. Ambos protocolos, OAuth 2.0 y OIDC, se combinan de forma complementaria en un mismo flujo de trabajo en el que OIDC aporta autenticación y OAuth 2.0 aporta autorización.

En OIDC, a diferencia de OAuth2.0, se denomina "Relying Party" (RP) a la aplicación cliente y se denomina "Identity Provider" (IDP) al servidor de autenticación. Además, en OIDC existe la posibilidad de disponer de un servicio "Discovery EndPoint" que sirve para que el RP pueda ver el catálogo de opciones y de funcionalidades del IDP.

El formato del servicio Discovery es como el siguiente:

```
https://[domain]/.well-known/openid-configuration
```

Compañía	URL del servicio Discovery
Google	https://accounts.google.com/.well-known/openid-configuration
Microsoft	https://login.microsoftonline.com/common/v2.0/.well-known/openid-configuration
Facebook	https://www.facebook.com/.well-known/openid-configuration

Tabla 2.1. Ejemplos de servicio Discovery en OIDC

En OAuth 2.0, la petición original que envía el Resource Owner al Authorization server contiene el campo “response_type”, cuyo valor posible es “code” o “token”, según el tipo de flujo de implementación. En OIDC se añade el posible valor “id_token”, de forma que si se incluye en el campo “request_type” de la petición a Identity Provider se conseguirá añadir el proceso de autenticación. Por tanto, para añadir el proceso de autenticación al flujo “Authorization Code Flow” descrito en la sección de O Auth2.0, será necesario que el campo “request_type” tome el valor “code id_token”.

En el caso de OIDC, el campo “scope” de la petición que envía al Identity Provider se ve modificado respecto a O Auth2.0 en que en OIDC se añaden los siguientes valores por defecto: openid, profile, email, address y phone. Para incluir el proceso de autenticación es necesario que en el campo “scope” se incluya el valor “openid” además de otros valores que se deseen. El Identity Provider devolverá en su respuesta un campo “id_token” que corresponderá con una cadena en formato JWT [24] en donde, si se descifra, se obtendrá la información sobre la identificación del usuario que ha hecho el inicio de sesión.

El formato JWT se compone de tres partes separadas entre sí por un carácter “.”:

- Header: es la primera parte del JWT y contiene el tipo de json (“JWT” para este caso) y el algoritmo utilizado. El header es un json codificado en base64.
- Payload: es la segunda parte del JWT y contiene los datos de usuario, los privilegios asociados y otra información adicional que se desee añadir. El payload es un json codificado en base64.
- Signature: es la tercera parte del JWT y contiene una firma que permite verificar si el token es válido. La firma se realiza sobre los campos header y payload.

A continuación, se muestra un ejemplo de una cadena JWT:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiw
iaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36Pok6yJV_adQssw5c
```

- El header “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9” se decodifica en base 64 y se obtiene: {“alg”:”HS256”,”typ”:”JWT”}.
- El payload “eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ” se decodifica en base 64 y se obtiene: {“sub”:”1234567890”,”name”:”John Doe”,”iat”:1516239022}
- Finalmente, la signatura es SflKxwRJSMeKKF2QT4fwpMeJf36Pok6yJV_adQssw5c y no procede a decodificación por tratarse de una firma.

Además, hay herramientas online que decodifican un JWT, por ejemplo: https://jwt.io/#debugger-io [25]

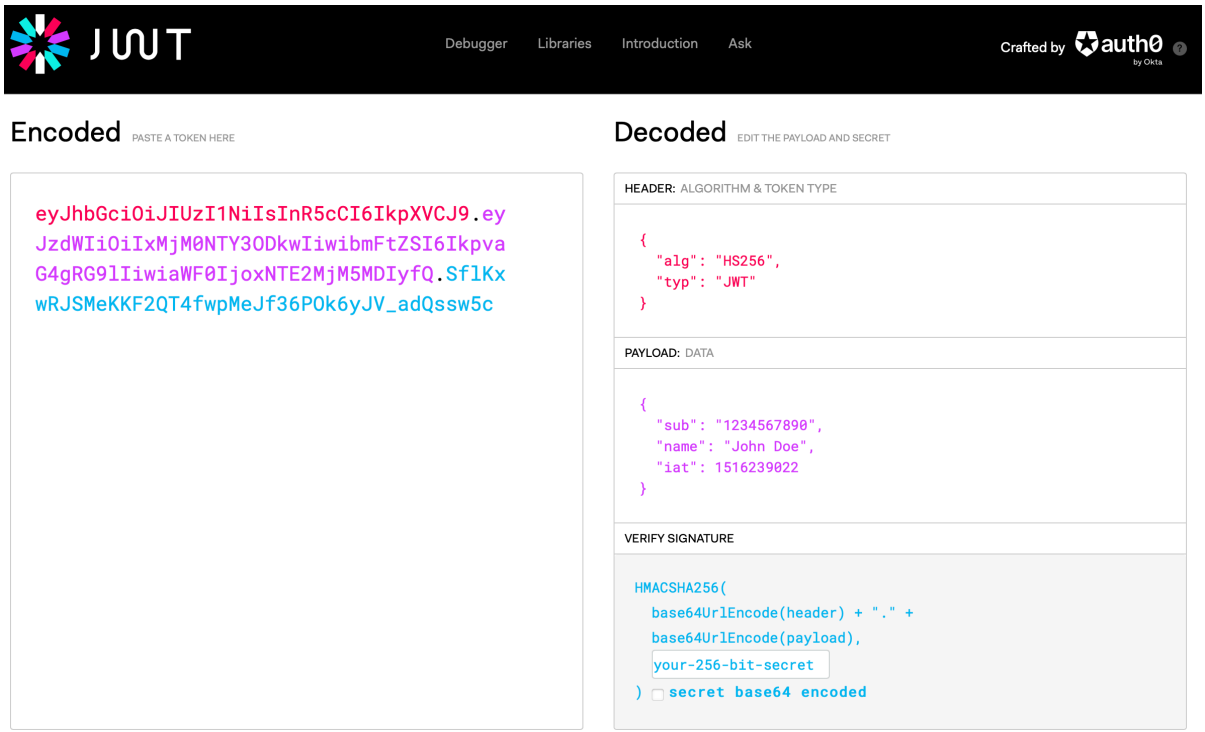

Figura 2.2: Decodificación de JWT en https://jwt.io [25]

2.2.4. Reforzado de la autenticación de usuarios

En la gestión de identidades es importante garantizar que solo las personas autorizadas tengan acceso a los recursos de la organización y que se cumplan las políticas de seguridad de la organización. Por tanto, es muy importante el proceso de autenticación. Para reforzar este proceso se puede implementar una técnica de autenticación multi-factor (MFA) [5] [6] [7]. El MFA es una técnica de autenticación que requiere que los usuarios proporcionen más de un método de autenticación para acceder a un recurso o sistema. Esto significa que los usuarios deben proporcionar más que solo su nombre de usuario y contraseña. Por ejemplo, también podrían proporcionar una clave de seguridad física, una huella dactilar o una contraseña de un solo uso generada por una aplicación móvil. Al combinar la gestión de identidades con el MFA, las organizaciones pueden mejorar significativamente su seguridad en la gestión de accesos. Al exigir múltiples factores de autenticación, se reduce la posibilidad de que los atacantes puedan acceder a las cuentas de usuario utilizando solo credenciales robadas o adivinadas.

Uno de los métodos más comunes de MFA es OTP (One Time Password), donde un usuario recibe un código de acceso de un solo uso en un dispositivo separado, como puede ser un teléfono móvil, para autenticar su identidad [26] [27]. OTP tiene dos protocolos de implementación [4]:

- HOTP: protocolo de OTP basado en eventos. El dispositivo de usuario y el servidor conocen una clave simétrica y un contador (que va cambiando con cada proceso de autenticación en el servidor). El dispositivo de usuario aplicará un algoritmo de tipo hash sobre el contador y lo firmará con la clave, posteriormente enviará esa contraseña resultante al servidor. Dado que el servidor conoce el

algoritmo, el valor del contador y de la clave, validará si la contraseña recibida es o no correcta.

- TOTP: protocolo OTP basado en el tiempo. El funcionamiento de TOTP es el mismo que el del caso HOTP excepto que en TOTP no se usa como factor dinámico un contador, sino que se usa el tiempo. Esto garantiza que la contraseña se cambia cada cierto tiempo.

El proceso de autenticación multi-factor con OTP generalmente involucra lo siguiente:

1. El usuario realiza un inicio de sesión en un sitio web o una aplicación que requiere autenticación.
2. El sitio web o la aplicación envía una solicitud de OTP al servidor de autenticación.
3. El servidor de autenticación genera una OTP única y la envía al usuario a través de un mensaje de texto, una aplicación móvil o un correo electrónico.
4. El usuario ingresa la OTP en el sitio web o la aplicación.
5. El sitio web o la aplicación envía la OTP ingresada por el usuario al servidor de autenticación para su verificación.
6. Si la OTP es válida, el usuario se autentica y se le permite acceder al sitio web o la aplicación.

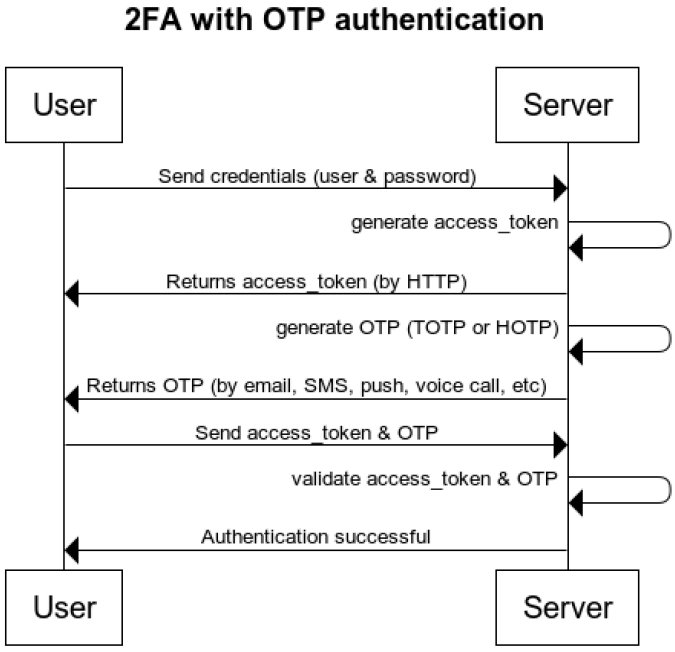

Figura 2.3: Proceso de autenticación multi-factor con OTP

2.3. Firma digital

2.3.1. Introducción

Cuando una persona o entidad ya posee una identidad electrónica que le permite distinguirse de forma segura del resto de usuarios dentro del mundo digital, entonces puede usar su identidad para realizar diferentes acciones. Una

de las acciones es la de firmar documentos de manera electrónica mediante una firma digital.

La firma es un mecanismo por el cual una persona de su propia mano escribe su nombre y apellidos de forma escrita para dar autenticidad o mostrar aprobación del contenido de un documento [28]. En el contexto de la documentación electrónica, la firma correspondiente a estos documentos es la firma electrónica. En el artículo 3 ("Firma electrónica, y documentos firmados electrónicamente") de la "Ley 59/2003, de 19 de diciembre, de firma electrónica" [29], se define la firma electrónica como "el conjunto de datos en forma electrónica, consignados junto a otros o asociados con ellos, que pueden ser utilizados como medio de identificación del firmante". Además, en el mismo artículo también se aporta la siguiente definición sobre la firma electrónica avanzada: "La firma electrónica avanzada es la firma electrónica que permite identificar al firmante y detectar cualquier cambio ulterior de los datos firmados, que está vinculada al firmante de manera única y a los datos a que se refiere y que ha sido creada por medios que el firmante puede utilizar, con un alto nivel de confianza, bajo su exclusivo control".

Cuando un documento es firmado digitalmente, se procede a realizar unas operaciones criptográficas que añaden una serie de datos cifrados al documento electrónico original. Esta serie de datos corresponderían a la firma digital. Por las características de la generación de la firma digital, se permite otorgar las propiedades de autenticación y de integridad al documento firmado. Es decir, permite garantizar que el documento no haya sido modificado desde el momento en que haya sido firmado y también permite identificar de manera inequívoca a la persona firmante.

2.3.2. Certificado electrónico

Los certificados electrónicos son unos ficheros digitales que resultan esenciales para realizar la validación de las firmas digitales y garantizar así la seguridad de los documentos electrónicos. Un certificado electrónico o certificado digital es un fichero digital que vincula una clave pública con la identidad de su propietario.

Para poder confiar en que la vinculación sea inequívoca, será necesario que una tercera parte en la que todos confíen haya dado fe de que la vinculación entre la identidad del propietario del certificado y los datos contenidos en el certificado electrónico sea correcta. Esta tercera parte se llama "prestador de servicios de confianza" o "Autoridad Certificadora" o "Trusted Third Party" y es la base de la definición de las Infraestructuras de Clave Pública (o, en inglés, Public Key Infrastructure o PKI). Una PKI es un conjunto de servicios que gestionan el ciclo de vida de los certificados digitales [33].

Los componentes de una PKI suelen ser los siguientes:

- Autoridad de Certificación (o, en inglés, Certificate Authority o CA): es la entidad encargada de emitir y revocar certificados digitales.

- Autoridad de Registro (o, en inglés, Registration Authority o RA): es la entidad responsable de verificar la vinculación entre los certificados y la identidad de sus titulares.
- Lista de revocación de certificados (o, en inglés, Certificate Revocation List o CRL): es el repositorio en el que están incluidos todos los certificados que hayan dejado de ser válidos antes de su fecha de expiración.
- Autoridad de Validación (o, en inglés, Validation Authority o VA): es la entidad que gestiona el listado de certificados emitidos, caducados y revocados. Un usuario puede consultar a una VA sobre si un certificado es o no válido.
- Autoridad de sellado de tiempo (o, en inglés, TimeStamp Authority o TSA): es la entidad que proporciona la certeza de la preexistencia de un documento electrónico en un momento dado.
- Los usuarios y entidades finales: son quienes explotan los servicios de la PKI y además quienes poseen un par de claves (pública y privada) y un certificado digital que está asociado a su clave pública.

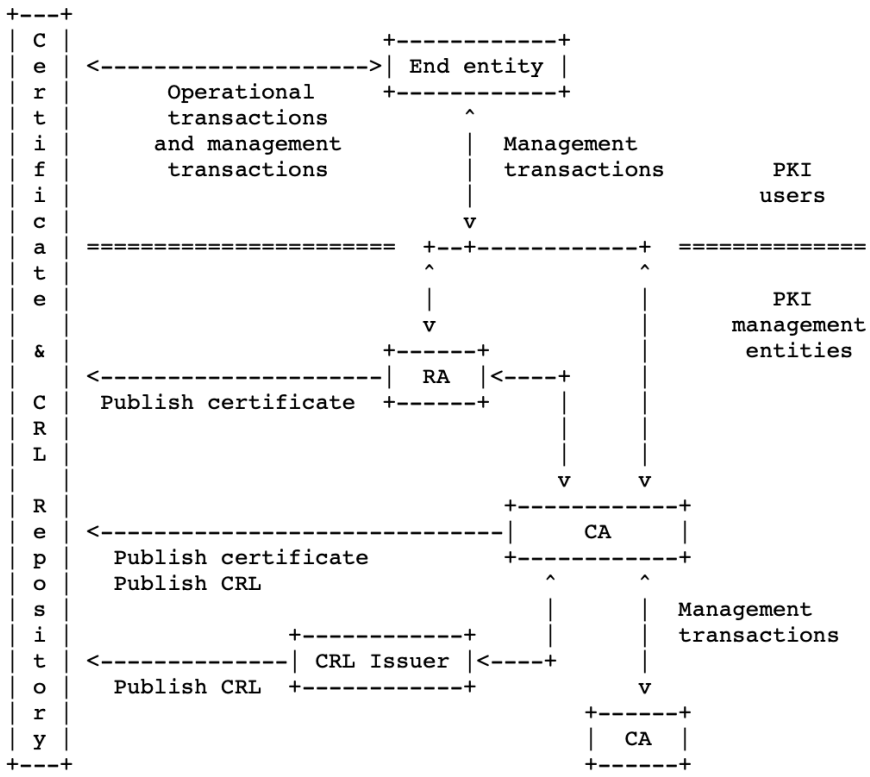

Figura 2.4: Entidades que componen una PKI (imagen del estándar RFC 5280) [33]

En una PKI realmente no hay una única CA, sino que existe una red jerárquica de CA donde una CA inicial (denominada como CA raíz) firma el certificado de una o varias CA intermedias y las autorizas para poder firmar certificados electrónicos de usuarios e incluso para firmar certificados de otras CA. El motivo de hacer esto es por seguridad, para exponer lo mínimo a la CA raíz ya que ésta es la CA en quien todos confían y por tanto la más importante.

Cuando se quiere validar un certificado, se determina qué CA lo ha firmado y si el certificado de esta CA firmante fue firmado por otra CA y así sucesivamente

hasta llegar a la CA raíz. El certificado de la CA raíz siempre estará firmado por la propia CA. Si la CA raíz es considerada de confianza y las firmas de los certificados son correctos, entonces se podrá considerar que la cadena de certificados sigue la cadena de confianza y por tanto todos los certificados implicados en la validación quedan así validados. Este proceso encadenado de validación es a lo que se denomina cadena de confianza.

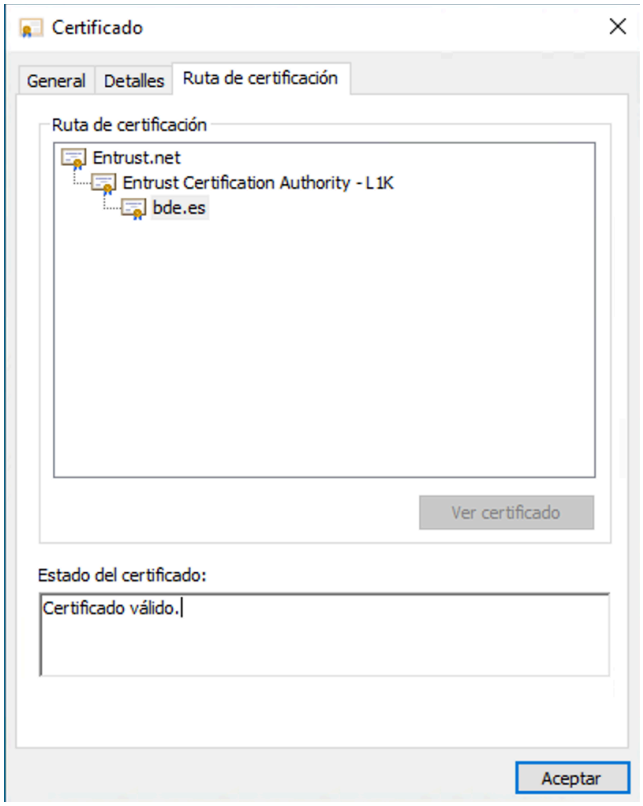

Figura 2.5: Ruta de certificación que muestra la cadena de confianza de certificados

El estándar actual para los certificados digitales es X.509 v3 y se especifica en la RFC 5280 “Internet X.509 Public Key Infrastructure Certificate Certificate Revocation List (CRL) Profile” de la “Internet Engineering Task Force (IETF) [33]. En este estándar se especifica la estructura de un certificado digital X.509, siendo su contenido el siguiente:

- Versión
- Número de serie del certificado
- ID del algoritmo que ha utilizado la CA para firmar el certificado (normalmente será RSA o DSA)
- Emisor (CA)
- Validez del certificado:
  - No antes de: [fecha]
  - No después de: [fecha]
- Sujeto: titular del certificado (persona física o jurídica, servidor o servicio), expresado en notación DN (Distinguished Name)
  - CN: Common Name
  - OU: Organization Unit
  - O: Organization
  - C: Country

- Información relativa a la clave pública del sujeto:
  - Algoritmo de clave pública
  - Clave pública
- Identificador único del emisor (parámetro opcional)
- Identificador único del sujeto (parámetro opcional)
- Extensiones (parámetro opcional)

A continuación, se muestra del contenido de un certificado electrónico real (certificado de bde.es [34]):

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            7a:af:23:b9:a9:a1:23:e1:f2:81:dc:03:91:80:3e:04
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Entrust, Inc., OU=See www.entrust.net/legal-terms, OU=(c) 2012
Entrust, Inc. - for authorized use only, CN=Entrust Certification Authority - L1K
        Validity
            Not Before: Jun  7 09:41:13 2022 GMT
            Not After : Jul  6 09:41:13 2023 GMT
        Subject: C=ES, ST=Madrid, L=Madrid, O=Banco de Espa\xC3\xB1a, CN=bde.es
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:d9:d8:1e:8f:c7:79:f9:f3:f3:35:61:34:87:87:
                    bb:f0:ec:d6:b4:54:46:ba:3a:4a:88:4c:20:1d:1a:
                    94:d7:a4:43:66:19:33:71:e9:2b:ff:ec:40:7e:6e:
                    39:ab:65:fb:0c:6d:d8:68:81:b0:5d:38:db:ef:da:
                    b4:85:ce:49:61:19:77:fc:a3:27:59:34:75:85:fd:
                    95:30:22:38:af:3b:65:d1:78:9e:29:62:61:53:d2:
                    f9:4f:12:ba:bf:9b:22:d3:5f:4b:e0:69:df:57:94:
                    c6:c2:31:b5:4a:1f:c4:d5:6e:9a:a9:22:97:dc:32:
                    a9:94:12:aa:df:5d:29:13:74:d1:19:2a:70:d6:aa:
                    30:51:37:be:98:c2:3c:ac:6a:3b:29:9c:bd:54:ca:
                    c7:c5:2b:aa:b8:df:d0:93:1d:41:3a:9c:b8:c0:58:
                    32:7f:80:e5:b5:47:bc:6f:05:1c:76:e7:5f:f7:79:
                    90:d9:97:f4:af:b1:ad:a2:1d:7c:82:be:ae:ec:60:
                    33:dc:da:55:98:bc:50:5d:4b:ad:5e:35:70:4d:82:
                    c7:58:7c:0c:26:e0:bc:5c:df:7b:f9:84:69:bf:ec:
                    a9:40:57:6b:f6:80:bf:5b:24:15:75:19:3a:48:9e:
                    d3:17:b5:35:4a:f9:dd:2f:46:d3:c8:ea:a7:78:ac:
                    6b:8d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                89:D7:F6:E4:90:4C:C5:59:EB:4B:A0:F0:B4:9C:B8:31:C8:62:EE:FE
            X509v3 Authority Key Identifier:
                keyid:82:A2:70:74:DD:BC:53:3F:CF:7B:D4:F7:CD:7F:A7:60:C6:0A:4C:BF

            Authority Information Access:
                OCSP - URI:http://ocsp.entrust.net
                CA Issuers - URI:http://aia.entrust.net/l1k-chain256.cer

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.entrust.net/level1k.crl

            X509v3 Subject Alternative Name:
                DNS:bde.es, DNS:www.bde.es
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Certificate Policies:
                Policy: 2.16.840.1.114028.10.1.5
                  CPS: https://www.entrust.net/rpa
```

```
                Policy: 2.23.140.1.2.2

            1.3.6.1.4.1.11129.2.4.2:

...l.j.w.U.....6.J...W<S...8xp%../...........=.(......H0F.!....\.>.>T.l.r..k..*4'xq.....
.(H..!..................$q.f(....ip.=.R.w..sw...P.c.......Jy-
.g.......y6......=.(......H0F.!...`.....;H.
..G.,/J_U^er...w3.q".!...P.i....?....u"...Z.GM...$H
..o.v.....|.....=..>.j.g)]...$...4........=.(......G0E.
N....N.wWO.P.J..o.Pv...H...{v.q..!......pM.tj@...K.;..[r..!6.J(u..4
    Signature Algorithm: sha256WithRSAEncryption
         7f:cc:20:c2:95:d0:a4:50:fd:37:4a:3e:6c:80:0a:d5:d2:36:
         24:d1:e0:85:e2:62:ff:d9:19:1d:b9:ad:02:38:45:75:a4:9d:
         55:06:4d:e5:2b:f5:3a:55:b5:3a:fc:ff:17:ec:0b:12:77:48:
         68:ac:6b:2d:50:51:d0:21:7c:57:41:b8:e3:ea:17:d5:76:42:
         ca:84:57:b2:ad:cf:23:2a:8a:63:e9:07:52:ac:15:94:47:10:
         cc:66:14:76:e3:6f:c2:9e:07:00:a0:32:be:b9:db:e4:e4:4d:
         eb:b1:54:d9:a0:33:dd:65:4b:8d:80:f2:98:0e:af:7e:10:04:
         f8:83:77:30:71:5a:f6:46:f8:e1:8c:ce:48:5c:29:53:89:ff:
         24:be:17:37:cc:a8:a9:82:a9:91:5e:a1:0c:97:62:78:33:bf:
         ad:ba:d2:8f:ab:93:0b:d5:ca:8b:5a:d4:b7:7d:67:6b:73:eb:
         79:d5:ba:1c:6d:1a:0c:4f:1d:f7:e8:c6:91:3c:f2:37:d4:c2:
         86:b4:fe:05:49:9f:16:a4:bb:b5:56:67:7a:88:e9:49:99:2d:
         e1:dd:1e:a1:1f:24:bd:11:45:b5:fd:cf:76:95:e2:5c:9a:3b:
         5a:4a:cc:5c:b9:5b:ab:17:83:a1:19:de:64:5b:82:50:ee:e2:
         7c:7a:a5:02
```

Los certificados electrónicos tienen una fecha de caducidad que suele ser inferior a cinco años. La Ley 59/2003, de 19 de diciembre, de firma electrónica [29], establece que la duración máxima de los certificados reconocidos será de cinco años. El motivo es obligar a renovar el certificado para adaptarlo a los cambios tecnológicos (evitando que haya en circulación certificados tecnológicamente vulnerables) y para que, en caso de que el certificado se haya visto comprometido, su uso indebido no pueda realizarse de forma ilimitada. Sin embargo, también pueden darse otras circunstancias por las que un certificado electrónico deba ser invalidado antes de su fecha de caducidad (por ejemplo: que la clave privada del usuario del certificado haya sido revelada), para estos casos existen los siguientes mecanismos:

- Lista de revocación de certificados (o, en inglés, o Certificate Revocation List o CRL): este mecanismo consiste en que las CA exponen una lista con los números de serie de los certificados que han sido revocados. De este modo, cuando un usuario quiere comprobar la validez de un certificado, descarga la lista CRL de la CA y comprueba si el certificado que quiere comprobar está o no en la lista de certificados revocados.

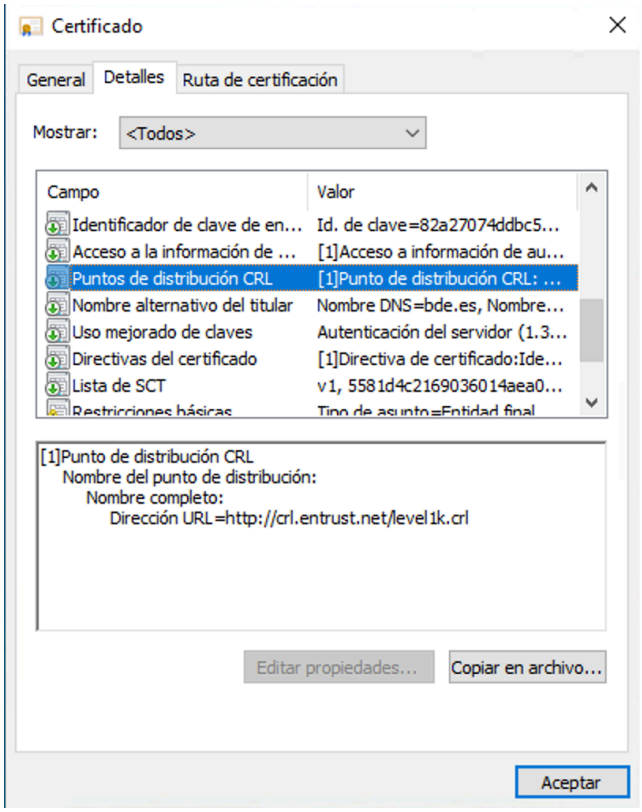

Figura 2.6: Detalle de lista CRL en un certificado electrónico

- Protocolo de comprobación del estado de un certificado en línea (o, en inglés, Online Certificate Status Protocol o OCSP): este mecanismo consiste en que las CA exponen un servicio online (que suele ser HTTP) donde se puede consultar si un certificado está o no revocado.

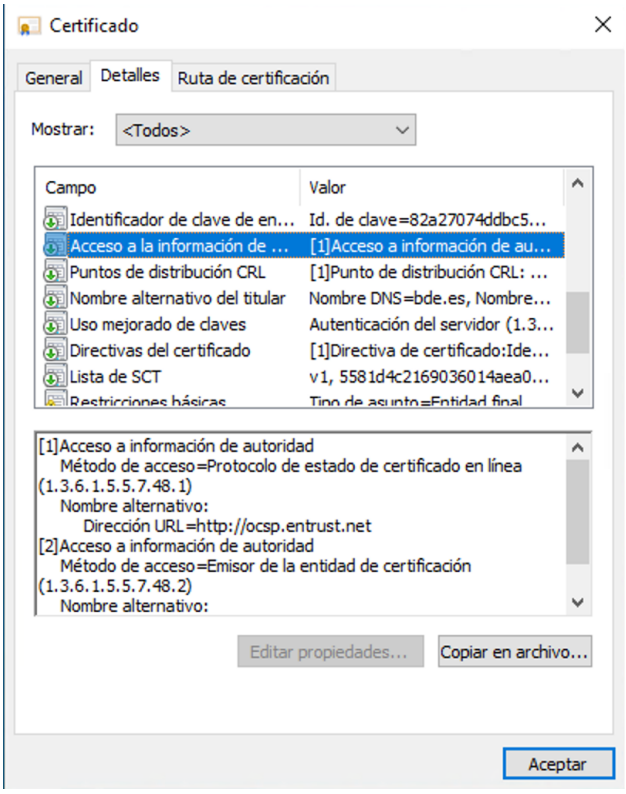

Figura 2.7: Detalle de OCSP en un certificado electrónico

Los sistemas informáticos poseen repositorios locales donde almacenan los certificados digitales en los que confían o que son utilizados para realizar firmas digitales. Según el sistema operativo, el lugar donde se almacenan los certificados es diferente. A continuación, se indica la ubicación de los almacenes de certificados para los sistemas operativos más comunes:

- Sistemas Linux basados en Debian [35]:
  - Los certificados instalados de CA de confianza se almacenan en: /etc/ssl/certs/
  - Los nuevos certificados generados se almacenan en: /usr/local/share/ca-certificates/

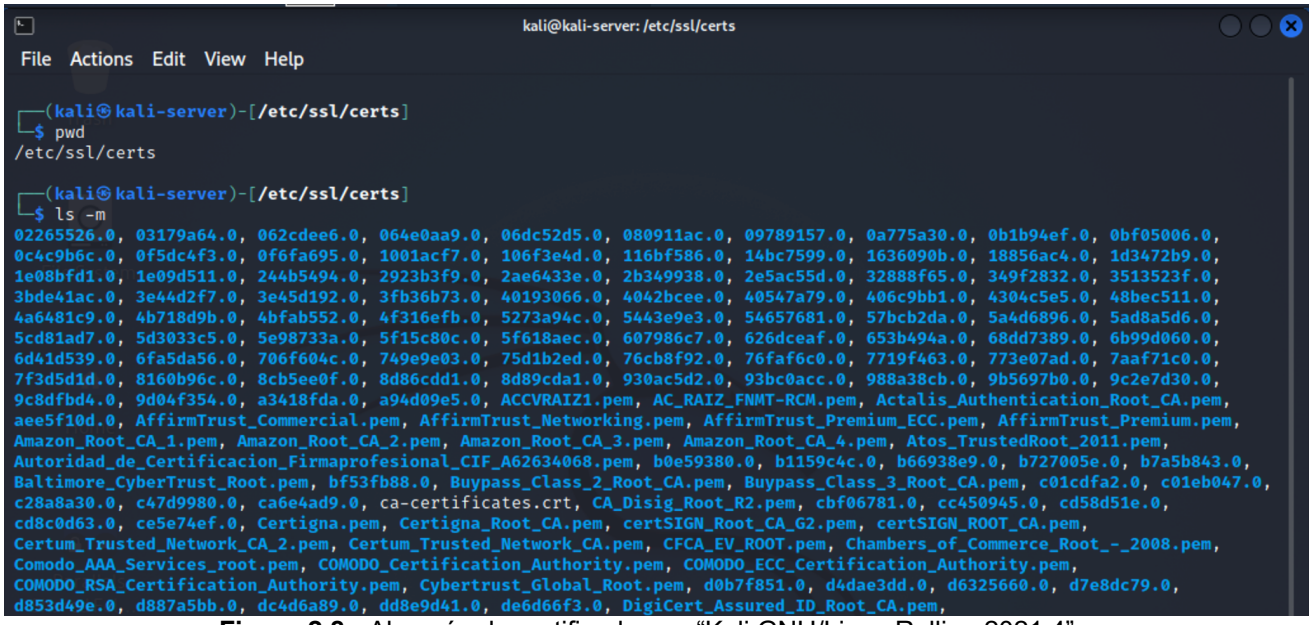

Figura 2.8: Almacén de certificados en "Kali GNU/Linux Rolling 2021.4"

- Sistemas Mac:
  - Los certificados pueden ser accedidos a través de la aplicación "Acceso a Llaveros" [36]. En el apartado "Certificados" se encuentran los certificados de CA instalados y en el apartado "Mis certificados" se encuentran los certificados propios instalados.

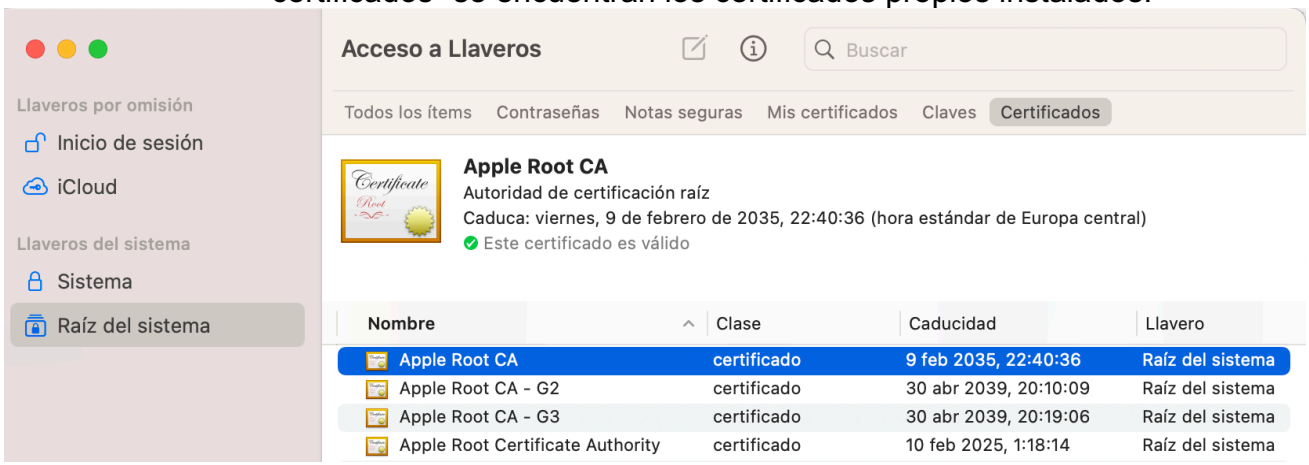

Figura 2.9: Almacén de certificados en macOS 13.2.1

- Sistemas Windows 10:
  - El acceso a los certificados se realiza ejecutando “certmgr.msc” en la línea de comandos [37]

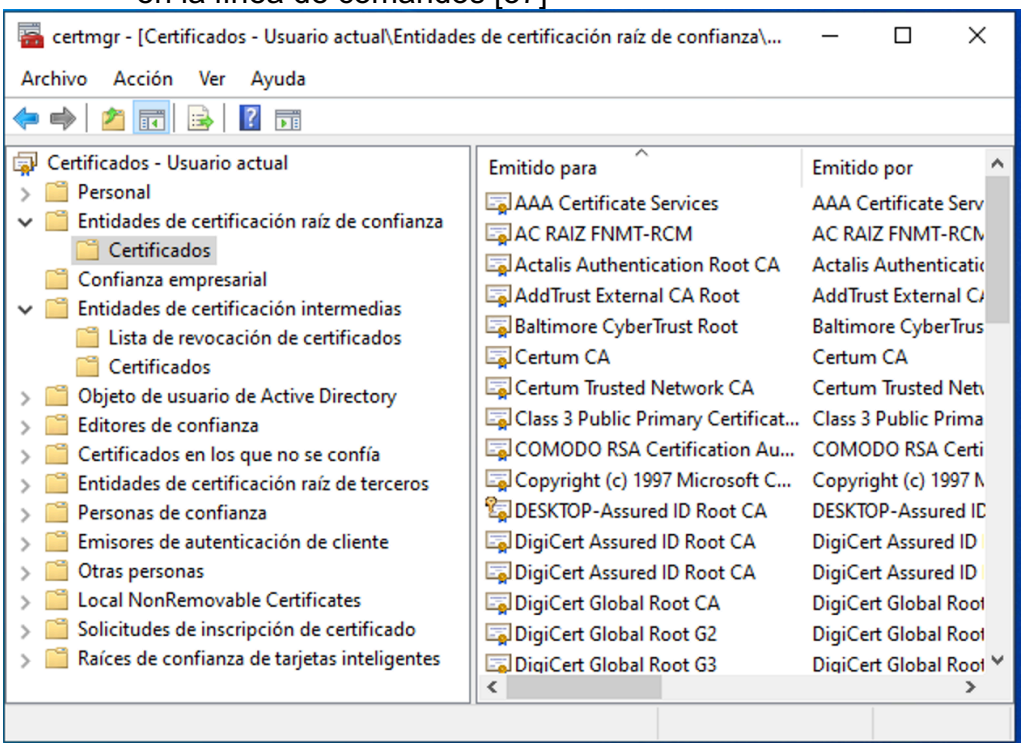

Figura 2.10: Almacén de certificados en Windows 10

Además del propio almacén de certificados de los sistemas operativos, los propios servicios dentro de un sistema informático pueden tener sus propios almacenes de certificados. En el caso de “Java”, la máquina virtual toma los certificados del almacén contenido en: $JAVA_HOME/lib/security/cacerts , siendo $JAVA_HOME la ubicación de la JVM [38].

2.3.3. Activación y protección de la clave privada para hacer firma digital

La firma digital utiliza un par de claves criptográficas, una clave privada y una clave pública, para garantizar que el contenido del documento no se haya alterado y que la persona que lo firma es quien dice ser. La clave privada es una parte fundamental del proceso de firma digital y una pieza de información confidencial que solo el firmante debe poseer. Es importante proteger esta clave privada para evitar que otras personas la utilicen de forma no autorizada.

La activación de la clave privada es el proceso de utilizar la clave privada para crear una firma digital. Es importante que este proceso se realice en un entorno seguro y confiable para garantizar la autenticidad y la integridad de la firma digital. La activación de la clave privada generalmente implica el uso de un software especializado de firma digital, que se utiliza para firmar electrónicamente documentos y otros tipos de información. El software suele requerir que el usuario introduzca su clave privada para poder crear una firma digital. Es importante que el software sea seguro y esté actualizado para garantizar que la firma digital sea segura.

El almacenamiento seguro de claves privadas es esencial para garantizar la seguridad de la firma digital y prevenir el acceso no autorizado a la información. A continuación, se presentan algunas de las formas más comunes de almacenar las claves privadas de forma segura:

- Dispositivos de hardware especializados. Se trata de dispositivos físicos que se conectan a un equipo informático y se utilizan para almacenar las claves privadas. Algunos ejemplos de este tipo de dispositivos son los tokens USB y las tarjetas inteligentes. Estos dispositivos suelen requerir algún tipo de contraseña para acceder a la clave privada, añadiendo así una capa adicional de seguridad. El DNIe [15] es un ejemplo de dispositivo hardware que almacena la clave privada.
- Almacenamiento en un contenedor criptográfico. Un contenedor criptográfico es un archivo o componente de software que se utiliza para almacenar claves privadas y otros datos de seguridad de forma segura. Los contenedores criptográficos están diseñados para proteger las claves privadas de accesos no autorizados y para facilitar su uso en aplicaciones que requieren seguridad. Un ejemplo de contenedor criptográfico es Keystore, que se utiliza en la plataforma Java para almacenar claves privadas, certificados y otros datos de seguridad.
- Almacenamiento en un servidor seguro. La elección de algunas organizaciones es almacenar sus claves privadas en un servidor seguro, como un HSM (Módulo de Seguridad de Hardware), que proporciona una seguridad adicional y control de acceso. En lugar de almacenar las claves privadas en el disco duro local de un ordenador, se almacenan en un servidor remoto que está protegido por medidas de seguridad avanzadas. Este enfoque tiene varias ventajas, como la centralización de la gestión de claves, el acceso controlado a las claves y la capacidad de escalar para soportar grandes volúmenes de claves.
- Almacenamiento en la nube con cifrado. En la nube también hay servicios específicos para el almacén de claves con alta seguridad y cifrado de extremo a extremo. Algunos ejemplos son: AWS Key Management Service [30], Cloud Key Management Service [31] y Azure Key Vault [32].

2.3.4. Tipos y formatos de firmas digitales

La firma digital puede tomar diferentes formas (firma digital simple, firma digital avanzada, firma digital biométrica, firma digital visible, etc). Sin embargo, dentro del contexto del reglamento eIDAS [2], las firmas digitales (o firmas electrónicas) pueden ser catalogadas en tres tipos, que dependerán de su nivel de seguridad:

- Firma electrónica simple. Se trata de la firma electrónica más simple de los tres tipos. Esta firma no proporciona información sobre la identidad del firmante y se puede crear fácilmente sin necesidad de utilizar un certificado digital emitido por una autoridad de certificación. Este tipo de firma es utilizada para firmar correos electrónicos o documentos electrónicos de bajo riesgo.

- Firma electrónica avanzada. Esta firma electrónica es como la firma electrónica simple, pero requiere una contraseña única para confirmar la identidad del firmante en la firma de documentos, por lo que se añade un nivel más de seguridad. Además, esta firma se basa en un certificado digital emitido por una autoridad de certificación.
- Firma electrónica cualificada. Es la firma con mayor nivel de seguridad de las tres, dentro del contexto eIDAS [2]. Esta firma es un tipo especial de firma electrónica avanzada que cumple con los requisitos de eIDAS [2]. Se requiere que el firmante incluya su información de identificación personal, una contraseña única y también un certificado cualificado que haya sido emitido por una autoridad de certificación confiable. Esta firma electrónica sirve para firmar documentos legales y transacciones financieras importantes.

Las firmas digitales pueden ser implementadas en diferentes formatos. Se entiende como formato de firma digital a un conjunto de estándares y reglas que pueden ser utilizados para codificar una firma digital en un formato específico, de manera que la firma digital pueda ser verificada por cualquier persona que tenga acceso a la clave pública correspondiente. Los formatos de firma digital a usar para firmar un documento electrónico dependerán del tipo de documento electrónico a firmar y de las normativas legales o técnicas que se deseen cumplir. Generalmente incluirán la información necesaria para verificar la identidad del firmante, la fecha y hora de la firma, la información del certificado digital utilizado y la propia firma digital codificada en un formato específico.

A continuación, se exponen varios de los formatos de firma digital más conocidos y que además cumplen con la normativa eIDAS [2]:

- XMLDsig (XML Digital signature) [39]. El formato XMLDsig está basado en XML y es utilizado para garantizar la integridad y la autenticidad de documentos electrónicos de tipo XML.
- XAdES (XML Advanced Electronic Signature) [40]. El formato XAdES, al igual que XMLDsig, también está basado en XML y es utilizado para garantizar la integridad y la autenticidad de documentos electrónicos de tipo XML. Este formato proporciona un alto nivel de seguridad y es utilizado para la firma de documentos legales y transacciones financieras.
- PAdES (PDF Advanced Electronic Signature) [41]. El formato PAdES es el formato de firma digital utilizado para firmar documentos electrónicos en formato PDF. Proporciona un alto nivel de seguridad y es utilizado para la firma de documentos legales y transacciones financieras.
- CAdES (CMS Advanced Electronic Signature) [42]. El formato CAdES es un formato de firma digital basado en CMS (Cryptographic Message Syntax) que cumple con la normativa eIDAS [2] y sirve para la firma de documentos legales y transacciones financieras de documentos electrónicos en diferentes formatos.
- PKCS (Public-Key Cryptography Standards) [43]. Las firmas digitales PKCS son un conjunto de estándares que establecen la forma en que se deben crear y verificar las firmas digitales. Estos estándares son desarrollados por RSA Security y otros colaboradores.

