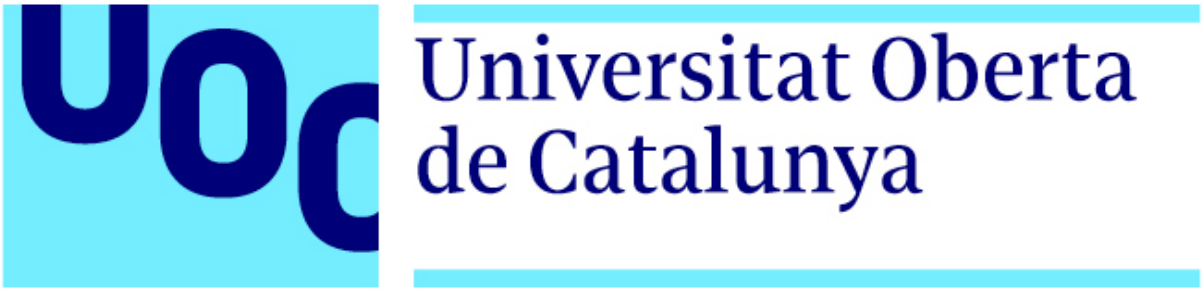

Identidad Descentralizada Aplicada.

María Ruiz Molina

Máster universitario Online de Ciberseguridad y Privacidad
Sistemas de Blockchain

Tutor
Juan Carlos Fernández Jara

Profesor Responsable de la Asignatura
Victor Garcia Font

Fecha de Entrega
13 de Junio de 2023

Ficha del Trabajo Final

Título del trabajo	Identidad Descentralizada Aplicada
Nombre del autor	María Ruiz Molina
Nombre del consultor	Juan Carlos Fernández Jara
Nombre del PRA	Victor Garcia Font
Fecha de entrega	06/2023
Titulación o Programa	Máster universitario Online de Ciberseguridad y Privacidad
Área del Trabajo Final	Sistemas de Blockchain
Idioma del trabajo	Castellano
Palabras clave	Blockchain, Identidad Descentralizada, Privacidad

Agradecimientos

Quiero agradecer a mi tutor Juan Carlos Fernández la atención recibida durante la ejecución de este proyecto.

También, quiero darle las gracias a mi familia, en especial a mi madre por su apoyo a lo largo de toda mi vida.

A Mercedes por introducirme y guiarme en esta especialidad.

Quiero dar las gracias a mis amigos, en especial a Willy, a Juan y a Susana, que han estado tan pendientes de mis avances con este trabajo.

And to Nat. With you, my life is so much better.

Resumen

La Identidad Descentralizada es una de las aplicaciones de la tecnología Blockchain y DLTs (*Distributed Ledger Technologies)*, permitiendo a los usuarios almacenar y presentar sus datos de manera mucho más privada que con los modelos centralizados existentes, además de proporcionándoles total control sobre los mismos. Debido a su novedad, se han llevado a cabo diversas propuestas, tanto para el modelado de los datos como para los protocolos de comunicación a utilizar.

Algunos de estos modelos son las Credenciales Verificables de la W3C o los SoulBound Tokens de la Ethereum Foundation. Entre los protocolos de comunicación podemos encontrar OpenID Connect[1], el cual integra los protocolos OpenID, protocolo de autenticación, con OAuth 2.0, protocolo de autorización; o DIDComm, protocolo creado desde sus inicios con la Identidad Descentralizada como modelo objetivo en todo momento. Además de estos, existen otros conceptos que bien pueden complementar los casos de uso de la identidad digital, como los Non Fungible Tokens (NFTs), aunque este tipo de tokens está orientado a activos transferibles, y por lo tanto son más aptos para adaptar al mundo de la Web3.0 características propias de tickets de avión o tren, coleccionables o entradas a conciertos.

Abstract

Decentralized Identity is one of the applications of Blockchain and DLTs (Distributed Ledger Technologies), allowing users to store and present their data in a much more private way than with the existing centralized models, while also providing them full control over it. Due to its novelty, various proposals have been made, both for data modeling and for the communication protocols to be used.