2.3.5. Procedimiento de firmado de documentos y verificación

A través de la firma digital pueden firmarse documentos electrónicos cuyos formatos más comunes son: documentos XML, documentos PDF y documentos binarios (imágenes, texto, etc). Sea cual sea el formato del documento electrónico y el formato de la firma, los procesos generales de firma y de verificación de la firma siguen los mismos procedimientos.

El proceso de firma electrónica de un documento electrónico es el siguiente:

1. El usuario posee una firma digital y un documento electrónico que desea ser firmado
2. Se aplica un algoritmo hash sobre el documento electrónico para calcular su resumen. El algoritmo suele ser SHA256 [44].
3. Se aplica un cifrado del resumen utilizando la clave privada del usuario. Como resultado se obtiene un resumen firmado.
4. Se inserta el resumen firmado en el documento original. Como resultado se obtiene el documento con firma electrónica.

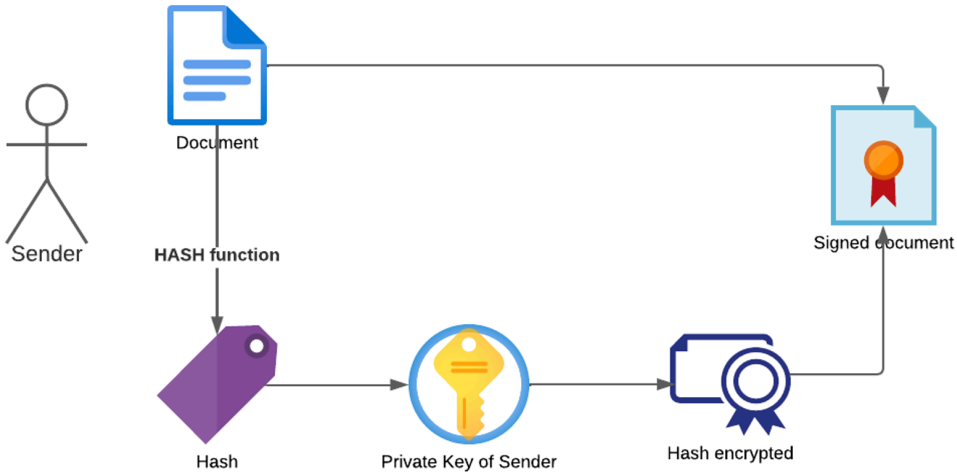

Figura 2.11: Proceso de firmado digital de un documento

Una vez que se obtiene un documento firmado, se puede enviar al receptor correspondiente. A su vez, el receptor correspondiente puede realizar un proceso de validación de la firma para determinar la integridad y autoría del documento.

El proceso de validación de firma de un documento electrónico firmado es el siguiente:

1. El receptor posee un documento firmado electrónicamente y la clave pública del firmante.
2. El receptor separa del documento firmado el resumen firmado del documento original.
3. Sobre el documento original, el receptor aplica un algoritmo hash para calcular su resumen. El algoritmo a aplicar debe ser el mismo que el que

haya utilizado el firmante para realizar su firma digital. Como resultado se obtiene un resumen firmado del documento.

4. Sobre el resumen firmado, el receptor aplica la clave pública del firmante para descifrar su contenido. Como resultado se obtiene el resumen del documento original que calculó el firmante durante su proceso de firma
5. El receptor compara los resúmenes calculados. Si son idénticos, entonces se demuestra que el documento no ha sufrido alteraciones durante su transporte y también la autoría del firmante. En caso de que ambos resúmenes no sean idénticos, la firma digital del documento no será válida porque significará que el documento ha sufrido alteraciones durante su transporte o que la autoría del documento no es correcta.

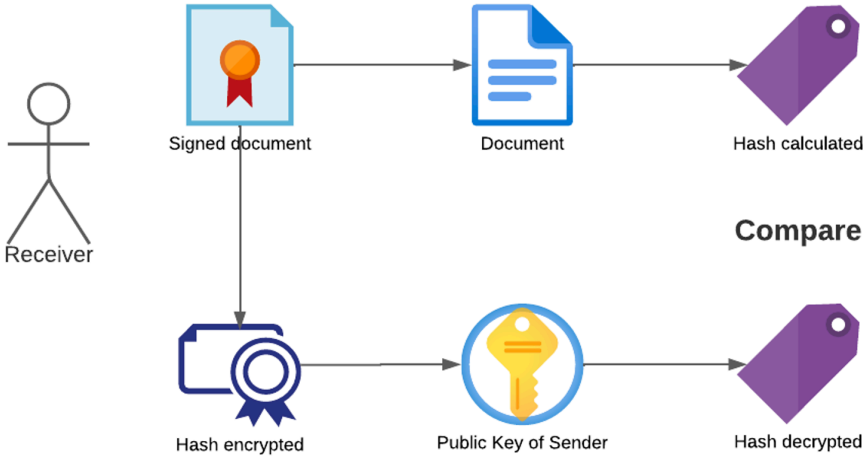

Figura 2.12: Validación de la firma digital de un documento

2.3.6. Procedimiento de firmado de documentos con XMLDSig

En el presente apartado se describe la firma XMLSDig y cómo utilizar este tipo de firma para firmar un documento XML.

La firma XMLSDig [39] se basa en añadir un elemento “Signature” al documento XML original que se desea firmar digitalmente. El elemento “Signature” contendrá los diferentes componentes de la firma y además estará dentro del espacio de nombres http://www.w3.org/2000/09/xmldsig#. La estructura básica del elemento “Signature” es la siguiente:

```
<Signature>
    <SignedInfo>
        <SignatureMethod />
        <CanonicalizationMethod />
        <Reference>
            <Transforms>
            <DigestMethod>
            <DigestValue>
        </Reference>
        <Reference />
    </SignedInfo>
    <SignatureValue />
    <KeyInfo />
    <Object />
</Signature>
```

A continuación, se describen los campos del elemento “Signature” de la firma XMLSDig:

- Signature: es el elemento padre de la firma digital
  - SignedInfo: incluye la información de cómo y qué se firma.
    - CanonicalizationMethod: indica cuál es el algoritmo de canonización para aplicar al elemento “SignedInfo” antes de que se le aplique el algoritmo de firma. El algoritmo de canonización es un algoritmo que permite normalizar los XML.
    - SignatureMethod: indica cuál es el algoritmo de firma para firmar el elemento “SignedInfo”
    - Reference: especifica el recurso que se firma mediante una referencia que se detalla en el atributo “ID” del elemento Reference.
      - Transforms: indica la transformación necesaria previa a la aplicación del algoritmo de cálculo de hash. Típicamente este algoritmo puede ser una canonización.
      - DigestMethod: indica el algoritmo de cálculo de hash que se aplicará después de haber aplicado las transformaciones descritas en el campo “Transforms”.
      - DigestValue: contiene el resultado en base64 de aplicar el algoritmo hash descrito en DigestMethod
  - SignatureValue: al elemento “SignedInfo” se le aplica la canonización descrita en “CanonicalizationMethod”, después se le aplica la firma digital descrita en “SignatureMethod”, el resultado se transforma en base64 y se inserta en el campo SignatureValue.
  - KeyInfo: contiene información sobre la clave pública que será utilizada para validar la firma.
  - Object: se trata de un campo opcional que contendría los datos firmados en caso de tratarse de una firma envolvente.

La implementación de la firma XMLDSig tiene tres tipos de formatos posibles, que pueden ser los siguientes:

- Enveloped Signature: la firma, es decir, el campo “Signature” se encuentra dentro del propio objeto firmado.
- Enveloping Signature: el objeto firmado se encuentra dentro del campo “Object” descrito anteriormente dentro de “Signature”.
- Detached Format: en este caso, la firma hace referencia a un objeto independiente. Es decir, que lo que se firma no está dentro del elemento “Signature” ni el elemento “Signature” está dentro de lo que se firma. En este caso, lo que se firma puede estar dentro de un mismo XML o incluso hacer referencia a una URL de un recurso completamente externo.

Para realizar una firma de un documento con XMLSDig, los pasos son los siguientes:

1. Determinar el objeto que se desea firmar

2. Indicar la referencia del objeto que se desea firmar en el atributo “ID” del objeto “Reference” de “Signature”.
3. Indicar en el campo “Transforms” las transformaciones que se deseen hacer al objeto antes de aplicar el algoritmo de hash. Por ejemplo, aplicar una canonización c14n.
4. Indicar en el campo “DigestMethod” cuál es el algoritmo que se va a aplicar para calcular el hash.
5. Realizar las transformaciones descritas en “Transforms” al objeto referenciado, posteriormente aplicarle el algoritmo de hash y transformarlo a base64. El resultado se añade a “DigestValue”.
6. Repetir los pasos anteriores para todos los campos “Reference” que haya indicados
7. Indicar el tipo de canonización a aplicar en el elemento “CanonicalizationMethod”
8. Indicar el tipo de firma que se va a aplicar en el campo “SignatureMethod”
9. Aplicar el algoritmo de canonización [45] descrito en “CanonicalizationMethod” sobre el campo “SignedInfo”. Esta canonización consiste en aplicar una transformación al fichero XML para normalizarlo. Es decir, que, aunque sean válidas diferentes sintaxis en un XML, la canonización transformaría el XML en solo una de las únicas formas posibles de expresar el XML. El algoritmo de canonización viene descrito en el campo “CanonicalizationMethod” dentro de “Signature” y también puede venir dentro de “Transforms”. El método de canonización más extendido es c14n (http://www.w3.org/2001/10/xml-exc-c14n#”).
10. Al resultado de la canonización anterior, aplicarle el algoritmo de firma indicado en “SignatureMethod”
11. Transformar el resultado anterior a base64 y poner el valor resultante en el elemento “SignatureValue”
12. Introducir en el elemento “KeyInfo” información relacionada con la clave de validación de la firma. Por ejemplo: una referencia a otro campo del XML en donde aparezca en base64 la clave pública del firmante.

Si un documento XML firmado ve modificado uno o más de sus caracteres respecto al documento XML original, su hash sería diferente, por lo que el elemento “DigestValue” correspondiente se vería alterado y por consiguiente también lo haría el valor de “SignatureValue”. Es decir, si un carácter del XML firmado se modifica, también lo hará su firma. Esto implica que si después de haber firmado un documento XML, se modifica uno o más caracteres, el proceso de validación de firma fallará debido a que los hashes del objeto no coincidirán. De este modo, la firma digital aseguraría la integridad de los datos.

2.4. Facturas electrónicas

2.4.1. Introducción

Una persona que posea una identidad electrónica tendrá la posibilidad de poseer una firma digital con un certificado electrónico que vincule la identidad del dueño de la firma con su clave pública. A través de su firma digital podrá

firmar documentos electrónicos con la misma validez que con una firma manuscrita. Debido a la transformación digital, estos documentos van abarcando cada vez más ámbitos tanto particulares como empresariales. Se pueden firmar contratos, declaraciones, formularios, etc.

Una de las utilidades prácticas de la firma de documentos electrónicos es la de la firma de facturas electrónicas. En el presente apartado se trata de hacer un repaso sobre las facturas electrónicas y su relación con las firmas digitales.

2.4.2. Regulación sobre facturación

Una factura es un elemento en formato físico (en papel) o digital (documento electrónico) que sirve para justificar un servicio prestado o la entrega de un bien y se entrega al cliente para exigir su pago [46].

En España, la regulación sobre las facturas y facturas electrónicas se hace en el "Real Decreto 1619/2012, de 30 de noviembre, por el que se aprueba el Reglamento por el que se regulan las obligaciones de facturación" (RD 1619/2012) [47]. En RD 1619/2012 se tratan los siguientes temas:

- Obligaciones sobre cuándo se tiene que realizar una factura
- Contenido de una factura
- Medios de expedición de una factura
- Factura electrónica y requisitos sobre su autenticidad e integridad
- Plazo para la expedición de facturas
- Otros aspectos: moneda y lengua en que se pondrán expresar y expedir las facturas, facturas recapitulativas, duplicados de las facturas, facturas rectificativas, particularidades de la obligación de documentar las operaciones en los regímenes especiales del Impuesto, remisión de facturas, conservación de facturas y otras disposiciones.

2.4.2.1. Obligaciones sobre cuándo se tiene que realizar una factura

Los empresarios o profesionales que realizan la entrega de un producto o la prestación de algún servicio deberán expedir y entregar facturas. Además, éstos a su vez también están obligados a conservan las facturas recibidas de otros empresarios. Existen algunas condiciones especiales contempladas en el artículo 3 de RD 1619/2012 [47] por las que se exime de la expedición de facturas. En el artículo 4 de RD 1619/2012 [47] se indica bajo qué casos determinados (como servicios de hostelería y restauración, peluquería, uso de autopistas, etc.) puede emitirse una factura simplificada en lugar de una factura completa.

2.4.2.2. Contenido de la factura

En el artículo 6 de RD 1619/2012 [47] se describe el contenido que deben tener las facturas. Los aspectos más importantes son los siguientes:

- El número de factura y, si procede, también la serie.
- Fecha de expedición de la factura

- Nombre y apellidos, razón o denominación social completa del expedidor de la factura
- Nombre y apellidos, razón o denominación social completa del destinatario de las operaciones de la factura
- Número de identificación Fiscal del expendedor de la factura
- Número de identificación Fiscal del destinatario
- Domicilio del obligado a expedir la factura
- Domicilio del destinatario de las operaciones
- Descripción de las operaciones, el precio unitario sin impuesto, así como cualquier descuento o rebaja que no esté incluido en el precio unitario
- El tipo impositivo o tipos impositivos, en su caso, aplicados a las operaciones
- La cuota tributaria que deberá consignarse por separado
- La fecha en la que se hayan efectuado las operaciones que se documentan en la factura
- Si una operación está exenta del Impuesto, deberá hacerse una referencia a la disposición correspondiente que así lo indica.
- En las entregas de medios de transporte referidas en el artículo 25 de la Ley del Impuesto, deberá indicarse información relativa a las características del medio de transporte, la fecha de su primera puesta en servicio y distancias recorridas u horas de navegación o vuelo realizadas hasta su entrega.
- En algunos casos particulares contemplados en el RD 1619/2012, deberá aparecer en la factura una mención correspondiente a cada caso, según lo establece el RD.

2.4.2.3. Medios de expedición de las facturas

En el artículo 8 de RD 1619/2012 [47] se indica que las facturas se podrán expedir tanto en formato de papel como en formato electrónico, pero que ya sea en un formato u otro, deberá poder garantizarse en la factura la autenticidad de su origen (garantizar la identidad del proveedor de los bienes o prestador de los servicios facturados, y la del emisor de la factura, en caso de no ser el mismo), la integridad de su contenido (garantizar que el contenido de la factura no ha sido modificado) y su legibilidad durante todo el periodo de conservación de ésta.

2.4.2.4. Factura electrónica

En el artículo 9 de RD 1619/2012 [47] se indica que las facturas electrónicas son facturas que se hayan expedido y recibido en formato electrónico y que se ajusten a lo establecido en el reglamento. Además, en el mismo artículo se indica que la expedición de la factura electrónica estará condicionada a que su destinatario haya dado su consentimiento.

2.4.2.5. Autenticidad e integridad de la factura electrónica

En el artículo 10 de RD 1619/2012 [47] se indican las opciones que tiene una factura electrónica para cumplir con los requisitos de autenticidad e integridad

descrito en el artículo 8 de RD 1619/2012 [47]. Las opciones que se indican para la factura electrónica son:

- Firma electrónica. Regulada en la "Ley 59/2003, de 19 de diciembre, de firma electrónica" [29].
- Intercambio electrónico de datos (EDI). Regulado bajo la "Recomendación de la Comisión de 19 de octubre de 1994 relativa a los aspectos jurídicos del intercambio electrónico de datos" [48].

La factura electrónica que utilice firma electrónica deberá usar un certificado electrónico reconocido que cumpla con la "Directiva 1999/93/CE del Parlamento Europeo y del Consejo, de 13 de diciembre de 1999, por la que se establece un marco comunitario para la firma electrónica" (Directiva 1999/93/CE) [49].

En el Anexo I de la Directiva 1999/93/CE [49], se establece que los certificados reconocidos deberán contener lo siguiente:

- La indicación de que el certificado se expide como certificado reconocido
- La indicación del proveedor de servicios de certificación y el Estado en que está establecido
- Nombre y apellidos del firmante o un seudónimo que conste como tal
- Un atributo específico del firmante, en caso de que fuera significativo en función de la finalidad del certificado
- Los datos de verificación de firma que correspondan a los datos de creación de firma bajo control del firmante
- Una indicación relativa al comienzo y fin del periodo de validez del certificado
- El código identificativo del certificado
- La firma electrónica avanzada del proveedor de servicios de certificación que expide el certificado
- Los límites de uso del certificado, si procede
- Los límites del valor de las transacciones para las que puede utilizarse el certificado, si procede

Por otro lado, en el Anexo II de la Directiva 1999/93/CE [49], se establecen los requisitos de los proveedores de servicios de certificados que expiden certificados reconocidos:

- Demostrar la fiabilidad necesaria para prestar servicios de certificación.
- Garantizar la utilización de un servicio rápido y seguro de guía de usuarios y de un servicio de revocación seguro e inmediato.
- Garantizar que pueda determinarse con precisión la fecha y la hora en que se expidió o revocó un certificado.
- Comprobar debidamente, de conformidad con el Derecho nacional, la identidad y, si procede, cualesquiera atributos específicos de la persona a la que se expide un certificado reconocido.
- Emplear personal que tenga los conocimientos especializados, la experiencia y las cualificaciones necesarias correspondientes a los servicios prestados.

- Utilizar sistemas y productos fiables que estén protegidos contra toda alteración y que garanticen la seguridad técnica y criptográfica de los procedimientos con que trabajan.
- Tomar medidas contra la falsificación de certificados y, en caso de que el proveedor de servicios de certificación genere datos de creación de firma, garantizar la confidencialidad durante el proceso de generación de dichos datos.
- Disponer de recursos económicos suficientes para operar de conformidad con lo dispuesto en la Directiva, en particular para afrontar el riesgo de responsabilidad por daños y perjuicios, por ejemplo, contratando un seguro apropiado.
- Registrar toda la información pertinente relativa a un certificado reconocido durante un período de tiempo adecuado, en particular para aportar pruebas de certificación en procedimientos judiciales. Esta actividad de registro podrá realizarse por medios electrónicos.
- No almacenar ni copiar los datos de creación de firma de la persona a la que el proveedor de servicios de certificación ha prestado servicios de gestión de claves.
- Antes de entrar en una relación contractual con una persona que solicite un certificado para apoyar a partir del mismo su firma electrónica, informar a dicha persona utilizando un medio de comunicación no perecedero de las condiciones precisas de utilización del certificado, incluidos los posibles límites de la utilización del certificado, la existencia de un sistema voluntario de acreditación y los procedimientos de reclamación y solución de litigios.
- Utilizar sistemas fiables para almacenar certificados de forma verificable.

2.4.3. Facturas electrónicas y las administraciones públicas

Con el objetivo de erradicar la morosidad en el sector público, se generó la “Ley 25/2013, de 27 de diciembre, de impulso de la factura electrónica y creación del registro contable de facturas en el Sector Público” (Ley 25/2013) [50]. Esta Ley obliga a facturar electrónicamente a todos los proveedores de las Administraciones Públicas (Estatal, Autonómica y Local) a partir del 15 de enero de 2015.

Los métodos de envío de facturas electrónicas desde los proveedores hasta las Administraciones Públicas son:

- A través del portal web del Punto General de Entrada de las Facturas Electrónicas.
- Mediante la conexión automática entre el servicio de facturación electrónica del proveedor y el Punto General de Entrada de las Facturas Electrónicas.

Las facturas electrónicas destinadas a las Administraciones Públicas tienen tres requisitos específicos (aparte de los requisitos generales de facturación establecidos en RD 1619/2012 [47]):

- Las facturas deberán estar escritas en un lenguaje informático determinado: Facturae 3.2 o Facturae 3.2.1 [51]

- Las facturas electrónicas deberán estar firmadas electrónicamente, bajo los requisitos establecidos en RD 1619/2012 [47] y Directiva 1999/93/CE [49].
- Indicar el destinatario de la factura, especificando: oficina contable, órgano gestor y unidad tramitadora. Estos datos los facilita la propia Administración Pública.

Cabe destacar que las facturas que no vayan dirigidas a las Administraciones Públicas pero que vayan dirigidas a otras empresas o consumidores, tienen las siguientes características diferentes respecto a las dirigidas a las Administraciones Públicas:

- La emisión de factura electrónica en lugar de hacerse en papel estará condicionada a que el destinatario de la factura acepte la recepción en formato electrónico. No como sucede con las Administraciones Públicas donde el formato electrónico es el obligatorio.
- El formato de la factura no tiene que estar escrito en formato “Facturae” sino que puede realizarse con programas ofimáticos. No obstante, también puede utilizarse el formato Facturae, especialmente en entornos B2B.

2.4.4. Formato de facturas electrónicas: Facturae

Conforme a la Ley 25/2013, de 27 de diciembre, de impulso de la factura electrónica y creación del registro contable de facturas en el Sector Público [50], a partir del 15 de enero de 2015 las facturan que se remitan a las Administraciones Públicas serán electrónicas y se ajustarán al formato Facturae (definido mediante XML) con firma electrónica XAdES [40].

La estructura de una factura electrónica en formato Facturae es las siguientes [51] [52]:

- FileHeader: contiene los metadatos de la factura. Se trata de información relativa a la modalidad de la factura (si es individual o por lotes), quién emite la factura (el proveedor, el emisor o una tercera parte), los datos relativos a una tercera parte en caso de que sea ésta quien la emita y firme, etc.
- Parties: contiene los datos relativos al emisor y receptor de la factura. Los datos del emisor quedarían contenidos dentro de “SellerParty” y los datos del receptor dentro de “BuyerParty”.
- Invoices: se trata del conjunto de facturas contenidas en el fichero. Cada factura estaría contenida dentro de “Invoice”
- Extensions: este campo permite que cuando el emisor y el receptor tengan interés conjunto puedan incorporar ahí nuevas definiciones estructuradas que no estén definidas previamente en el esquema de la factura.
- Ds:Signature: contiene la firma electrónica que garantizará la identidad del emisor y la integridad de la factura electrónica. La firma electrónica puede ser realizada directamente por el proveedor o puede estar delegada en una tercera parte, en cuyo caso deberá figurar con los datos correspondientes en el apartado “FileHeader”.

2.5. Firmado de facturas electrónicas

El proceso de firmado de una factura electrónica implica la aplicación de una firma digital al documento electrónico que contenga la factura, garantizando así su integridad y autenticidad. Según cuál sea el destinatario de la factura, se aplicarán unas características de firma u otras. Sin embargo, el procedimiento general para firmar una factura electrónica es similar al de firmar cualquier otro documento, adecuando la firma al formato correspondiente de la firma electrónica y su destinatario.

Si la factura va destinada a las Administraciones Públicas, entonces el formato de la factura deberá ser Facturae (fichero de tipo XML), la firma a aplicar será una firma electrónica avanzada o firma electrónica cualificada y el método de firma será XAdES. En este caso, el procedimiento de firma de una factura sería el siguiente:

1. Aseguramiento de que el formato de la factura a firmar sea Facturae
2. Seleccionar el certificado y la clave privada con la que se realizará la firma de la factura
3. Aplicar canonización al fichero de factura recibida y calcular el hash del fichero de factura electrónica
4. Firmar el hash calculado con la clave privada y XAdES
5. Insertar la firma y el certificado utilizado para la firma en el XML de la factura

Si la factura no va destinada a las Administraciones Públicas, entonces el formato podrá ser XML, PDF u otro formato binario (pudiendo ser una imagen resultante del escaneo de una factura en papel). Se podrá utilizar una firma electrónica simple, avanzada o cualificada. Además, el método de firma podrá ser XMLDsig, XAdES, PAdES, CAdES, PKCS u otro método [53]. Por lo general, las facturas emitidas a particulares o entre empresas se emitirán en PDF utilizando CAdES, PAdES o PKCS para firmarlas. El procedimiento general será el siguiente:

1. Aseguramiento de que el formato de la factura a firmar sea el deseado por el destinatario.
2. Seleccionar el certificado y la clave privada con la que se realizará la firma de la factura
3. Calcular el hash del fichero de factura electrónica. En el caso de las facturas en XML, se aplicaría primero una canonización al contenido del fichero.
4. Firmar el hash calculado con la clave privada y el formato de firma deseado, que sea compatible con el formato de la factura electrónica a firmar.
5. Insertar la firma y el certificado público correspondiente a la firma aplicada en el documento electrónico resultante, que corresponderá con la factura electrónica firmada digitalmente.

El proceso de validación de la firma de una factura electrónica será el siguiente:

1. Se separa el documento original de la parte de firma

2. Se calcula el hash del documento original
3. Se obtiene el certificado público del firmante, que posiblemente estará embebido en la parte de la firma
4. Se obtiene el resumen firmado con la clave privada del emisor
5. Se aplica el certificado público del firmante al resumen firmado recibido, obteniendo como resultado un hash
6. Se compara el hash calculado del documento original con el resultado de la operación anterior. Si la comparación entre ambos hashes es idéntica, se valida la firma del documento.

2.6. Soluciones y herramientas de software

2.6.1. Software para la gestión de identidades

La gestión de identidades es un componente crítico de la seguridad de la información y existen diversas soluciones y herramientas disponibles para ayudar a las organizaciones a administrar las identidades de usuario de manera segura y eficiente. Al elegir una herramienta de gestión de identidad, es importante considerar varias características clave para asegurar que la elegida sea adecuada para las necesidades de la organización donde se implante. A continuación, se presentan algunas de las características más importantes a tener en cuenta:

- Funcionalidad: la herramienta debería ser capaz de proporcionar todas las necesidades de gestión de identidad y acceso de la organización, incluyendo autenticación, autorización, aprovisionamiento de usuarios, administración de contraseñas, etc.
- Cumplimiento normativo: la herramienta debería ser capaz de cumplir con los requisitos de seguridad (MFA, cifrado de datos, auditoría, etc) tanto de la organización como de la legislación bajo las que ésta se encuentre (GDRP, HIPAA, etc.).
- Escalabilidad: la herramienta debería ser capaz de manejar el posible crecimiento y expansión de la organización sin perder funcionalidad o velocidad.
- Integración: la herramienta debería ser capaz de integrarse con los sistemas y aplicaciones de la organización, incluyendo aplicaciones en la nube y locales. Además, la herramienta debería poseer la flexibilidad suficiente para adaptarse a necesidades específicas de la organización.
- Soporte técnico: la herramienta debería disponer de un soporte técnico sólido para proporcionar ayuda a los diferentes problemas que puedan surgir en el uso de la herramienta.
- Implementación: deberá considerarse si el servicio se ofrecerá desde la nube (“cloud”) o desde las instalaciones de la organización (“on premise”). La ventaja principal de un servicio “on premise” es que la organización dispone de mayor control sobre los datos y la seguridad. Por otro lado, la ventaja principal de un servicio “on cloud” es que se dispondrá de mayor flexibilidad y escalabilidad.
- Costes: deberán considerarse los posibles costes de la herramienta asociados a su instalación o explotación o asociados al soporte técnico.

Según las características de la organización, este aspecto puede resultar decisivo a la hora de considerar una solución.

En la siguiente tabla se muestran algunas de las principales soluciones y herramientas disponibles que pueden implementar las organizaciones para administrar las identidades de usuarios utilizando mecanismos de autenticación, autorización e incluso SSO y federación de identidades [53]-[71].

Producto	**Cloud**	**On premise**	**Coste**
Okta	SI	NO	De pago
Microsoft Azure Active Directory	SI	NO	De pago
Microsoft Active Directory	NO	SI	De pago
OneLogin	SI	SI	De pago
Ping Identity	SI	SI	De pago
Salesforce Identity	SI	NO	De pago
Google Cloud Identity	SI	NO	De pago
IBM Cloud Identity	SI	NO	De pago
IBM Security Identity Manager	NO	SI	De pago
ForgeRock Identity Platform	SI	SI	De pago
FreeIPA	NO	SI	Gratuita (pago por soporte)
Keycloak	NO	SI	Gratuita (pago por soporte)
Gluu Cloud	SI	NO	De pago
Gluu Server Community Edition	NO	SI	Gratuita (pago por soporte)
Oracle Identity Manager	NO	SI	De pago
Oracle Identity Cloud Service	SI	NO	De pago
OpenAM	SI	SI	Gratuita (pago por soporte)
WSO2 Identity Server	SI	SI	Gratuita (pago por soporte)

Tabla 2.2. Soluciones para la gestión de identidades

En apartados anteriores se había hecho mención a la posibilidad de integrar OTP a los sistemas de gestión de identidades con el objetivo de mejorar el proceso de autenticación. Algunas de las aplicaciones más comunes para generar OTP que son gratuitas y están disponibles tanto para dispositivos Android como para dispositivos iOS son las siguientes [72]-[79]:

- Google Authenticator
- Microsoft Authenticator
- Authy
- Duo Security
- LastPass Authenticator
- Yubico Authenticator
- FreeOTP
- Aegis Authenticator

2.6.2. Soluciones para la generación de certificados electrónicos

La generación de certificados electrónicos es esencial para posibilitar a un usuario a realizar firmas digitales. Existen diferentes soluciones para la generación de los certificados y la elección de una solución u otra dependerá de las necesidades que tenga la organización o del usuario. Algunas de las

soluciones más comunes para generar certificados electrónicos son las siguientes [80]-[87]:

- OpenSSL: es una herramienta de código abierto multiplataforma que permite la generación y administración de certificados electrónicos. También permite la creación de autoridades de certificación
- DigiCert: es una autoridad de certificación que ofrece una amplia gama de soluciones de certificación digital, incluyendo certificados SSL/TLS, firma electrónica y autenticación de identidad. Además, ofrece certificados electrónicos eIDAS [2] para autenticación, firma y cifrado que cumplen con los requisitos legales y normativos de la UE.
- GlobalSign: es una autoridad de certificación similar a DigiCert, que también ofrece servicios similares y es capaz de emitir certificados electrónicos eIDAS [2].
- Fábrica Nacional de Moneda y Timbre: es la entidad pública responsable de la emisión de certificados digitales en España. Ofrece certificados digitales para personas físicas, personas jurídicas y entidades sin personalidad jurídica. Sus certificados son compatibles con la normativa eIDAS [2] y se utilizan para autenticación, firma electrónica y cifrado.
- Camerfirma: es una entidad de certificación española que ofrece soluciones de certificación digital para empresas y organizaciones. Ofrece certificados digitales para autenticación, firma electrónica y cifrado que cumplen con los requisitos de la normativa eIDAS [2].
- Izenpe: es la entidad de certificación digital del Gobierno Vasco. Ofrece certificados digitales para personas físicas, personas jurídicas y entidades sin personalidad jurídica. Sus certificados son compatibles con la normativa eIDAS [2] y se utilizan para autenticación, firma electrónica y cifrado
- Agencia Notarial de Certificación: es una entidad de certificación española especializada en la emisión de certificados digitales notariales. Sus certificados son utilizados en la firma de documentos notariales y son compatibles con la normativa eIDAS [2].
- Firmaprofesional: es una entidad de certificación digital que ofrece soluciones de firma electrónica y certificación digital para empresas y particulares. Ofrece certificados digitales para autenticación, firma electrónica y cifrado que cumplen con la normativa eIDAS [2].

2.6.3. Soluciones para el firmado de facturas electrónicas

El software para el firmado de documentos electrónicos es una herramienta que permite a los usuarios la agregación de firmas digitales a documentos electrónicos para validar su autenticidad e integridad de los mismos. Esta tecnología permite el reemplazo de los procesos manuales y basados en papel de firmar y enviar documentos físicos por correo o fax, ahorrando tiempo, costes y recursos.

Existen diferentes soluciones para firmar documentos electrónicos. Algunas de las soluciones más comunes son las siguientes [88]-[94]:

- Adobe Acrobat Reader DC: Es un software de lectura de PDF que también permite a los usuarios firmar documentos electrónicos. La firma de documentos PDF está disponible en su versión gratuita.
- Adobe Sign: Es una plataforma de firma electrónica de Adobe que permite a los usuarios enviar, firmar y rastrear documentos electrónicos. Se trata de un servicio de pago que también ofrece algunas opciones de prueba gratuitas.
- LibreOffice: Es un paquete de software ofimático gratuito que incluye una herramienta de edición de documentos PDF llamada LibreOffice Draw. Esta herramienta permite a los usuarios agregar firmas electrónicas a los documentos PDF.
- Foxit Reader: Es un software de lectura de PDF que permite a los usuarios agregar firmas electrónicas a los documentos PDF. La firma de documentos PDF está disponible en su versión gratuita.
- GnuPG: Es una herramienta de cifrado y firma digital de código abierto. Permite a los usuarios agregar firmas electrónicas a los documentos PDF y otros tipos de archivos.
- SignNow: Es una plataforma de firma electrónica que permite a los usuarios enviar, firmar y rastrear documentos electrónicos. Se trata de un servicio de pago que también ofrece algunas opciones de prueba gratuitas.
- PandaDoc: Es una plataforma de gestión de documentos que permite a los usuarios crear, enviar y firmar documentos electrónicos. Se trata de un servicio de pago que también ofrece algunas opciones de prueba gratuitas.

En España, la firma de facturas electrónicas debe cumplir con los requisitos establecidos por la Agencia Tributaria. A continuación, se presentan algunas opciones para firmar facturas electrónicas en España [95]-[98]:

- FACe: Es una plataforma gratuita que permite la presentación electrónica de facturas a las administraciones públicas de España. La plataforma cuenta con un sistema de firma electrónica para validar la autenticidad y la integridad de las facturas.
- Signaturit: Es una herramienta que permite a los usuarios firmar y enviar facturas electrónicas. Se trata de un servicio de pago que también ofrece algunas opciones de prueba gratuitas.
- Factura Directa: Es una plataforma gratuita que permite a los usuarios enviar y recibir facturas electrónicas. La plataforma cuenta con un sistema de firma electrónica para validar la autenticidad y la integridad de las facturas.
- FacturaScripts: Es un software de facturación electrónica que permite a los usuarios crear, enviar y firmar facturas electrónicas.

3. Implementación de un caso práctico

3.1. Introducción

El servicio de firmado digital de facturas electrónicas con control de acceso basado en MFA es una solución de seguridad informática que permite a las empresas garantizar la autenticidad, integridad y confidencialidad de sus facturas electrónicas. En este caso práctico, se considera la implementación de un servicio de firmado de facturas electrónicas para mejorar la seguridad de las transacciones comerciales.

3.2. Análisis y diseño

3.2.1. Alcance

El alcance del servicio de firmado digital de facturas electrónicas con control de acceso basado en MFA del caso práctico es proporcionar a las empresas una solución segura y confiable para la firma digital de sus facturas electrónicas.

A continuación, se enumeran una serie de requisitos que se consideran dentro del alcance de la implementación del caso práctico:

- Implementar un sistema de control de acceso basado en OIDC que permita a los usuarios acceder al servicio de firmado digital de facturas electrónicas utilizando su identidad.
- Implementar un sistema de autenticación multi-factor que requiera que los usuarios puedan proporcionar un segundo factor de autenticación para verificar su identidad, lo que garantiza un nivel de seguridad adicional para la firma digital de las facturas electrónicas. El sistema MFA a implementar será OTP.
- Generar un certificado digital para la firma digital de facturas electrónicas. Se implementará una PKI para la generación de certificados para el propósito del presente TFM.
- Almacenar las claves de firma en un almacén de claves protegido.
- Implementar un servicio con API REST que se encargue del firmado de facturas electrónicas en formato PDF.
- El servicio de firmado de facturas electrónicas validará que la petición de firma haya sido emitida por un usuario que se haya identificado satisfactoriamente en el sistema de control de acceso.
- El servicio de firmado de facturas electrónicas se integrará con el sistema de almacenamiento de claves.
- Para la activación de claves de firma será necesario que el usuario haga un proceso de autenticación específico a tal efecto. Para conseguir esto, el servicio de firmado solo activará la clave de firma si la petición entrante contiene un token generado por el sistema de control de accesos que demuestre que el sistema se ha autenticado correctamente. Además, se establecerá un mecanismo basado en base de datos para garantizar que el token sea de un solo uso y que de esa

forma no se pueda usar un mismo token para activar la clave de firma varias veces.
- El servicio de firmado de facturas electrónicas almacenará en una base de datos la relación entre los usuarios y el alias del certificado digital que servirá para firmar digitalmente las facturas electrónicas.

A continuación, se detallan algunas consideraciones sobre la implementación del caso práctico y lo que se considera fuera del alcance definido:
- El certificado electrónico será emitido por una CA propia y no será emitido por una Autoridad de Certificación confiable que cumpla con la normativa eIDAS.
- El proceso de verificación de identidad de los usuarios durante la generación de su certificado electrónico es un aspecto crítico dentro del ciclo de vida de la identidad electrónica. Sin embargo, en la implementación del caso práctico no se realizará mediante un procedimiento específico.
- El sistema de control de accesos será desplegado en modo “desarrollo” y no se le aplicarán configuraciones exhaustivas que si aplicarían si se tratara de un entorno de producción.
- El sistema de control de accesos no estará integrado con un directorio de usuarios externo.
- El sistema de almacenamiento de claves y mecanismo de acceso a éstas debe realizarse mediante un sistema seguro y confiable. Además, debe garantizarse que solo pueda acceder a las claves el dueño de las mismas. Sin embargo, en la implementación del caso práctico se abordará el almacenamiento de claves en un almacén seguro pero que estará custodiado con una contraseña maestra que será la que utilice el servicio de firmado.
- En la implementación práctica solo se abordará el caso del firmado de facturas en formato PDF y no se abordarán las facturas en formato Facturae.
- En el alcance de la implementación práctica no se utilizará una herramienta de facturación para realizar la integración con el servicio a desarrollar. En su lugar, se simulará el comportamiento a través de una herramienta software genérica de cliente de API REST.

3.2.2. Identificación de los componentes

El servicio incluirá los siguientes componentes:
- Sistema de control de identidad: sistema de control de identidad y acceso encargado de la autenticación y autorización, basado en OIDC.
- Sistema de autenticación multi-factor: sistema de OTP complementario al sistema de control de identidad para dotar de mayor securización a la autenticación de usuarios.
- Servicio PKI: servicio utilizado para crear certificados electrónicos.
- Certificados digitales: certificados que serán utilizados para firmar digitalmente las facturas electrónicas.

- Sistema de almacenamiento de claves: sistema que almacenará de forma segura las claves que serán usadas para firmar digitalmente las facturas electrónicas.
- Servicio de firma digital: servicio con API REST securizado con OIDC, que recibe facturas electrónicas para que sean firmadas digitalmente.
- Base de datos: base de datos para almacenamiento de relación entre usuarios y alias de certificado de firma digital, y también utilizado para almacenar un registro de los tokens que ya hayan sido usados para firmar digitalmente una firma electrónica.
- Cliente del servicio de firma digital.

3.2.3. Casos de uso

En el presente apartado se realiza un análisis de los diferentes casos de uso de la implementación del caso práctico del TFM. El objetivo principal es el de describir cómo un usuario interactúa con los diferentes componentes y cómo interactúan entre sí los propios componentes.

Nombre	01 – La aplicación cliente realiza login satisfactorio con el sistema de control de identidad
Actores	Dueño del recurso Aplicación cliente Sistema de control de identidad Sistema de MFA
Precondiciones	El dueño del recurso tiene su usuario ya configurado en el sistema de control de identidad. El dueño del recurso tiene configurado su mecanismo de OTP. La aplicación cliente ya está configurada en el sistema de control de identidad.
Secuencia principal	1. El dueño del recurso obtiene su password de OTP a través de su sistema de MFA 2. Se envía petición de login a través de la aplicación cliente 3. Se redirige al dueño del recurso a la página de login del sistema de control de identidad para que se autentique 4. El dueño del recurso se autentica correctamente 5. El sistema de control de identidad valida las peticiones recibidas 6. El sistema de control de identidad responde a la aplicación cliente con un token
Resultado	La aplicación cliente posee un token de acceso

Nombre	02– La aplicación cliente realiza login insatisfactorio con el sistema de control de identidad por introducir mal sus credenciales
Actores	Dueño del recurso Aplicación cliente Sistema de control de identidad Sistema de MFA
Precondiciones	El dueño del recurso tiene su usuario ya configurado en el sistema de control de identidad. El dueño del recurso tiene configurado su mecanismo de OTP. La aplicación cliente ya está configurada en el sistema de control de identidad.