Some of these models include Verifiable Credentials from W3C or SoulBound Tokens from the Ethereum Foundation. Among the communication protocols, we can find OpenID Connect, which integrates OpenID, an authentication protocol, with OAuth 2.0, an authorization protocol; or DIDComm, a protocol created from its inception with Decentralized Identity as its main model since the beginning. In addition to these, there are other concepts that can complement the use cases of digital identity, such as Non-Fungible Tokens (NFTs), although this type of tokens is focused on transferable assets and, therefore, it is more suitable for adapting to the Web3.0 world the characteristics of travelling tickets, collectibles, or concert tickets.

Índice

Índice de figuras

1. Introducción

La reciente evolución tecnológica en el apartado de la Identidad Digital, gracias a la popularización de las redes descentralizadas y el enfoque en la privacidad del usuario, ha dado lugar a diversas soluciones entorno a la tecnología blockchain. Veamos en qué circunstancias esto se da lugar y el impacto actual que tiene en el desarrollo tecnológico de soluciones.

1.1. Contexto

La Identidad Digital está cada vez más integrada en nuestra vida, llegando al punto en el que la gran mayoría de nuestra vida, datos e información, está almacenada de manera digital. Este incremento de nuestra exposición en la web ha dado lugar a modelos económicos basados en la recopilación, compra y venta de datos personales de los usuarios. Para la obtención de los mismos se emplean técnicas extremadamente invasivas con la privacidad, y son unas pocas empresas las que poseen el mayor control de nuestra información. Esta centralización de los datos en los servidores de dichas organizaciones da lugar a una serie de problemas.

- **Venta de datos a terceros.** Los usuarios cuando hacen uso de algún servicio supuestamente gratuito, en muchas ocasiones están cediendo a cambio sus datos. Estos, normalmente, no solamente se comparten con la entidad dueña de dicho servicio, sino que posteriormente los vende a entidades terceras. A partir de dicho punto, el usuario pierde completamente el control de sus datos, llegando a desconocer completamente quién posee qué información de ellos.
- **Dificultades para uniformar datos de un mismo usuario.** El uso de protocolos centralizados implica además la falta de uniformidad. Esto da lugar a la elaboración de diversos perfiles sobre un mismo individuo, desconectados entre sí y, que además, dificulta a los usuarios ejercer sus derechos a la modificación de datos incorrectos sobre sí mismos. Esto lleva además al siguiente punto.
- **Dificultad de los individuos para ejercer sus derechos expresados en el RGPD.** Dada la pérdida de control sobre sus datos, así como de sus perfiles generados, los usuarios se encuentran con diversas dificultades a la hora de querer ejercer el derecho al olvido o rectificación. Esto se debe a que si un individuo reclama a una organización para ello, esta se verá obligada a eliminar dicha información, pero en caso de que haya vendido la información a otras entidades, estos terceros no tienen por qué hacerlo. Se debería por lo tanto seguir la cadena de venta de datos por todas las organizaciones que los hayan adquirido, dificultando muchísimo la eliminación de información en la red de un usuario determinado.

Además, los usuarios deben ser capaces de recordar múltiples contraseñas para cada uno de los servicios que emplean, llevando esta práctica a la creación de claves sencillas de recordar y no lo suficientemente resistentes a ataques de fuerza bruta o técnicas similares.

Producto de todos estos problemas que la Identidad Centralizada trae, han surgido distintas propuestas, cada vez más descentralizadas. Lo que comenzó como un modelo de identidad aislada, donde cada servicio necesitaba de su credencial individual, fue evolucionando, primero a un modelo de Arquitecturas de Administración de Identidades (*Identity Management Architectures*, donde el usuario podía identificarse con un mismo par de credencial-usuario en una serie de proveedores de dentro de un conjunto de dominios de confianza; luego expandiéndose el concepto a la Identidad Federada (*Federated Identity)*, donde el usuario se identificaba ante dicha federación y esta le asociaría un token el cual puede presentar a los proveedores de servicios dentro del contexto de dicha federación. Finalmente, el último paso antes de la descentralización pasa por un modelo extremadamente centralizado, donde los individuos eligen un proveedor, como Google o Facebook, y se identifican ante otros proveedores con dichas credenciales.

De esto puede verse que a la hora de solucionar las incidencias que la centralización de la identidad daba, se priorizó inicialmente el acabar con la necesidad de los usuarios de recordar múltiples contraseñas. Es ahora el

momento en el que, además de tener esto en mente, se busca llegar a un modelo de Identidad Descentralizada donde el usuario es el único dueño de sus propios datos y el único con capacidad para enviarlos a aquellas entidades que seleccione.

Debido al crecimiento que la Identidad Descentralizada está experimentando, entidades de gran importancia han comenzado a integrarla en sus ecosistemas. Empresas privadas y orientadas a la emisión de firmas digitales o certificados, Autoridades Certificadoras, pero también el sector público, siendo uno de los movimientos más grandes la búsqueda de la adopción masiva de esta tecnología por parte de la ciudadanía de Europa [2].

De aquí encontramos que en la Unión Europea comienza a haber movimientos muy definidos en torno a esta adopción de un sistema descentralizado para preservar la información de sus ciudadanos. Es por ello que en 2018 se fundó la *European Blockchain Service Infraestructure* (EBSI), una iniciativa que busca establecer una estructura blockchain que permita la comunicación entre diferentes servicios en la Unión Europea, identificándose sus integrantes y usuarios de manera privada y confiable. Más adelante veremos que esta iniciativa ha optado por el modelo de datos de la W3C, las Credenciales Verificables, posee su propia red blockchain permisionada y hace uso del protocolo OpenID para comunicaciones como primer paso para luego realizar una adaptación a DIDComm, si bien como veremos, dando lugar a cuestiones sobre la centralización del estándar durante la fase de uso de OpenID.

Por otro lado, estos estándares tienen sus ventajas y desventajas frente a otros. Es por ello necesario evaluar propuestas alternativas a las VCs, como los SoulBound Tokens (SBTs), modelo de datos inspirado en los Non-Fungible Tokens, diseñado por la Ethereum Foundation, pero sin la capacidad de transferencia. Estos SBTs poseen características diferenciadoras de las Credenciales Verificables a niveles criptográfico y de privacidad, y por lo tanto, ventajosos sobre algunos casos de uso y desventajosos sobre otros.

Cabe mencionar de forma paralela los NFTs y también los Proof Of Attendance Protocol (POAPs), tokens y estructuras de datos que llevan un tiempo en el ámbito blockchain y que pueden llegar a complementar ciertas áreas que estos anteriores no puedan abarcar dadas sus características de transferibilidad.

1.2. Motivación

Actualmente este área de investigación y desarrollo se encuentra en un estado muy temprano, habiendo organizaciones e infraestructuras formándose para su adopción, pero sin todavía un modelo de datos, protocolo de comunicación o red específicos elegidos. En general, no hay un estándar definido para ello ni tampoco intercomunicación posible entre las distintas propuestas.

La correcta convivencia de todos estos protocolos, así como de una acertada selección de los mismos como pilares de la blockchain conformada por los países miembros de la Unión Europea, y que albergará la información y transacciones de los ciudadanos de la misma, es un paso fundamental para la adopción masiva de la descentralización y la llamada Web3.0.

Esta estandarización debe ser definida para que entidades de todas partes utilicen los mismos medios para comunicarse y permitir así una intercompatibilidad.

1.3. Planteamiento del Problema

Sin la cuestión previamente planteada aclarada, no será posible realizar los avances y adaptaciones necesarios a nivel legal para regular y gestionar acorde a la legislación de la Unión Europea las tecnologías blockchain, pues actualmente existen diversas dicotomías, sobre todo con respecto al Reglamento General de Protección de Datos (RGPD).

Esta necesidad es básica, pues sin una clara regulación de la tecnología, la fiabilidad de los ciudadanos usuarios no será suficiente como para poder lograr la adopción masiva de la misma.

Además, es necesaria la existencia de una estandarización no solo para poder regular, sino también educar y promover el modelo de datos y protocolo de comunicación en las distintas entidades. La educación y formación en el área será necesaria para la correcta implementación y desarrollo de productos funcionando con esta tecnología.