Secuencia principal	1. El dueño del recurso obtiene su password de OTP a través de su sistema de MFA 2. Se envía petición de login a través de la aplicación cliente 3. Se redirige al dueño del recurso a la página de login del sistema de control de identidad para que se autentique 4. El dueño del recurso se autentica incorrectamente 5. El sistema de control de identidad no valida las peticiones recibidas 6. La aplicación cliente no consigue obtener un token de acceso válido
Resultado	La aplicación cliente no finaliza correctamente el proceso de login y no posee un token de acceso válido

Nombre	03– La aplicación cliente realiza login insatisfactorio con el sistema de control de identidad por introducir mal su OTP
Actores	Dueño del recurso Aplicación cliente Sistema de control de identidad Sistema de MFA
Precondiciones	El dueño del recurso tiene su usuario ya configurado en el sistema de control de identidad. El dueño del recurso tiene configurado su mecanismo de OTP. La aplicación cliente ya está configurada en el sistema de control de identidad.
Secuencia principal	1. El dueño del recurso obtiene su password de OTP a través de su sistema de MFA 2. Se envía petición de login a través de la aplicación cliente 3. Se redirige al dueño del recurso a la página de login del sistema de control de identidad para que se autentique 4. El dueño del recurso introduce correctamente sus credenciales, pero introduce código OTP erróneo 5. El sistema de control de identidad no valida las peticiones recibidas 6. La aplicación cliente no consigue obtener un token de acceso válido
Resultado	La aplicación cliente no finaliza correctamente el proceso de login y no posee un token de acceso válido

Nombre	04– La aplicación cliente envía una petición al servicio de firmado y obtiene una factura electrónica firmada digitalmente
Actores	Aplicación cliente Servicio de firma digital Sistema de almacenamiento de claves Certificados digitales Sistema de control de identidad
Precondiciones	La aplicación cliente debe poseer un token de acceso válido El servicio de firma digital tiene acceso correcto al sistema de control de identidad La aplicación cliente posee una factura electrónica en formato PDF El servicio de firma digital tiene acceso correcto al sistema de almacenamiento de claves El usuario, dueño del recurso, tiene asignado el rol correspondiente

	para poder firmar facturas electrónicas La base de datos tiene configurado el usuario y su correspondiente alias de certificado que está almacenado en el Keystore El dueño del recurso posee un certificado válido configurado en el sistema de almacenamiento de claves
Secuencia principal	1. La aplicación cliente envía una petición al servicio de firma digital en el que incluye el token de acceso y la factura electrónica que desea que sea firmada 2. El servicio de firma digital recibe la petición desde la aplicación cliente 3. El servicio de firma digital valida la firma del token de acceso utilizando la clave pública del sistema de control de identidad que ha generado y firmado el token 4. El servicio de firma digital valida con una base de datos que el token no haya sido usado previamente para firmar una factura electrónica 5. El servicio de firma digital determina con el token está dentro de su tiempo de vida y no ha expirado, si el usuario tiene permisos suficientes para realizar una firma y si el usuario ha otorgado los permisos para firmar facturas electrónicas a la aplicación cliente 6. El servicio de firma digital valida el formato de la factura electrónica recibida 7. El servicio de firma digital accede al almacén de claves y toma las que corresponden para el usuario que realiza la petición 8. El servicio de firma digital procede a realizar la firma digital de la factura electrónica recibida 9. El servicio de firma digital registra en la base de datos el token y el resultado de la firma de la factura 10. El servicio de firma digital responde a la aplicación cliente con la factura electrónica firmada
Resultado	La aplicación cliente posee una factura electrónica firmada digitalmente

Nombre	05– La aplicación cliente envía una petición con token de acceso no válido al servicio de firmado
Actores	Aplicación cliente Servicio de firma digital Sistema de control de identidad
Precondiciones	La aplicación cliente debe poseer un token de acceso no válido El servicio de firma digital tiene acceso correcto al sistema de control de identidad
Secuencia principal	1. La aplicación cliente envía una petición al servicio de firma digital en el que incluye el token de acceso que no es válido y la factura electrónica que desea que sea firmada 2. El servicio de firma digital recibe la petición desde la aplicación cliente 3. La validación del token de acceso que realiza el servicio de firma digital resulta no satisfactoria porque el token no es válido 4. El servicio de firma digital responde a la aplicación cliente con un mensaje de error
Resultado	La aplicación cliente recibe un mensaje de error

Nombre	06– La aplicación cliente envía una petición al servicio de firmado utilizando un token que ya ha sido utilizado previamente para fimar una factura electrónica
Actores	Aplicación cliente Servicio de firma digital Sistema de almacenamiento de claves Certificados digitales Sistema de control de identidad
Precondiciones	La aplicación cliente debe poseer un token de acceso válido, pero que ya haya sido usado previamente para firmar una factura electrónica El servicio de firma digital tiene acceso correcto al sistema de control de identidad La aplicación cliente posee una factura electrónica en formato PDF El servicio de firma digital tiene acceso correcto al sistema de almacenamiento de claves El dueño del recurso posee un certificado válido configurado en el sistema de almacenamiento de claves
Secuencia principal	1. La aplicación cliente envía una petición al servicio de firma digital en el que incluye el token de acceso y la factura electrónica que desea que sea firmada 2. El servicio de firma digital recibe la petición desde la aplicación cliente 3. El servicio de firma digital valida la firma del token de acceso utilizando la clave pública del sistema de control de identidad que ha generado y firmado el token 4. El servicio de firma digital intenta validar con la base de datos que el token no haya sido usado previamente para firmar una factura electrónica, pero la validación es incorrecta 5. El servicio de firma digital responde a la aplicación cliente con un mensaje de error
Resultado	La aplicación cliente recibe un mensaje de error

Nombre	07– La aplicación cliente envía una petición al servicio de firmado con token de acceso de un usuario que no tiene permisos para firmar
Actores	Aplicación cliente Servicio de firma digital Sistema de control de identidad
Precondiciones	La aplicación cliente debe poseer un token de acceso válido El servicio de firma digital tiene acceso correcto al sistema de control de identidad La aplicación cliente posee una factura electrónica en formato PDF El servicio de firma digital tiene acceso correcto al sistema de almacenamiento de claves El usuario, dueño del recurso, no tiene asignado el rol correspondiente para poder firmar facturas electrónicas
Secuencia principal	1. La aplicación cliente envía una petición al servicio de firma digital en el que incluye el token de acceso y la factura electrónica que desea que sea firmada 2. El servicio de firma digital recibe la petición desde la aplicación cliente 3. El servicio de firma digital valida la firma del token de acceso

	utilizando la clave pública del sistema de control de identidad que ha generado y firmado el token 4. El servicio de firma digital valida con una base de datos que el token no haya sido usado previamente para firmar una factura electrónica 5. El servicio de firma digital descifra el token de acceso y determina que el rol del usuario no tiene el permiso para realizar firma de facturas electrónicas 6. El servicio de firma digital registra en la base de datos el token y el resultado no satisfactorio de la firma de la factura 7. El servicio de firma digital responde a la aplicación cliente con un mensaje de error
Resultado	La aplicación cliente recibe un mensaje de error

Nombre	08– La aplicación cliente envía una petición con un formato de factura no válido al servicio de firmado
Actores	Aplicación cliente Servicio de firma digital Sistema de control de identidad
Precondiciones	La aplicación cliente debe poseer un token de acceso válido El servicio de firma digital tiene acceso correcto al sistema de control de identidad La aplicación cliente posee una factura electrónica en formato diferente a PDF
Secuencia principal	1. La aplicación cliente envía una petición al servicio de firma digital en el que incluye el token de acceso y la factura electrónica que desea que sea firmada, pero en formato que no sea PDF 2. El servicio de firma digital recibe la petición desde la aplicación cliente 3. El servicio de firma digital valida el token de acceso utilizando la clave pública del sistema de control de identidad que ha generado y firmado el token 4. El servicio de firma digital no valida el formato de la factura electrónica recibida 5. El servicio de firma digital registra en la base de datos el token y el resultado no satisfactorio de la firma de la factura 6. El servicio de firma digital responde a la aplicación cliente con un mensaje de error
Resultado	La aplicación cliente recibe un mensaje de error

Nombre	09– La aplicación cliente envía una petición al servicio de firmado pero el servicio de firmado no tiene acceso correcto al almacén de claves
Actores	Aplicación cliente Servicio de firma digital Sistema de almacenamiento de claves Sistema de control de identidad
Precondiciones	La aplicación cliente debe poseer un token de acceso válido El servicio de firma digital tiene acceso correcto al sistema de control de identidad La aplicación cliente posee una factura electrónica en formato PDF

	El servicio de firma digital tiene acceso correcto al sistema de almacenamiento de claves El dueño del recurso posee un certificado válido configurado en el sistema de almacenamiento de claves
Secuencia principal	1. La aplicación cliente envía una petición al servicio de firma digital en el que incluye el token de acceso y la factura electrónica que desea que sea firmada 2. El servicio de firma digital recibe la petición desde la aplicación cliente 3. El servicio de firma digital valida el token de acceso utilizando la clave pública del sistema de control de identidad que ha generado y firmado el token 4. El servicio de firma digital valida con una base de datos que el token no haya sido usado previamente para firmar una factura electrónica 5. El servicio de firma digital determina que el token está dentro de su tiempo de vida y no ha expirado, si el usuario tiene permisos suficientes para realizar una firma y si el usuario ha otorgado los permisos para firmar facturas electrónicas a la aplicación cliente 6. El servicio de firma digital valida el formato de la factura electrónica recibida 7. El servicio de firma digital no consigue acceder correctamente al almacén de claves 8. El servicio de firma digital registra en la base de datos el token y el resultado no satisfactorio de la firma de la factura 9. El servicio de firma digital responde a la aplicación cliente con un mensaje de error
Resultado	La aplicación cliente recibe un mensaje de error

Nombre	10– La aplicación cliente envía una petición al servicio de firmado pero el servicio de firmado no encuentra claves válidas para el usuario
Actores	Aplicación cliente Servicio de firma digital Sistema de almacenamiento de claves Sistema de control de identidad
Precondiciones	La aplicación cliente debe poseer un token de acceso válido El servicio de firma digital tiene acceso correcto al sistema de control de identidad La aplicación cliente posee una factura electrónica en formato PDF El servicio de firma digital tiene acceso correcto al sistema de almacenamiento de claves El dueño del recurso no posee un certificado válido configurado en el sistema de almacenamiento de claves
Secuencia principal	1. La aplicación cliente envía una petición al servicio de firma digital en el que incluye el token de acceso y la factura electrónica que desea que sea firmada 2. El servicio de firma digital recibe la petición desde la aplicación cliente 3. El servicio de firma digital valida el token de acceso utilizando la clave pública del sistema de control de identidad que ha generado y firmado el token 4. El servicio de firma digital valida con una base de datos que el

	token no haya sido usado previamente para firmar una factura electrónica 5. El servicio de firma digital determina que el token está dentro de su tiempo de vida y no ha expirado, si el usuario tiene permisos suficientes para realizar una firma y si el usuario ha otorgado los permisos para firmar facturas electrónicas a la aplicación cliente 6. El servicio de firma digital valida el formato de la factura electrónica recibida 7. El servicio de firma digital accede al almacén de claves 8. El servicio de firma digital no consigue tomar del almacén de claves las que corresponden para el usuario que realiza la petición 9. El servicio de firma digital registra en la base de datos el token y el resultado no satisfactorio de la firma de la factura 10. El servicio de firma digital responde a la aplicación cliente con un mensaje de error
Resultado	La aplicación cliente recibe un mensaje de error

A continuación, se presenta el diagrama de secuencia del caso práctico a implementar, considerando que se sigue el ciclo completo, sin ningún error y con resultado satisfactorio.

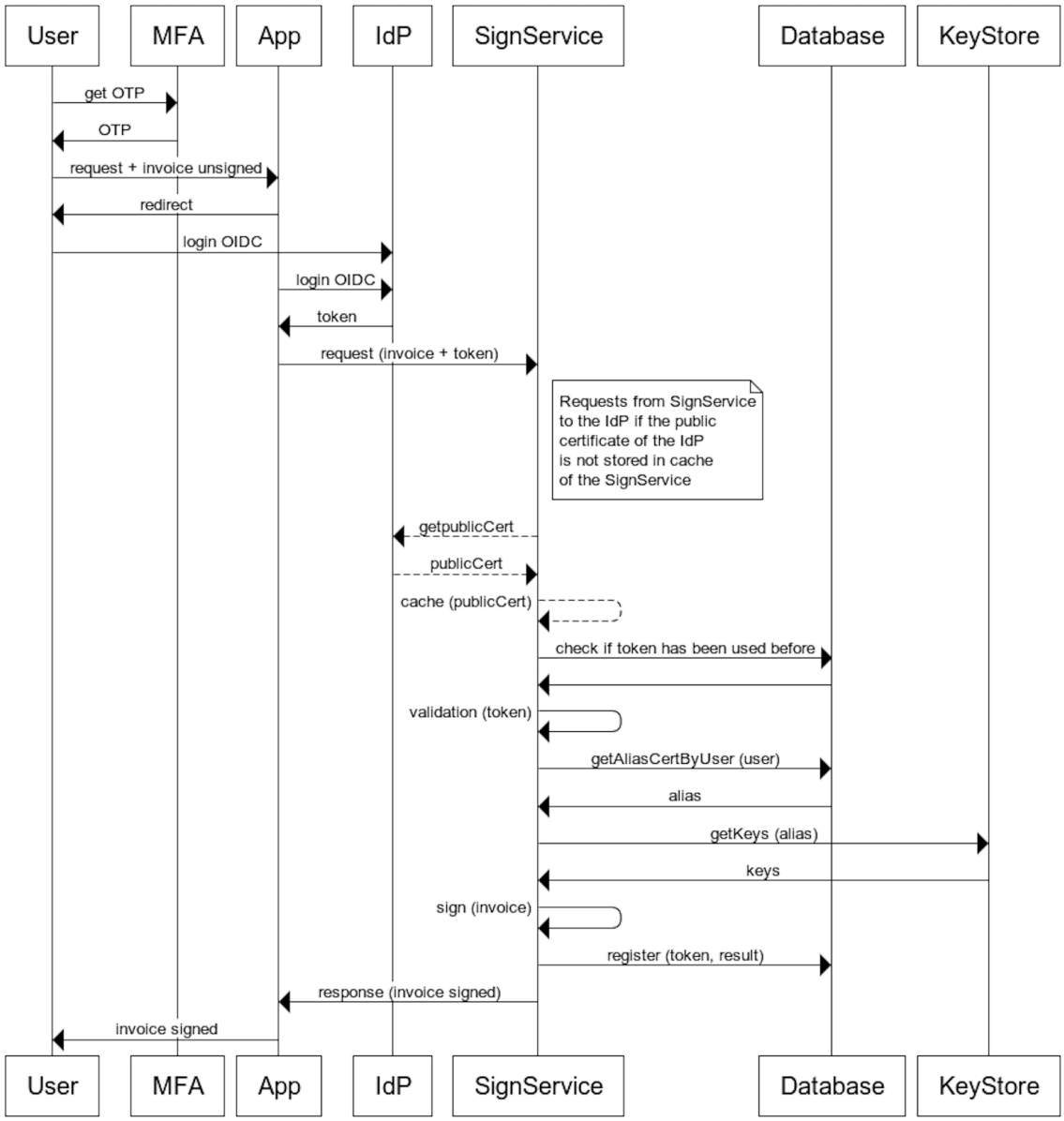

Figura 3.1: diagrama de secuencia del flujo completo

3.2.4. Selección del stack tecnológico

Según el alcance, la identificación de los componentes y los casos de uso, se realiza una selección del stack tecnológico a utilizar en la implementación del caso práctico del TFM. En la siguiente tabla se muestra la tecnología seleccionada para cada uno de los componentes identificados en el servicio y la justificación de la elección de cada uno de ellos [65] [72] [80] [99] [100] [101] [102].

Componente	**Tecnología**	**Justificación**
Sistema de control de identidad y acceso	Keycloak	Cubre las necesidades del alcance definido Gratuito Amplia documentación disponible
Sistema MFA	Google Authenticator	Cubre las necesidades del alcance definido

	(autenticación OTP)	Gratuito Amplia documentación disponible Respaldado por Google
Servicio PKI	OpenSSL	Cubre las necesidades del alcance definido Gratuito Amplia documentación disponible Experiencia previa con este software
Certificados digitales	OpenSSL	Cubre las necesidades del alcance definido Gratuito Amplia documentación disponible Experiencia previa con este software
Almacén de claves	Java KeyStore	Cubre las necesidades del alcance definido Gratuito Amplia documentación disponible Experiencia previa con este software
Servicio de firma digital	Java Spring Boot	Cubre las necesidades del alcance definido Gratuito Amplia documentación disponible Apoyado por una gran comunidad Experiencia previa con este software
Base de datos	H2	Cubre las necesidades del alcance definido para un entorno de desarrollo Gratuito Amplia documentación disponible Apoyado por una gran comunidad Experiencia previa con este software
Aplicación cliente	Postman	Cubre las necesidades del alcance definido, considerando que el desarrollo de una aplicación cliente queda fuera del alcance del presente TFM Gratuito Amplia documentación disponible Apoyado por una gran comunidad Experiencia previa con este software

Tabla 3.1. Stack tecnológico

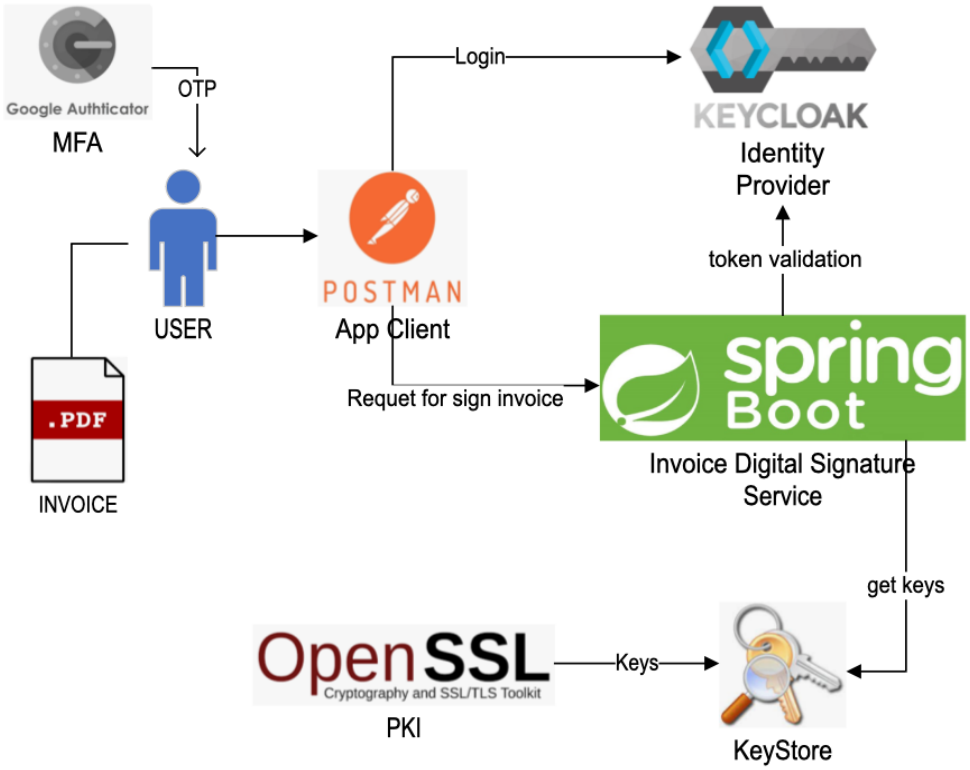

Figura 3.2: Diagrama de componentes del caso práctico a implementar

3.3. Desarrollo e implementación

3.3.1. Sistema de control de acceso con MFA

La implantación de un sistema de control de identidad y acceso con autenticación multi-factor (MFA) es una necesidad que cada vez aumenta más de importancia para dotar de seguridad en el acceso a los servicios. Keycloak es una solución de gestión de identidades de código abierto que permite la implantación de un sistema de control de identidades y acceso con MFA de manera sencilla. Por otro lado, Google Authenticator es una aplicación de autenticación MFA que se puede integrar con Keycloak y así conseguir añadir una capa adicional de seguridad en el acceso a las aplicaciones. En el presente apartado se muestra la implantación del sistema de control de identidad y acceso con MFA para el caso práctico del TFM.

Dentro del contexto de Keycloak, algunos de los conceptos fundamentales son los siguientes:

- Realm. Un Realm es un espacio aislado dentro de Keycloak en el que se definen los usuarios, clientes, roles, grupos y las políticas asociadas a éstos. Un Realm es lo que en otros sistemas se denomina como dominio.
- Users. Los usuarios (o, en inglés, users) son las entidades que pueden iniciar sesión en el sistema. En el contexto de OAuth 2.0 corresponden con los propietarios de los recursos. Los usuarios pueden tener aplicadas diferentes configuraciones, entre las más destacables estarían la asignación a grupos y roles, credenciales y los atributos. Los atributos

de usuario son datos del propio usuario que se almacenan en su perfil de usuario y describen su nombre completo, correo electrónico, número de teléfono, etc. Además, también se pueden añadir atributos adicionales que permiten su uso en las políticas de acceso y en flujos de autenticación.

- Roles. Los roles son etiquetas que se asignan a un usuario o grupo de usuarios y se utilizan dentro de las aplicaciones para controlar el acceso de los usuarios a los diferentes recursos. También hay roles para aplicaciones cliente y roles generales en los Realm.
- Grupos de usuario. En Keycloak, al igual que en otros sistemas, los grupos de usuario son colecciones de usuarios que se utilizan para administrar y organizar usuarios en función de sus necesidades de acceso y atributos. Se pueden asignar roles a grupos y usuarios a grupos, de este modo, los usuarios heredarían los roles a través de estar dentro de determinados grupos.
- Clients. Un cliente (o, en inglés, client) es una aplicación o servicio que desea acceder a un recurso protegido en nombre de un usuario. Un cliente utiliza Keycloak para autenticar y autorizar a los usuarios. En el contexto de OAuth 2.0 corresponde con la figura homónima de aplicación cliente. En Keycloak se pueden aplicar diferentes configuraciones a los clientes tales como URLs de redirección válidas, credenciales de autenticación, flujos de autorización permitidos, roles, client scopes, etc.
- Client scopes. Un client scope es el conjunto de atributos y permisos que se le pueden asignar a un cliente. Los client scopes sirven para definir sobre qué recursos de los usuarios pueden acceder los clientes. De este modo, si un cliente tiene configurado en su scope atributos de usuario tales como nombre o email, podrá solicitar a un usuario el acceso a éstos durante el proceso de autorización.
- Flujo de autenticación. Un flujo de autenticación es la secuencia de pasos de un usuario para conseguir autenticarse en una aplicación. En Keycloak se definen diferentes flujos de autenticación y pueden asociarse a los clientes para que puedan usar uno o más tipos de flujos para los procesos de autenticación.
- Sessions. Tras el login de un usuario o cliente en Keycloak, se almacena su sesión en Keycloak para que pueda ser administrada, pudiéndose revocar si fuera necesario.
- Identity Providers. Un proveedor de identidad (o, en inglés, identity provider) es un sistema que permite autenticar y autorizar a usuarios y está ligado al concepto de federación de identidad. La federación de identidad permite que los usuarios puedan autenticarse en una aplicación utilizando sus credenciales de inicio de sesión de otro proveedor de identidad. En Keycloak se admite la federación de identidad pudiendo utilizar un proveedor de identidad externo como Facebook, GitHub, Google, Linkedin, Twitter, etc.

Para realizar la implementación de Keycloak se realizar los siguientes pasos:

1. Preparación del entorno: aseguramiento de que el sistema anfitrión posea JVM y generación de certificado para permitir que Keycloak atienda las peticiones bajo protocolo seguro HTTPS

2. Obtención del software Keycloak e inicio del servicio Keycloak con escucha en HTTPS
3. Configuración de Realm, roles, usuarios, client scopes y clients dentro de Keycloak.
4. Realización de pruebas básicas de funcionamiento

3.3.1.1. Preparación del entorno

Keycloak es una aplicación de servidor que se ejecuta en una JVM (Java Virtual Machine). Por tanto, puede ser desplegado en servidores con diferentes características y sistemas operativos. Para el caso práctico del presente TFM se despliega Keycloak en un sistema con las siguientes características: procesador M1 de Apple, 16 GB de RAM y sistema operativo mac OS Ventura 13.2.1.

Se realiza un aseguramiento sobre la JVM del equipo anfitrión con los siguientes comandos desde una terminal:

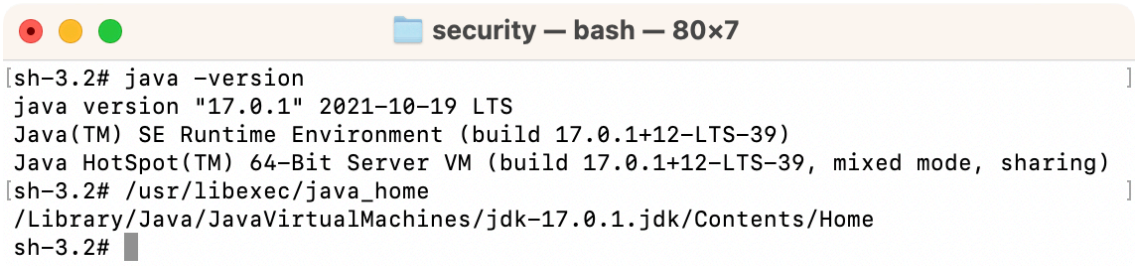

Figura 3.3: Comprobación JVM del equipo anfitrión

Se desea que el sistema Keycloak pueda atender peticiones bajo protocolo seguro HTTPS. Para poder hacer esto es necesario disponer de un certificado. En un entorno productivo se utilizaría un certificado generado por una entidad reconocida. Sin embargo, para el despliegue a realizar en el presente TFM se aplicará un certificado autofirmado.

Los diferentes componentes a utilizar en el caso práctico del TFM se ejecutarán bajo el mismo sistema anfitrión ya descrito por lo que los servicios se apuntarán entre sí a través de “localhost”. Debido a ello es necesario que el certificado autofirmado a generar posea una CN con el nombre “localhost” y que su clave pública esté instalada en el almacén de claves “cacerts” de la JVM.

A continuación, se muestra el procedimiento de generación del certificado para que Keycloak atienda peticiones HTTPS:

security — bash — 80×24

```
sh-3.2# openssl req -newkey rsa:2048 -nodes \
>   -keyout keycloak-server.key.pem -x509 -days 3650 -out keycloak-server.crt.pe
m
Generating a 2048 bit RSA private key
.....................................................................
.....................................................+++++
......+++++
writing new private key to 'keycloak-server.key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:ES
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:TFM
Organizational Unit Name (eg, section) []:
Common Name (eg, fully qualified host name) []:localhost
Email Address []:
sh-3.2#
```

Figura 3.4: Generación de certificado autofirmado para Keycloak

A continuación, se importa la clave pública del certificado autofirmado en el almacén de certificados de la JVM y se verifica que el contenido del certificado corresponde con los datos introducidos en su proceso de generación.

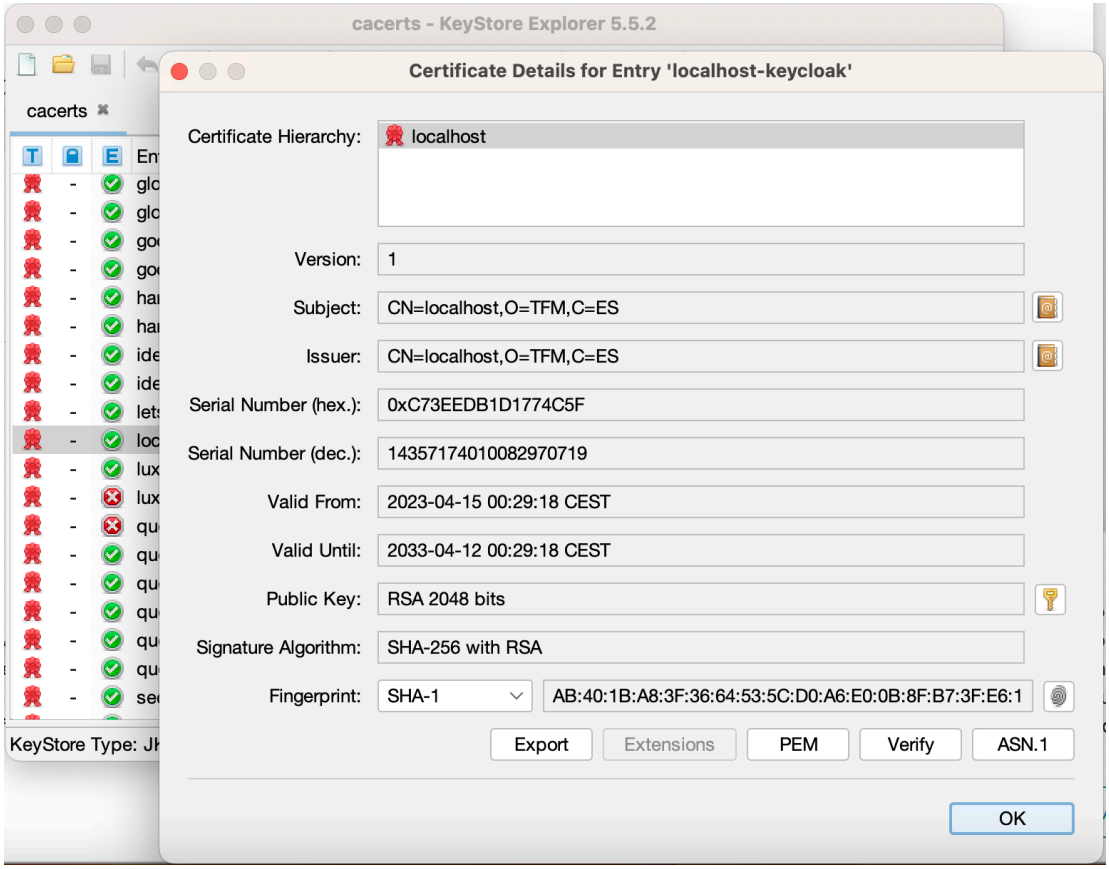

Figura 3.5: Detalle de la clave pública del certificado autofirmado para Keycloak

3.3.1.2. Inicio del servicio Keycloak

La obtención de la aplicación Keycloak se consigue a través de su descarga del sitio oficial de Keycloak. La versión que se descarga es 21.0.2. Al tratarse de un servicio que corre bajo JVM, su instalación consiste en descomprimir el fichero descargado y ubicarlo en el lugar deseado con los permisos de ejecución correspondientes.

El inicio del servicio se realiza por comando desde la terminal. Para el caso práctico del TFM se realiza un inicio del servicio en modo desarrollo para simplificar su arranque. Además, se le indican por argumentos de entrada las claves a usar para atender peticiones por HTTPS. En un entorno productivo se aplicarían configuraciones específicas tales como arrancar el servicio en modo producción, arranque del servicio con un usuario bajo política de privilegios mínimos, configurar un nombre del servidor, restringir el acceso por HTTP para que solo sea accesible por HTTPS, configurar nivel de trazas de logs, etc.

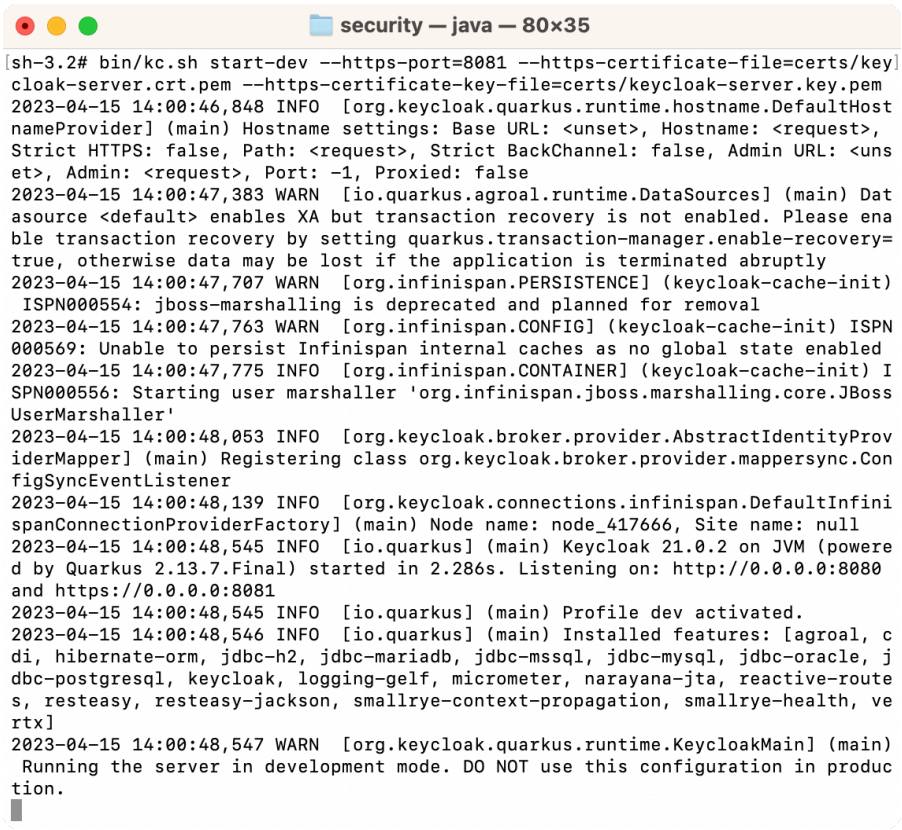

Figura 3.6: Arranque por comando del servicio Keycloak desde terminal

3.3.1.3. Configuración de componentes en Keycloak

Una vez arrancado el servicio Keycloak es necesario aplicar una serie de configuraciones desde la consola de administrador. Para el caso práctico a implementar, se aplican configuraciones en los siguientes elementos:

- Realm
- Roles

- Usuarios
- Client scopes
- Clientes

Se crea un nuevo Realm (o, en español, reino) para generar un espacio aislado para los componentes dedicados al caso práctico del presente TFM. El nombre elegido para el nuevo Realm es “digsign4einvoices”. Para el propósito del TFM no se aplica configuración específica en el Realm.

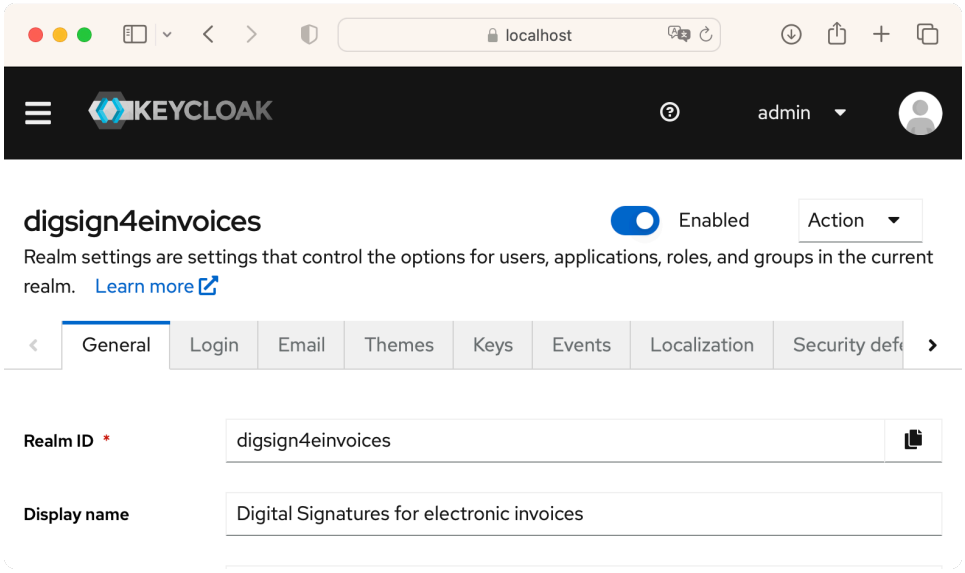

Figura 3.7: Realm dedicado para el caso práctico a implementar

Se crea un nuevo rol de usuario denominado “SIGNER_ROLE” y se asociará a este nuevo rol a los usuarios que deban tener permitida la capacidad de firmar facturas electrónicas. El servidor de recursos comprobará si un usuario posee este rol para permitirle o no el firmado de facturas electrónicas.

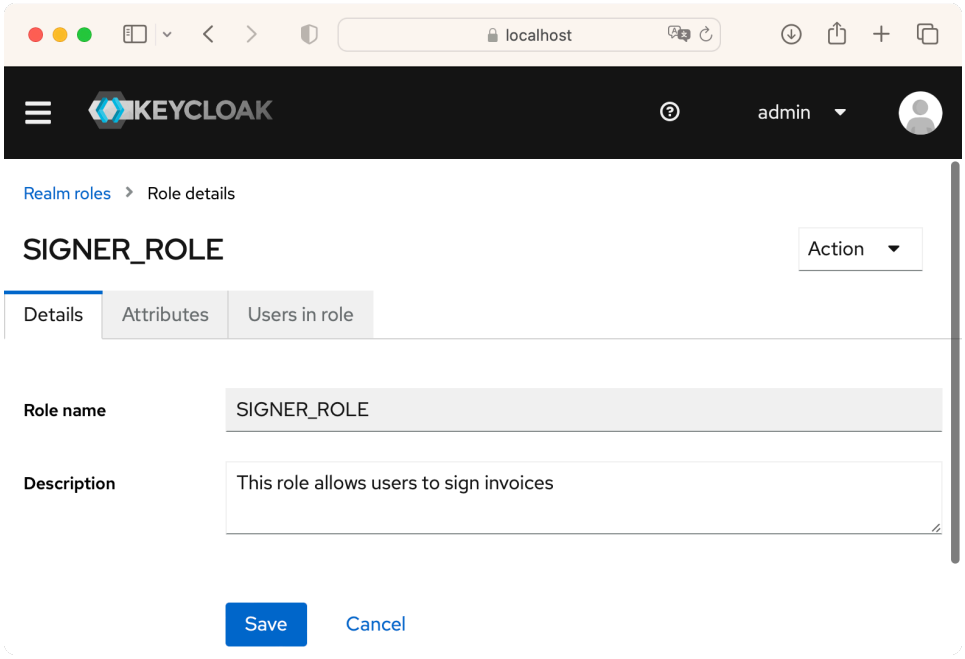

Figura 3.8: Rol SIGNER_ROLE

Después de haberse creado el Realm y el rol de usuario, se procede a la configuración de usuarios. Para el caso práctico del TFM se configuran los usuarios con acceso por OTP. De este modo se aplica MFA y se aumenta el nivel de securización. A los usuarios que deban tener permitida la capacidad de firmar facturas electrónicas se les asignaráa el rol “SIGNER_ROLE”.

Para ilustrar el caso práctico del TFM se crean varios usuarios con diferentes configuraciones:

- Usuario “fgonzalezhernandez”
  - Usuario con credenciales por Password y doble factor de autenticación por OTP
  - Rol asignado: “SIGNER_ROLE”
- Usuario “user-no-role”
  - Usuario con credenciales por Password y doble factor de autenticación por OTP
  - Sin rol asignado
- Usuario “user-no-cert”
  - Usuario con credenciales por Password y doble factor de autenticación por OTP
  - Rol asignado: “SIGNER_ROLE”

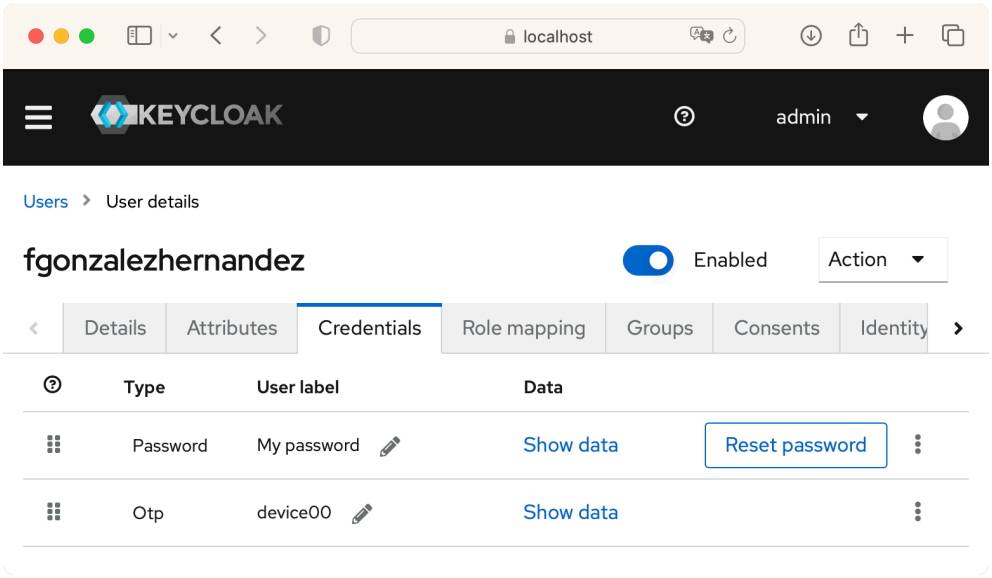

Figura 3.9: Configuración de credenciales del usuario “fgonzalezhernandez”

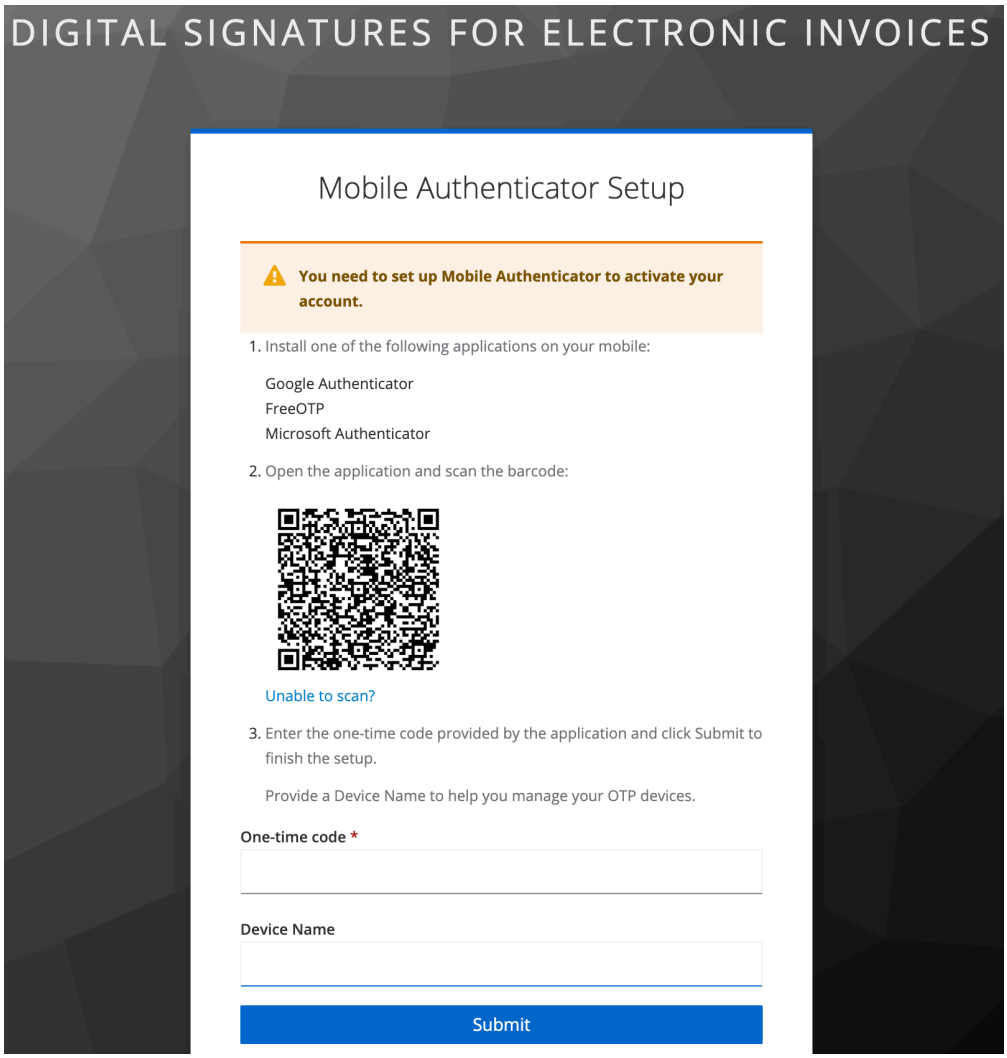

Figura 3.10: Proceso de alta de autenticación MFA

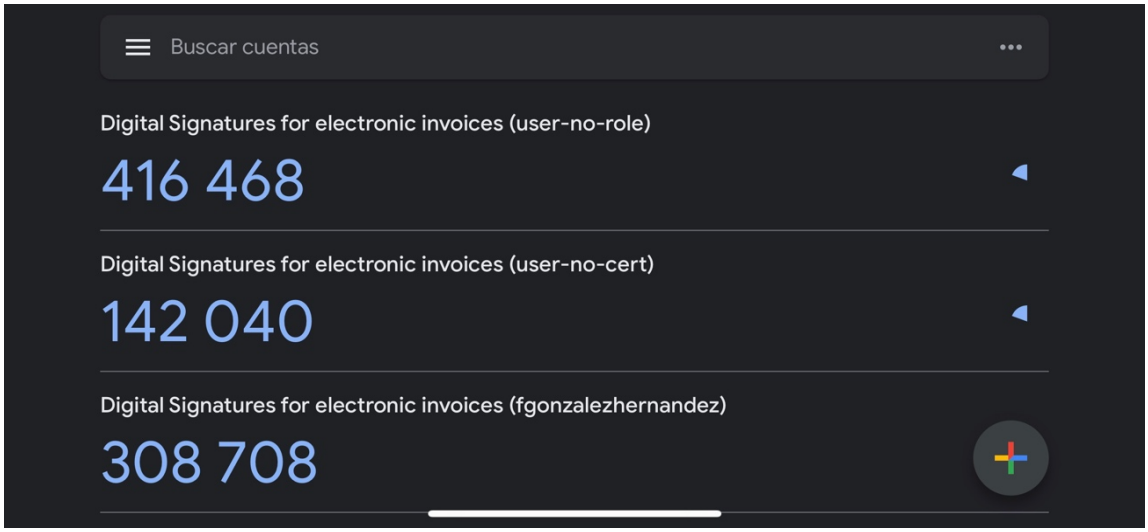

Figura 3.11: Cuentas de usuario en la aplicación Google Authenticator

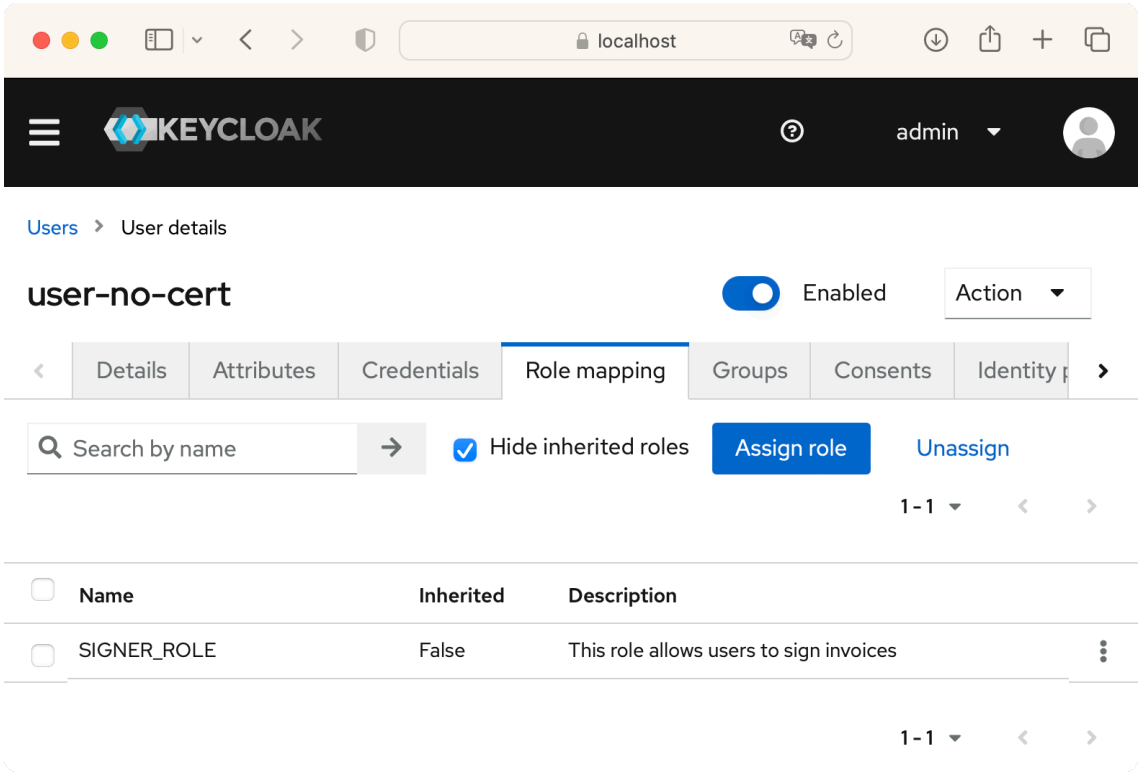

Figura 3.13: Roles asignados al usuario “user-no-cert”

Se crea un client scope denominado “invoices-scope”. Este client scope tendrá mapeado los roles a los que pertenezca un usuario. El objetivo es que estos datos puedan ser provisionados al servidor de recursos para que pueda utilizarlos para determinar si un usuario puede o no firmar facturas electrónicas. El client scope se asocia al cliente de aplicación para que cuando éste solicite a Keycloak el token de acceso, este token permita al proveedor de recursos enviar una petición a Keycloak para obtener estos datos.