1.3.1. Objetivos

Analizar el Estado del Arte de la Identidad Descentralizada, tanto de los modelos de datos existentes, como protocolos de comunicación e infraestructuras blockchain. Se analizarán sus casos de uso actuales, qué frameworks y entidades se han decantado por unos u otros, así como las ventajas y desventajas del uso de cada uno.

Tras ello se hará una prueba de concepto, empleando uno de los frameworks disponibles, para generar una credencial, intransferible y que contenga supuesta información personal de un usuario.

Este trabajo servirá como base para una futura investigación como tesis doctoral, donde el trabajo futuro tal que la indagación en las comparativas entre los distintos modelos, así como en las problemáticas derivadas de cada uno de sus enfoques criptográficos y de privacidad, será ampliada. También se explorarán maneras de comunicar los diferentes modelos entre sí, así como la posibilidad de conversión entre modelos de datos en caso de ser necesario.

2. Planificación y Metodología

2.1. Metodología

La metodología a emplear en este proyecto, debido a la estructura que posee donde se definen tres grupos principales de tareas, será la metodología en cascada. Esto es debido a que cada grupo depende del inmediatamente anterior y de su correcta finalización. El primer grupo consiste en el análisis del **Estado del arte**, donde se exploran e investigan las principales opciones existentes actualmente. Tras ello, se hará una **Comparativa**, con lo que es necesario haber comprendido y evaluado correctamente el apartado anterior. Finalmente, se realizará una **Prueba de concepto**, donde se aplicarán los conocimientos analizados previamente y se elegirá de entre las tecnologías preseleccionadas en la comparación.

Dentro de la primera fase las tareas se podrán realizar de forma paralela, evaluando cada protocolo existente para cada capa del modelo analizada. Del mismo modo ocurrirá con la comparativa. Sin embargo, en la demostración práctica, de nuevo se optará por este modelo en cascada, pues cada tarea (**Elección del Framework**, **Diseño**, **Ejecución** y **Conclusiones**) depende del cierre de la inmediatamente anterior.

2.2. Planificación

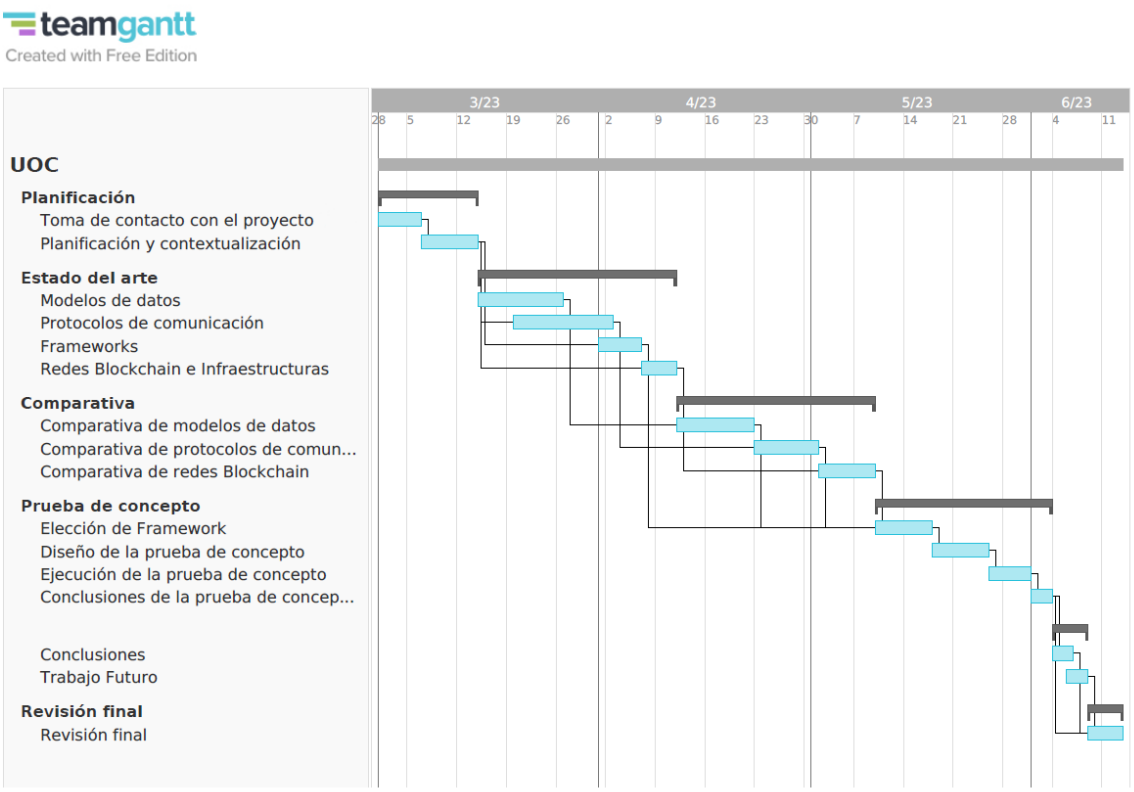

Figura 1: Planificación inicial

Vemos que la duración del proyecto ha sido organizada en distintas fases a lo largo de 16 semanas, entre inicios de marzo hasta mediados de junio. El proyecto finaliza tras la Revisión Final del mismo, tarea última

antes de la presentación y defensa del mismo.

- **Planificación.** Desde la semana 1 a la semana 3. Se subdivide en:
  - **Toma de contacto con el proyecto.**
  - **Planificación y contextualización.**
- **Estado del arte.** Desde la semana 3 a la semana 7. Se subdivide en:
  - **Modelos de datos.**
  - **Protocolos de comunicación.**
  - **Frameworks.**
  - **Redes Blockchain e infraestructuras.**
- **Comparativa.** Desde la semana 7 a la semana 11. Se subdivide en:
  - **Comparativa de modelos de datos.**
  - **Comparativa de protocolos de comunicación.**
  - **Comparativa de redes Blockchain.**
- **Prueba de concepto.** Desde la semana 11 a la semana 14. Se subdivide en:
  - **Elección de Framework**
  - **Diseño de la prueba de concepto.**
  - **Ejecución de la prueba de concepto.**
  - **Conclusiones de la prueba de concepto.**
- **Conclusiones y Trabajo Futuro.** Durante la semana 14. Se subdivide en:
  - **Conclusiones.**
  - **Trabajo Futuro.**
- **Revisión final.** Durante la semana 15.

2.3. Camino crítico

Dentro del proyecto encontramos dependencia entre tareas, por lo que un retraso en una supondría lo mismo para las subsiguientes. Estas son la **Planificación y contextualización** del proyecto, la cual determina el comienzo de todas las tareas que conforman el **Estado del arte**. Tras ello, la última tarea del mimo, **Redes Blockchain e Infraestructura**, determina cuándo podrá comenzar la última tarea, **Comparativa de redes Blockchain**, del siguiente bloque. Esta deberá terminarse, junto a las demás tareas englobadas bajo la comparación, antes de poder comenzar a realizar la **Prueba de concepto**. Esto es debido a que para poder llevar a cabo el demostrador es necesario haber analizado y comparado las diferentes opciones existentes. Una vez hecho, todas las tareas a llevar a cabo durante la parte práctica serán dependientes la una de la anterior debido a la metodología en cascada empleada. Para terminar, vemos una dependencia directa de las **Conclusiones**, el **Trabajo a Futuro** y la **Revisión final** con la finalización a tiempo del desarrollo del demostrador. Esto es lógico, pues para poder cerrar el proyecto es necesario

2.4. Presupuesto

Dado que todas las pruebas a realizar se elija la tecnología blockchain que sea se harán en redes de prueba y nunca en redes principales con criptomonedas con valor económico, el presupuesto será únicamente el siguiente.

Ordenador	700€
Conexión a Internet	25€ al mes, 4 meses ->100€

2.5. Seguimiento

El calendario se llevó a cabo prácticamente tal y como se planificó con las siguientes diferencias.