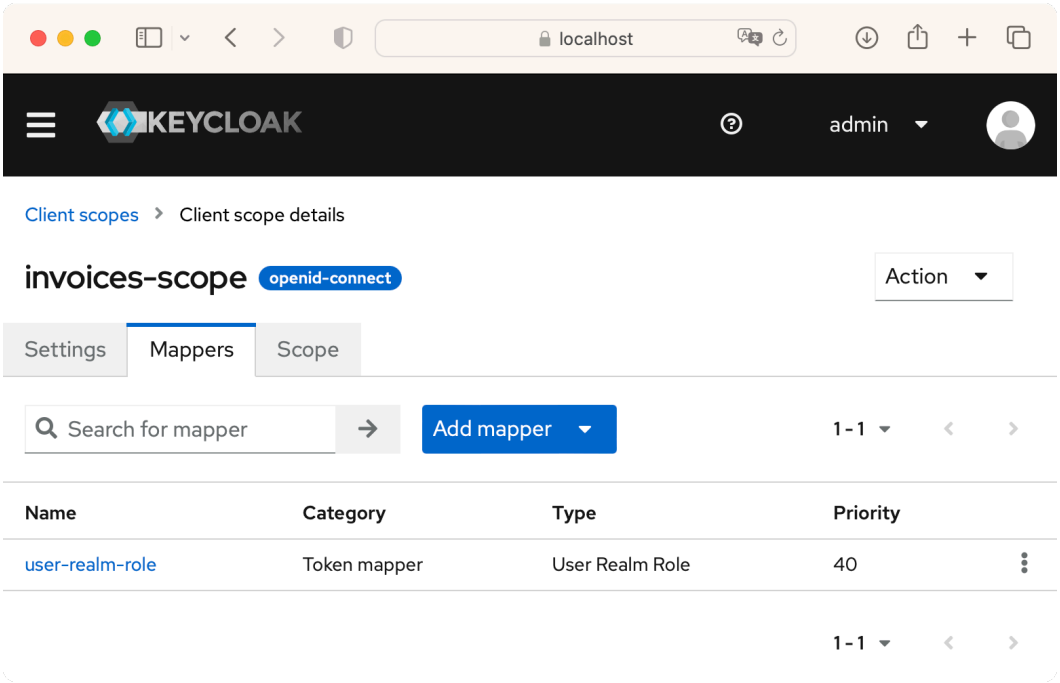

Figura 3.14: Mapeo de atributos en el client scope “invoices-scope”

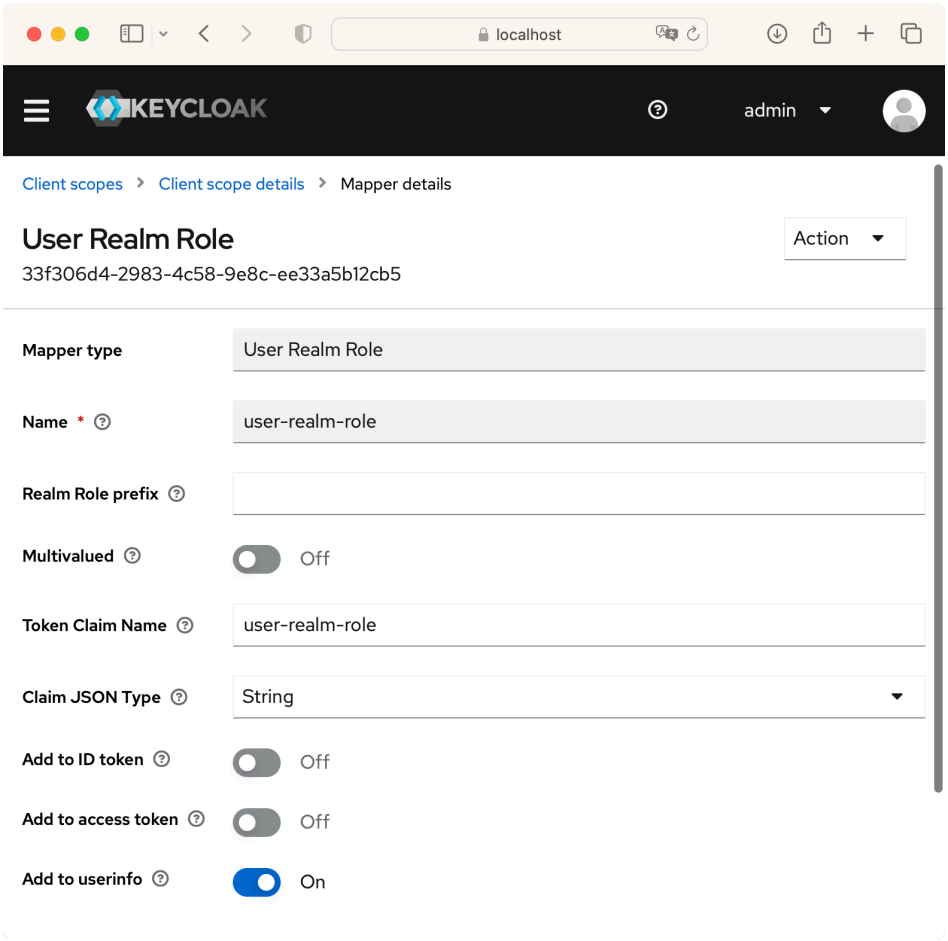

Figura 3.16: Configuración del mapeo en “invoices-scope” para conseguir los roles de usuario

Se crea la aplicación cliente con el Client ID “app-client”. A esta aplicación cliente se le configura el flujo “Standard Flow” que corresponde con el flujo “Authorization Code Flow” de OAuth 2.0. No se le configuran más flujos debido a que éste es el flujo más completo y seguro y el resto de flujos no se consideran necesarios para el caso de implementación del TFM. El cliente “app-client” posee una clave secreta, que será necesaria para completar el flujo “Standard Flow”. Además, se añade al cliente el client scope “invoice-scope” para que puedan solicitar la información requerida para el proceso de firmado.

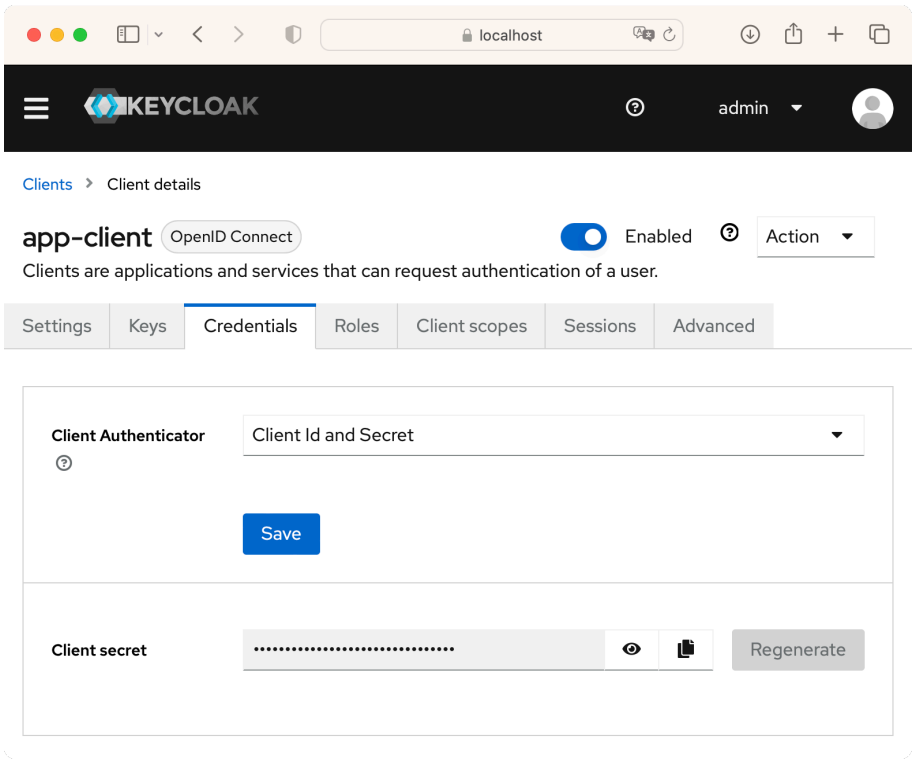

Figura 3.17: Detalle de configuración "credentials" del cliente "app-client"

3.3.1.4. Pruebas básicas de Keycloak

En el presente apartado se realiza una prueba básica de funcionamiento de Keycloak. Como queda fuera del alcance del presente TFM la implementación de una aplicación cliente, se utilizará Postman para la simulación del comportamiento de una aplicación cliente. La versión de Postman utilizada para la realización de las pruebas es 10.12.13.

Debido a que la conexión con el servicio Keycloak se realiza a través de HTTPS y a que para ello se ha utilizado un certificado autofirmado, es necesario especificar en la configuración de Postman que no se aplique verificación de los certificados SSL. En caso contrario, desde Postman surgiría un error de validación de certificados y no se podrían emitir peticiones hacia el servicio Keycloak implementado.

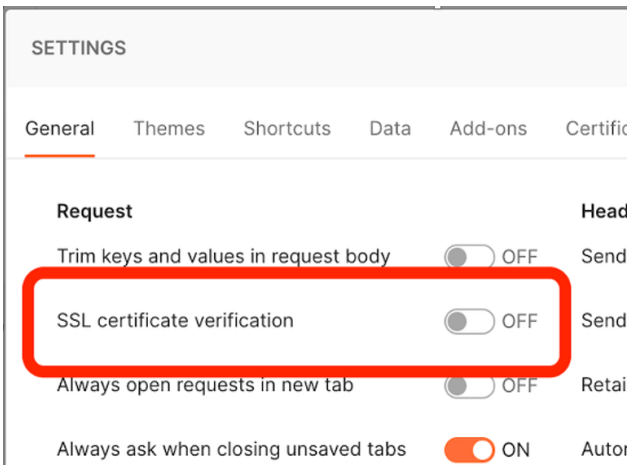

Figura 3.18: Configuración de Postman para la comunicación con certificados autofirmados

En Postman se puede utilizar un recurso para provocar el flujo “Authorization Code Flow” de una manera sencilla y así conseguir obtener el token acceso. Este método consiste en generar un nuevo “request” en cuyo apartado “Authorization” seleccionar “Type = OAuth 2.0” y rellenar los valores del formulario que aparece a continuación con los datos correspondientes:

- Token Name: este campo no es necesario rellenarlo y sirve para identificar el token que se genere al ejecutar el flujo.
- Grant Type: este campo sirve para determinar qué tipo de flujo OAuth 2.0 se va a seguir. Para el caso del presente TFM el flujo deseado es “Authorization Code”.
- Callback URL: este campo sirve para determinar qué URL de callback utilizará el cliente para recibir el flujo desde Keycloak durante el proceso de autorización. Para el caso del presente TFM se indica el siguiente valor: “http://localhost:8082/test-callback”
- Auth URL: este campo sirve para obtener el código de autorización. Para el caso del presente TFM y el servicio de Keycloak desplegado, el valor es el siguiente: “https://localhost:8081/realms/digsign4einvoices/protocol/openid-connect/auth”
- Access Token URL: este campo sirve para obtener el código de acceso a partir del código de autorización. Para el caso del presente TFM y el servicio de Keycloak desplegado, el valor es el siguiente: “https://localhost:8081/realms/digsign4einvoices/protocol/openid-connect/token”
- Client ID: este campo sirve para determinar el identificador de la aplicación cliente. Para el caso del presente TFM se indica el siguiente valor: “app-client”
- Client Secret: este campo sirve para indicar el secreto de la aplicación cliente. El secreto se obtiene a partir de la consola de administración de Keycloak y solo debe ser conocido por el cliente.
- Scope: este campo sirve para determinar los scopes a los que el cliente quiere tener acceso. Para el caso del presente TFM se indica que se quiere tener el siguiente valor: “openid invoices-scope”. Es decir, se indica el valor “openid” para seguir el flujo OpenId Connect y el valor “incoices-scope” para tener acceso a los atributos específicos necesarios para el servicio a implementar.
- State: este campo sirve para mejorar el proceso de seguridad del flujo de autenticación. Se puede introducir un código cualquiera.
- ClientAuthentication: este campo sirve para determinar la forma en la que enviar las credenciales durante el flujo, si se envían en las cabeceras de la petición o en el cuerpo de la petición.

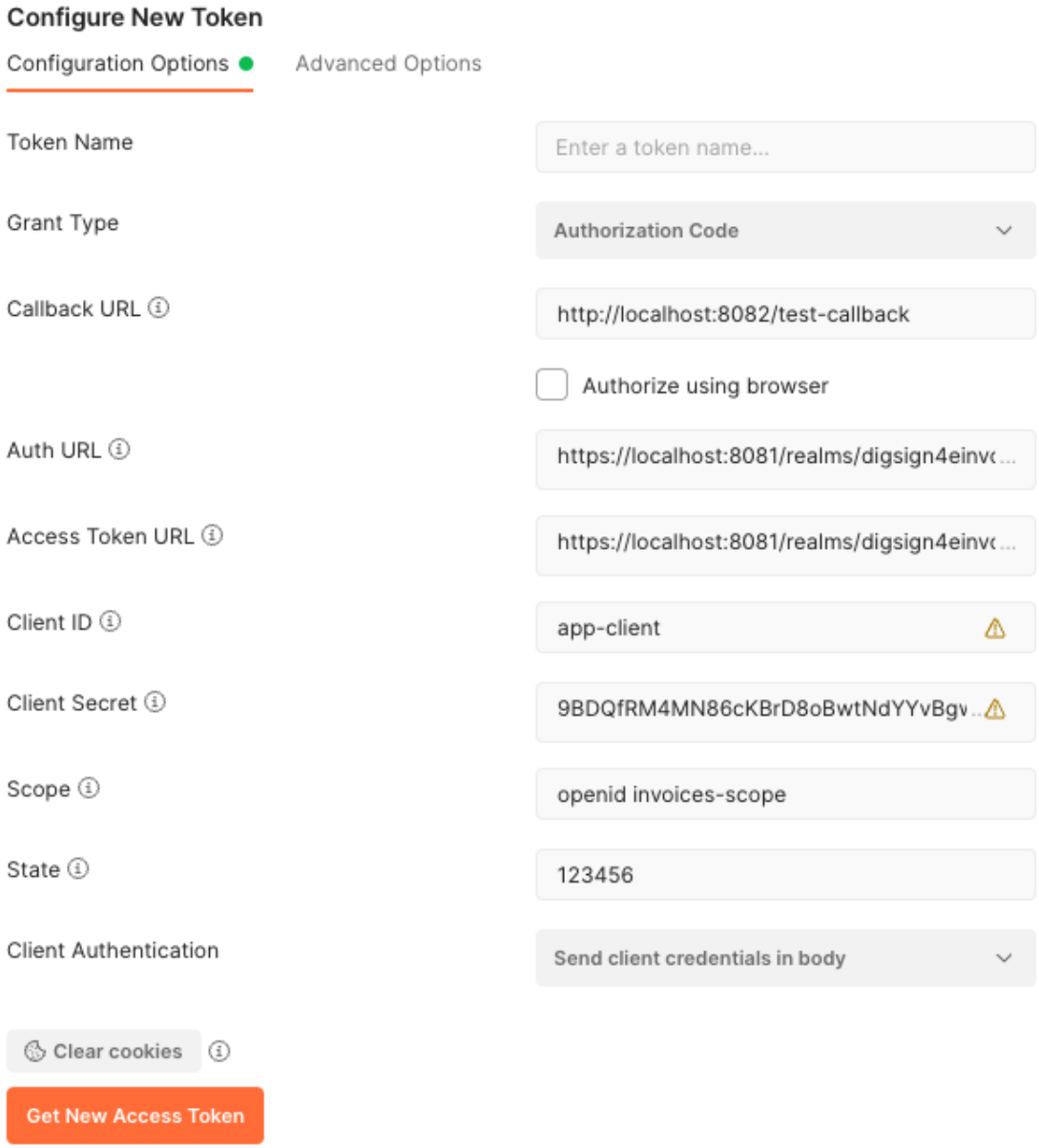

Figura 3.19: Configuración de Postman para generar flujo “Authorization Code”

A continuación, se realiza una prueba para solicitar un token de acceso a Keycloak utilizando Postman con la configuración citada anteriormente. Se inicia el flujo pulsando el botón “Get New Acess Token”, posteriormente el usuario (dueño del recurso) debe introducir sus credenciales y la contraseña OTP. De este modo se conseguirá en Postman el token de acceso.

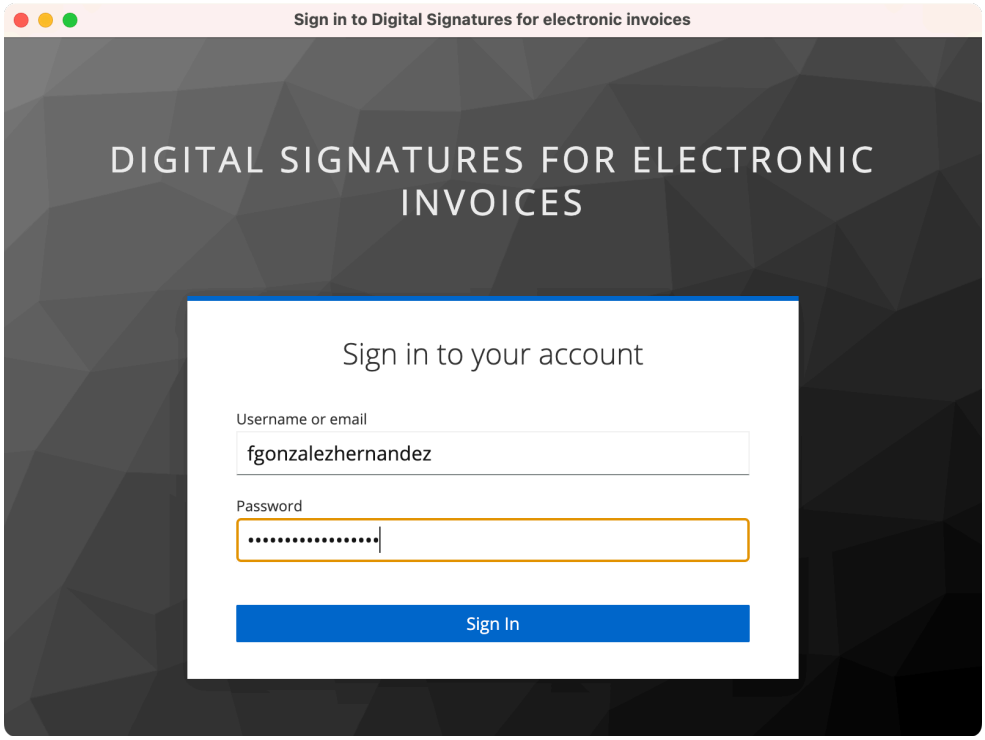

Figura 3.20: Formulario de login de usuario

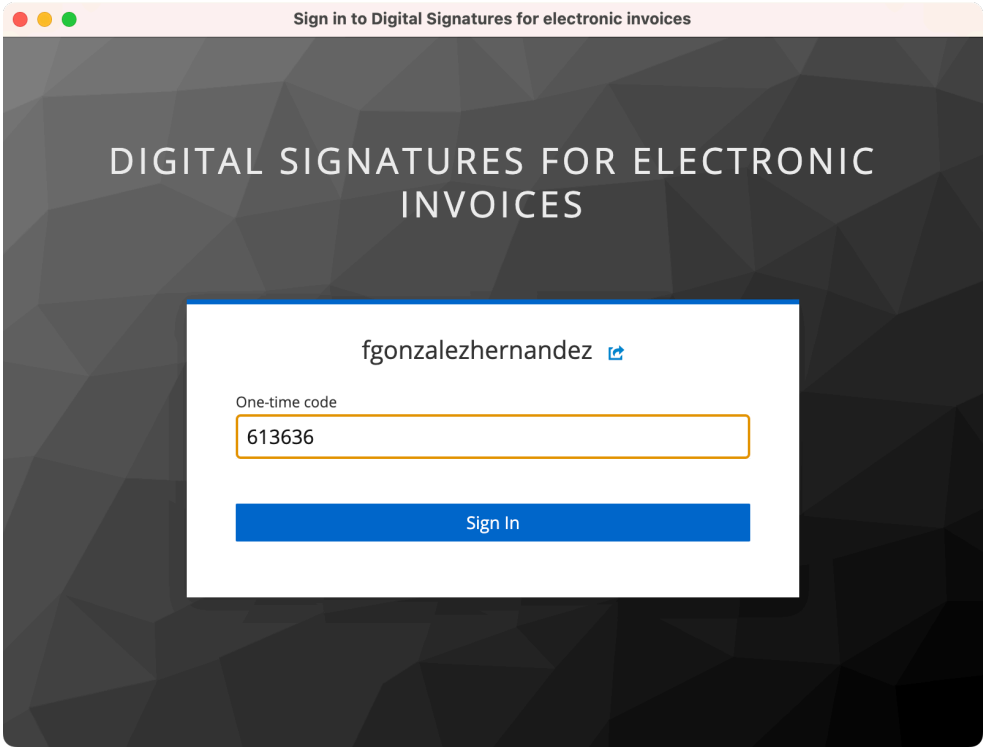

Figura 3.21: Formulacio para introducir código OTP

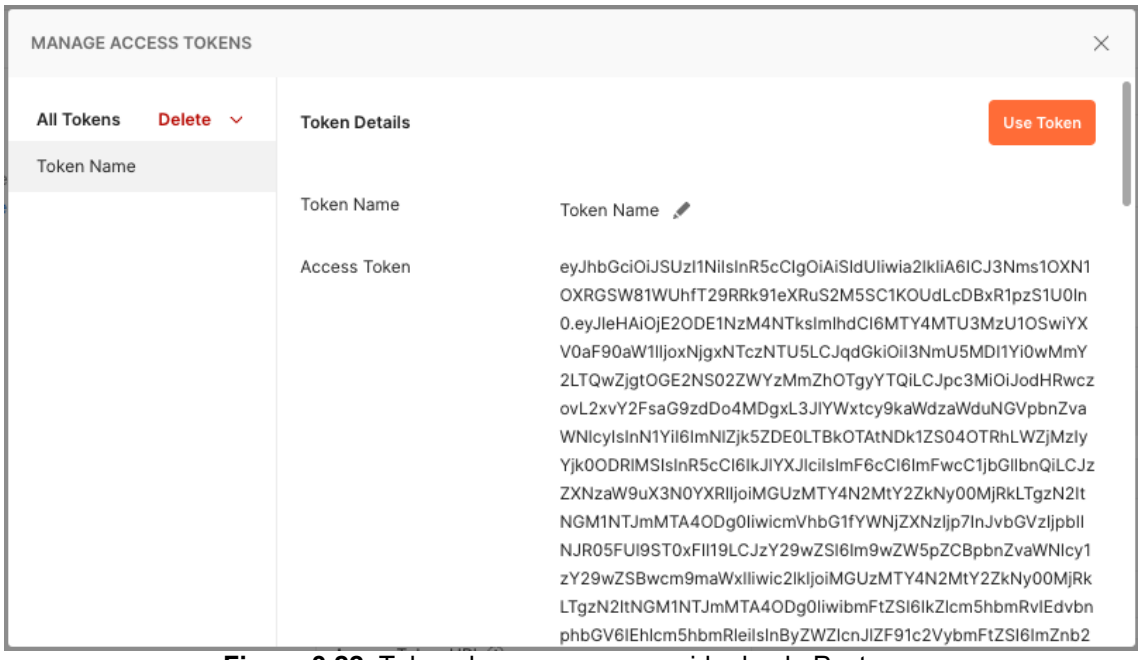

Figura 3.22: Token de acceso conseguido desde Postman

Del ejemplo ejecutado se consiguen los siguientes datos:

Key	Value
Access Token	eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJ3Nms1OXN1OXRGSW81WUhfT29RRk91eXRuS2M5SC1KOUdLcDBxR1pzS1U0In0.eyJleHAiOjE2ODE2Mjk0MDMsImlhdCI6MTY4MTYyOTEwMywiYXV0aF90aW1lIjoxNjgxNjI5MTAyLCJqdGkiOiJmMDUxZWRiMy0yNDY0LTRkY2UtOWRiNS0zMTY1NzQ5ZmViMTEiLCJpc3MiOiJodHRwczovL2xvY2FsaG9zdDo4MDgxL3JlYWxtcy9kaWdzaWduGVpbnZvaWNlcyIsInN1YiI6ImNlZjk5ZDE0LTBkOTAtNDk1ZS04OTRhLWZjMzIyYjk0ODRlMSIsInR5cCI6IkJIYXJlciIsImF6cCI6ImFwcC1jbGllbnQiLCJzZXNzaW9uX3N0YXRlIjoiZTkwYmJlNDEtM2M4OC00YmZiLWE4YzItOGIxMmM4ODhmOTBlliwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbIlNJR05FUl9ST0xFIl19LCJzY29wZSI6Im9wZW5pZCBpbnZvaWNlcy1zY29wZSBwcm9maWxlIiwic2lkIjoiZTkwYmJlNDEtM2M4OC00YmZiLWE4YzItOGIxMmM4ODhmOTBlliwibmFtZSI6IkZlcm5hbmRvIEdvbnphbGV6IEhlcm5hbmRleiIsInByZWZlcnJlZF91c2VybmFtZSI6ImZnb256YWxlemhlcm5hbmRleiIsImdpdmVuX25hbWUiOiJGZXJuYW5kbyIsImZhbWlseV9uYW1lIjoiR29uemFsZXogSGVybmFuZGV6In0.eWb8gxFtju5y_2yXXzJ4OxYElhTQp3FAvbQ8LRw1S8sm4Nt0ZTIBqsuOlCJLSSnP4i61V8PD-BxGtZOGzR3Ks8sbUGaX5FqTHmYq-kUIR9obpaid7-9_Z9QV_wB0mKe5P5dInZk2qsscwFVbkzvV_fIWEiFwQc0KoyYIftfhl8d61xd0uAvwpUHuBXvGsJnOf3xSOfW2eAxpjO__STndYUurR7lNZqafKv_LRuNqGzUqd_D4Ylu6sD4i3K7TsCdwgHLWZeQHJlTsAhLbpZ4fU_hUq47JwKFHRHDyZ0v7YCLzqXSp3Wie1yneoL7hrclY68w8pMbk4eNIus7k4RvLTA
Token Type	Bearer
expires_in	300
refresh_expires_in	1800
refresh_token	eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI0MTc0MGEyZS0xNTFhLTRhZTgtYTk5NS0wMTRiOGUyNWEwYmYifQ.eyJleHAiOjE2ODE2MzA5MDMsImlhdCI6MTY4MTYyOTEwMywianRpIjoiMDMyOGQ1ODktZWQwNy00YjZlLThmM2YtODE1MjZiZDMwZTIwIiwiaXNzIjoiaHR0cHM6Ly9sb2NhbGhvc3Q6ODA4MS9yZWFsbXMvZGlnc2lnbjRlaW52b2ljZXMiLCJhdWQiOiJodHRwczovL2xvY2FsaG9zdDo4MDgxL3JlYWxtcy9kaWdzaWduGVpbnZvaWNlcyIsInN1YiI6ImNlZjk5ZDE0LTBkOTAtNDk1ZS04OTRhLWZjMzIyYjk0ODRlMSIsInR5cCI6IlJlZnJlc2giLCJhenAiOiJhcHAtY2xpZW50Iiwic2Vzc2lvbl9zdGF0ZSI6ImU5MGJiZTQxLTNjODgtNGJmYi1hOGMyLThiMTJjODg4ZjkwZSIsInNjb3BlIjoib3BlbmlkIGludm9pY2VzLXNjb3BlIHByb2ZpbGUiLCJzaWQiOiJlOTBiYmU0MS0zYzg4LTRiZmItYThjMi04YjEyYzg4OGY5MGUifQ.H9B3z6BNuQu3xyPPxUFQETUdnviSwzQahtUWf-mg7r8
id_token	eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJ3Nms1OXN1OXRGSW81WUhfT29RRk91eXRuS2M5SC1KOUdLcDBxR1pzS1U0In0.eyJleHAiOjE2ODE2Mjk0MDMsImlhdCI6MTY4MTYyOTEwMywiYXV0aF90aW1lIjoxNjgxNjI5MTAyLCJqdGkiOiI5MGVlMjBjMy0zN2U0LTRjZTgtODRhNC0zMmE2YWEwMTU5NzAiLCJpc3MiOiJodHRwczovL2xvY2FsaG9zdDo4MDgxL3JlYWxtcy9kaWdzaWduGVpbnZvaWNlcyIsImF1ZCI6ImFwcC1jbGllbnQiLCJzdWIiOiJjZWY5OWQxNC0wZDkwLTQ5NWUtODk0YS1mYzMyMmI5NDg0ZTEiLCJ0eXAiOiJJRCIsImF6cCI6ImFwcC1jbGllbnQiLCJzZXNzaW9uX3N0YXRlIjoiZTkwYmJlNDEtM2M4OC00YmZiLWE4YzItOGIxMmM4ODhmOTBlliwiYXRfaGFzaCI6InpsYTVxNExPTnRQN3JjZ1ZQZk15Y3ciLCJzaWQiOiJlOTBiYmU0MS0zYzg4LTRiZmItYThjMi04YjEyYzg4OGY5MGUiLCJuYW1lIjoiRmVybmFuZG8gR29uemFsZXogSGVybmFuZGV6IiwicHJlZmVycmVkX3VzZXJuYW1lIjoiZmdvbnphbGV6aGVybmFuZGV6IiwiZ2l2ZW5fbmFtZSI6IkZlcm5hbmRvIiwiZmFtaWx5X25hbWUiOiJHb256YWxleiBlZXJuYW5kZXoifQ.HfX9xR6Mr

	PnrT9pQuGOAt9Jq3-TUsGXPTgbRdiT-aSz80ERTAHavXaim1vqevyeochX14aaYyq7utqjy-n739ozeW74NMHJpQI-A5e1SZUKKUaRaILZCSdL9ebKrRTY65-AJSTR7dH2-h-3eB3-puxbOPMYLAziCJZVbg-slpjy4t2EWgFOfjyt_rS14Z7TDoS2VLvZo-09ve2c5pfo1-3jkqPLICZx7R4YJwwxuKT3MVMqn68emd5gSEuZ1IBZVcmEFZbT9H6irjNy8tEnPVlnGXwJ9zDYEMlGVxE23ECe_n_pXi0vImD4LDx8f4otCAr8gvqvQ7vjsUzdwMXmZkQ	
not-before-policy	0
session_state	e90bbe41-3c88-4bfb-a8c2-8b12c888f90e
scope	openid invoices-scope profile
access_token_url	https://localhost:8081/realms/digsign4einvoices/protocol/openid-connect/token
client_id	app-client
client_secret	9BDQfRM4MN86cKBrD8oBwtNdYYvBgwQH
timestamp	1681629103194

Tabla 3.2. Ejemplos de servicio Discovery en OIDC

Los tokens de acceso recibidos vienen en formato JWT. Esto quiere decir que pueden descifrarse y obtenerse a partir de ellos algunos valores. A continuación, se muestra el descifrado del cuerpo del token “Access Token”, que es el token de acceso que usará el cliente para enviar su solicitud al servidor de recursos:


```
{
  "exp": 1681629403,
  "iat": 1681629103,
  "auth_time": 1681629102,
  "jti": "f051edb3-2464-4dce-9db5-3165749feb11",
  "iss": "https://localhost:8081/realms/digsign4einvoices",
  "sub": "cef99d14-0d90-495e-894a-fc322b9484e1",
  "typ": "Bearer",
  "azp": "app-client",
  "session_state": "e90bbe41-3c88-4bfb-a8c2-8b12c888f90e",
  "realm_access": {
    "roles": [
      "SIGNER_ROLE"
    ]
  },
  "scope": "openid invoices-scope profile",
  "sid": "e90bbe41-3c88-4bfb-a8c2-8b12c888f90e",
  "name": "Fernando Gonzalez Hernandez",
  "preferred_username": "fgonzalezhernandez",
  "given_name": "Fernando",
  "family_name": "Gonzalez Hernandez"
}
```


A partir del “Access Token” recibido, se pueden determinar algunos datos relevantes para el servicio a implementar tales como como:

- iss: indica cuál es la URL del servidor de autenticación.
- Exp: fecha de expiración del token
- realm_access: indica en su interior el campo “roles” con los roles asignados al usuario.
- preferred_username: alias del usuario.
- name: nombre y apellidos del usuario.
- given_name: nombre del usuario.
- family_name: apellidos del usuario.

Para validar el token recibido y conseguir información de los atributos del usuario, se puede realizar una petición de tipo POST a la siguiente URL de Keycloak: “https://localhost:8081/realms/digsign4einvoices/protocol/openid-connect/userinfo”. A continuación, se muestra una petición lanzada desde

Postman al servicio Keycloak para validar el token y obtener información de los atributos del usuario:

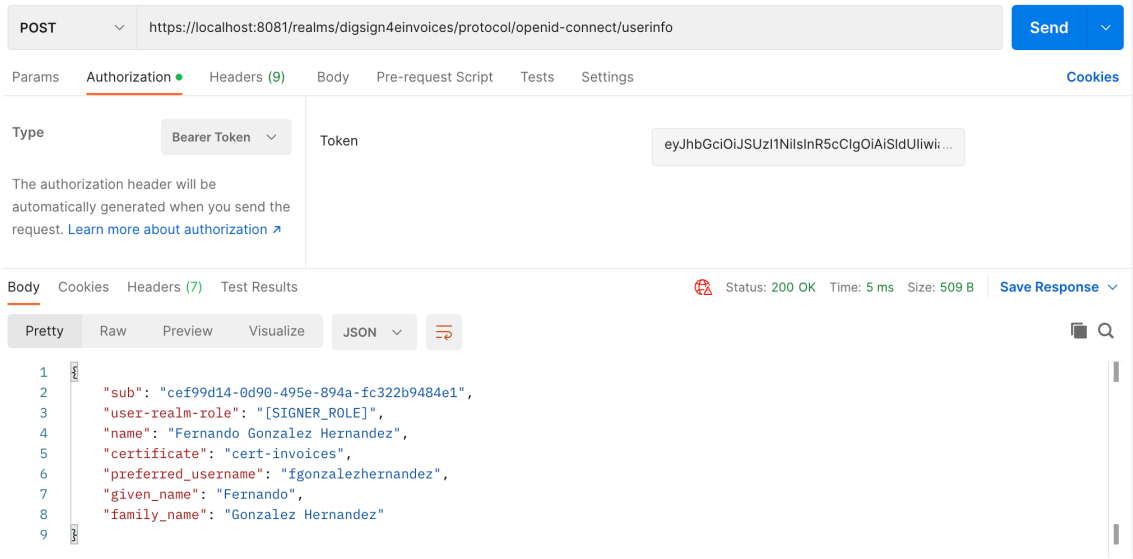

Figura 3.23: Petición enviada desde Postman a Keycloak para validar token y conseguir información sobre atributos de usuario

3.3.2. Generación de claves de firma y almacenamiento

Para el caso práctico del TFM se necesita la generación y almacenamiento de claves de firma. En el presente apartado se aborda la generación de claves de firma a través de la creación de una PKI, el almacenamiento de certificados a través de un keystore y la validación de las firmas generadas a través de una prueba que consiste en firmar un documento PDF.

3.3.2.1. Generación de PKI y de claves de firma

La PKI (acrónimo de “Public Key Infrastructure”, o, en español, Infraestructura de Clave Pública) es una tecnología que permite la gestión segura de claves públicas y privadas para su uso en la autenticación, el cifrado y la firma digital. Una de las aplicaciones más comunes de la PKI es la firma digital de documentos.

A continuación, se explican los pasos seguidos para la implementación de una PKI que consta de un certificado de CA raíz, un certificado de CA subordinada y un certificado de firma digital para la firma de documentos. La implementación ha sido realizada mediante la generación de unos scripts de elaboración propia que ejecutan comandos de la herramienta OpenSSL.

En primer lugar, se ha generado la entidad CA raíz utilizando un fichero de configuración y un script que genera los directorios de la CA y ejecuta los comandos OpenSSL necesarios para la generación del certificado.

El fichero de configuración utilizado para la generación de la CA raíz tiene el siguiente contenido:

```
# OpenSSL root CA configuration file.

[ ca ]
# Se indica a openssl que tome la configuración de CA_default para config la CA
default_ca = CA_default

[ CA_default ]
# Configuración por defecto para la CA

# Localización de directorios y ficheros
# "dir" es el directorio raíz sobre el que se depliegan los recursos de la CA
dir               = /Users/fer/PKI/ca
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/private/.rand
# par de claves de la CA
private_key       = $dir/private/ca.key.pem
certificate       = $dir/certs/ca.cert.pem
# Lista de revocación CRL
crlnumber         = $dir/crlnumber
crl               = $dir/crl/ca.crl.pem
crl_extensions    = crl_ext
default_crl_days  = 30
# algoritmo de cifrado
default_md        = sha256
# Valores por defecto
name_opt          = ca_default
cert_opt          = ca_default
default_days      = 375
preserve          = no
policy            = policy_strict

[ policy_strict ]
# Parámetros que deben coincidir entre certificados para que la CA lo firme
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ policy_loose ]
# Se disminuye la restricción en coincidencia de parámetros para que una CA intermedia
firme un certificado
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
default_bits        = 2048
distinguished_name  = req_distinguished_name
string_mask         = utf8only
default_md          = sha256

# Extension to add when the -x509 option is used.
x509_extensions     = v3_ca

[ req_distinguished_name ]
# Descripción de los parámetros
countryName                     = Country Name (2 letter code)
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
```

```
organizationalUnitName          = Organizational Unit Name
commonName                      = Common Name
emailAddress                    = Email Address

# Valores por defecto
countryName_default             = ES
stateOrProvinceName_default     = Spain
localityName_default            =
0.organizationName_default      = fgonzalezhernandez
organizationalUnitName_default  =
emailAddress_default            = fgonzalezhernandez@uoc.edu
commonName_default              = ROOT CA


[ v3_ca ]
# Configuración para CA raíz
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
# Configuración para CA intermedia/subordinada
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
# Configuración para los certificados para usuarios
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Configuración para los certificados para servidores
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Configuración para lista de revocación CRL
authorityKeyIdentifier=keyid:always

[ ocsp ]
# Configuración para el protocolo "Online Certificate Status Protocol" (OCSP).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

El resultado de la ejecución del script que genera la entidad CA raíz es el siguiente:

```
----------------------------------------------------------------
> Create directories for the new root CA
 ----------------------------------------------------------------
> Copy the base configuration file for the creation of a root CA
 ----------------------------------------------------------------
> Create the root CA certificates
Generating RSA private key, 4096 bit long modulus
.....................................................++++
```

```
.....................................................++++
e is 65537 (0x10001)
Enter pass phrase for private/ca.key.pem:
Verifying - Enter pass phrase for private/ca.key.pem:
Enter pass phrase for private/ca.key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name [Spain]:
Locality Name []:
Organization Name [fgonzalezhernandez]:
Organizational Unit Name []:
Common Name [ROOT CA]:
Email Address [fgonzalezhernandez@uoc.edu]:
 ----------------------------------------------------------------------
> Show certificate
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            b4:eb:8f:47:f0:c6:e0:1f
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=ES, ST=Spain, O=fgonzalezhernandez, CN=ROOT
CA/emailAddress=fgonzalezhernandez@uoc.edu
        Validity
            Not Before: Apr 18 15:43:13 2023 GMT
            Not After : Apr 17 15:43:13 2024 GMT
        Subject: C=ES, ST=Spain, O=fgonzalezhernandez, CN=ROOT
CA/emailAddress=fgonzalezhernandez@uoc.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:a5:a0:86:0d:7f:30:a8:90:8a:ce:cd:14:00:9a:
                    05:28:18:d5:e2:18:e1:8c:41:5c:42:d7:70:85:d0:
                    36:14:ab:1f:f5:5c:4c:f8:14:05:84:2f:9f:a5:60:
                    d9:d4:42:36:61:40:30:ac:fe:d1:13:cf:03:2d:4d:
                    7c:c1:69:31:21:09:0b:f1:5a:4e:cf:5f:ca:af:0d:
                    8f:5f:22:31:ce:ac:6c:85:b3:ae:d6:22:f7:77:18:
                    ae:81:40:47:3c:97:80:72:3b:91:d4:b0:9e:c3:52:
                    21:d2:d5:6f:7d:f1:82:22:b4:89:ec:5a:45:5d:50:
                    47:ce:81:50:bc:60:04:6e:23:e1:82:59:53:16:45:
                    3b:3e:d0:e7:f5:b8:e0:83:aa:da:97:57:81:ee:f5:
                    2b:c8:ec:13:dd:1c:22:28:7c:b0:b1:19:69:5f:08:
                    50:e0:64:5c:e4:d7:2b:46:25:d8:16:ab:ef:b4:c0:
                    ac:a5:cc:b9:37:67:64:b9:53:c2:2a:7b:2f:7e:c5:
                    36:bf:d9:95:86:1c:77:42:a0:c0:7b:2c:54:28:bf:
                    ab:1f:f7:69:c2:bc:a8:68:01:23:f0:c1:67:9e:4d:
                    e1:25:c6:98:c0:2c:98:43:b2:ea:0b:66:b7:d4:58:
                    e0:05:45:46:60:a3:1b:a9:a6:a3:75:b0:ca:3b:7c:
                    4f:67:12:a7:57:86:39:3b:99:e4:31:be:8b:b2:eb:
                    87:14:02:69:18:5f:46:ce:f6:7f:53:72:b9:3a:19:
                    48:a0:95:63:e5:d0:f4:38:95:78:47:ab:95:78:35:
                    5e:e6:0f:3c:f5:d4:be:90:f6:36:19:9e:19:8b:87:
                    a5:04:58:ce:66:bd:69:bf:42:da:92:0f:fd:12:21:
                    69:25:35:12:47:01:d3:c3:2f:12:c6:72:59:1d:93:
                    e0:ba:e5:ed:bd:1e:18:87:65:89:c1:27:23:f0:91:
                    ca:f0:27:88:04:62:27:65:d9:ee:a2:27:a7:6a:a0:
                    03:a7:7f:2f:98:cd:be:23:70:40:16:d3:27:15:4b:
                    8b:41:6e:1a:90:72:38:3b:70:68:93:f4:0a:37:b9:
                    e1:89:ef:ac:a7:cd:5f:68:7a:50:e0:aa:43:dd:12:
                    df:15:a0:16:f0:91:db:90:dd:47:11:da:08:2f:2a:
                    b1:a7:1f:22:b1:8e:07:2f:2e:35:b4:07:00:e3:87:
                    04:f9:4c:5b:d6:46:90:a4:e9:42:db:4c:8d:0a:50:
                    91:f4:1c:b2:ce:64:01:9c:60:31:4a:74:85:b2:98:
                    6e:fa:88:6b:8c:59:51:16:c2:d2:2e:55:55:75:49:
                    42:5d:69:6d:c8:a7:35:55:55:23:68:00:cb:ae:8b:
                    22:4e:3b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
```

```
            X509v3 Subject Key Identifier:
                64:DA:9F:9C:36:9A:00:F6:D2:C0:73:CC:18:CB:28:6D:34:D2:C6:76
            X509v3 Authority Key Identifier:
                keyid:64:DA:9F:9C:36:9A:00:F6:D2:C0:73:CC:18:CB:28:6D:34:D2:C6:76

            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
         3a:9e:57:a5:c9:ed:23:57:15:18:59:55:09:97:44:13:b2:18:
         43:7e:c0:89:5f:a0:20:eb:41:45:8c:0c:14:72:61:2c:8c:12:
         fb:93:3a:76:19:eb:1a:28:23:0f:97:1d:d1:bc:e5:bd:70:7d:
         a2:c5:72:ef:d8:38:de:55:d9:a1:92:89:69:2a:6d:11:19:c0:
         88:2e:8a:cc:f5:63:3b:5a:c6:f6:cd:43:22:3c:48:0a:97:ea:
         4a:12:aa:17:42:e7:24:b7:89:e1:8f:ab:48:d0:14:f3:72:46:
         0c:88:c8:58:23:d1:e6:46:3e:5c:74:b8:68:c0:bb:60:19:f4:
         ab:a0:62:f6:c4:f2:26:8b:a3:cd:c2:f9:94:62:fd:1a:ab:0c:
         a8:e5:41:b1:6d:7b:cd:86:4e:a7:6e:2f:1d:9b:07:29:44:fa:
         9b:61:7e:b0:bf:2d:0d:c7:14:79:60:5a:be:fb:3e:8c:a6:49:
         5b:57:ae:62:b4:e4:f7:09:73:87:c7:21:36:1e:d1:95:a3:95:
         c8:c3:b3:99:e0:3c:fd:4b:3a:c0:db:b6:05:c4:a1:c8:15:d1:
         9c:d5:fa:dd:43:cc:82:7f:66:d2:42:9a:f2:92:74:9d:b0:63:
         07:98:9f:b1:bb:01:75:22:93:ad:6f:1c:81:85:5e:01:95:26:
         54:7d:40:ae:f8:b0:4a:d1:43:b9:09:d1:36:4f:c5:2c:ea:ed:
         66:e7:18:41:05:84:03:93:43:ab:b3:76:f0:e2:97:93:6e:98:
         e4:05:cd:87:4f:fc:1e:0f:45:41:90:6a:0d:3c:8c:85:31:6a:
         14:32:cf:9a:9a:5a:80:f2:d6:e5:5a:3a:5e:f3:35:c5:7d:40:
         5a:e2:b1:21:22:78:4b:8f:5a:db:2c:41:76:50:eb:c9:a7:82:
         ff:f1:12:28:15:3a:67:6b:88:80:62:4d:56:92:e0:f6:14:f3:
         55:6d:10:29:ce:f9:bb:39:f7:ec:d2:eb:38:5f:bb:fa:cc:4c:
         db:84:b3:65:8c:59:d9:e8:db:82:40:38:f0:db:27:56:02:76:
         ef:dc:21:36:35:b2:7d:ea:93:88:e2:4d:2c:74:0d:b3:fe:de:
         65:5c:6f:18:68:ce:24:35:aa:97:c9:e5:3c:36:d9:df:50:f8:
         4e:66:29:0a:7b:d4:35:01:d4:e8:d6:49:cf:60:7d:0b:c8:fb:
         45:34:a4:cc:d1:72:d7:f5:d4:72:48:e4:f9:fe:2a:37:57:ba:
         b0:3a:b8:6d:77:59:59:af:ab:21:0c:55:17:27:37:2e:78:71:
         cb:96:e7:73:2c:3a:e5:36:90:3f:fd:cd:32:13:c0:72:1f:25:
         50:c2:e4:08:c6:8d:dc:2a
-----------------------------------------------------------------------
```

Se realiza la emisión de la CA subordinada a partir de la CA raíz generada. Al igual que con la CA raíz, la implementación de la CA subordinada se ha llevado a cabo a partir de un fichero de configuración y un script que genera los directorios de la CA y ejecuta los comandos OpenSSL necesarios para la generación del certificado.

El fichero de configuración utilizado para la generación de la CA subordinada tiene el siguiente contenido:

```
# OpenSSL intermediate CA configuration file.