- En el apartado del Estado del arte, la subtarea de **Frameworks** fue pospuesta, realizándose justo tras la finalización de la tarea de **Comparativa** y previamente al desarrollo de la **Prueba de concepto**. Esta nueva planificación surgió de la falta de tiempo durante la fase de análisis del **Estado del arte**, la posibilidad de posponer esta subtarea dado que ninguna del siguiente bloque dependía de la misma, y a su mayor relación con la **Elección de Framework**, tarea con la que fue paralelizada durante el periodo de la **Prueba de concepto**.
- También, se tuvo que volver sobre la tarea **Protocolos de comunicación** durante el periodo dedicado a la **Comparativa**. Esto fue debido a una idea errónea, donde se equivocó uno de los protocolos OpenID4VC con OpenID Connect. Es por ello que se tuvo que dedicar tiempo a este subapartado durante el periodo equivalente a **Comparativa de protocolos de comunicación**. Por suerte, dada la similitud y dependencia de una tarea con la otra, pudo paralelizarse el trabajo, si bien dedicando alguna hora extra a lo previsto.
- Otro cambio con respecto a la planificación inicial fue la denominación y desarrollo de la parte de la **Prueba de concepto.** Debido a las características del demostrador empleado, no fue necesario dedicar tanto tiempo al **Diseño de la prueba de concepto**, pues se hizo uso de un demostrador ya preparado a nivel de diseño bajo el framework elegido para ello (como se verá más adelante este corresponde al de Hyperledger Aries Cloud Agent - Python). De este modo, **Ejecución de la prueba de concepto** abarcó la mayoría del tiempo de la subtarea inmediatamente anterior.
- Cabe destacar el desarrollo del Glosario como una tarea paralela durante todo el proyecto, anexo donde se han ido recopilando diferentes conceptos que pudieran necesitar de una definición pero no un apartado completo dentro del proyecto.

3. Estado del Arte

Para la realización del análisis del estado del arte, se plantearán los principales componentes relativos a la Identidad Digital Descentralizada en diferentes capas, siguiendo una estructura basada en el orden propuesto por entidades como DIF[3] y TrustOverIP[4].

Primero se tratará qué es el concepto de DIDs (*Decentralized Identifiers*). Tras ello, se construirá el apartado en torno a aquellos protocolos de comunicación que permiten la comunicación de identificadores sobre el ecosistema construido, la capa dos. Luego se presentará la capa tres, donde se muestran posibles modelos de intercambio de datos para dichos identificadores, como son las Credenciales Verificables de la W3C o los SoulBound Tokens de la Ethereum Foundation, los cuales definen las características y datos que se almacenarán y tratarán en el modelo. Finalmente, se explican algunos ejemplos pertenecientes a la cuarta capa, los frameworks.

Además, dado que en este caso la tecnología a utilizar para la descentralización del modelo será la cadena de bloques, se tratará una sección más referente a las posibles tecnologías blockchain a utilizar. Cabe destacar que esta sección no es contemplada en el modelo por capas de TrustOverIP, pues este no concreta en qué tecnología DLT hay que usar.

3.1. DID

Un DID (*Decentralized Identifier* o *Identificador Descentralizado*), es un identificador, es decir, una serie de datos asociados a una entidad, sea un individuo, colectivo u objeto, que no se encuentra custodiado de forma centralizada por una autoridad de registro. Esto implica que su almacenamiento y tratamiento se realiza de manera descentralizada, es decir, tiene lugar en una red descentralizada, la cual puede ser una DLT (Decentralized Ledger Technology), como por ejemplo, una red blockchain. Esta especificación en la tecnología a utilizar no se realiza todavía a este nivel de capa, pues se hace más adelante en el diseño de cualquier modelo.

Estos DIDs deben poseer, sin embargo, una serie de propiedades fundamentales para poder establecerse como Identificadores **Descentralizados**.

- **Permanencia.** Un DID ha de funcionar de tal modo que una vez asignado a una entidad, no ha de ser transferible y reasignado a otra entidad diferente a esta.
- **Resolubilidad.** El DID resuelve a un documento asociado a este, el cual contiene una serie de datos en un formato específico, donde no solo se refiere a la información propia del identificador, sino también las definiciones de la clave pública y servicios endpoint necesarios para establecer comunicaciones de confianza con el propietario de este DID en el momento que lo use, mediante la verificación del mismo.
- **Verificación Criptográfica.** La prueba criptográfica en el documento DID al que el DID resuelve que demuestra que el usuario dueño de un DID tiene el absoluto control sobre el mismo.
- **Descentralización.** Dado que los DID son generados y verificados mediante pruebas criptográficas involucrando a las diversas partes del modelo al que pertenecen, no se requiere una autoridad central de registro que lleve control de todo ello, sino simplemente un lugar de almacenamiento y comprobación de la emisión de dichas transacciones.