[ ca ]
# Se indica a openssl que tome la configuración de CA_default para config la CA
default_ca = CA_default

[ CA_default ]
# Configuración por defecto para la CA

# Localización de directorios y ficheros
# "dir" es el directorio raíz sobre el que se depliegan los recursos de la CA intermedia
dir               = /Users/fer/PKI/ca/intermediate
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
```

```
serial            = $dir/serial
RANDFILE          = $dir/private/.rand
# Par de claves para la CA intermedia
private_key       = $dir/private/intermediate.key.pem
certificate       = $dir/certs/intermediate.cert.pem
# Lista de revocación CRL
crlnumber         = $dir/crlnumber
crl               = $dir/crl/intermediate.crl.pem
crl_extensions    = crl_ext
idefault_crl_days  = 30
# Algortirmo de cifrado
default_md        = sha256
# Valroes por defecto
name_opt          = ca_default
cert_opt          = ca_default
default_days      = 375
preserve          = no
policy            = policy_loose

[ policy_strict ]
# Parámetros que deben coincidir entre certificados para que la CA lo firme
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ policy_loose ]
# Se disminuye la restricción en coincidencia de parámetros para que una CA intermedia
firme un certificado
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
# Options for the `req` tool (`man req`).
default_bits        = 2048
distinguished_name  = req_distinguished_name
string_mask         = utf8only
default_md          = sha256

# Extension to add when the -x509 option is used.
x509_extensions     = v3_ca

[ req_distinguished_name ]
# Descripción de los parámetros
countryName                     = Country Name (2 letter code)
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
organizationalUnitName          = Organizational Unit Name
commonName                      = Common Name
emailAddress                    = Email Address

# Valores por defecto
countryName_default             = ES
stateOrProvinceName_default     = Spain
localityName_default            =
0.organizationName_default      = fgonzalezhernandez
organizationalUnitName_default  =
emailAddress_default            = fgonzalezhernandez@uoc.edu


[ v3_ca ]
# Configuración para CA raíz
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

```
[ v3_intermediate_ca ]
# Configuración para CA intermedia/subordinada
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
# Configuración para los certificados para usuarios
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Configuración para los certificados para servidores
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Configuración para lista de revocación CRL
authorityKeyIdentifier=keyid:always

[ ocsp ]
# Configuración para el protocolo "Online Certificate Status Protocol" (OCSP).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

El resultado de la ejecución del script que genera la entidad CA subordinada es el siguiente:

```
----------------------------------------------------------------------
> Create directories for the new subordinate CA
 ----------------------------------------------------------------------
> Copy the base configuration file for the creation of a subordinate CA
 ----------------------------------------------------------------------
> Create the subordinate CA certificates
Generating RSA private key, 4096 bit long modulus
.....................................................................................
.....................................................+++
.....................................................................................
.....................................................+++
e is 65537 (0x10001)
Enter pass phrase for intermediate/private/intermediate.key.pem:
Verifying - Enter pass phrase for intermediate/private/intermediate.key.pem:
Enter pass phrase for intermediate/private/intermediate.key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name [Spain]:
Locality Name []:
Organization Name [fgonzalezhernandez]:
Organizational Unit Name []:
Common Name []:SUBORDINATE CA
```

```
Email Address [fgonzalezhernandez@uoc.edu]:
Using configuration from openssl.cnf
Enter pass phrase for /Users/fer/PKI/pki2/ca/private/ca.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Apr 18 15:44:08 2023 GMT
            Not After : Apr 17 15:44:08 2024 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = Spain
            organizationName          = fgonzalezhernandez
            commonName                = SUBORDINATE CA
            emailAddress              = fgonzalezhernandez@uoc.edu
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                CD:22:A4:46:0E:7A:C2:50:BE:25:A5:93:6F:A3:0B:1E:7B:D4:38:70
            X509v3 Authority Key Identifier:
                keyid:64:DA:9F:9C:36:9A:00:F6:D2:C0:73:CC:18:CB:28:6D:34:D2:C6:76

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
Certificate is to be certified until Apr 17 15:44:08 2024 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
 ----------------------------------------------------------------------
> Show certificate
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=ES, ST=Spain, O=fgonzalezhernandez, CN=ROOT
CA/emailAddress=fgonzalezhernandez@uoc.edu
        Validity
            Not Before: Apr 18 15:44:08 2023 GMT
            Not After : Apr 17 15:44:08 2024 GMT
        Subject: C=ES, ST=Spain, O=fgonzalezhernandez, CN=SUBORDINATE
CA/emailAddress=fgonzalezhernandez@uoc.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:a3:80:0b:9e:a0:c0:b0:07:14:92:18:c9:f6:d2:
                    e0:13:00:85:74:85:65:28:3d:2c:91:ab:c5:c0:9c:
                    70:18:83:be:d5:00:9a:74:4a:16:67:9d:b5:ed:a5:
                    34:33:6b:e1:f4:ad:bf:5d:09:15:69:4f:5d:a2:84:
                    6a:11:8e:47:28:60:95:3a:18:04:30:63:69:4a:63:
                    60:2e:bb:c4:55:7b:f0:eb:29:c4:35:7d:ce:d6:c0:
                    13:bc:b8:0f:48:db:ba:bd:0f:72:4f:7b:4c:db:4e:
                    bf:ef:0d:34:78:2b:ce:2e:84:89:c6:89:45:09:92:
                    22:b9:de:30:6a:7f:47:54:0c:a2:ff:f7:38:81:57:
                    24:6d:fd:ff:f9:31:45:99:22:e0:f9:47:96:36:52:
                    81:91:57:c1:9b:d3:cc:16:65:91:51:56:cb:13:90:
                    fd:7f:d5:5c:b4:68:c4:4b:40:5a:c6:2a:3f:55:b5:
                    b2:6b:b3:a9:b2:ec:9d:59:80:a2:f6:59:d4:b4:c5:
                    e8:6f:9e:9a:81:eb:7a:83:c4:8d:70:f9:b3:e4:d2:
                    7b:dd:a8:62:a0:0f:9a:2a:36:27:00:09:1c:42:aa:
                    f7:53:e8:82:01:d9:60:4e:84:fd:75:c2:5e:f8:f9:
                    96:89:f0:5a:70:b1:88:b7:66:1f:54:4b:03:f6:f4:
                    4a:e5:01:d5:cc:21:e9:9c:b9:c8:0b:d6:bf:27:42:
                    82:d4:32:15:b2:c0:7d:2a:32:df:08:d9:e5:07:5c:
                    4c:46:75:60:c1:61:3a:34:33:8e:74:d7:7a:95:ce:
                    7d:6a:66:b2:c3:11:c3:ca:e5:f1:3a:e1:cb:76:89:
                    5c:1f:8a:29:f6:0c:c9:dd:a3:5e:35:58:7a:eb:49:
                    39:b9:07:15:56:01:4b:74:57:13:e6:ff:3a:c3:90:
                    8f:f7:b3:1d:52:2f:15:ed:ef:61:6a:b3:36:0c:3f:
```

```
                    1c:80:5d:eb:4a:32:d7:28:fd:3d:85:99:86:1b:4d:
                    41:1b:b1:ef:91:e0:b8:90:95:42:01:d0:dd:0c:be:
                    cb:c9:0b:e3:f1:43:65:fc:98:0e:35:e5:1e:39:48:
                    77:ea:4d:4d:74:96:4c:76:bb:8d:38:ca:4f:6d:7d:
                    56:ad:87:46:e8:c4:22:da:b3:56:6b:52:5e:6f:73:
                    33:5e:fa:91:30:a6:a9:b1:f3:1a:4e:81:bb:7e:2e:
                    9c:ce:ee:c5:0e:43:0a:b2:be:82:8b:e3:78:0c:01:
                    d0:42:01:8f:00:2d:13:0e:37:e2:61:e8:81:df:32:
                    03:c5:9a:9a:3e:b6:3b:1c:c1:9a:e5:e4:17:17:9c:
                    0b:bc:5f:ee:8a:ae:b3:18:1d:35:fc:fe:a6:ff:d6:
                    52:30:51
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                CD:22:A4:46:0E:7A:C2:50:BE:25:A5:93:6F:A3:0B:1E:7B:D4:38:70
            X509v3 Authority Key Identifier:
                keyid:64:DA:9F:9C:36:9A:00:F6:D2:C0:73:CC:18:CB:28:6D:34:D2:C6:76

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
         0a:ce:ef:dc:ae:10:b8:b8:f3:45:bf:42:a2:cb:83:89:aa:1e:
         ee:5c:74:91:0f:9d:62:03:42:28:e6:c7:99:3f:76:cc:9e:f4:
         2b:64:02:b8:fd:0a:89:fa:49:30:80:33:c5:16:70:88:94:61:
         5d:06:79:2d:24:51:8f:c3:aa:b2:63:22:c3:f7:de:19:57:4b:
         43:fa:12:6b:51:20:c2:a0:d9:fa:3f:49:60:ad:f2:b1:1f:7d:
         f4:6a:07:cb:d6:b6:9d:49:42:54:88:b6:ed:b4:cb:ad:fb:ea:
         b5:17:ab:03:61:93:ec:85:c5:4d:82:c2:23:4f:e5:f7:b4:c0:
         8d:a0:4b:fa:fb:3b:c2:97:ed:b4:83:80:58:6e:9a:7b:58:13:
         2c:23:b9:fe:02:e4:38:8b:45:ff:4b:05:ab:88:bc:cb:78:ac:
         3e:b2:60:f7:23:e7:70:5d:6e:fb:62:b9:1e:5c:5a:8d:4a:2a:
         d1:a6:3b:dc:5c:db:ac:4e:c4:1a:50:c6:91:5e:c2:8c:60:a7:
         73:02:dc:27:4a:3c:b5:08:a3:7c:1d:87:ae:30:26:8f:88:64:
         cd:cb:a1:5f:b2:e2:87:0b:d0:75:69:98:09:bf:39:9c:95:94:
         10:99:b4:0a:e5:27:21:fb:60:bb:84:51:a7:01:48:76:fd:1e:
         b3:8e:04:6c:ba:cf:84:b6:73:5b:c6:85:7b:36:9d:97:cb:ea:
         d8:ac:f8:b6:26:61:d0:f2:18:9e:5f:60:3c:26:97:a4:af:94:
         23:1b:b3:81:6e:bd:de:9b:7f:1c:7d:6f:06:5f:46:46:9f:d8:
         df:0c:d7:d7:7a:d0:f6:82:22:8a:89:e7:c1:18:a6:c3:a6:7b:
         36:59:4a:15:b6:06:50:a5:35:20:12:1e:f3:4f:6d:99:a4:dd:
         ba:a8:66:6a:6f:2a:52:b4:61:25:cb:63:9e:c5:68:40:4c:62:
         49:bc:df:0c:3b:70:46:82:79:27:41:e3:c7:73:f4:92:c1:e4:
         9b:9f:b0:97:53:39:e5:5f:6b:fd:8d:78:80:82:b1:25:2e:3c:
         dd:b9:eb:4d:0f:11:15:20:00:4a:41:a1:e9:a8:08:5e:34:96:
         e2:39:ce:59:86:5b:04:5c:43:a1:0b:a1:db:4e:54:8c:aa:69:
         ed:85:fe:5c:35:05:2c:0c:dd:ad:ad:1b:7a:16:4a:17:d1:04:
         a0:41:d9:b0:2e:70:4e:ec:44:2f:70:d0:74:ee:a3:b5:ad:f1:
         43:bd:46:09:17:3b:d1:87:4a:64:80:0c:2d:d5:7b:4f:03:9c:
         91:c8:6b:3b:5d:66:a6:4e:fd:4f:c9:53:fd:97:1c:cc:c8:3d:
         01:55:aa:ee:86:af:30:d5
 ----------------------------------------------------------------------
> Verify CA subordinate
intermediate/certs/intermediate.cert.pem: OK

> Create chain cert
```

Finalmente se genera un certificado de firma digital utilizando la CA subordinada creada anteriormente. Para la generación de este certificado también se ha generado un script con los comandos OpenSSL necesarios. Además, en el último paso se exporta un fichero de tipo p12 que será utilizado para la firma digital de documentos.

El resultado de ejecución del script que genera el certificado de firma digital es el siguiente:

```
-----------------------------------------------------------------------
> Create a user certificate for digital signature signed by subordinate CA
Generating RSA private key, 2048 bit long modulus
.....................................+++++
.................................+++++
e is 65537 (0x10001)
Enter pass phrase for intermediate/private/fgonzalezhernandez.key.pem:
Verifying - Enter pass phrase for intermediate/private/fgonzalezhernandez.key.pem:
Enter pass phrase for intermediate/private/fgonzalezhernandez.key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [ES]:
State or Province Name [Spain]:
Locality Name []:
Organization Name [fgonzalezhernandez]:
Organizational Unit Name []:
Common Name []:FERNANDO GONZALEZ HERNANDEZ
Email Address [fgonzalezhernandez@uoc.edu]:
Using configuration from intermediate/openssl.cnf
Enter pass phrase for /Users/fer/PKI/pki2/ca/intermediate/private/intermediate.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Apr 18 15:45:01 2023 GMT
            Not After : Apr 27 15:45:01 2024 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = Spain
            organizationName          = fgonzalezhernandez
            commonName                = FERNANDO GONZALEZ HERNANDEZ
            emailAddress              = fgonzalezhernandez@uoc.edu
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Client, S/MIME
            Netscape Comment:
                OpenSSL Generated Client Certificate
            X509v3 Subject Key Identifier:
                58:EA:28:10:43:4C:5D:EC:D6:88:9B:0C:84:B3:94:CB:95:BF:5C:3F
            X509v3 Authority Key Identifier:
                keyid:CD:22:A4:46:0E:7A:C2:50:BE:25:A5:93:6F:A3:0B:1E:7B:D4:38:70

            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Client Authentication, E-mail Protection
Certificate is to be certified until Apr 27 15:45:01 2024 GMT (375 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
 ---------------------------------------------------------------------
> Show certificate
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=ES, ST=Spain, O=fgonzalezhernandez, CN=SUBORDINATE
CA/emailAddress=fgonzalezhernandez@uoc.edu
        Validity
            Not Before: Apr 18 15:45:01 2023 GMT
            Not After : Apr 27 15:45:01 2024 GMT
        Subject: C=ES, ST=Spain, O=fgonzalezhernandez, CN=FERNANDO GONZALEZ
HERNANDEZ/emailAddress=fgonzalezhernandez@uoc.edu
```

```
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:a6:1f:83:04:12:aa:66:eb:a3:f7:9a:4a:7a:a5:
                    be:66:6a:fa:fd:82:ba:88:12:d5:f1:1b:e9:20:b6:
                    ee:38:3d:a9:64:85:23:10:30:d2:0f:74:62:43:1f:
                    82:4e:3b:40:2e:e2:35:d1:33:d0:4e:1c:e8:33:0b:
                    65:77:04:64:80:20:05:96:45:93:a0:d2:59:80:42:
                    23:5d:8d:c9:9c:b4:93:98:e1:d9:6a:fd:46:f6:a6:
                    7c:d4:6e:2e:e3:4c:78:62:f1:59:1a:f6:3b:e1:67:
                    64:29:73:00:b2:f0:67:2c:ff:32:87:72:0b:20:a5:
                    fd:b1:83:9f:70:8d:6b:9b:a7:1c:33:8d:84:07:92:
                    07:87:4a:dd:7e:b0:f3:17:61:61:92:07:89:07:81:
                    36:4e:c6:53:f9:32:c1:82:03:a3:b3:a3:1f:23:c1:
                    17:8a:de:c9:7d:50:a3:d6:08:7c:09:cc:ad:43:dc:
                    18:ea:3d:b9:5c:5c:00:44:3b:cc:13:23:bd:24:5d:
                    55:fa:ce:ae:67:34:6e:ba:32:70:3e:f3:fd:6b:ea:
                    0d:a5:58:fe:5c:f5:7c:bf:9c:8c:aa:6f:f1:ee:13:
                    8c:41:d3:d5:6c:77:ab:50:aa:c2:dc:c8:69:79:3c:
                    71:ad:e6:97:64:21:85:6f:ed:fe:25:fd:69:9f:d0:
                    c2:b1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Client, S/MIME
            Netscape Comment:
                OpenSSL Generated Client Certificate
            X509v3 Subject Key Identifier:
                58:EA:28:10:43:4C:5D:EC:D6:88:9B:0C:84:B3:94:CB:95:BF:5C:3F
            X509v3 Authority Key Identifier:
                keyid:CD:22:A4:46:0E:7A:C2:50:BE:25:A5:93:6F:A3:0B:1E:7B:D4:38:70

            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Client Authentication, E-mail Protection
    Signature Algorithm: sha256WithRSAEncryption
         02:fa:1b:e6:5e:4f:64:2b:ea:f3:95:04:c9:5f:fa:a6:1c:06:
         91:22:8f:13:8f:3e:cd:44:09:ed:a6:08:2e:4a:bf:d1:80:03:
         e4:1a:4b:e3:71:39:44:65:08:13:2d:0e:5d:11:55:d8:98:e1:
         c3:58:70:ed:fb:e3:c8:b6:bb:97:5b:c9:cf:8d:1a:5d:75:9c:
         80:d2:da:e6:f1:42:8e:bd:14:9c:5a:45:bd:ba:24:e9:44:1f:
         ef:34:43:fa:03:1c:83:63:67:c0:1b:ca:f0:fc:02:e2:8e:fb:
         5d:8e:88:e7:7d:18:b9:94:08:c2:37:b1:09:73:6e:86:a6:93:
         6c:d5:6a:ef:c5:74:14:d1:d7:b0:02:b8:dd:12:84:52:f0:a0:
         3d:16:c8:00:ee:6f:9a:b5:2f:4e:63:8e:eb:bc:8a:17:16:37:
         97:39:9d:9d:5c:5d:d2:4c:63:01:e0:68:0a:32:23:82:26:0b:
         8b:06:41:0e:76:f8:ac:f6:37:16:e5:17:f9:4f:c1:6a:60:96:
         b6:ba:ff:4d:8f:52:e0:cb:96:d3:2f:cb:36:98:d7:3b:df:aa:
         b9:5c:43:f4:45:2d:0d:25:cc:e9:15:20:c4:17:a4:54:fb:39:
         47:a6:8a:21:80:10:b7:18:52:bf:aa:e4:9d:37:8f:b6:fa:11:
         bf:81:26:b2:84:ed:ec:fb:38:c0:02:3d:50:cd:be:4d:2e:54:
         82:e2:e8:e3:2e:ac:2a:db:77:22:b0:1b:06:08:d0:fe:c4:23:
         62:ef:35:56:b6:a4:51:66:c0:b3:21:63:93:f4:54:b0:88:76:
         9a:15:25:87:d3:cc:ca:fc:64:ec:2d:2c:aa:4d:72:9f:88:0f:
         3d:12:99:97:6f:22:ce:3b:84:a1:2e:df:ea:39:17:67:ba:4d:
         ae:5c:44:4a:fb:4c:f9:d7:a8:26:9e:8e:0a:e3:37:74:6b:ab:
         c0:58:fc:6f:9e:ba:21:ff:a8:40:bb:04:fc:3a:73:c9:bd:76:
         51:df:eb:52:a8:0d:0d:0c:b5:1e:58:19:e6:06:de:3b:04:76:
         87:05:ed:e1:50:8e:22:5d:f2:fa:5f:fa:fa:39:a6:5a:47:21:
         cd:fd:37:51:7e:12:3a:6d:5a:2e:d2:bd:42:71:d0:91:89:de:
         43:5f:e7:5c:91:bb:fc:3b:af:5d:0d:83:04:4c:c6:a5:ed:10:
         0f:7a:4a:0d:8e:a0:c7:5a:e5:42:96:54:17:d2:34:10:75:0c:
         01:ef:3f:ab:c6:20:9d:aa:83:68:45:e2:70:e0:49:97:76:b4:
         fd:b8:98:dd:99:cb:be:fa:92:da:7d:18:20:77:8b:c5:9c:3f:
         5b:23:6d:ad:4f:8a:c2:d5
 -----------------------------------------------------------------
> Verify certificate
intermediate/certs/fgonzalezhernandez.cert.pem: OK
 -----------------------------------------------------------------
> Create p12 file from the user's certificate
Enter pass phrase for intermediate/private/fgonzalezhernandez.key.pem:
```

3.3.2.2. Almacenamiento de certificados y de claves de firma

Un keystore, también conocido como almacén de claves, es una base de datos segura que se utiliza para proteger y gestionar claves privadas y certificados digitales que se utilizan para la autenticación el cifrado de aplicaciones de seguridad. Existen diferentes tipos de keystore, que dependerán de la plataforma o aplicación que se utilice.

Para el propósito del presente caso práctico se crea un keystore en formato Java KeyStore (JKS) para almacenar los certificados creados desde el servidor Linux. Se utiliza la herramienta KeyStore Explorer 5.5.2 para la creación del keystore.

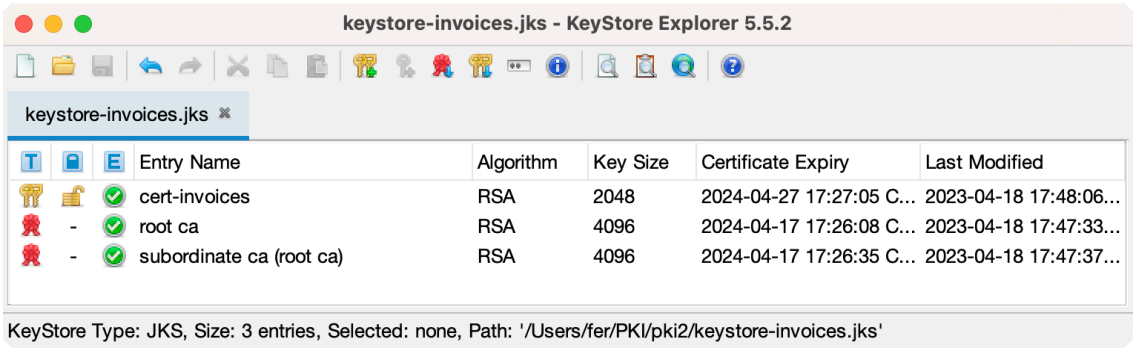

Figura 3.24: Almacén de certificados creado

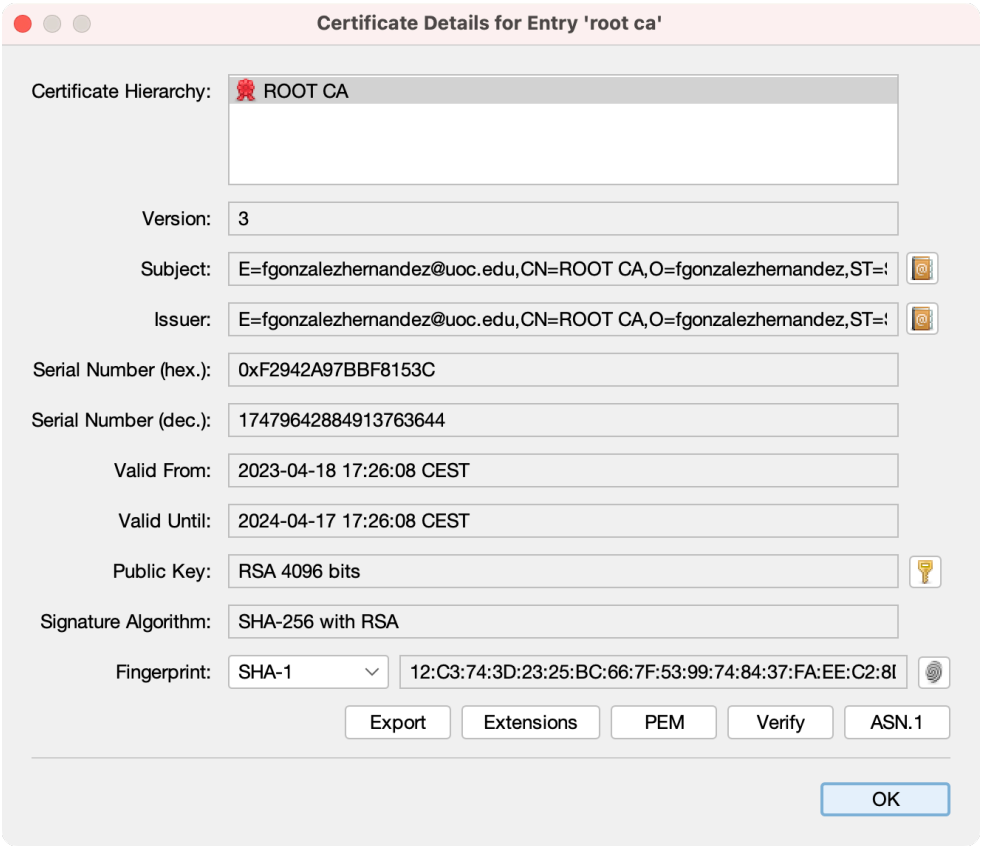

Figura 3.25: Detalle del certificado de CA raíz

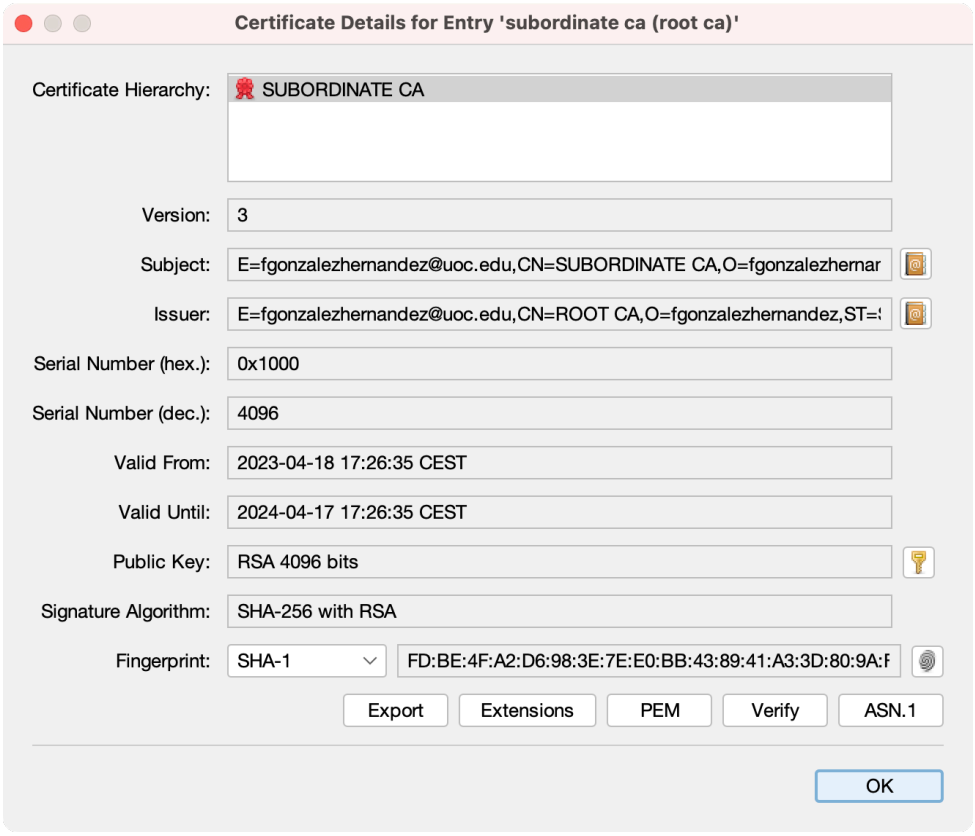

Figura 3.26: Detalle del certificado de CA subordinada

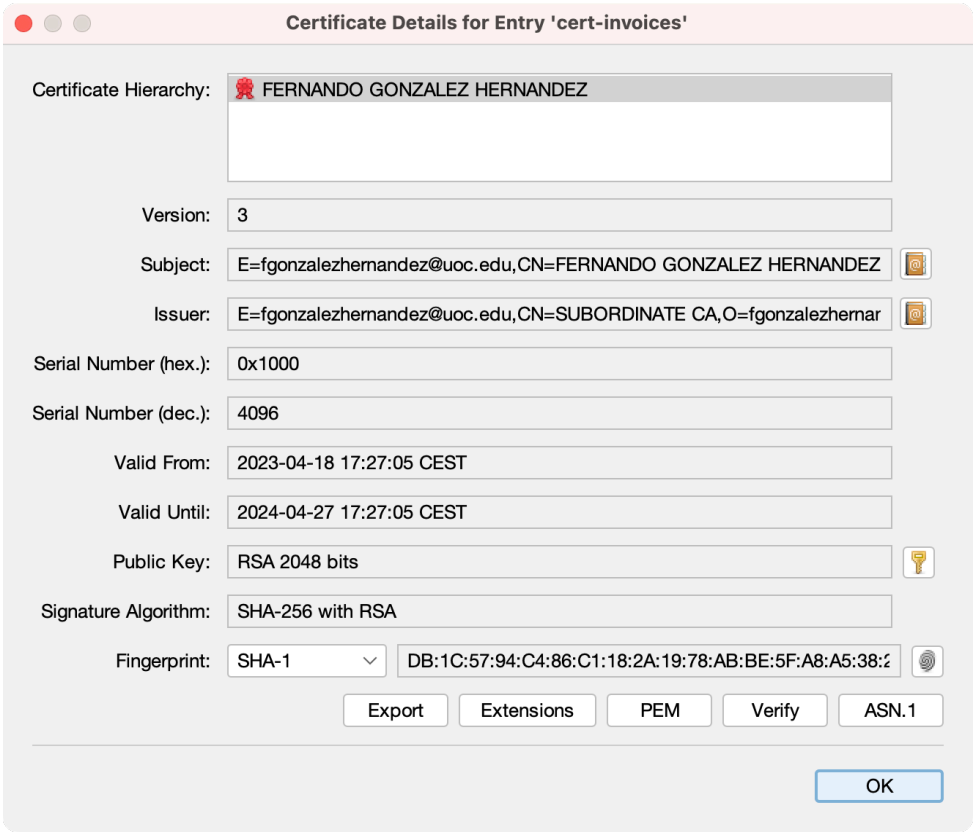

Figura 3.27: Detalle del certificado de firma digital

3.3.2.3. Validación de la clave de firma digital generada

A continuación, se realiza una prueba de validación de los certificados generados. La prueba consiste en utilizar el certificado de firma digital para firmar digitalmente un documento PDF utilizando la herramienta Adobe Reader DC de Acrobat. Una vez que la firma está aplicada al documento PDF, se verificará también con la herramienta Adobe Reader DC de Acrobat.

Para realizar estos pasos se han exportado los certificados generados a un equipo Windows con el software Adobe Reader DC de Acrobat instalado. Además, se han instalado los certificados de CA raíz y CA subordinada en el equipo para que así la firma a aplicar sea considerada de confianza.

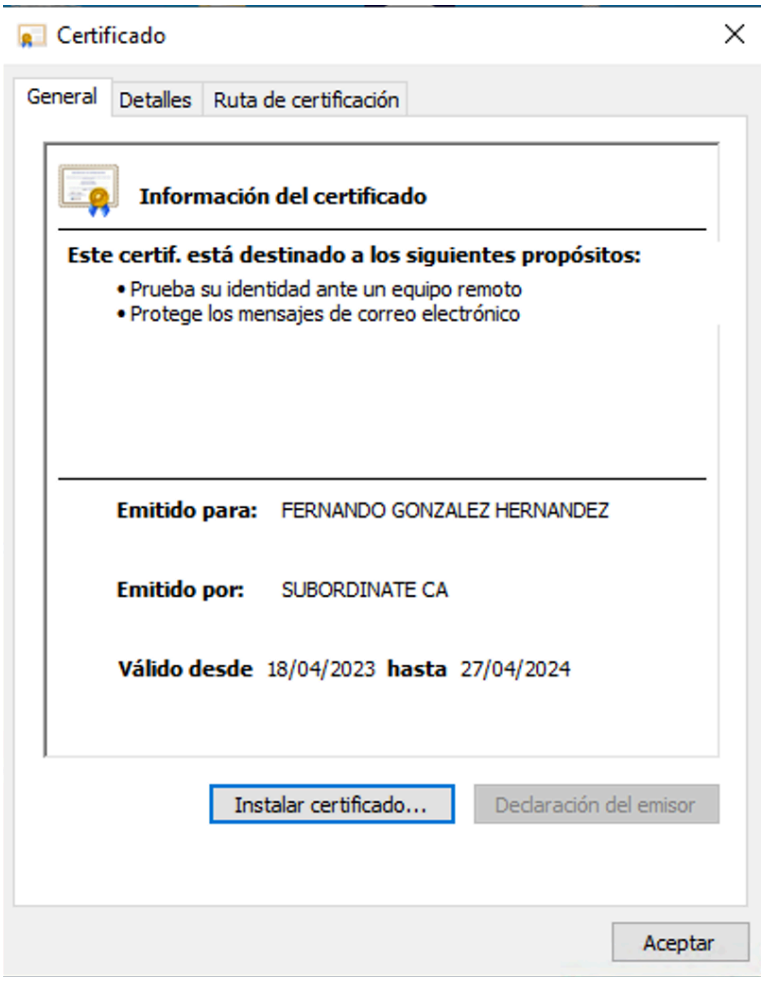

Figura 3.28: Detalle del certificado de firma digital en Windows (I)

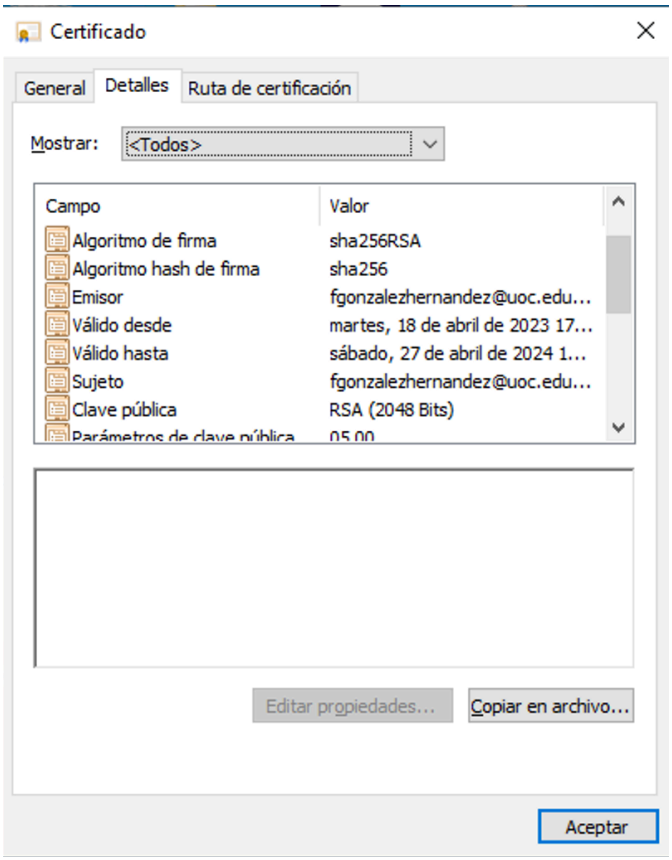

Figura 3.29: Detalle del certificado de firma digital en Windows (II)

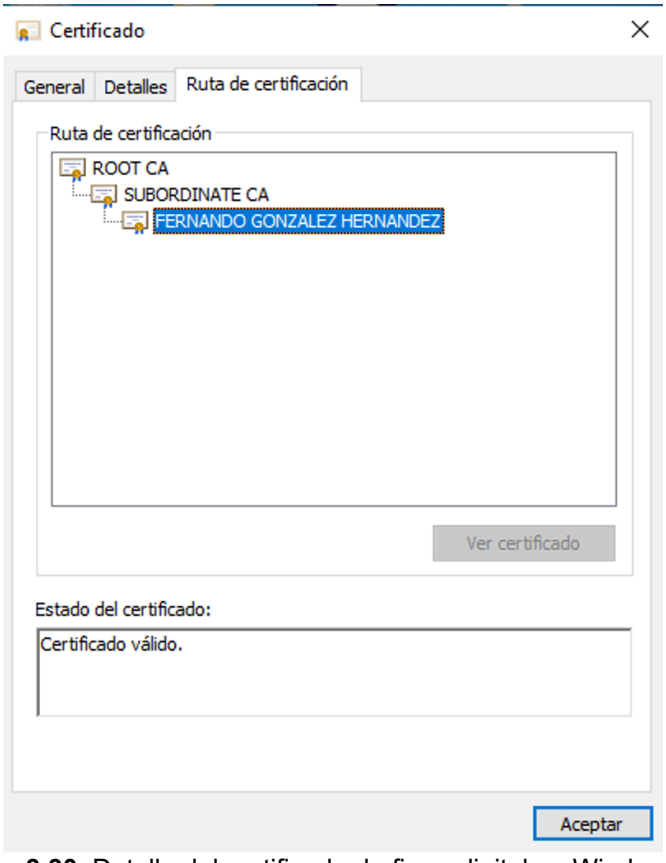

Figura 3.30: Detalle del certificado de firma digital en Windows (III)

A continuación, se utiliza el software Adobe Reader DC de Acrobat para firmar un documento PDF de prueba. Para firmar el documento, se utiliza la herramienta “Firmar digitalmente” del software y se selecciona el fichero p12 que contiene el certificado de firma digital.

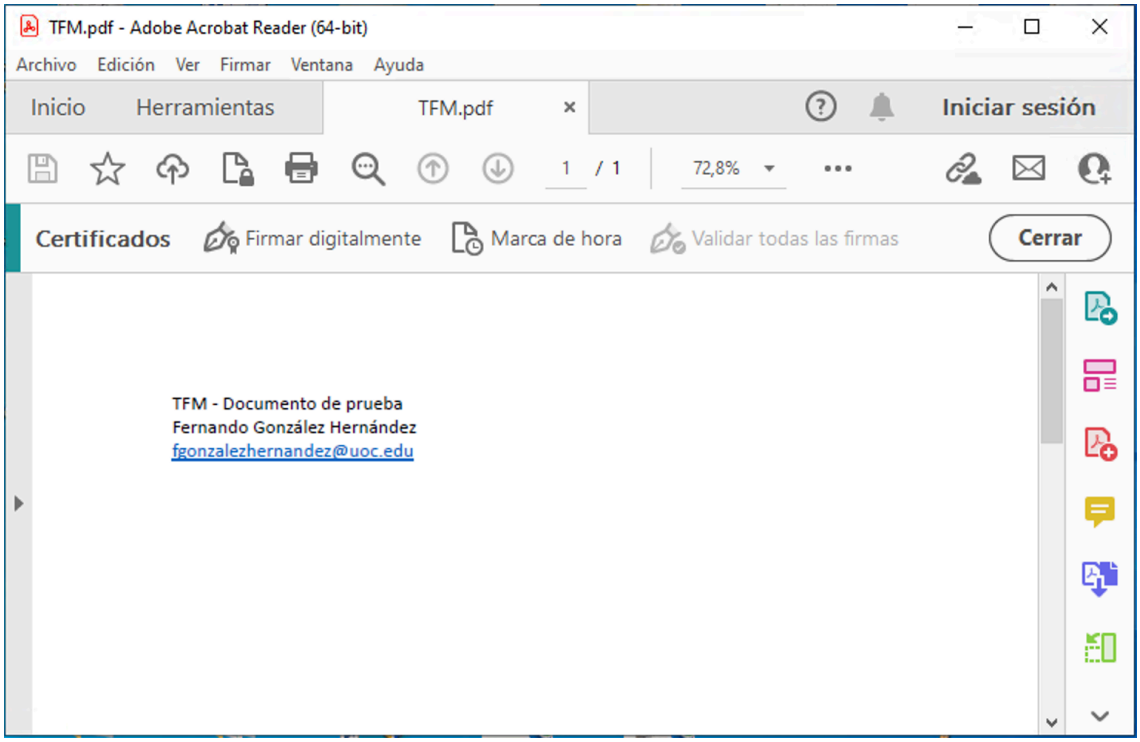

Figura 3.31: Detalle del documento PDF de prueba antes de ser firmado

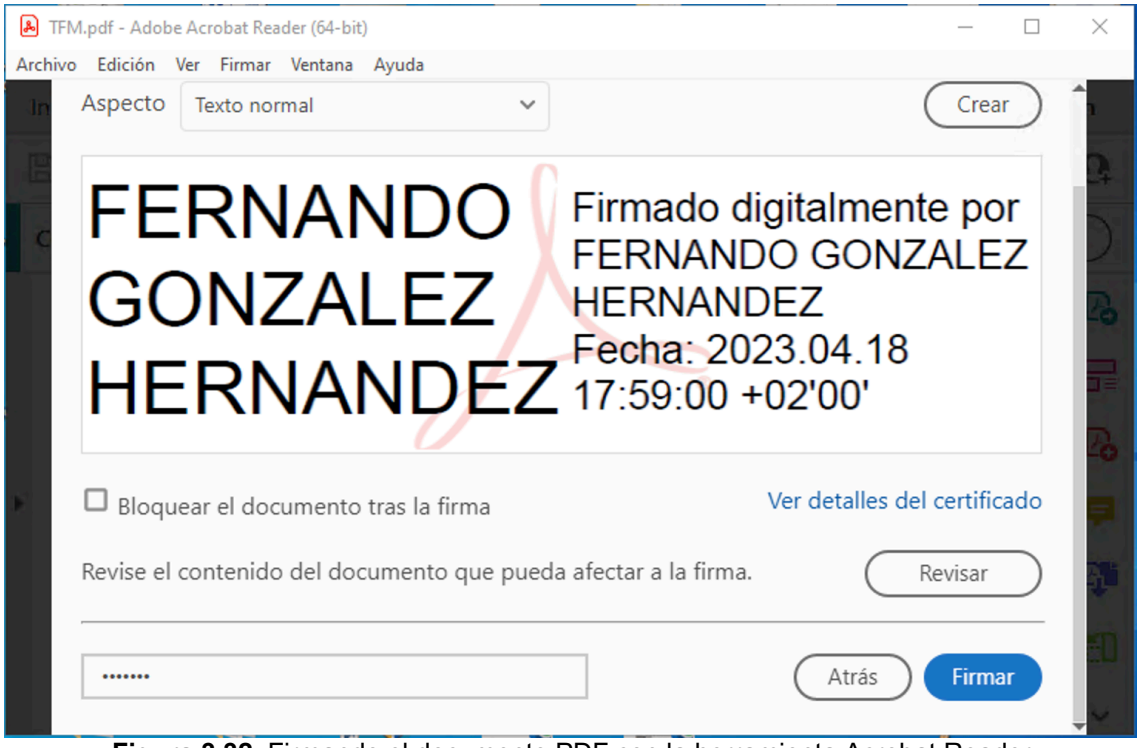

Figura 3.32: Firmando el documento PDF con la herramienta Acrobat Reader

Una vez que se ha firmado el documento, se guarda con otro nombre para conservar el documento original no firmado y posteriormente se vuelve a abrir con Acrobat Reader para verificar la firma. En la siguiente figura se observa el resultado de la apertura del fichero con la firma incrustada y el detalle de la validación de la firma.

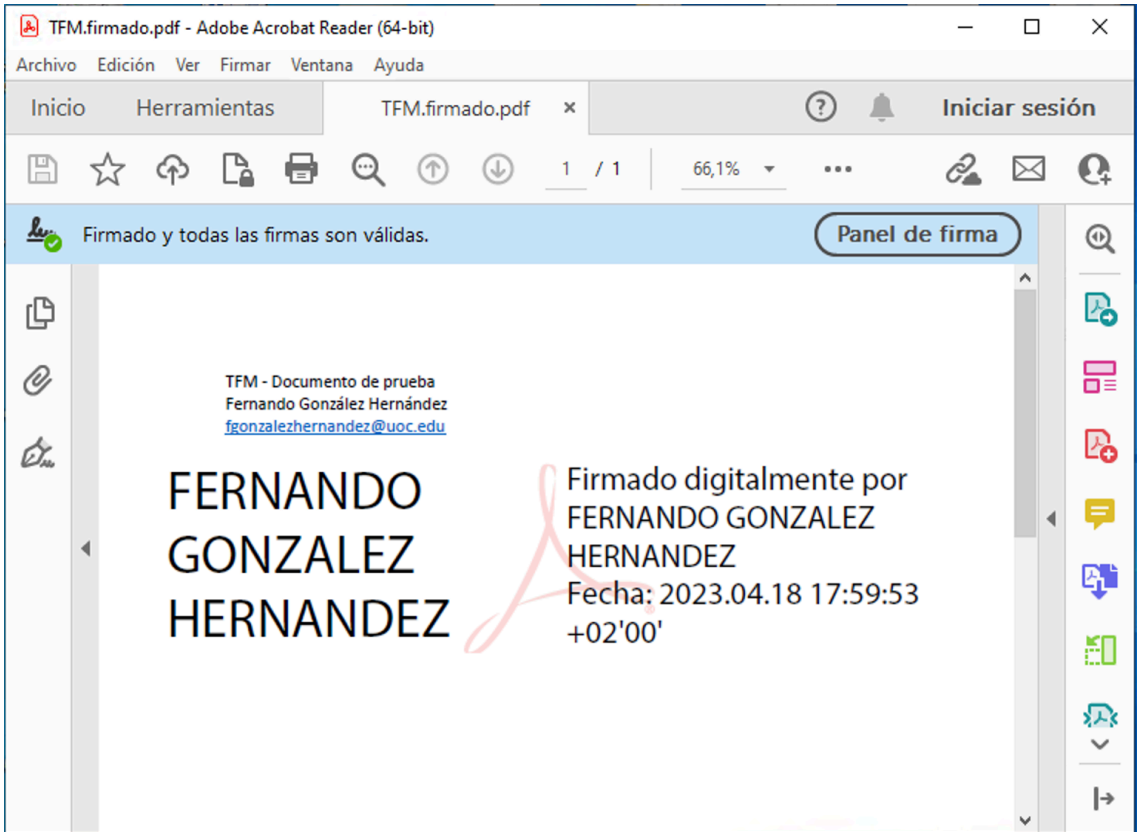

Figura 3.33: Documento PDF de prueba con firma digital incrustada

3.3.3. Servicio de firmado de facturas

Un servicio de firmado de facturas es una solución tecnológica que permite la generación y firma digital de facturas electrónicas. En el presente trabajo se realiza la implementación del servicio de firmado de facturas utilizando Spring Boot.

Las características principales del servicio a implementar son las siguientes:

- El servicio de firmado posee una interfaz de tipo API REST.
- El servicio de firmado se integra con el servicio Keycloak para obtener su certificado de clave pública y así validar la firma de los tokens recibidos en formato JWT
- El servicio de firmado se integra con una base de datos embebida en memoria (H2) para comprobar si un token de acceso ya ha sido usado previamente para intentar firmar una factura electrónica. Además, el servicio de firmado también se integra con la base de datos para obtener el alias del certificado digital correspondiente del usuario que ha autorizado la firma de la factura electrónica.

- El servicio de firmado valida que la información contenida en el token JWT sea correcta: que no se haya superado el tiempo de expiración del token, que el usuario posea el rol necesario para poder firmar facturas electrónicas y que el usuario haya dado permiso específico a la aplicación cliente para poder firmar facturas electrónicas en su nombre.
- El servicio de firmado digital comprueba que el formato de las facturas que se desean firmar sea PDF.
- El servicio de firmado se integra con un keystore de tipo JKS para acceder a sus certificados de firma digital y firmar con ellos facturas electrónicas.
- El servicio de firmado digital firma las facturas en PDF con un certificado de firma digital.

La secuencia de procesos del servicio de firmado de facturas es el siguiente:

En caso de ocurrir algún tipo de error en alguno de los procesos o de que alguna de las validaciones no se cumpliera, entonces se respondería a la petición con el error correspondiente.

3.3.3.1. Interfaz API REST del servicio de firmado

El tipo de interfaz elegido para poder interactuar con el servicio de firmado de facturas es API REST. API REST (Representational State Transfer) es una arquitectura de software para el desarrollo de servicios web que permite a los usuarios interactuar con una aplicación a través de una interfaz basada en el protocolo HTTP.

Para diseñar una API REST de forma sencilla y eficiente se utiliza la herramienta swagger. Esta herramienta permite describir la estructura y el comportamiento de una API en un formato legible y fácil de entender, lo que facilita la integración y el desarrollo de clientes que consuman la API.

A continuación, se muestra el swagger generado para definir la API REST del servicio de firmado:

```
swagger: "2.0"
info:
  title: API REST to sign PDF invoices
  version: "1.0.0"
paths:
  /sign-invoice:
    post:
      summary: Sign a PDF invoice
      description: Sign a PDF invoice and return the invoice signed
      consumes:
        - multipart/form-data
      produces:
        - application/pdf
      parameters:
        - in: header
          name: Authorization
          description: Token OpenID Connect in Bearer format
          required: true
          type: string
        - in: formData
          name: invoice
          description: PDF invoice to be signed
          required: true
          type: file
      responses:
        200:
          description: PDF invoice signed
          schema:
            type: file
            format: binary
        400:
          description: Bad request
        401:
          description: Token invalid or expired
        500:
          description: Internal Server Error
```

3.3.3.2. Integración con el servicio Keycloak

El servicio Keycloak dispone de una API REST con diferentes endpoints accesibles por HTTPS. Cada endpoint sirve a una funcionalidad diferente. Los principales endpoints de Keycloak son los siguientes:

- /auth/realms/{realm-name}: Este endpoint se utiliza para acceder a una instancia específica de un Realm en Keycloak. El {realm-name} se

reemplaza con el nombre del reino, que en el caso del presente TFM es igual a “digsign4einvoices”.

- /auth/realms/{realm-name}/protocol/openid-connect/certs: Este endpoint es utilizado para obtener las claves públicas de los certificados de Keycloak.
- /auth/realms/{realm-name}/protocol/openid-connect/auth: Este endpoint se utiliza para iniciar el proceso de autenticación en Keycloak.
- /auth/realms/{realm-name}/protocol/openid-connect/token: Este endpoint se utiliza para obtener un token de acceso para acceder a recursos protegidos.
- /auth/realms/{realm-name}/protocol/openid-connect/userinfo: Este endpoint se utiliza para obtener información del usuario después de que se ha autenticado.
- /auth/admin/realms/{realm-name}/users: Este endpoint se utiliza para administrar usuarios dentro de un reino.
- /auth/admin/realms/{realm-name}/clients: Este endpoint se utiliza para administrar clientes dentro de un reino.

Los clientes de Keycloak no tienen la necesidad de utilizar todos sus endpoints sino solamente utilizarán los que cubran sus necesidades. Por ejemplo, un cliente de aplicación probablemente si necesitará acceder al endpoint de obtención de token de acceso, pero posiblemente no necesite acceder al endpoint para administrar los usuarios de un reino.

Del mismo modo, el servicio de firmado implementado no necesita acceder a muchos de los recursos de Keycloak y solamente necesita acceder al listado de claves públicas de los certificados que usa Keycloak para firmar los tokens JWT. Debido a que estas claves no cambian con frecuencia alta, se utiliza un recurso del framework Spring para cachear las respuestas y así reducir el número de interacciones entre el servicio de firmado implementado y Keycloak. La URL y el identificador del certificado al que el servicio de firmado deberá acceder están configurados en un fichero de propiedades (application.properties) con el siguiente contenido:

```
keycloak.jwt-set-uri=https://localhost:8081/realms/digsign4einvoices/protocol/openid-connect/certs
keycloak.cert-id=w6k59su9tFIo5YH_OoQFOuytnKc9H-J9GKp0qGZsKU4
```

3.3.3.3. Integración con base de datos para comprobar si el token recibido ha sido utilizado para firmar documentos con anterioridad

La normativa eIDAS obliga a que cada vez que active una clave, el servicio debe tener constancia de que el usuario dueño de la clave haya autorizado su uso. Es decir, que, aunque se autentique un usuario una sola vez en el sistema de control de identidad y se genere un token, este token deberá ser de un solo uso de cara al firmado de facturas electrónicas.

Para conseguir que los tokens generados por Keycloak tengan un solo uso en lo que respecta al servicio de firmado de facturas electrónicas, se ha delegado la responsabilidad del aseguramiento de unicidad de uso del token al propio

servicio de firmado. El servicio de firmado registra los tokens usados en una base de datos, de este modo, cuando se recibe una petición se comprueba si el token ya había sido utilizado previamente o no. En caso de que ya hubiera sido usado previamente, entonces se devolvería un error de tipo 401-Unauthorized.

La base de datos utilizada es una base de datos en memoria de tipo H2 y el servicio de firmado se conecta a esta base de datos a través de JPA. El servicio de firmado registra en la tabla correspondiente el resultado de uso de los tokens (siempre que sean tokens bien formado y firmado correctamente), tanto si el resultado es satisfactorio como si es no satisfactorio.

El esquema de la tabla de base de datos es el siguiente:

```
create table tokens (
       id bigint generated by default as identity,
        expiry_time varchar(255),
        token varchar(2000),
        username varchar(255),
        client_app varchar(255),
        result varchar(255),
        description varchar(255),
        primary key (id)
    );
```

3.3.3.4. Comprobación del contenido del token JWT

El token generado por Keycloak es un token de tipo JWT. Este token contiene información relevante para el servicio de firmado, que utilizará para determinar si el tiempo de expiración del token ya ha transcurrido, si el usuario tiene el rol necesario para poder firmar facturas electrónicas en su nombre y si el usuario otorgó el permiso necesario a la aplicación cliente para poder solicitar la firma de facturas electrónicas en su nombre.

El servicio de firmado de facturas electrónicas posee un fichero de propiedades (application.properties) en el que se ha configurado el nombre del rol necesario que deberá disponer un usuario para poder firmar facturas electrónicas. Además, en el mismo fichero de propiedades también se indica el scope necesario que deberá tener una aplicación cliente para tener autorizada la solicitud de firma de facturas electrónicas en nombre del usuario. La configuración incluida es la siguiente:

```
user.role.authorized=SIGNER_ROLE
user.scope.authorized=invoices-scope
```

3.3.3.5. Integración con base de datos para obtener el alias del certificado digital para firmar facturas electrónicas

Como se ha comentado en el apartado anterior, el token que recibe el servicio de firmado de facturas electrónicas es un token de tipo JWT y contiene cierta información, entre la que se encuentra el nombre del usuario que autoriza la firma de la factura electrónica. A partir de este nombre de usuario el servicio de firmado debe determinar qué certificado digital elegir para firmar digitalmente

las facturas electrónicas. El mecanismo utilizado para hacer esa correlación es el uso de una tabla en base de datos que relaciona el nombre de usuario y el alias del certificado electrónico almacenado en el almacén de claves.

De este modo, el servicio de firmado consultará a la base de datos cuál es el alias del certificado a partir del nombre de usuario que ha autorizado la firma. Además, también se ha incluido un campo en la base de datos para determinar si un usuario está habilitado o no. En caso de que no se encuentre un alias de certificado electrónico asociado a un usuario o que el usuario se encuentre configurado como deshabilitado, entonces se devolvería un error de tipo 401-Unauthorized.

El esquema de la tabla de base de datos es el siguiente:

```
create table alias_certificate (
       id bigint generated by default as identity,
        alias varchar(255),
        create_at varchar(255),
        enabled varchar(255),
        username varchar(255),
        primary key (id)
    );
```

Las inserciones en base de datos iniciales para configurar a dos usuarios son las siguientes:

```
INSERT INTO ALIAS_CERTIFICATE (USERNAME, ALIAS, ENABLED, CREATE_AT)
VALUES('fgonzalezhernandez', 'cert-invoices', 'true', '1684261747');
INSERT INTO ALIAS_CERTIFICATE (USERNAME, ALIAS, ENABLED, CREATE_AT) VALUES('user-no-
cert', 'fake', 'true', '1684261748');
```

3.3.3.6. Comprobación del formato de la factura

En otros apartados del presente TFM se ha indicado que las facturas electrónicas pueden tener diferentes formatos. Sin embargo, para el caso de la implementación práctica se ha considerado únicamente el firmado de las facturas electrónicas en formato 'pdf'. Por ese motivo, las facturas deberán estar en formato "pdf" y con adjuntadas con el nombre de clave "invoice", aunque el nombre del fichero de la factura adjunta será indiferente. En caso de recibirse una factura en otro formato o con otro nombre de clave diferente a "invoice", el servicio de firmado devolverá una respuesta de tipo 400-Bad request.

3.3.3.7. Acceso al almacenamiento de certificados digitales

El almacén de claves elegido para la implementación del caso práctico es de tipo JKS. Este almacén de claves está protegido con contraseña y contiene el certificado digital necesario para firmar facturas electrónicas, que está en formato p12 y a su vez también está protegido con contraseña.

El servicio de firmado debe tener acceso al directorio donde se encuentre el fichero de tipo JKS. A través de un fichero de propiedades se determina la ruta

del almacén de claves y su contraseña de acceso. Según el entorno de despliegue del servicio de firmado, podría aplicarse un sistema de almacén de secretos para guardar las contraseñas de forma segura. Sin embargo, para el alcance del presente caso práctico no se ha considerado tal implementación. Además, dentro del almacén de claves se encuentra el certificado de firma digital en formato “p12”, que está protegido por contraseña la cual también se almacena en un fichero de propiedades. El servicio de firmado accederá a un certificado u otro según el atributo de usuario “certificate” obtenido a través de la consulta al servicio de Keycloak con el token de acceso.

3.3.3.8. Firma digital de facturas

La factura electrónica que se desea firmar es recibida por el servicio de firmado a través de un adjunto en una petición de tipo POST, pero antes de proceder al firmado de la factura electrónica, el servicio de firmado ya habrá realizado varias validaciones: comprobación del token de acceso, comprobación del rol de usuario, comprobación del formato de la factura electrónica y acceso al almacén de claves y al certificado de firma digital. Cuando el servicio de firmado ha completado todas las validaciones procederá a aplicar la firma digital y anexarla en un nuevo pdf que devolverá como adjunto en la respuesta de la petición de tipo POST.

El tipo de firmado que se aplica al fichero PDF es CAdES. Además, se genera un cuadro en la esquina superior derecha de la primera página del documento con datos básicos sobre la firma. De este modo, la firma permitirá ser verificada visualmente de forma rápida además de poder utilizarse otros métodos de verificación de firma.

3.3.3.9. Detalles de la implementación

El servicio de firmado se ha implementado en Java bajo el framework Spring Boot, JDK 17 y se ha utilizado para el desarrollo el IDE Eclipse, en su versión 2022-06 (4.24.0). Además de usar las librerías del propio Spring Boot, se han utilizado librerías externas de bouncycastle y de itextpdf.

A continuación, se expone el contenido del fichero “pom.xml” con la configuración y dependencias del proyecto:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.0.3</version>
		<relativePath />
	</parent>
	<groupId>edu.uoc.fgonzalezhernandez</groupId>
	<artifactId>InvoiceDigitalSign</artifactId>
	<version>1.0.0-RELEASE</version>
	<name>InvoiceDigitalSign</name>
```

```
	<description>TFM - InvoiceDigitalSign</description>

	<properties>
		<java.version>17</java.version>
	</properties>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>com.auth0</groupId>
			<artifactId>jwks-rsa</artifactId>
			<version>0.12.0</version>
		</dependency>
		<dependency>
			<groupId>com.auth0</groupId>
			<artifactId>java-jwt</artifactId>
			<version>3.8.3</version>
		</dependency>
		<dependency>
			<groupId>org.bouncycastle</groupId>
			<artifactId>bcprov-jdk15on</artifactId>
			<version>1.69</version>
		</dependency>
		<dependency>
			<groupId>org.bouncycastle</groupId>
			<artifactId>bcpkix-jdk15on</artifactId>
			<version>1.69</version>
		</dependency>
		<dependency>
			<groupId>com.itextpdf</groupId>
			<artifactId>itextpdf</artifactId>
			<version>5.5.13.3</version>
		</dependency>
		<dependency>
			<groupId>com.itextpdf</groupId>
			<artifactId>kernel</artifactId>
			<version>7.2.5</version>
		</dependency>
		<dependency>
			<groupId>com.itextpdf</groupId>
			<artifactId>sign</artifactId>
			<version>7.2.5</version>
		</dependency>


		<dependency>
			<groupId>com.h2database</groupId>
			<artifactId>h2</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>

	</dependencies>
</project>
```

Se han implementado los siguientes paquetes:

- edu.uoc.fgonzalezh.digsign: paquete padre en donde se encuentra la clave encargada del arranque del servicio.
- edu.uoc.fgonzalezh.digsign.controller: paquete que contiene el controlador del servicio
- edu.uoc.fgonzalezh.digsign.utils: paquete que contiene utilidades varias
- edu.uoc.fgonzalezh.digsign.mapper: paquete que contiene una clase de tipo “mapper”
- edu.uoc.fgonzalezh.digsign.service: paquete que contiene los servicios desarrollados tales como accesos a bases de datos, acceso a almacén de claves, procesado de token JWT y firma digital de documentos electrónicos
- edu.uoc.fgonzalezh.digsign.repository: paquete que contiene los repositorios de acceso a base de datos
- edu.uoc.fgonzalezh.digsign.entity: paquete que contiene las entidades para el acceso a base de datos

Además de los paquetes indicados con sus clases correspondientes, también se tiene preparado un fichero de propiedades con las configuraciones necesarias para la aplicación. El contenido del fichero “application.properties” es el siguiente:

```
server.port=8088
spring.application.name=signinvoices
logging.level.root=INFO

user.role.authorized=SIGNER_ROLE
user.scope.authorized=invoices-scope

sign.creator=SignInvoiceService
sign.reason=Invoice validated
sign.location=Spain/Madrid

keystore.filepath=/InvoiceDigitalSign/src/main/resources/
keystore.filename=keystore-invoices.jks
keystore.password=uoc2022
keystore.cert.password=uoc2022

keycloak.jwt-set-uri=https://localhost:8081/realms/digsign4einvoices/protocol/openid-
connect/certs
keycloak.cert-id=w6k59su9tFIo5YH_OoQFOuytnKc9H-J9GKp0qGZsKU4

# Enabling H2 Console
spring.datasource.url: jdbc:h2:mem:testdb
spring.datasource.driverClassName: org.h2.Driver
spring.datasource.username: sa
spring.datasource.password: sa
spring.jpa.database-platform: org.hibernate.dialect.H2Dialect
spring.h2.console.enabled: true
# Show all queries
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

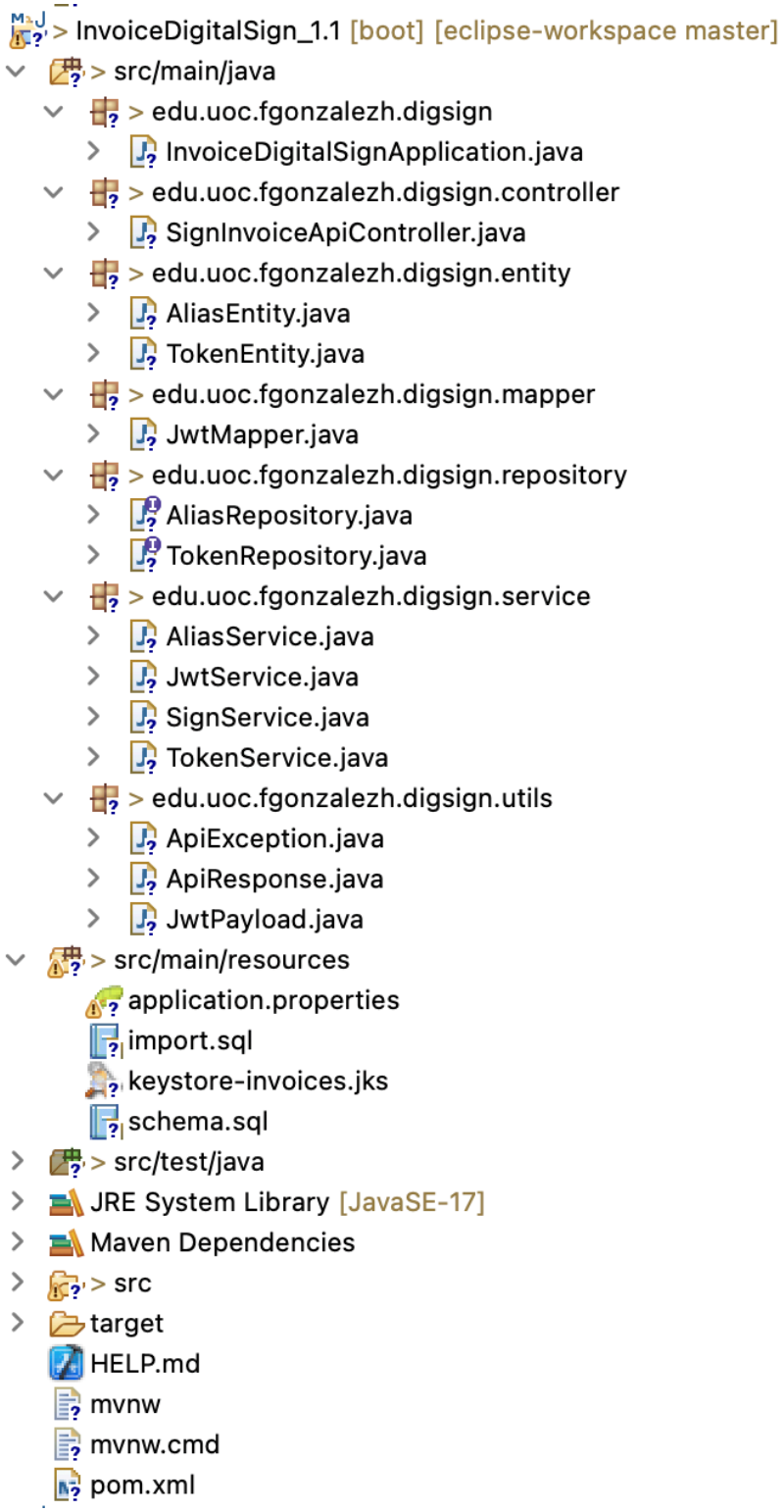

Figura 3.35: Componentes del proyecto del servicio de firmado de facturas

3.3.3.10. Validación del servicio de firmado

A continuación, se realiza una prueba básica del funcionamiento del servicio de firmado. La prueba consiste en enviar una petición de tipo POST al servicio de firmado con un token de acceso válido y adjuntando una factura en formato PDF. Posteriormente se comprobarán las trazas emitidas por el servicio de

firmado, la respuesta obtenida a la petición de tipo POST y se abrirá la factura firmada por el servicio desde la herramienta Acrobat Reader para la verificación de la firma.

Los pasos seguidos para la realización de la prueba son los siguientes:

1. Generación de una factura electrónica en formato PDF
2. Obtención de un token de acceso en Keycloak con un usuario y una aplicación correctamente configurados en Keycloak.
3. Preparación de una petición desde el cliente Postman en la que se incluye en la cabecera “Authorization” el token de acceso y se adjunta en el body la factura en formato PDF
4. Envío de la petición al servicio de firmado
5. Obtención de respuesta en Postman del servicio de firmado
6. Guardar la factura electrónica firmada en un directorio local el PDF respondido por el servicio de firmado
7. Exportar la factura electrónica firmada a un equipo con Acrobat Reader y los certificados digitales de confianza instalados.
8. Abrir la factura electrónica firmada y visualizar el detalle de la verificación de firma

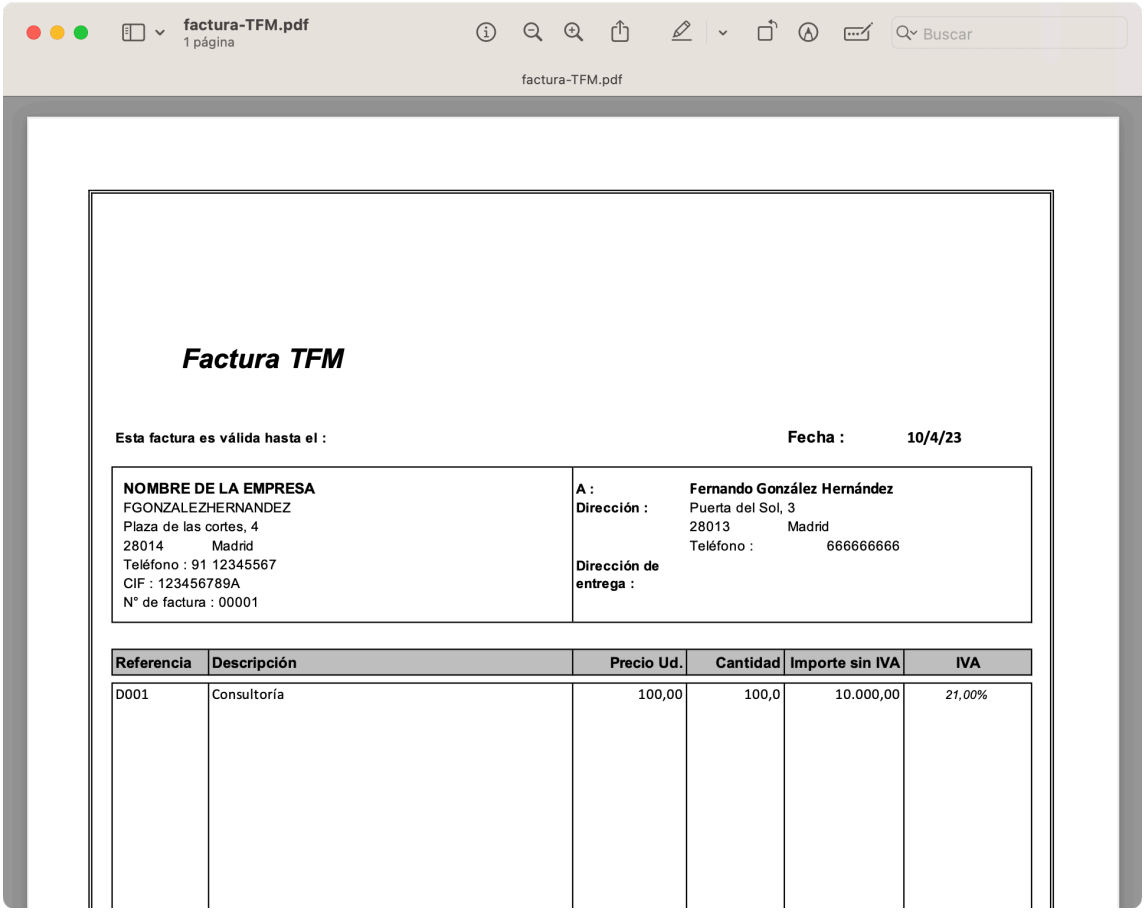

Figura 3.36: Factura en formato PDF sin firmar (paso 1)

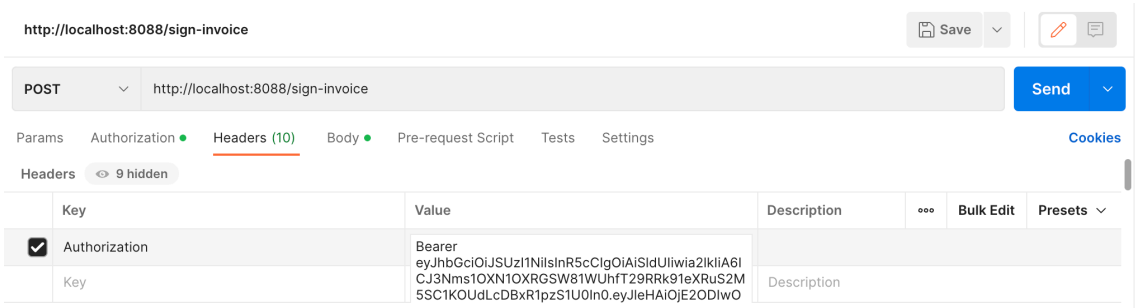

Figura 3.37: Preparación de petición con Postman: añadir cabecera Authorization (paso 3)

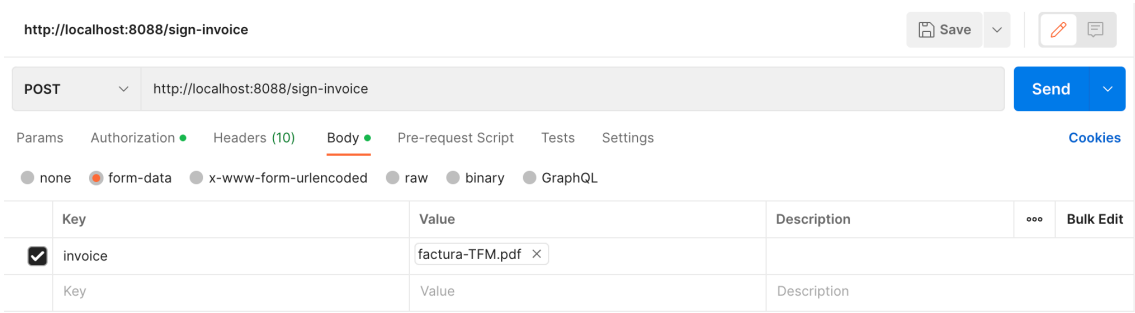

Figura 3.38: Preparación de petición con Postman: adjuntar factura electrónica (paso 3)

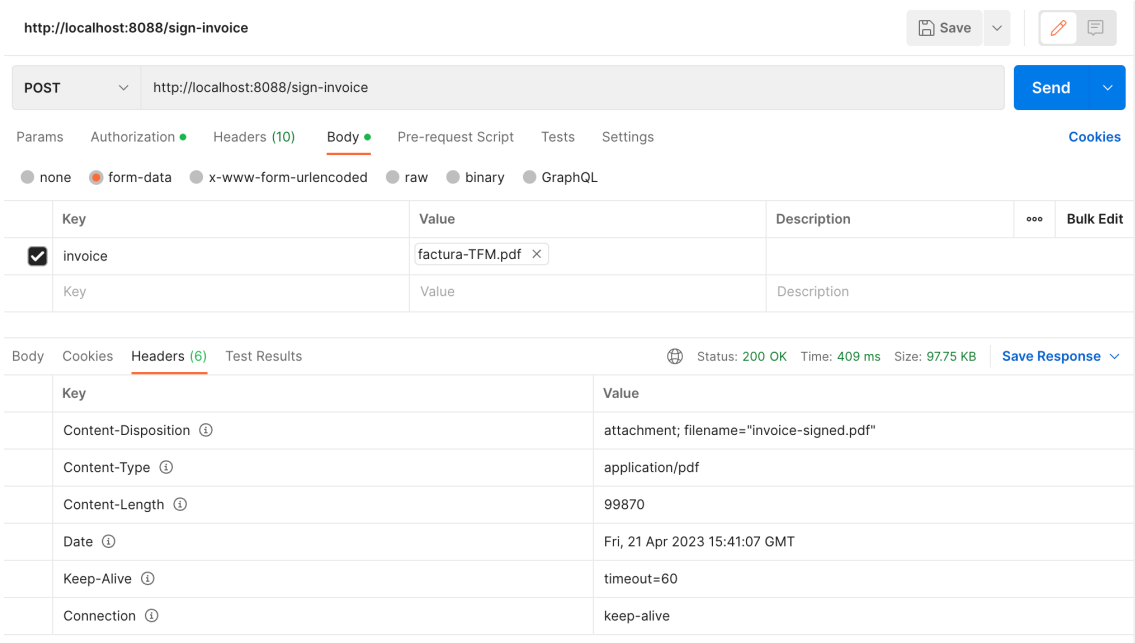

Figura 3.39: Respuesta recibida en Postman (headers)

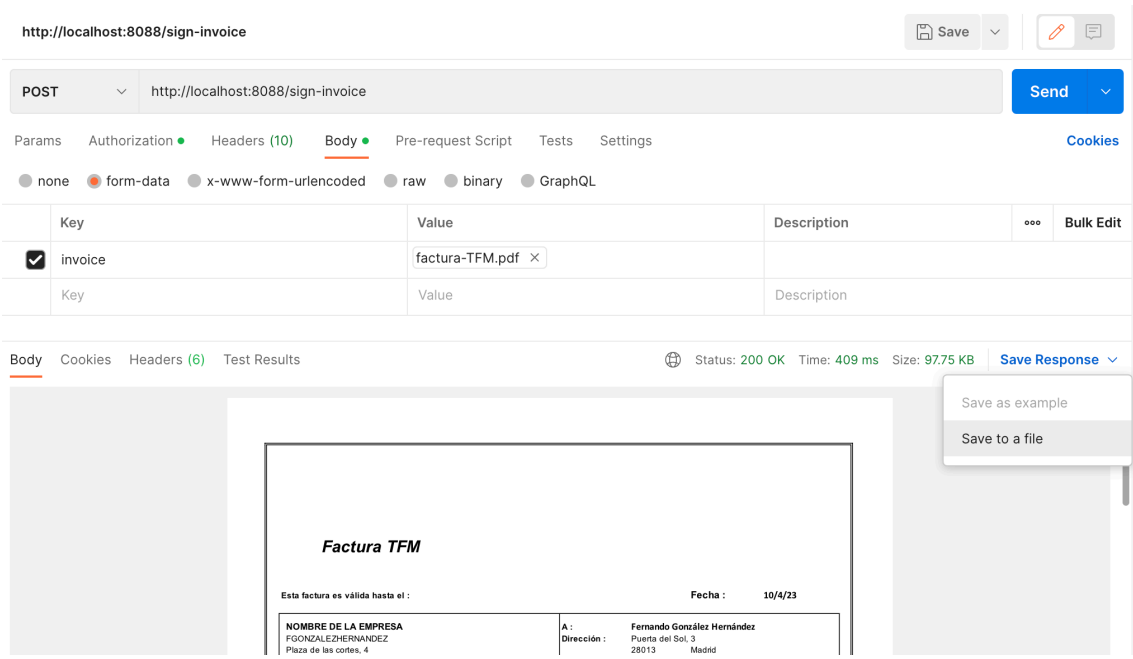

Figura 3.40: Respuesta recibida en Postman (body) y almacenado de factura firmada

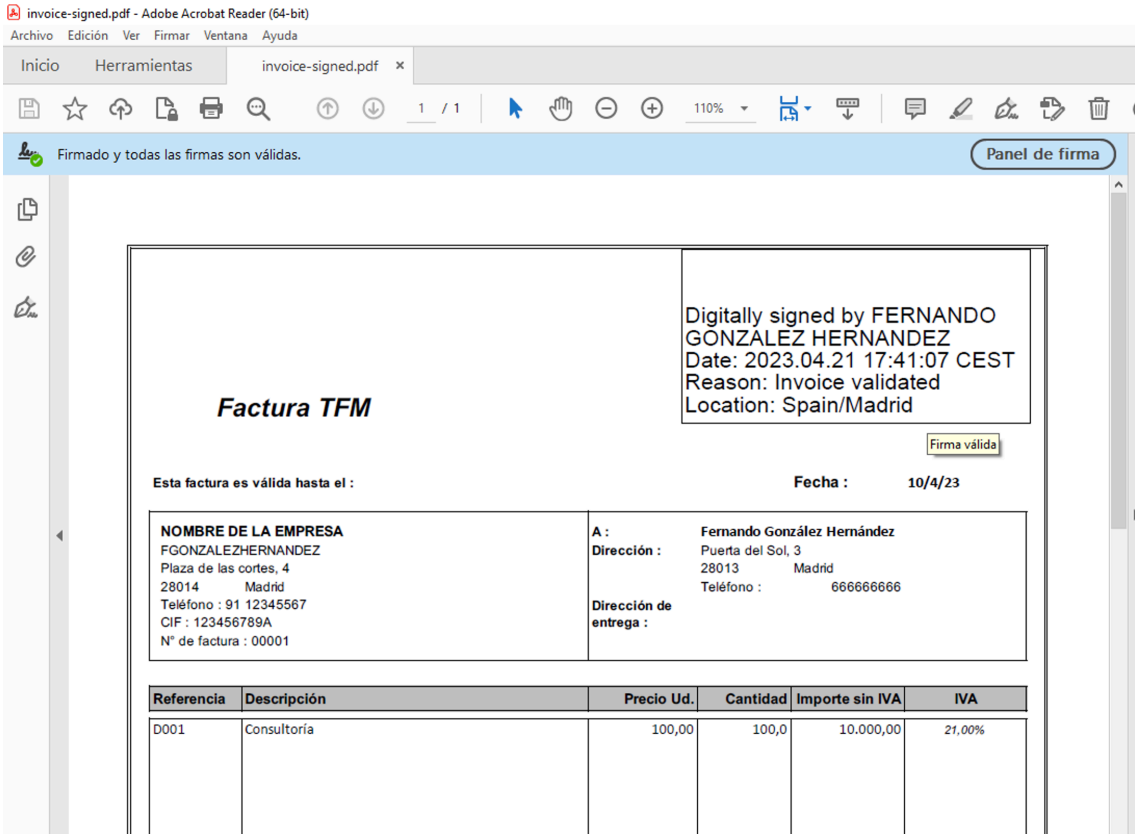

Figura 3.41: Apertura de la factura firmada desde un equipo con Acrobat Reader instalado

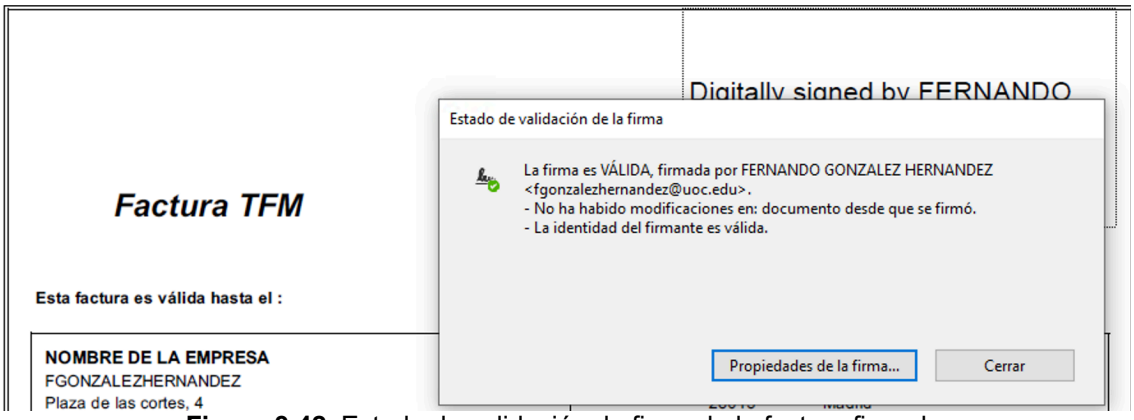

Figura 3.42: Estado de validación de firma de la factura firmada

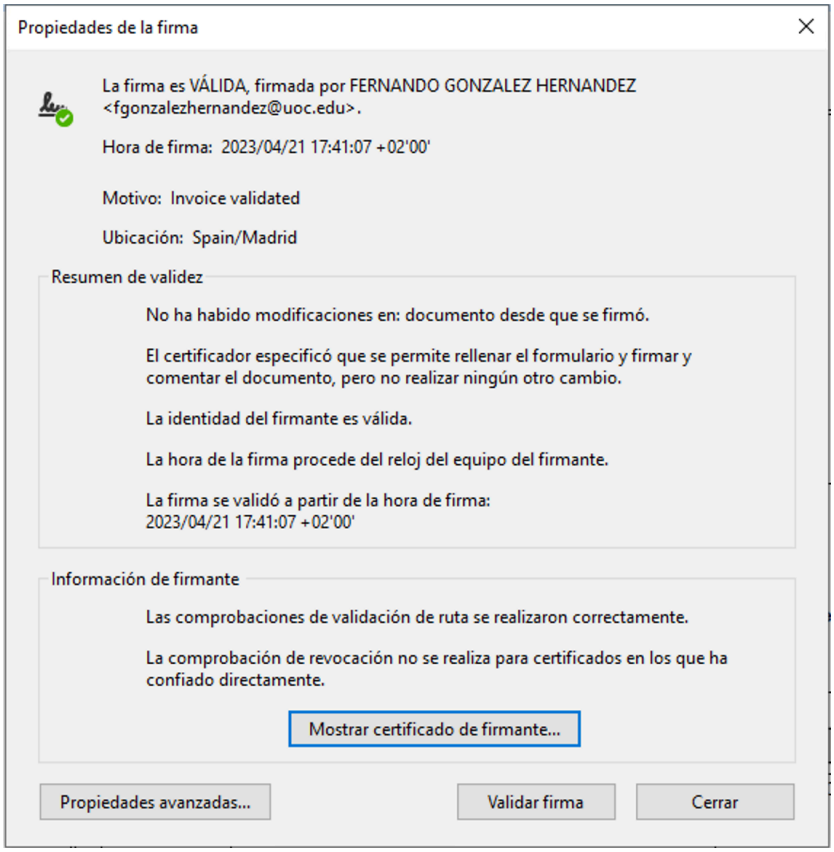

Figura 3.43: Propiedades de la firma de la factura firmada

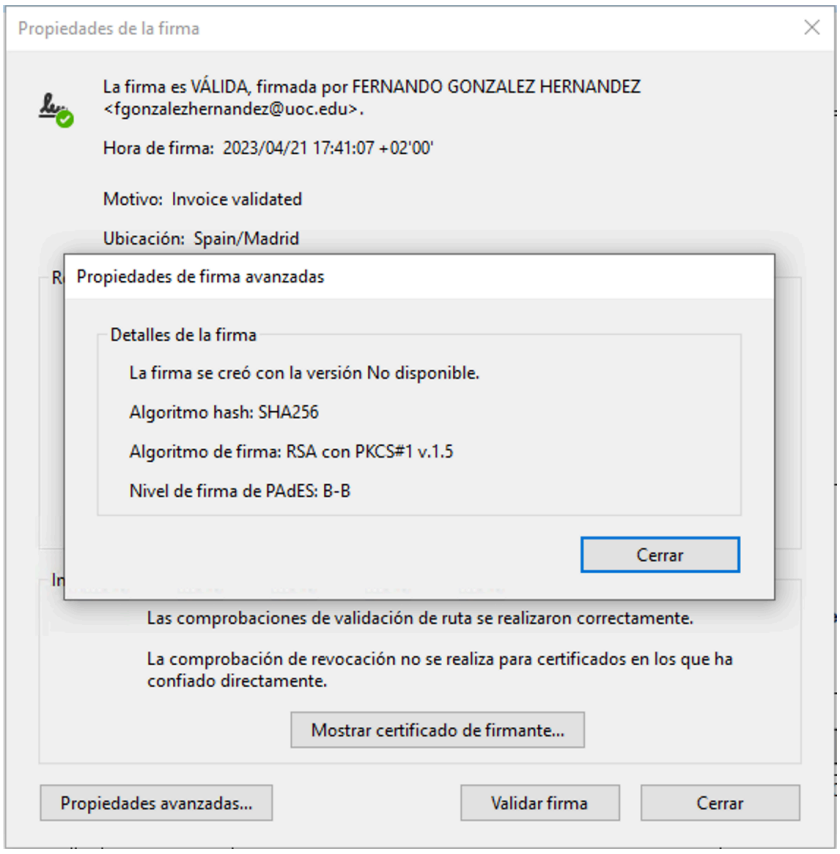

Figura 3.44: Propiedades de la firma avanzadas de la factura firmada

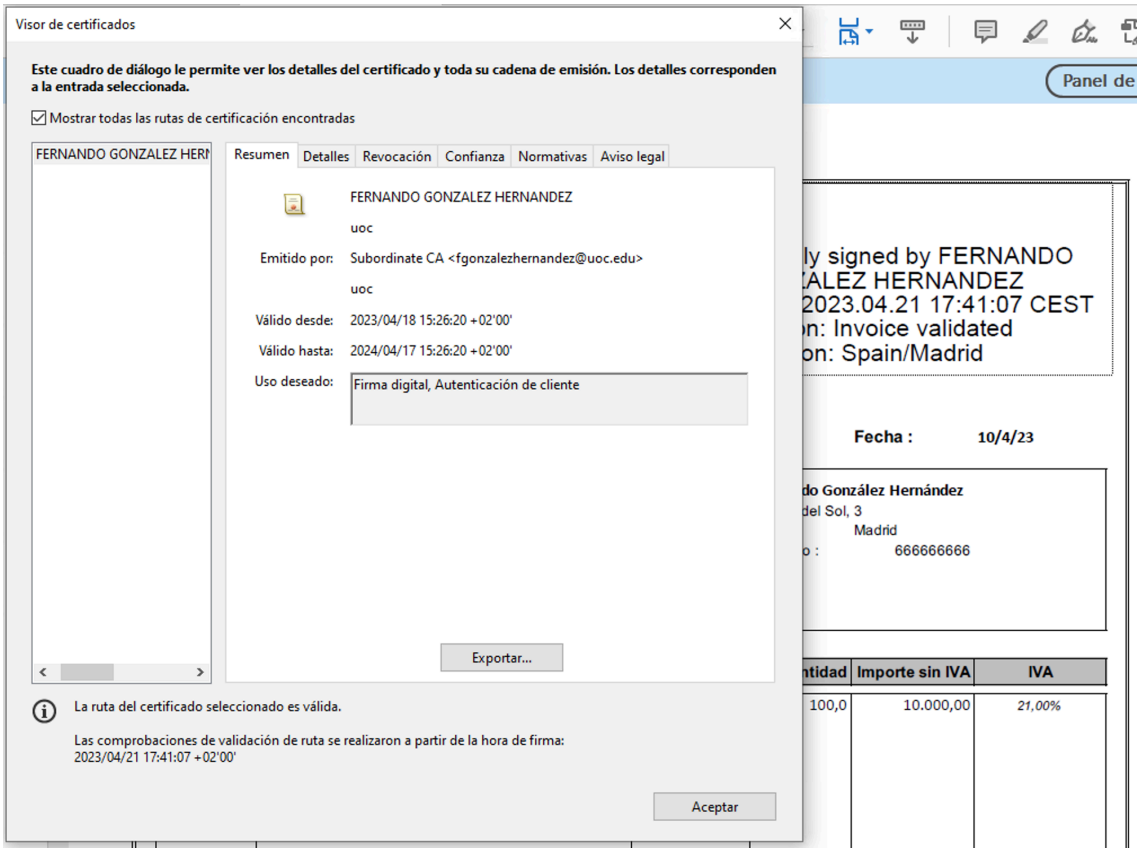

Figura 3.45: Visión del certificado de firma de la factura firmada

3.4. Pruebas

La finalidad de las pruebas es verificar que la implementación del caso práctico cumpla con los requisitos iniciales establecidos y que funcione correctamente. Durante el desarrollo de la implementación del caso práctico se han ido realizando pruebas parciales de los componentes del sistema y se han documentado en los apartados de desarrollo correspondientes. Por ese motivo, en el presente apartado se documentan las pruebas integrales del sistema a partir de la definición de los diferentes casos de uso. Es decir, por cada caso de uso se realiza una prueba que demuestra el comportamiento del servicio desarrollado.

3.4.1. Prueba 01: la aplicación cliente realiza un login satisfactorio con el sistema de control de identidad

El objetivo de la prueba '01' es probar la funcionalidad del caso de uso "01 – La aplicación cliente realiza login satisfactorio con el sistema de control de identidad". Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita. Además, se utiliza la herramienta Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo "Authorization Code".

La prueba a realizar es la misma que la desarrollada en el apartado "3.3.1.4 Pruebas básicas de Keycloak" del presente TFM. Por ese motivo, en el presente apartado se repite la prueba, pero los detalles explicativos de la misma pueden ser consultados en el apartado "3.3.1.4 Pruebas básicas de Keycloak".

Se utiliza un recurso de Postman para provocar el flujo "Authorization Code Flow" de una manera sencilla y así conseguir obtener el token de acceso. Este método consiste en generar un nuevo "request" en cuyo apartado "Authorization" se selecciona "Type = OAuth 2.0" y se rellenan los valores del formulario con los datos correspondientes:

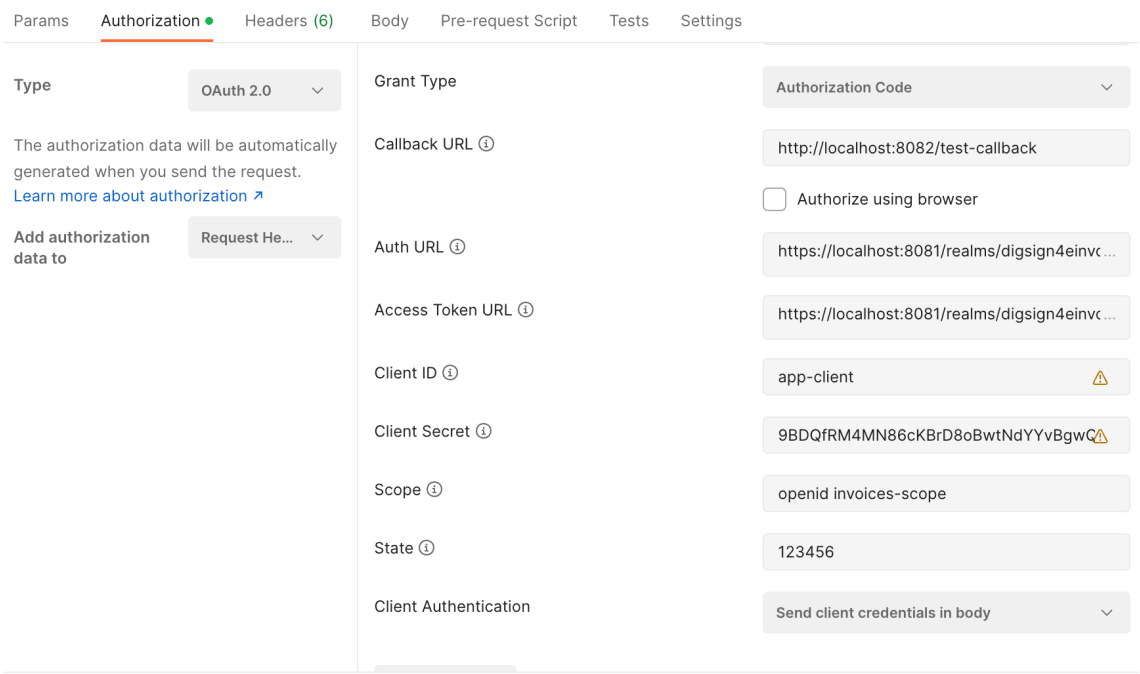

Figura 3.46: Configuración de Postman para obtener token de acceso con el flujo "Authorization Code"

Se pulsa el botón "Get New Access Token" de Postman y, a continuación, se abre una ventana en la que aparece el formulario de Keycloak para que el usuario introduzca sus datos de acceso al sistema de control de identidad.

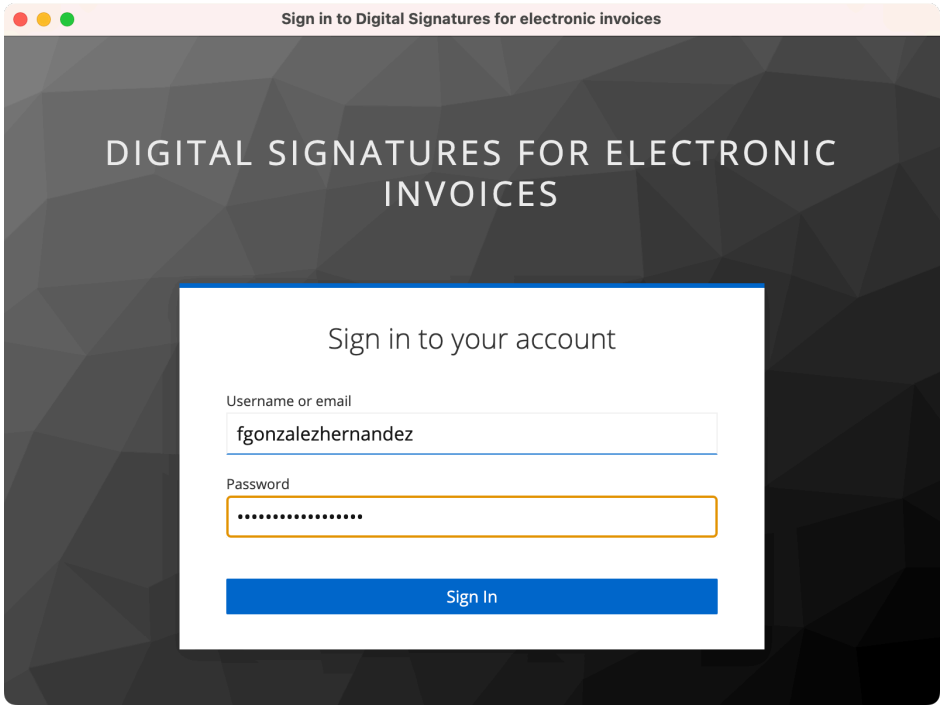

Figura 3.47: Formulario para que el usuario introduzca sus credenciales de usuario

Una vez que el usuario ha introducido correctamente sus datos de acceso, Keycloak devuelve a Postman un token de acceso. A continuación, se muestra el token de acceso devuelto:


```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJ3Nms1OXN1OXRGSW81WUhfT29RRk91eXRuS2M5
SC1KOUdLcDBxR1pzS1U0In0.eyJleHAiOjE2ODIyNDQ2ODYsImlhdCI6MTY4MjI0NDM4NiwiYXV0aF90aW1lIjox
NjgyMjQ0Mzg2LCJqdGkiOiIxMDViMzk5Zi00YjRlLTQ0ODYtYTMwOC1kMDdmNThhMmY0MmEiLCJpc3MiOiJodHRw
czovL2xvY2FsaG9zdDo4MDgxL3JlYWxtcy9kaWdzaWduNGVpbnZvaWNlcyIsInN1YiI6ImNlZjk5ZDE0LTBkOTAt
NDk1ZS04OTRhLWZjMzIyYjk0ODRlMSIsInR5cCI6IkJlYXJlciIsImF6cCI6ImFwcC1jbGllbnQiLCJzZXNzaW9u
X3N0YXRlIjoiZmRlMDUwNmUtYmMwOC00NDVjLWE1MTktZjlkZGYzMmU2OTFmIiwicmVhbG1fYWNjZXNzIjp7InJv
bGVzIjpbIlNJR05FUl9STOxFIl19LCJzY29wZSI6Im9wZW5pZCBpbnZvaWNlcy1zY29wZSBwcm9maWxlIiwic2lk
IjoiZmRlMDUwNmUtYmMwOC00NDVjLWE1MTktZjlkZGYzMmU2OTFmIiwibmFtZSI6IkZlcm5hbmRvIEdvbnphbGV6
IEhlcm5hbmRleiIsInByZWZlcnJlZF91c2VybmFtZSI6ImZnb256YWxlemhlcm5hbmRleiIsImdpdmVuX25hbWUi
OiJGZXJuYW5kbyIsImZhbWlseV9uYW1lIjoiR29uemFsZXogSGVybmFuZGV6In0.Hjmxqfx0Y4Jk8FPRSgZKZhkG
twhtcSGQlK-bRarOtlFOgyR3SpYIhpNsR_nTzdBfeAqoKL-nSIz-fM75PFeN-
uHrz3QCD5FlU3_WdO6lp1e2BAm1ZkXg0-
gFyGtXizrN0_q8Epzx00oClkJOk_YdgjdsgxSW5VBi3RwIxJdo0XiZq68QCUBIaw1B_RSZrasYHg2YdonwbCu3WI
hVJhDB5NP145u-GSnH5ypBFkXkHKpuO2KIv1M99bc-WEckXXGhtggADxXQ21DAXBHl67Tz0qm7oI5aJ-
l958QwKueKe-LiMJwdRi-kz9FhNXnkEUyYRAzDFkBWfp8BbZFtO_gfOQ
```


El token de acceso tiene formato JWT. Descifrando su contenido se consigue lo siguiente:

Header	{
	"alg": "RS256",
	"typ": "JWT",
	"kid": "w6k59su9tFIo5YH_OoQFOuytnKc9H-J9GKp0qGZsKU4"
	}

Payload	{
	"exp": 1682244686,
	"iat": 1682244386,
	"auth_time": 1682244386,
	"jti": "105b399f-4b4e-4486-a308-d07f58a2f42a",
	"iss": "https://localhost:8081/realms/digsign4einvoices",
	"sub": "cef99d14-0d90-495e-894a-fc322b9484e1",
	"typ": "Bearer",
	"azp": "app-client",
	"session_state": "fde0506e-bc08-445c-a519-f9ddf32e691f",
	"realm_access": {
	"roles": [
	"SIGNER_ROLE"
	]
	},
	"scope": "openid invoices-scope profile",
	"sid": "fde0506e-bc08-445c-a519-f9ddf32e691f",
	"name": "Fernando Gonzalez Hernandez",
	"preferred_username": "fgonzalezhernandez",
	"given_name": "Fernando",
	"family_name": "Gonzalez Hernandez"
	}

El resultado de ejecución de la prueba ‘02’ resulta satisfactorio y verifica el cumplimiento del caso de uso ‘02’.

3.4.2. Prueba 02: la aplicación cliente realiza un login satisfactorio con el sistema de control de identidad

El objetivo de la prueba ‘02’ es probar la funcionalidad del caso de uso “02– La aplicación cliente realiza login insatisfactorio con el sistema de control de identidad por introducir mal sus credenciales”. Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a

ejecutar los pasos de la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza la herramienta Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo “Authorization Code”.

Para la ejecución de esta prueba, se introducen incorrectamente las credenciales de usuario durante el proceso de login. Como resultado se obtiene que el sistema Keycloak no valida las credenciales y vuelve a solicitar al usuario que introduzca las correctas.

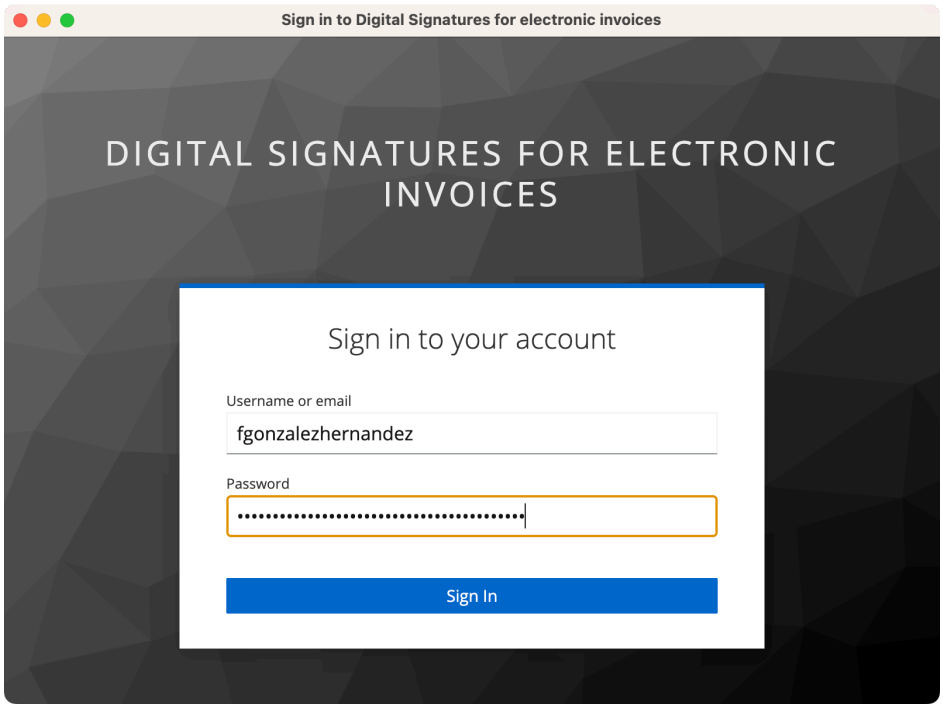

Figura 3.48: Introducción incorrecta de credenciales de usuario

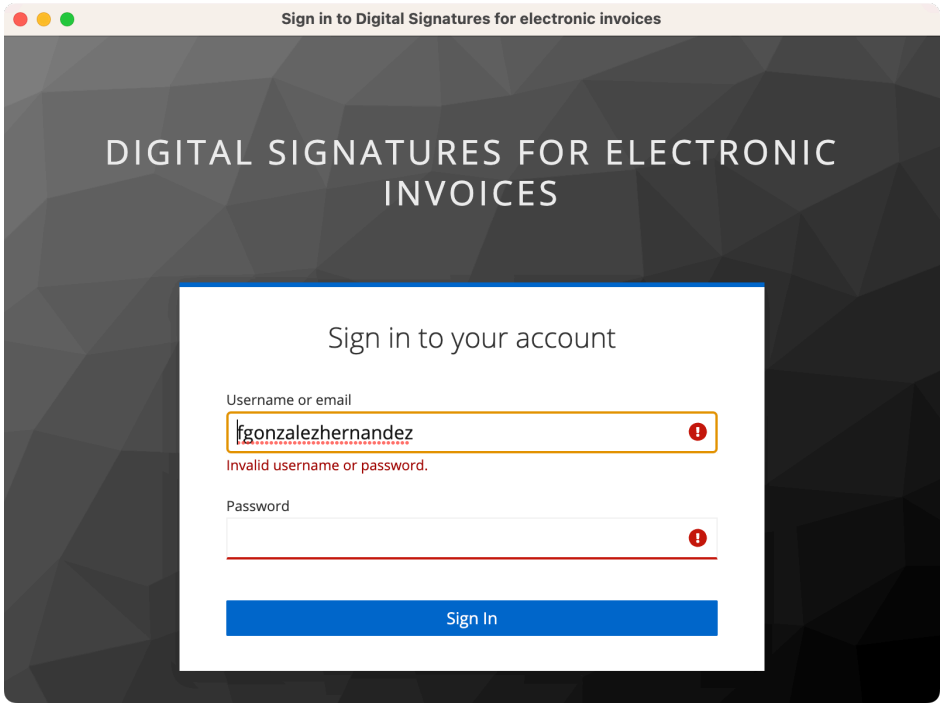

Figura 3.49: El sistema Keycloak no valida la introducción incorrecta de credenciales

El resultado de ejecución de la prueba ‘02’ resulta satisfactorio y verifica el cumplimiento del caso de uso ‘02’.

3.4.3. Prueba 03: la aplicación cliente realiza login insatisfactorio con el sistema de control de identidad por introducir mal su OTP

El objetivo de la prueba ‘03’ es probar la funcionalidad del caso de uso “03– La aplicación cliente realiza login insatisfactorio con el sistema de control de identidad por introducir mal su OTP”. Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza la herramienta Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo “Authorization Code”.

Para la ejecución de esta prueba, se introducen correctamente las credenciales de usuario durante el proceso de login, pero se introduce incorrectamente el código OTP. Como resultado se obtiene que el sistema Keycloak no valida las credenciales y vuelve a solicitar al usuario que introduzca las correctas.

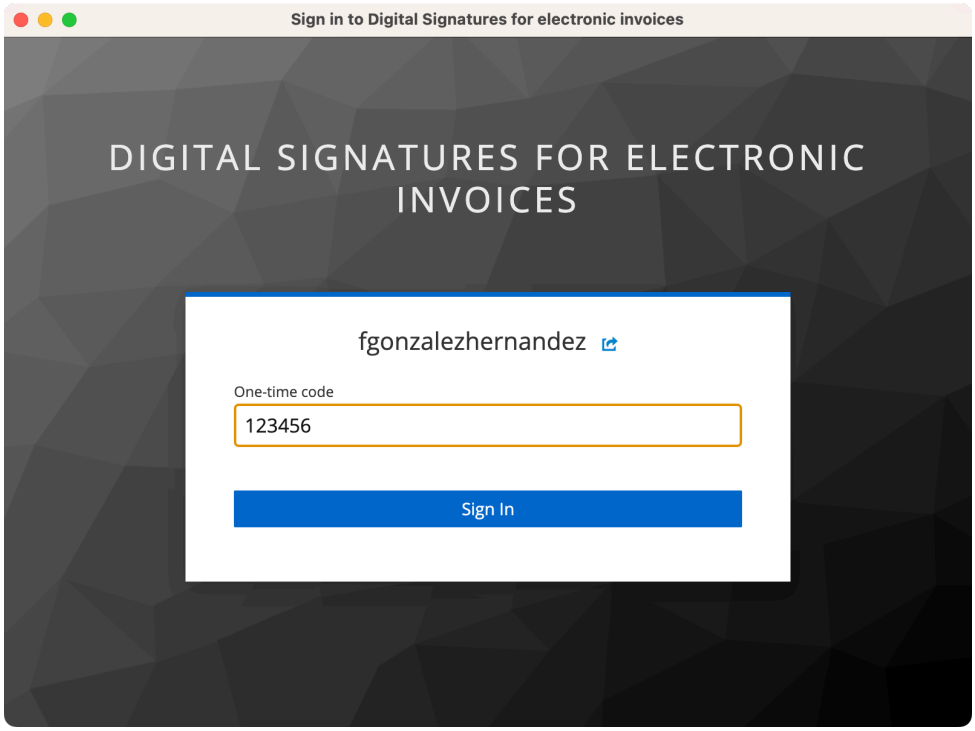

Figura 3.50: Introducción incorrecta del código OTP

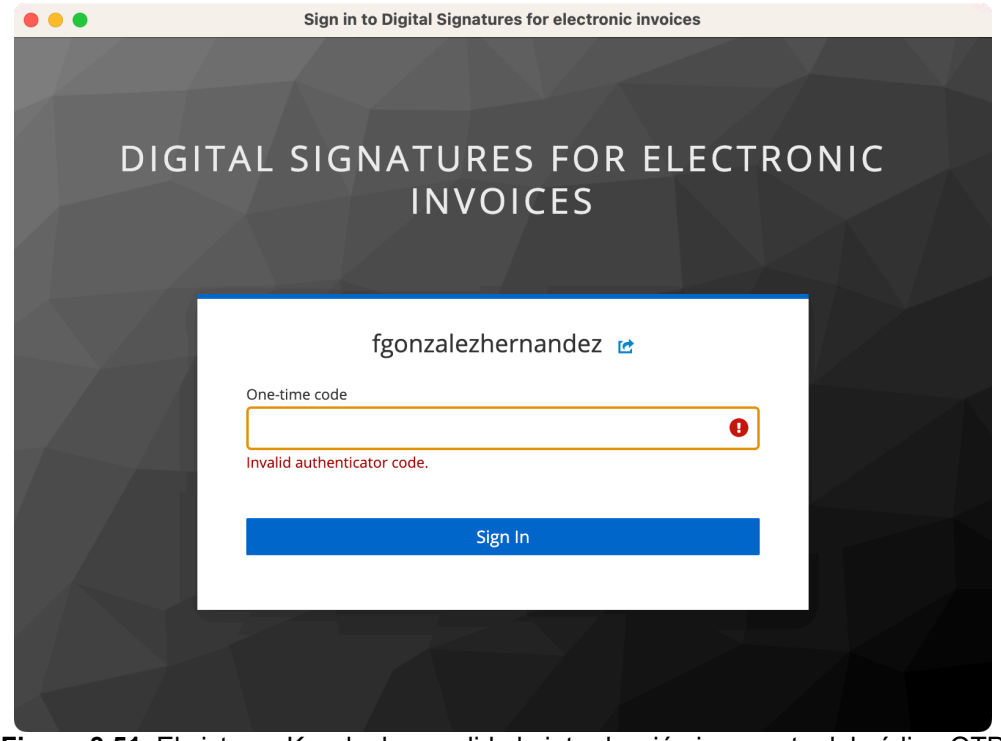

Figura 3.51: El sistema Keycloak no valida la introducción incorrecta del código OTP

El resultado de ejecución de la prueba '03' resulta satisfactorio y verifica el cumplimiento del caso de uso '03'.

3.4.4. Prueba 04: la aplicación cliente envía una petición al servicio de firmado y obtiene una factura electrónica firmada digitalmente

El objetivo de la prueba '04' es probar la funcionalidad del caso de uso "04– La aplicación cliente envía una petición al servicio de firmado y obtiene una factura electrónica firmada digitalmente". Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita.

La prueba a realizar es la misma que la desarrollada en el apartado "3.3.3.7 Validación del servicio de firmado" del presente TFM. Por ese motivo, en el presente apartado se repite la prueba, pero los detalles explicativos de la misma pueden ser consultados en el apartado "3.3.3.7. Validación del servicio de firmado".

Para la ejecución de esta prueba, se introducen correctamente las credenciales de usuario, el código OTP, se genera correctamente la petición al servicio de firmado desde Postman y se envía la petición al servicio de firmado adjuntando una factura sin firmar en formato PDF. Como resultado se obtiene que el servicio de firmado devolverá la factura firmada en formato PDF.

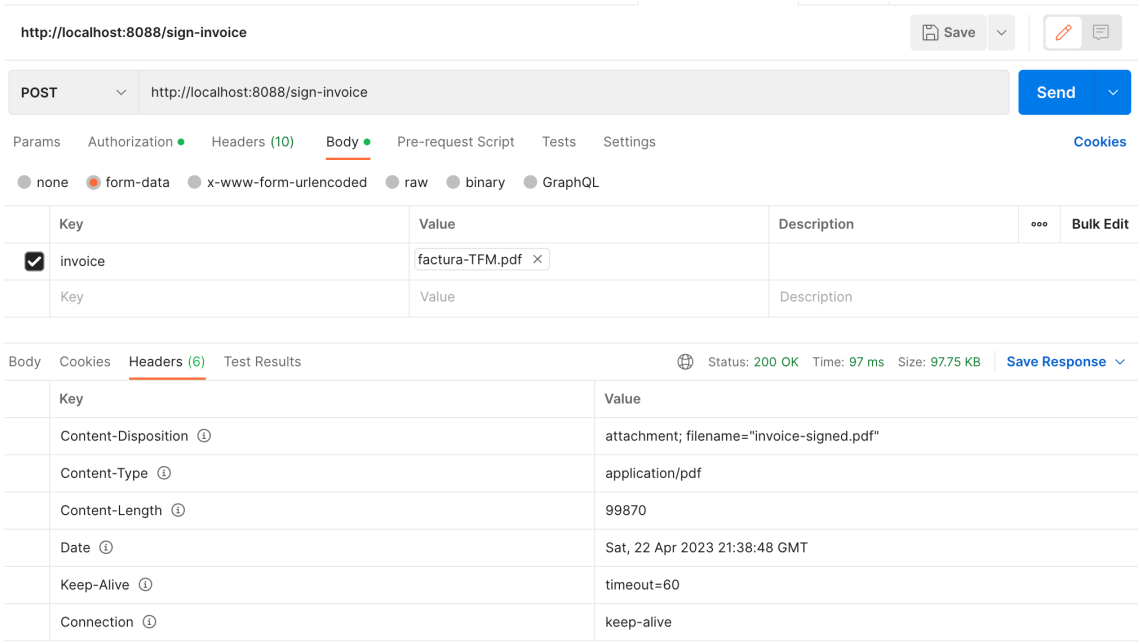

Figura 3.52: Respuesta de la petición enviada al servicio de firmado

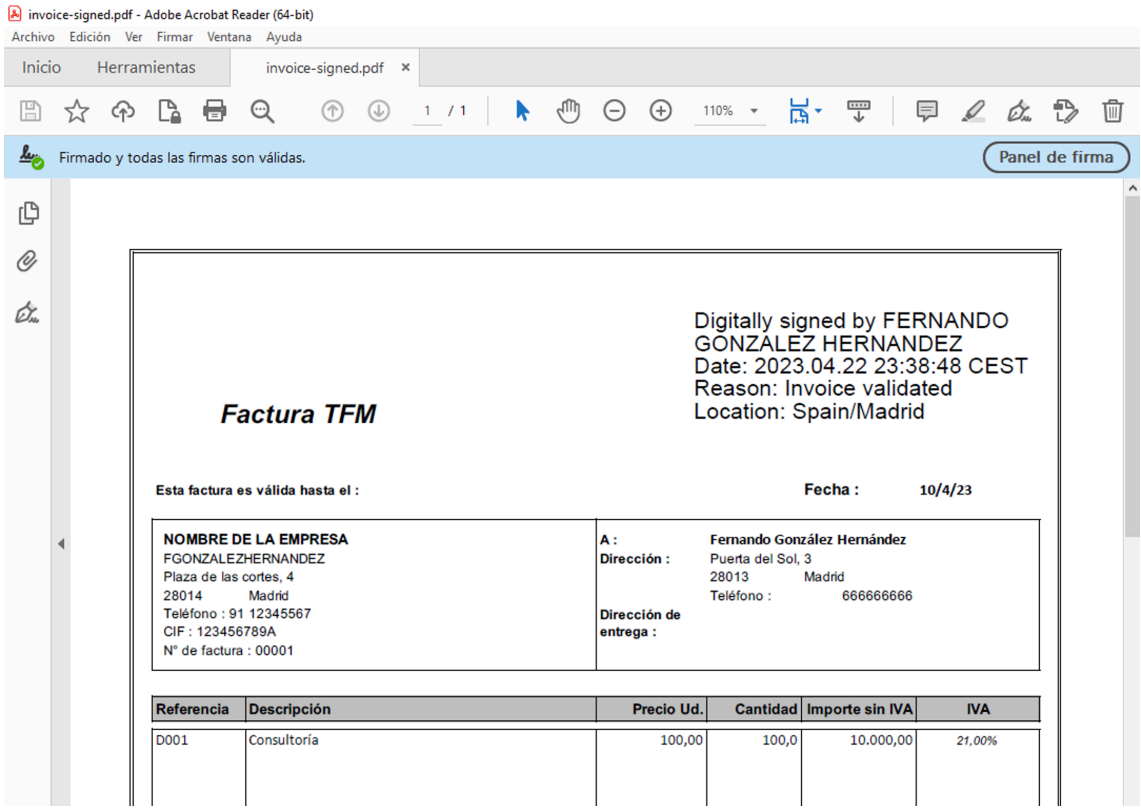

Figura 3.52: Comprobación de la factura firmada

El resultado de ejecución de la prueba ‘04’ resulta satisfactorio y verifica el cumplimiento del caso de uso ‘04’.

3.4.5. Prueba 05: la aplicación cliente envía una petición con token de acceso no válido al servicio de firmado

El objetivo de la prueba '05' es probar la funcionalidad del caso de uso "05– La aplicación cliente envía una petición con token de acceso no válido al servicio de firmado". Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo "Authorization Code" para obtener el token de acceso.

Para la ejecución de esta prueba, se adjunta correctamente la factura a la petición, pero se introduce incorrectamente el token de acceso en la cabecera "Authorization". Finalmente se envía la petición al servicio de firmado. Como resultado se obtiene que el servicio de firmado devuelve un error de tipo 400-Bad Request.

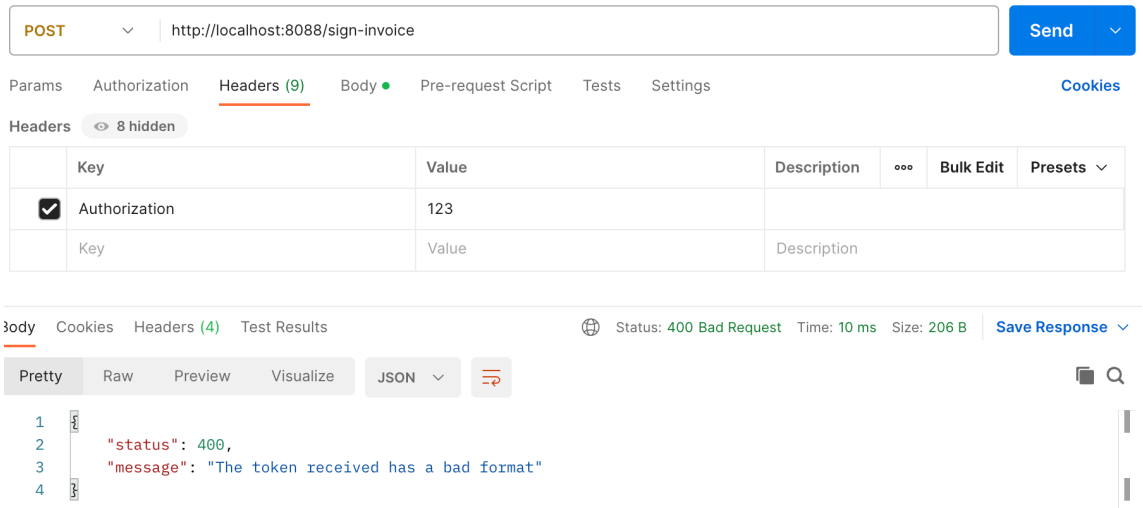

Figura 3.53: Respuesta del servicio de firmado cuando recibe un token inválido

El resultado de ejecución de la prueba '05' resulta satisfactorio y verifica el cumplimiento del caso de uso '05'.

3.4.6. Prueba 06: la aplicación cliente envía una petición al servicio de firmado utilizando un token que ya ha sido utilizado previamente para firmar una factura electrónica

El objetivo de la prueba '06' es probar la funcionalidad del caso de uso "06– La aplicación cliente envía una petición al servicio de firmado utilizando un token que ya ha sido utilizado previamente para firmar una factura electrónica". Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo "Authorization Code" para obtener el token de acceso.

Para la ejecución de esta prueba, se introducen correctamente las credenciales de usuario, el código OTP, se genera correctamente la petición al servicio de firmado desde Postman y se envía la petición al servicio de firmado adjuntando una factura sin firmar en formato PDF. Como resultado de la primera petición se obtiene la factura firmada en formato PDF.

Posteriormente de la ejecución de la primera petición se accede a la base de datos para comprobar que el token utilizado haya sido almacenado correctamente en la base de datos. De este modo, en un siguiente lanzamiento de la petición con el mismo token, éste ya resultaría inválido. Al acceder a la base de datos se encuentra que el token está almacenado en el registro con "ID=2".

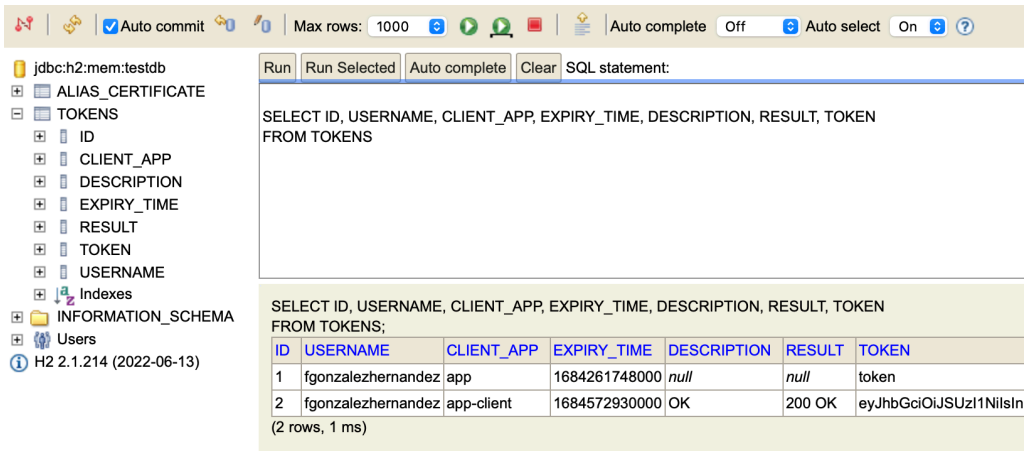

Figura 3.54: Acceso a la base de datos para comprobar que el token haya sido almacenado en la base de datos

Una vez realizado los pasos anteriores, se vuelve a lanzar la petición con el mismo token de autenticación, es decir, intentando reutilizar el token de autenticación. Como resultado de esta segunda petición se obtiene que el servicio de firmado devuelve un error de tipo 401-Unauthorized con un mensaje descriptivo.

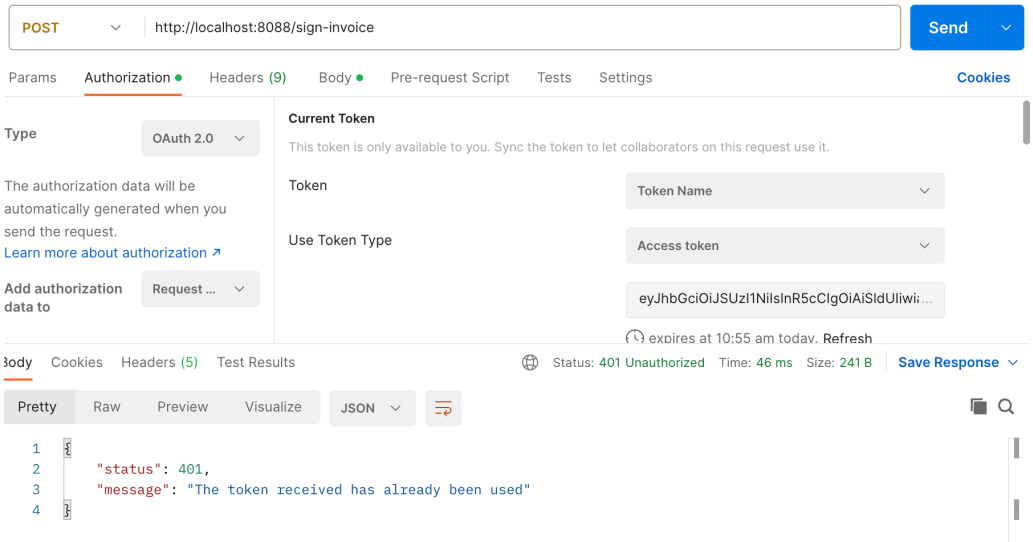

Figura 3.55: Respuesta del servicio de firmado cuando recibe un token que ya ha sido usado previamente

3.4.7. Prueba 07: la aplicación cliente envía una petición al servicio de firmado con token de acceso de un usuario que no tiene permisos para firmar

El objetivo de la prueba '07' es probar la funcionalidad del caso de uso "07– La aplicación cliente envía una petición al servicio de firmado con token de acceso de un usuario que no tiene permisos para firmar". Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo "Authorization Code" para obtener el token de acceso.

Para la ejecución de esta prueba, se hace login con el usuario "user-no-role", que es un usuario dado de alta en Keycloak y que no tiene ningún rol asignado. Una vez que se ha obtenido el token de acceso con ese usuario, se compone la petición Postman y se envía al servicio de firmado. El servicio de firmado el formato del token, su firma y comprueba que no ha sido utilizado previamente para intentar firmar una factura electrónica. Sin embargo, devuelve una respuesta de tipo "401-Unauthorized" a Postman debido a que el usuario no posee el rol necesario para poder firmar facturas electrónicas.

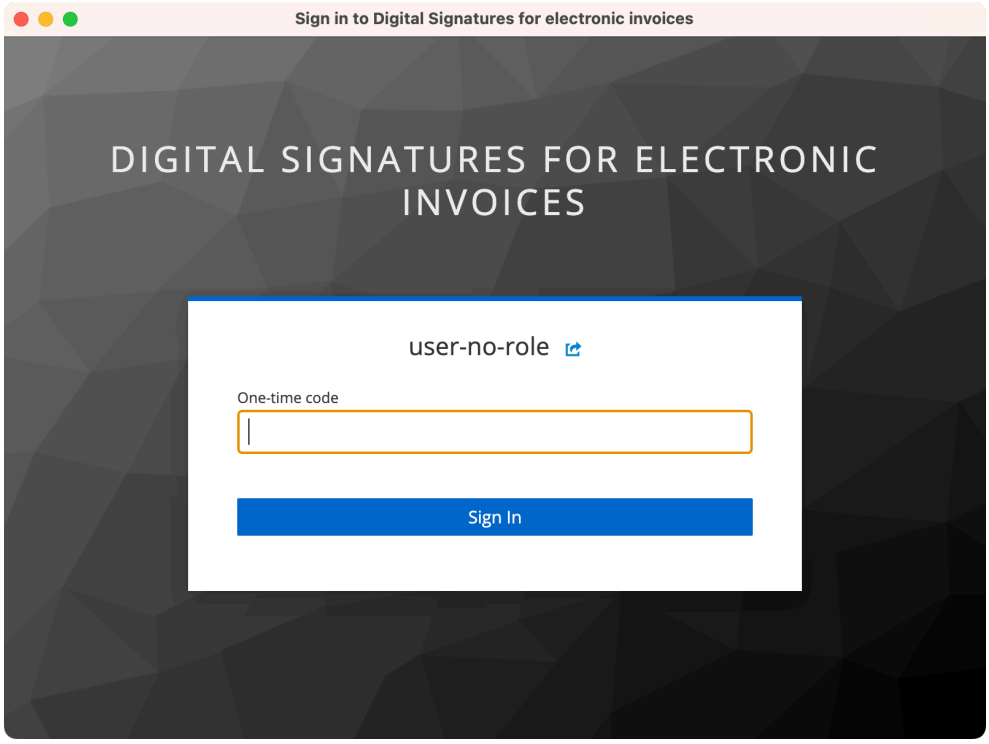

Figura 3.56: Login con el usuario "user-no-role"

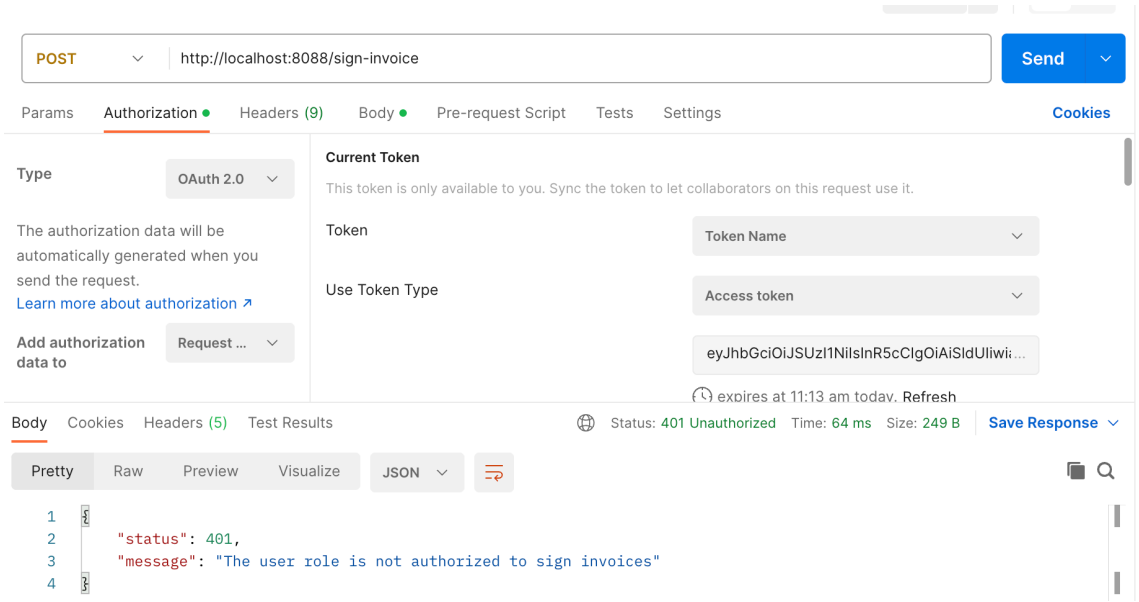

Figura 3.57: Respuesta del servicio de firmado cuando recibe una petición de un usuario que no el rol para poder firmar facturas electrónicas

Además, se accede a la base de datos para comprobar que el token haya sido almacenado en la base de datos con el resultado de la ejecución de la petición. Al acceder a la base de datos se encuentra que el token está almacenado en el registro con “ID=3” y se indica correctamente el resultado de la operación y su descripción.

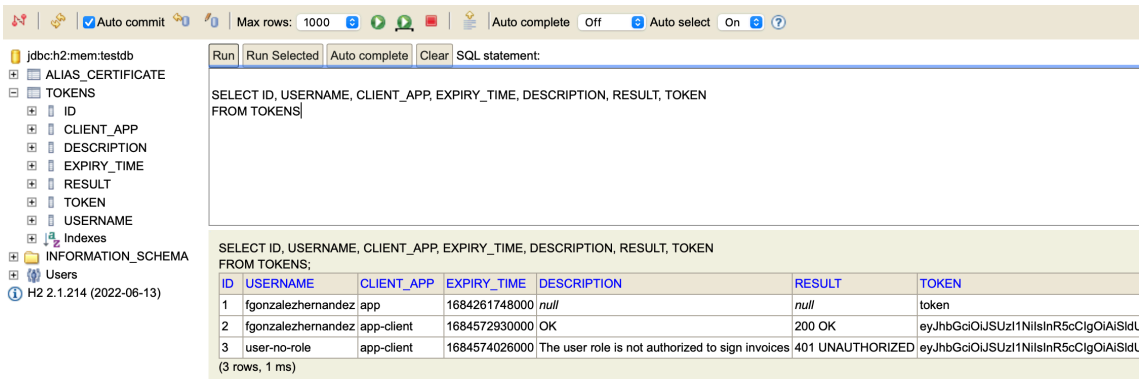

Figura 3.58: Acceso a la base de datos para comprobar que el token haya sido almacenado en la base de datos

El resultado de ejecución de la prueba ‘07’ resulta satisfactorio y verifica el cumplimiento del caso de uso ‘07’.

3.4.8. Prueba 08: la aplicación cliente envía una petición con un formato de factura no válido al servicio de firmado

El objetivo de la prueba ‘08’ es probar la funcionalidad del caso de uso “08– La aplicación cliente envía una petición con un formato de factura no válido al servicio de firmado”. Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de

la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo “Authorization Code” para obtener el token de acceso.

Para la ejecución de esta prueba, se introducen correctamente las credenciales de usuario, el código OTP, se genera correctamente la petición al servicio de firmado desde Postman y se envía la petición al servicio de firmado adjuntando una factura sin firmar en formato que no es PDF. Como resultado se obtiene que el servicio de firmado devuelve un error de tipo 400-Bad request.

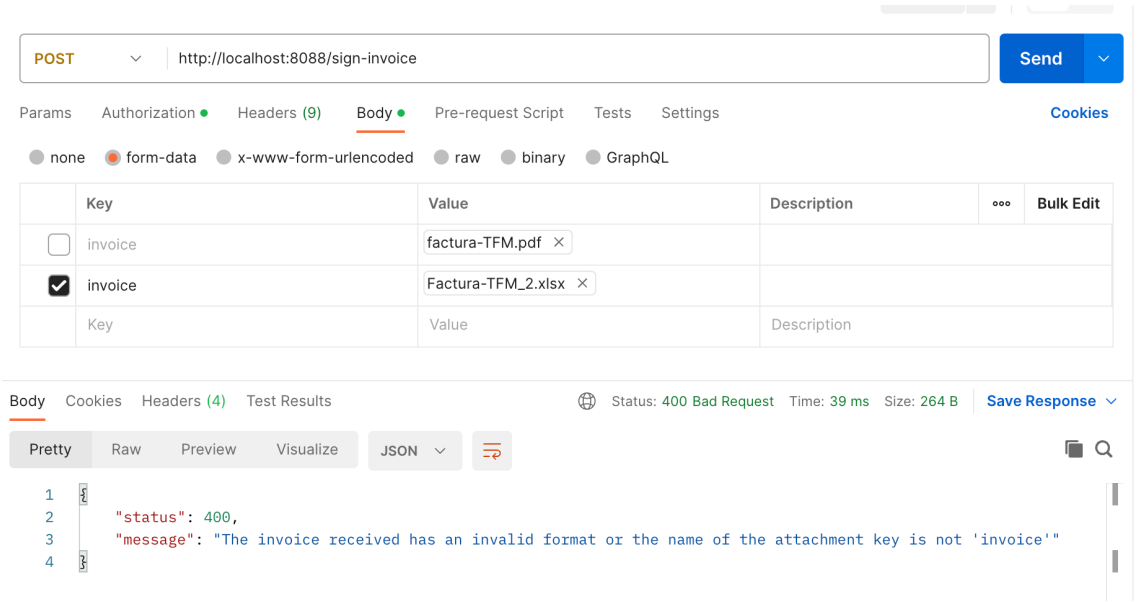

Figura 3.59: Respuesta del servicio de firmado al recibir una factura en un formato no válido

Además, se accede a la base de datos para comprobar que el token haya sido almacenado en la base de datos con el resultado de la ejecución de la petición. Al acceder a la base de datos se encuentra que el token está almacenado en el registro con “ID=4” y se indica correctamente el resultado de la operación y su descripción.

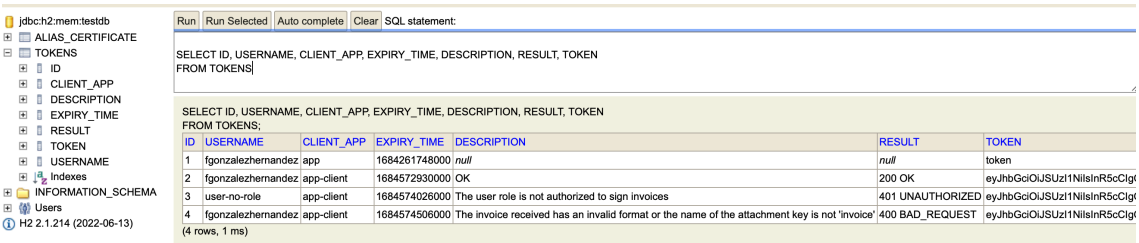

Figura 3.60: Acceso a la base de datos para comprobar que el token haya sido almacenado en la base de datos

El resultado de ejecución de la prueba ‘08’ resulta satisfactorio y verifica el cumplimiento del caso de uso ‘08’.

3.4.9. Prueba 09: la aplicación cliente envía una petición al servicio de firmado pero el servicio de firmado no tiene acceso correcto al almacén de claves

El objetivo de la prueba '09' es probar la funcionalidad del caso de uso "09– La aplicación cliente envía una petición al servicio de firmado pero el servicio de firmado no tiene acceso correcto al almacén de claves". Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo "Authorization Code" para obtener el token de acceso.

Para conseguir que el servicio de firmado no tenga acceso al almacén de claves, lo que se hace es cambiar el nombre del fichero de almacén de claves para que cuando intente acceder a él, no pueda hacerlo porque éste no exista con ese nombre. De este modo se consigue reproducir el comportamiento del servicio de firmado cuando no puede acceder al almacén de claves. En concreto se pasa del nombre de almacén "keystore-invoices.jks" a "keystore-invoices.NEW_NAME.jks".

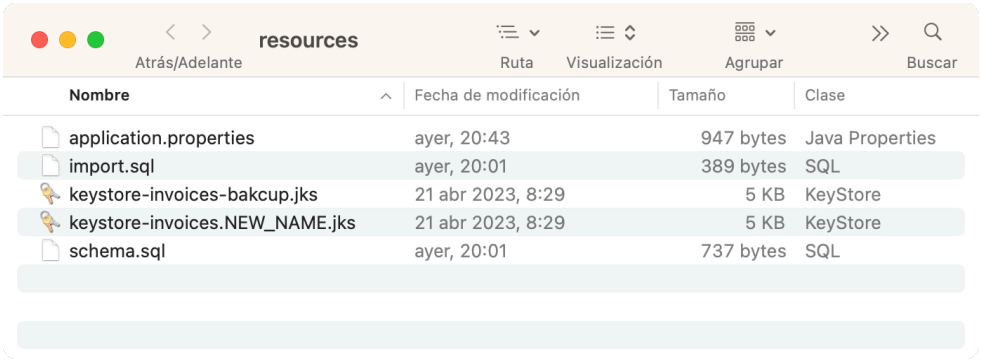

Figura 3.61: Cambio de nombre del fichero de almacén de claves

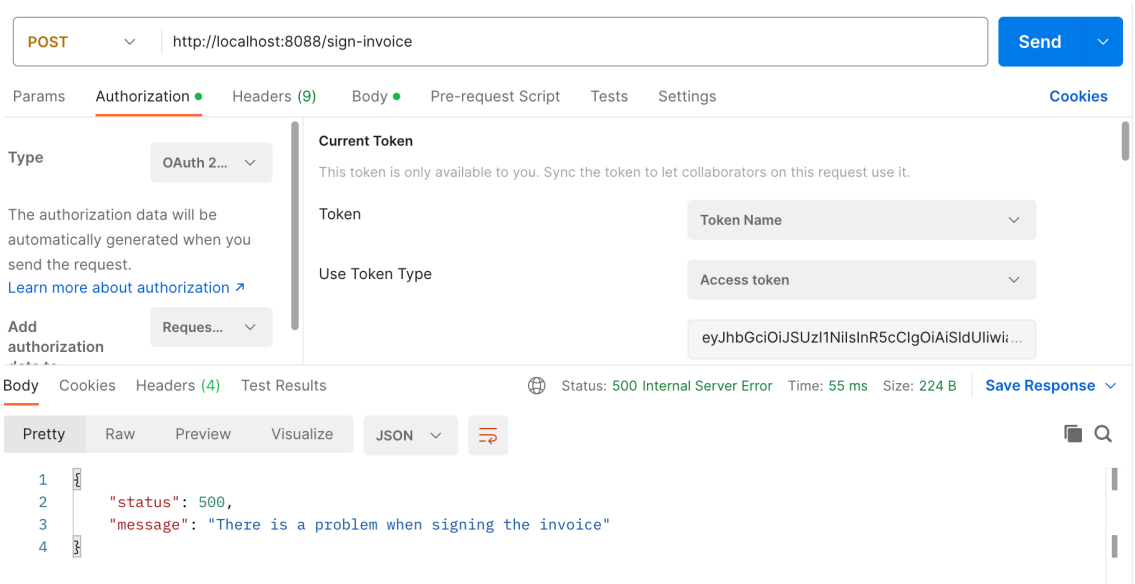

Figura 3.62: Resultado de ejecución de la petición cuando el keystore ha cambiado de nombre

Además, se accede a la base de datos para comprobar que el token haya sido almacenado en la base de datos con el resultado de la ejecución de la petición. Al acceder a la base de datos se encuentra que el token está almacenado en el registro con “ID=3” y se indica correctamente el resultado de la operación y su descripción.

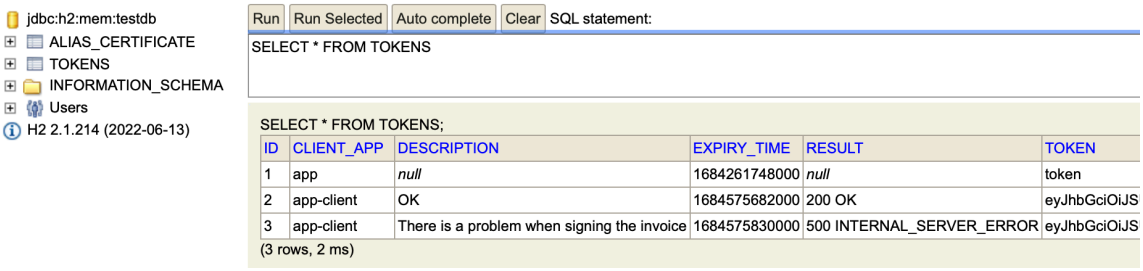

Figura 3.63: Acceso a la base de datos para comprobar que el token haya sido almacenado en la base de datos

A continuación, se muestra la traza de log del servicio de firmado donde se tiene el detalle de lo sucedido:

```
2023-05-20T11:38:52.934+02:00 ERROR 25479 --- [nio-8088-exec-5]
e.u.f.digsign.service.SignService        : [SignService] There is a problem
when signing the invoice. Exception: /Users/fer/eclipse-
workspace/InvoiceDigitalSign/src/main/resources/keystore-invoices.jks (No such
file or directory)
2023-05-20T11:38:52.934+02:00 ERROR 25479 --- [nio-8088-exec-5]
e.u.f.d.c.SignInvoiceApiController       : Internal Server Error: There is a
problem when signing the invoice
```

El resultado de ejecución de la prueba ‘09’ resulta satisfactorio y verifica el cumplimiento del caso de uso ‘09’.

3.4.10. Prueba 10: la aplicación cliente envía una petición al servicio de firmado pero el servicio de firmado no encuentra claves válidas para el usuario

El objetivo de la prueba ‘10’ es probar la funcionalidad del caso de uso “10– La aplicación cliente envía una petición al servicio de firmado pero el servicio de firmado no encuentra claves válidas para el usuario”. Para la realización de la prueba se cumplen las precondiciones del caso de uso y también se procede a ejecutar los pasos de la secuencia descrita. Además, al igual que en la prueba anterior, se utiliza Postman como aplicación cliente para la prueba y su funcionalidad de forzar el flujo “Authorization Code” para obtener el token de acceso.

Para la ejecución de esta prueba, se hace login con el usuario “user-no-cert”, que es un usuario dado de alta en Keycloak, si tiene el rol correcto para poder firmar facturas electrónicas, pero en la base de datos este usuario tiene asignado un rol de certificado digital que no existe y que, por tanto, no puede ser encontrado en el almacén de claves.

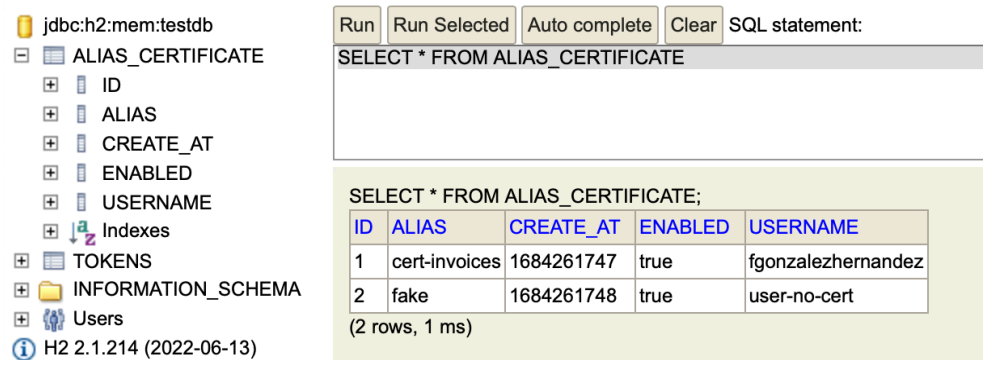

Figura 3.64: Relación de usuario y alias de certificado digital almacenados en la base de datos

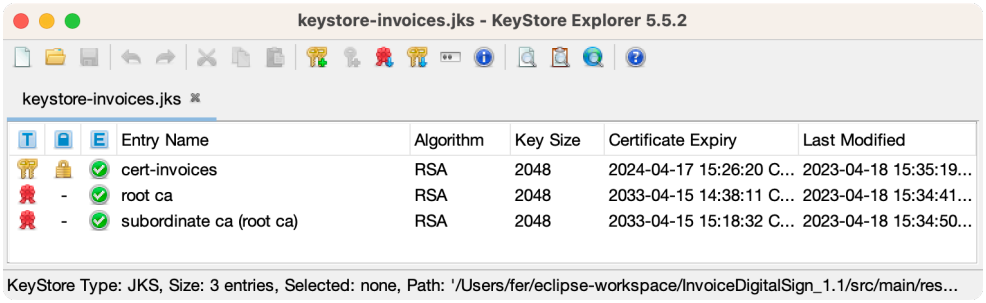

Figura 3.65: Certificados digitales almacenados en el almacén de claves

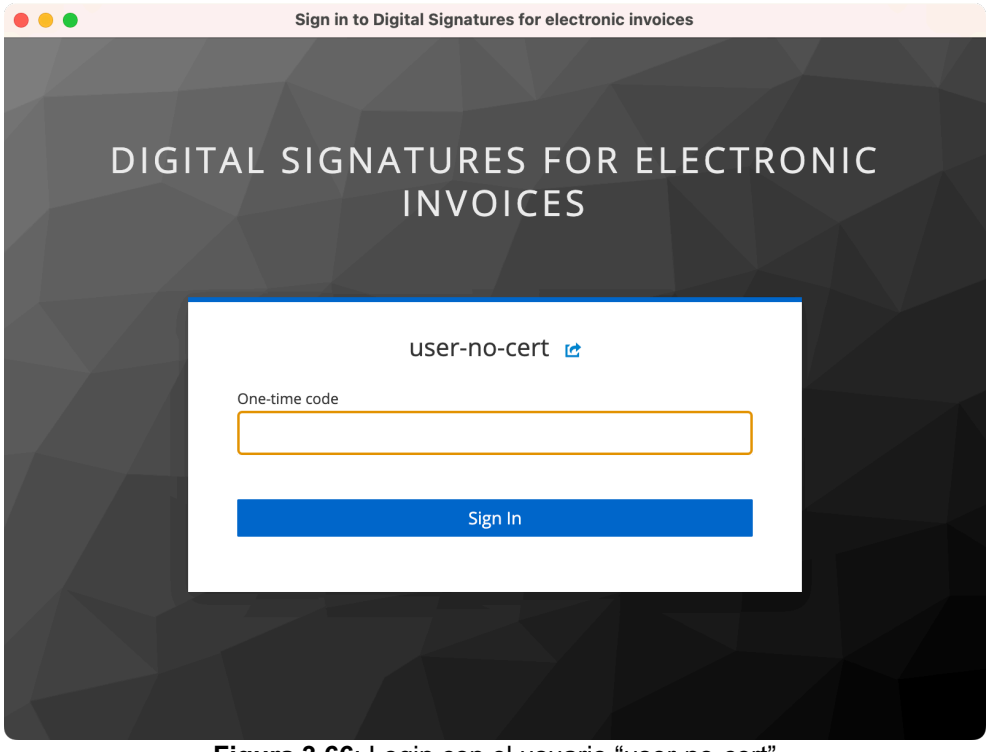

Figura 3.66: Login con el usuario “user-no-cert”

Una vez que se ha obtenido el token con ese usuario, se compone la petición Postman y se envía al servicio de firmado. El servicio de firmado valida el token de acceso, pero devuelve una respuesta de tipo “500-Internal Server Error” a

Postman debido a que no se ha encontrado un certificado digital con el que poder emitir la firma de la factura.

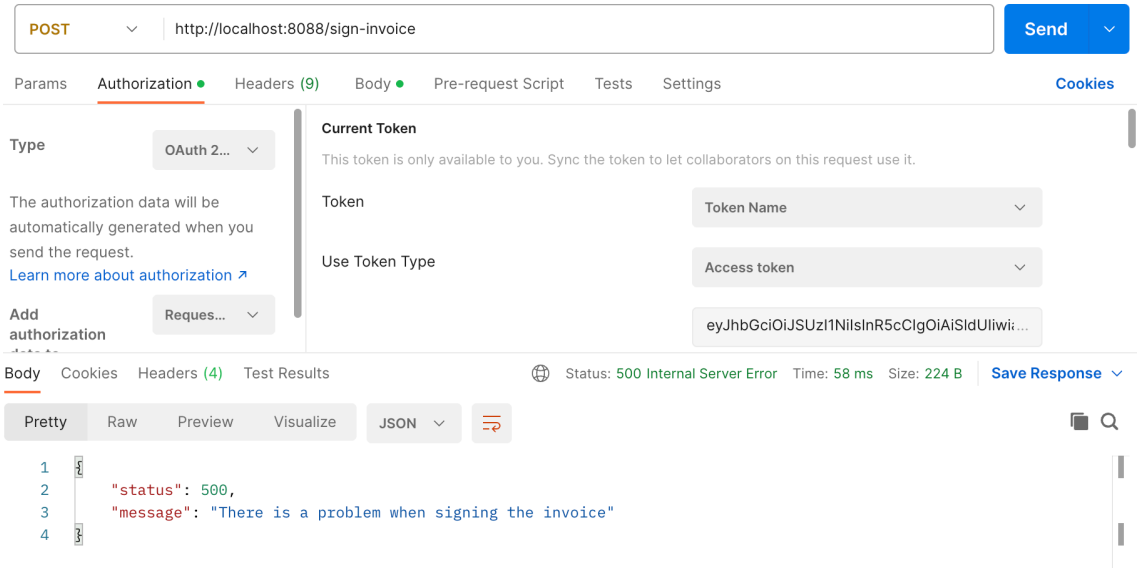

Figura 3.67: Respuesta del servicio de firmado al no poder acceder al certificado electrónico en el almacén de claves

Además, se accede a la base de datos para comprobar que el token haya sido almacenado en la base de datos con el resultado de la ejecución de la petición. Al acceder a la base de datos se encuentra que el token está almacenado en el registro con “ID=5” y se indica correctamente el resultado de la operación y su descripción.

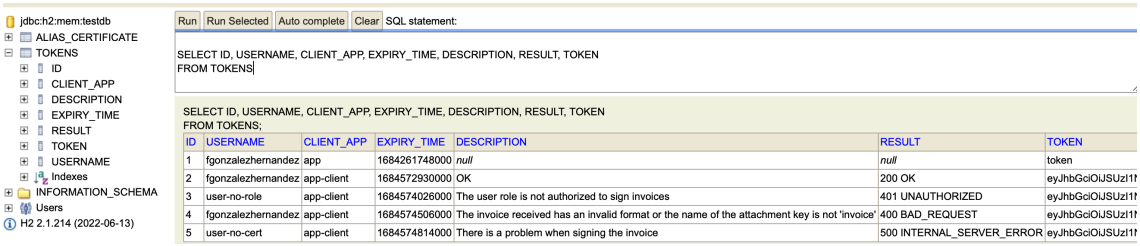

Figura 3.68: Acceso a la base de datos para comprobar que el token haya sido almacenado en la base de datos

El resultado de ejecución de la prueba ‘10’ resulta satisfactorio y verifica el cumplimiento del caso de uso ‘10’.

4. Conclusiones y trabajos futuros

En un mundo cada vez más digitalizado, donde las transacciones comerciales y administrativas se realizan de manera electrónica, la seguridad y la autenticación de los documentos se convierten en aspectos fundamentales. En este contexto, el presente trabajo ha abordado el diseño y desarrollo de un servicio de firmado digital de facturas electrónicas, fortalecido por un control de acceso basado en autenticación multi-factor (MFA). A lo largo de este estudio, se ha explorado la importancia de la seguridad y la integridad de las facturas electrónicas, así como la necesidad de implementar mecanismos que garanticen la autenticidad de los usuarios. En esta sección de conclusiones y trabajos futuros, se presentan los principales puntos clave que aporta este trabajo, destacando su relevancia en el entorno empresarial actual y también se incluyen algunas posibles direcciones futuras de investigación derivadas de este trabajo.

En el presente trabajo se ha demostrado que el uso de la firma digital en las facturas electrónicas puede permitir agilizar los procesos administrativos, reduciendo los costos asociados con la manipulación y almacenamiento de documentos físicos. La implementación de un sistema de control de acceso basado en MFA garantiza que solo los usuarios autorizados puedan acceder y realizar acciones dentro del sistema, minimizando así el riesgo de accesos no autorizados y suplantación de identidad. Además, la firma digital proporciona una forma legalmente válida de autenticación, lo que facilita el intercambio de facturas entre las partes involucradas. Esto no solo mejora la eficiencia en la gestión de documentos, sino que también asegura la integridad de los datos y reduce el riesgo de falsificación. También es importante destacar la estrecha relación entre el servicio de firmado digital de facturas electrónicas con control de acceso basado en MFA y el concepto de identidad digital. En un entorno donde las transacciones y comunicaciones se realizan de manera virtual, la identidad digital se convierte en un elemento fundamental para establecer la confianza entre las partes involucradas. Mediante la implementación de un sistema de autenticación con MFA, se fortalece la seguridad de la identidad digital de los usuarios, asegurando que solo aquellos con credenciales válidas puedan acceder y firmar electrónicamente las facturas.

Los objetivos planificados y conseguidos en el TFM son los siguientes:

- Se ha realizado un estudio sobre los conceptos fundamentales relacionados con el control de accesos, la identificación electrónica y el firmado de facturas electrónicas.
- Se han analizado las normativas y regulaciones aplicables a la firma electrónica en las facturas electrónicas.
- Se ha descrito el procedimiento de firmado de facturas electrónicas.
- Se han revisado soluciones comerciales y herramientas de software disponibles para el firmado de facturas electrónicas.
- Se ha logrado desarrollar un servicio de firmado de facturas electrónicas seguro y eficiente, utilizando tecnologías de control de acceso basado en MFA.

Se han alcanzado los objetivos iniciales del TFM. En el alcance inicial de la implementación de un caso práctico no se determinó el formato de facturas electrónicas a firmar debido a la incertidumbre del esfuerzo necesario para abarcar los diferentes tipos. Finalmente, se acotó el alcance a facturas electrónicas en formato PDF, dejando fuera del alcance las facturas en formato Facturae.

La planificación inicial del TFM se ha seguido con el mismo orden de secuenciación de las tareas definidas. Sin embargo, las fechas de consecución de las tareas no se han seguido de forma estricta, sino que en muchos casos de han mejorado mediante una intensificación para la compresión de tiempos a través de la dedicación de horas extra en la ejecución del TFM en las fases iniciales.

La metodología seguida en el desarrollo del TFM ha sido la expuesta en el apartado introductorio. Esta metodología ha resultado adecuada para la ejecución satisfactoria y eficiente porque se han podido adquirir los conocimientos fundamentales en el capítulo “marco teórico conceptual y normativo” y se han aplicado durante el desarrollo de la implementación del caso práctico.

Respecto a los impactos ético-sociales, de sostenibilidad y de diversidad, el desarrollo del TFM ha conseguido la implementación de un sistema con cuya integración de sistemas de autenticación y autorización en el servicio ha permitido asegurar la identidad de los usuarios y proteger los datos sensibles. Además, la implementación del servicio de firmado de facturas electrónicas permite optimizar los procesos internos de una empresa y reducir el tiempo y los costos asociados a la firma de facturas en papel.

Durante el desarrollo del caso práctico del TFM se han encontrado tres principales imprevistos: la corrupción en la generación de certificados de firma, la corrupción de facturas en formato PDF para ser firmadas y la aplicación de un aspecto relativo al reglamento eIDAS en el que por cada activación de una clave de firma sea necesaria la intervención del usuario

- La generación de certificados digitales para la firma digital de las facturas electrónicas se tenía previsto realizar desde una máquina virtual con sistema operativo Linux. Sin embargo, un problema de corrupción de ficheros provocó que los certificados generados por la máquina virtual Linux no fueran accesibles correctamente por otros equipos. Para solventar el inconveniente, se procedió a generar los certificados desde otro equipo (con sistema operativo macOS) y los certificados si pudieron ser accesibles correctamente por otros equipos.
- En la corrupción de facturas en formato PDF, también se encontró un inconveniente similar al de la corrupción de certificados en el que una factura de ejemplo si podía ser abierta por lectores de PDF, pero su procesamiento desde el servicio Java advertía con un warning de un problema de corrupción del fichero. El problema se solucionó volviendo a generar la factura en formato PDF a partir de una plantilla en formato Excel.

- La aplicación del reglamento eIDAS respecto a que por cada activación de clave de firma sea precedida de la intervención del usuario fue un inconveniente respecto al diseño inicial ya que este aspecto se consideró una vez estaba iniciada la fase de desarrollo. Sin embargo, se solucionó el problema mediante la comprensión de la necesidad del reglamento respecto a la activación de clave de firma, aplicando cambios en los requisitos iniciales, modificando el diseño análisis funcional en base a los nuevos requisitos y finalmente modificando la implementación con los nuevos cambios.

Algunos de los posibles trabajos futuros derivados de este TFM podrían ser los siguientes:

- Ampliar el alcance del servicio de firmado de facturas electrónicas para incluir otros formatos de facturas electrónicas y la verificación de la firma de facturas electrónicas firmadas.
- Analizar métodos de verificación de identidad de los usuarios para poder asociarles su identidad electrónica de forma segura.
- Analizar métodos seguros de transmisión de las claves desde una PKI hasta el dueño de las claves.
- Analizar métodos de almacenamiento y custodia de certificados electrónicos para asegurar que su acceso esté completamente restringido y no sean accesibles por contraseñas maestras a las que puedan tener acceso terceras personas, como un administrador de sistemas.
- Investigar y evaluar nuevas tecnologías de control de acceso y autenticación, como la biometría o la autenticación basada en blockchain.
- Integración del servicio de firmado de facturas electrónicas con otros sistemas empresariales, como sistemas de gestión documental o de contabilidad.
- Realizar pruebas de penetración y auditorías de seguridad para garantizar la protección de los datos y la privacidad de los usuarios.

5. Bibliografía

[1] Ministerio de Asuntos Económicos y Transformación Digital. (consultado en marzo 2023). Identidad digital. Disponible en: https://firmaelectronica.gob.es/Home/Empresas/Identidad-Digital.html
[2] Unión Europea. (2014). Reglamento (UE) 910/2014 del Parlamento Europeo y del Consejo, de 23 de julio de 2014, relativo a la identificación electrónica y los servicios de confianza para las transacciones electrónicas en el mercado interior y por el que se deroga la Directiva 1999/93/CE. Diario Oficial de la Unión Europea, L 257/73-L 257/127. Disponible en: https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32014R0910
[3] Binance. (2019). ¿Qué es una firma digital?. Disponible en: https://academy.binance.com/es/articles/what-is-a-digital-signature
[4] Fernández Jara J.C.; Martínez-Ballesté, A.; Solanas, A.; Castellà-Roca, J. (2022). Protocolos de autenticación, autorización y control de acceso. Fundació Universitat Oberta de Catalunya- PID_00287478
[5] AWS. (consultado en marzo 2023). ¿Qué es la autenticación multifactor (MFA)? Disponible en: https://aws.amazon.com/es/what-is/mfa/
[6] Microsoft. (consultado en marzo 2023). Qué es: Autenticación multifactor. Disponible en: https://support.microsoft.com/es-es/topic/qué-es-autenticación-multifactor-e5e39437-121c-be60-d123-eda06bddf661
[7] Cloud Google. (consultado en marzo 2023). Aplicar la autenticación multifactor (MFA) de manera uniforme a los recursos de la empresa. Disponible: https://cloud.google.com/identity/solutions/enforce-mfa?hl=es
[8] Ley de Identificación Nacional de 2005, Public Law Number 109-13, 119 Statutes at Large 231 (2005). Disponible en: https://www.congress.gov/109/plaws/publ13/PLAW-109publ13.pdf
[9] Ley de Protección de la Información Personal y Documentos Electrónicos (PIPEDA), Statutes of Canada. 2000, c. 5. Disponible en: https://laws-lois.justice.gc.ca/eng/acts/p-8.6/
[10] Ley de Identificación y Verificación de Identidad, 2013 (Commonwealth of Australia). Disponible en: https://www.legislation.gov.au/Details/C2013C00663
[11] Ley de Firma Electrónica, Ley n.º 102 de 2000 (Japón). Disponible en: https://www.japaneselawtranslation.go.jp/law/detail/?ft=1&re=02&dn=1&co=01&ia=03&y=2000&no=102
[12] Ley de Certificación Electrónica, Ley n.º 64 de 2000 (Japón). Disponible en: https://www.japaneselawtranslation.go.jp/law/detail/?id=2038&vm=04&re=
[13] Ley 6/2020, de 11 de noviembre, reguladora de determinados aspectos de los servicios electrónicos de confianza. Boletín Oficial del Estado, núm. 275, de 12 de noviembre de 2020, pp. 1-33. Disponible en: https://www.boe.es/buscar/act.php?id=BOE-A-2020-14203
[14] clave.gob.es. (consultado en marzo 2023). Cl@ve - Claves de Acceso Seguras para Ciudadanos. Disponible en: https://clave.gob.es/clave_Home/clave.html
[15] Dirección General de la Policía. (consultado en marzo 2023). DNI electrónico. Disponible en: https://www.dnielectronico.es/
[16] Fernández Jara J.C.; Palazón Romero, J.M.; Felguera A.; Castellà-Roca, J. (2022). Single sign-on y federación de identidades. Fundació Universitat Oberta de Catalunya- PID_00287476

[17] Campbell, B., Mortimore, C., Jones M.; IETF (2015) RFC 7522: Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants. Disponible en: https://datatracker.ietf.org/doc/rfc7522/
[18] Anicas M.; (2018) “Una introducción a OAuth 2”. Disponible en: https://www.digitalocean.com/community/tutorials/una-introduccion-a-oauth-2-es
[19] Hardt, D, Microsoft; IETF (2012) RFC 6749: The OAuth 2.0 Authorization Framework. Disponible en: https://www.ietf.org/rfc/rfc6749.txt
[20] Pari J.; (2021) “Todos los flujos de OAuth 2.0”. Disponible en: https://arquitecturaibm.com/todos-los-flujos-de-oauth-2-0/
[21] OpenID.net; “OpenID Connect specification”. Disponible en: https://openid.net/connect/
[22] García A.; (2022). OAuth2 y OpenID/OIDC. Disponible en: https://www.enmilocalfunciona.io/oauth2-y-openid/
[23] Torres G.; (2019) “OAuth 2.0, OpenID Connect y JSON Web Tokens (JWT) ¿Qué es qué?”. Disponible en: https://www.returngis.net/2019/04/oauth-2-0-openid-connect-y-json-web-tokens-jwt-que-es-que/
[24] Jones M.; IETF (2015) RFC 7519 JSON Web Token (JWT). Disponible en: https://www.rfc-editor.org/rfc/rfc7519
[25] JWT.IO; Disponible en: https://jwt.io/#debugger-io
[26] Okta (consultado en marzo 2023) What is a One-Time Password (OTP)? Disponible en: https://www.okta.com/blog/2020/06/what-is-a-one-time-password-otp/
[27] RSA (consultado en marzo 2023). DAUTH: Secure Offline Verification of One Time Passwords. Disponible en: https://community.rsa.com/yfcdo34327/attachments/yfcdo34327/securid-knowledge-base/1195/2/Secure%20Offline%20Verification%20of%20OTP.pdf
[28] Real Academia Española. Definiciones de “firma” y “firma digital”. Disponible en: https://dle.rae.es/firma
[29] Ley 59/2003, de 19 de diciembre, de firma electrónica. Disponible en: https://www.boe.es/buscar/pdf/2003/BOE-A-2003-23399-consolidado.pdf
[30] Amazon Web Services. (2023). AWS Key Management Service. Disponible en: https://aws.amazon.com/kms/
[31] Google Cloud. (2023). Cloud Key Management Service. Disponible en: https://cloud.google.com/kms
[32] Microsoft. (2023). Azure Key Vault. Disponible en: https://azure.microsoft.com/en-us/services/key-vault/
[33] Boeyen S., Santesson S., Polk T., Housley R., Farrell S., Cooper D.; IETF (2008) RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Disponible en: https://www.ietf.org/rfc/rfc5280.txt
[34] Banco de España (consultado en marzo 2023). Certificados digitales y firma electrónica. Disponible en: https://www.bde.es/bde/es/secciones/servicios/Particulares_y_e/Certificados_y_f/Certificados_di_58797f3710fd821.html
[35] Kaspersky (consultado en marzo 2023). Adding the self-signed certificate as trusted to a browser (Linux) Disponible en: https://support.kaspersky.com/ScanEngine/1.0/en-US/182984.htm

[36] Apple (consultado en marzo 2023). Obtener información sobre un certificado en Acceso a Llaveros en el Mac. Disponible en: https://support.apple.com/es-es/guide/keychain-access/kyca15178/mac
[37] Microsoft (2002). Almacenes de certificados. Disponible en: https://learn.microsoft.com/es-es/windows-hardware/drivers/install/certificate-stores
[38] Dvdcr (2022). Consultar y gestionar almacenes de certificados de confianza Java. Disponible en: https://dvdcr.com/posts/2022/02/java-truststore/
[39] Eastlake, D., Reagle, J., & Solo, D. (2002). XML-Signature Syntax and Processing. Disponible en: https://www.w3.org/TR/xmldsig-core/
[40] W3C. (2013). XML Advanced Electronic Signatures (XAdES). Disponible en: https://www.w3.org/TR/xmldsig-xades/
[41] ETSI (2009). Electronic Signatures and Infrastructures (ESI) - PDF Advanced Electronic Signature Profiles, Part 1: PAdES Overview - A Framework for PAdES. Disponible en: https://www.etsi.org/deliver/etsi_ts%5C102700_102799%5C10277801%5C01.01.01_60%5Cts_10277801v010101p.pdf
[42] ETSI (2021). Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures. Disponible en: https://www.etsi.org/deliver/etsi_en/319100_319199/31912201/01.02.01_60/en_31912201v010201p.pdf
[43] Jonsson J., Kaliski, B.; IETF (2003) RFC 3447: Public-Key Cryptography Standars (PKCS) #1: RSA Cryptography Specifications Version 2.1. Disponible en: https://datatracker.ietf.org/doc/html/rfc3447
[44] SHA256: National Institute of Standards and Technology. (2015). Secure Hash Standard (SHS). Disponible en: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
[45] Moreno, J.M. (2018). XML y canonización. Disponible en: https://josemmo.medium.com/xml-y-canonicalización-7002c42442dd
[46] Real Academia Española. Definición de “factura”. Disponible en: https://dle.rae.es/factura
[47] Real Decreto 1619/2012, de 30 de noviembre, por el que se aprueba el Reglamento por el que se regulan las obligaciones de facturación. Disponible en: https://www.boe.es/buscar/pdf/2012/BOE-A-2012-14696-consolidado.pdf
[48] Recomendación de la Comisión de 19 de octubre de 1994 relativa a los aspectos jurídicos del intercambio electrónico de datos. Disponible en: https://www.boe.es/doue/1994/338/L00098-00117.pdf
[49] Directiva 1999/93/CE del Parlamento Europeo y del Consejo, de 13 de diciembre de 1999, por la que se establece un marco comunitario para la firma electrónica. Disponible en: https://www.boe.es/doue/2000/013/L00012-00020.pdf
[50] Ley 25/2013, de 27 de diciembre, de impulso de la factura electrónica y creación del registro contable de facturas en el Sector Público. Disponible en: https://www.boe.es/buscar/pdf/2013/BOE-A-2013-13722-consolidado.pdf
[51] Ministerio de Asuntos Económicos y Transformación digital, Ministerio de Hacienda. (consultado en marzo 2023) “Factura-e”. Disponible en: https://www.facturae.gob.es/Paginas/Index.aspx
[52] b2brouter. (2022) “Todo sobre Facturae”. Disponible en: https://www.b2brouter.net/es/facturae/

[53] Valls C, Holded. (2023) “Facturación electrónica ¿por qué es tan importante y por qué deberías familiarizarte con ella?”. Disponible en: https://www.holded.com/es/blog/facturacion-electronica
[54] Okta. (consultado en marzo 2023). Identity for the internet. https://www.okta.com/
[55] Microsoft Corporation. (consultado en marzo 2023). Azure Active Directory. https://azure.microsoft.com/en-us/services/active-directory/
[56] Microsoft Corporation. (consultado en marzo 2023). Active Directory Domain Services Overview. https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview
[57] OneLogin (consultado en marzo 2023). Market-Leading Identity and Access Management Solutions. Disponible en: https://www.onelogin.com
[58] Ping Identity. (consultado en marzo 2023). Identity Defined Security. Disponible en: https://www.pingidentity.com/es.html
[59] Salesforce. (consultado en marzo 2023). Streamline user access with a single login from Salesforce Customer Identity. Disponible en: https://www.salesforce.com/products/platform/products/identity/
[60] Google Cloud. (consultado en marzo 2023). Google Cloud Identity. Disponible en: https://cloud.google.com/identity
[61] IBM. (consultado en marzo 2023). Cloud identity and access management (IAM) solutions. Disponible en: https://www.ibm.com/products/verify-identity/cloud
[62] IBM. (consultado en marzo 2023). Visión general de IBM Security Identity Manager. Disponible en: https://www.ibm.com/docs/es/sim/7.0.1.13?topic=overview-security-identity-manager
[63] ForgeRock. (consultado en marzo 2023). ForgeRock Identity Platform. Disponible en: https://www.forgerock.com
[64] FreeIPA. (consultado en marzo 2023). FreeIPA. https://www.freeipa.org/page/Main_Page
[65] Red Hat. (consultado en marzo 2023). Keycloak. https://www.Keycloak.org/
[66] Gluu. (consultado en marzo 2023). Gluu Cloud. Disponible en: https://www.glucloud.es
[67] Gluu (consultado en marzo 2023). Gluu Server 4.0 Documentation. Disponible en: https://gluu.org/docs/gluu-server/4.0/
[68] Oracle (consultado en marzo 2023). Identity and Access Management (IAM). Disponible en: https://www.oracle.com/es/security/identity-management/
[69] Oracle (consultado en marzo 2023). Oracle Identity Cloud Service. Disponible en: https://docs.oracle.com/en/cloud/paas/identity-cloud/index.html
[70] OpenAM (consultado en marzo 2023). Open Identity Platform. Disponible en: https://www.openidentityplatform.org/openam
[71] WSO2 (consultado en marzo 2023). WSO2 Identity Server. Disponible en: https://wso2.com/es/identity-server/
[72] Google LLC. (consultado en marzo 2023). Google Authenticator. Disponible en: https://support.google.com/accounts/answer/1066447
[73] Microsoft Corporation. (consultado en marzo 2023). Microsoft Authenticator. Disponible en: https://www.microsoft.com/en-us/account/authenticator

[74] Twilio. (consultado en marzo 2023). Authy. Disponible en: https://authy.com/
[75] Duo Security. (consultado en marzo 2023). Duo Security. Disponible en: https://duo.com/
[76] LastPass. (consultado en marzo 2023). LastPass Authenticator. Disponible en: https://www.lastpass.com/authenticator
[77] Yubico. (consultado en marzo 2023). Yubico Authenticator. Disponible en: https://www.yubico.com/products/services-software/download/yubico-authenticator/
[78] FreeOTP. (consultado en marzo 2023). In GitHub. Disponible en: https://github.com/freeotp/freeotp-ios
[79] Aegis (consultado en marzo 2023). Aegis Authenticator. Disponible en: https://getaegis.app
[80] OpenSSL. (consultado en marzo 2023). OpenSSL wiki. Disponible en: https://wiki.openssl.org/index.php/Main_Page
[81] DigiCert, Inc. (consultado en marzo 2023). About DigiCert. Disponible en: https://www.digicert.com/about/
[82] GlobalSign. (consultado en marzo 2023). Digital Certificates. https://www.globalsign.com/en
[83] Fábrica Nacional de Moneda y Timbre (FNMT). (consultado en marzo 2023). FNMT-CERES. Disponible en: https://www.sede.fnmt.gob.es/ceres/index.html
[84] Camerfirma. (consultado en marzo 2023). Digital Transformation. Trusted Solutions. Disponible en: https://www.camerfirma.com/
[85] Izenpe. (consultado en marzo 2023). Izenpe. Disponible en: https://www.izenpe.eus/
[86] ANCERT (consultado en marzo 2023). Agencia Notarial de Certificación. Disponible en: Agencia Notarial de Certificación
[87] Firmaprofesional. (consultado en marzo 2023). Servicios de firma electrónica avanzada. Disponible en: https://www.firmaprofesional.com/
[88] Adobe Systems Incorporated. (consultado en marzo 2023). Adobe Acrobat Reader DC. Disponible en: https://acrobat.adobe.com/us/en/acrobat/pdf-reader.html
[89] Adobe. (consultado en marzo 2023). Adobe Sign. Disponible en: https://acrobat.adobe.com/us/en/sign.html
[90] The Document Foundation. (consultado en marzo 2023). LibreOffice. https://www.libreoffice.org/
[91] Foxit Software. (consultado en marzo 2023). Foxit Reader. Disponible en: https://www.foxitsoftware.com/pdf-reader/
[92] GnuPG. (consultado en marzo 2023). GnuPG. Disponible en: https://gnupg.org/index.html
[93] SignNow. (consultado en marzo 2023). SignNow. Disponible en: https://www.signnow.com/
[94] PandaDoc. (consultado en marzo 2023). All-in-One Docuement Automation Software. Disponible en: https://www.pandadoc.com/
[95] Face (consultado en marzo 2023). Punto General de Entrada de Facturas Electrónicas. Disponible en: https://face.gob.es/es
[96] Signaturit. (consultado en marzo2023). Signaturit | Soluciones de firma electrónica y contratación digital. Disponible en: https://www.signaturit.com/es

[97] FacturaDirecta. (consulta en 2023). FacturaDirecta. Disponible en: https://www.facturadirecta.com
[98] FacturaScripts. (consultado en marzo 2023). FacturaScripts contabilidad y facturación libre. Disponible en: https://facturascripts.com
[99] OpenSSL Project. (consultado en abril 2023). OpenSSL. Disponible en: https://www.openssl.org/
[100] Keystore Explorer (consultado en abril 2023). Keystore Explorer. Disponible en: https://keystore-explorer.org/
[101] Postman (consultado en abril 2023). Postman. Disponible en: https://www.postman.com/
[102] Privotal software (consultado en abril 2023). Spring boot. Disponible en: https://spring.io/projects/spring-boot