Cabe destacar que además, eidas2 establece en el ARF "*The Common Union Toolbox for a Coordinated Approach Towards a European Digital Identity Framework - The European Digital Identity Wallet Architecture and Reference Framework*", apartado 5.1.2 [**arf**], que los identificadores para cumplir con la legislación y reglamentos establecidos en el marco de la Unión Europea, han de poseer siempre la siguiente información, mucha de ella directamente relacionada con los requisitos establecidos anteriormente.

- Ha de contener la información requerida para identificar al proovedor de identificadores.
- Ha de contener la información requerida para realizar una comprobación sobre la integridad de los datos.
- Ha de contener la información requerida para verificar su autenticidad.
- Ha de contener la información requerida para validar el estado del identificador, si este ha podido expirar o haber sido revocado.
- Ha de contener toda la información requerida para poder realizar la verificación del usuario poseedor en el momento en el que se ponga en contacto con un proveedor de servicios que le solicite dicho identificador.

Esta recomendación también establece una serie de protocolos a utilizar en las distintas capas establecidas por TrustOverIP con el fin de lograr una uniformidad en el tratamiento de los datos contenidos en los DIDs.

3.2. Protocolos de Comunicación

3.2.1. OpenID Connect

OpenID Connect es un protocolo de comunicación creado como una capa añadida sobre OAuth 2.0 con el fin de dar soporte a técnicas de identificación ante entidades, las cuales pueden mediante ello asegurarse de que el usuario final es quien dice ser. En el protocolo el usuario envía a la entidad verificadora información sobre sí mismo usando una aplicación REST.

Debido a ello, es posible emplear este protocolo junto a alguno de los modelos de datos, expuestos en el apartado anterior, para que el usuario pueda enviar mediante dicha interacción atributos integrados en dichos modelos.

Es necesario resaltar que OpenID Connect parte inicialmente de OAuth2. Este protocolo se centra en la capacidad para otorgar autorizaciones sobre recursos de usuarios a aplicaciones que los requieran, sin que estos usuarios tengan que proporcionarles información confidencial o delicada para poder acceder a dicha información sabiendo de forma certera que son del usuario.

Un flujo habitual del uso de esta tecnología en una página consistiría en el siguiente esquema, el cual es el caso de uso más empleado del protocolo: *Authorization Code Flow with Proof Key for Code Exchange (PKCE).*[5]

Tenemos tres actores en el esquema:

- **Aplicación.** Servicio que desea obtener una pieza de información sobre un usuario sin necesidad de identificarle.
- **Proveedor.** Servicio completamente ajeno a la aplicación que posee la información solicitada por la aplicación sobre el usuario. Otorgará una autorización a esta sin identificar a la aplicación en su plataforma. Una metáfora para comprender este concepto puede ser un ticket para acceder a una sala de cine, el cual permite a un individuo (aquí sería la aplicación) acceder a una proyección de una película (el recurso) bajo ciertas condiciones (horario, día, si es en 3D o en 2D...), sin necesidad de identificar al usuario en el ticket.
- **Usuario.** El usuario de la aplicación el cual va a otorgar el acceso a la aplicación a cierta información suya almacenada en el proveedor.

Como podemos ver, el flujo más habitual de OAuth2.0, *Code Flow + PKCE*, consiste en las siguientes interacciones:[6]

1. La aplicación que solicita el recurso del usuario genera una URL de *callback* que asocia a la aplicación solicitante con el proveedor, un reto criptográfico (datos con los que se deberá hacer un cálculo) y un método para resolver dicho reto (la operación o cálculo a emplear con los datos previos) que la aplicación deberá resolver más adelante.
2. Tras ello, la aplicación redirige al usuario al proveedor que va a proporcionar el acceso a la información solicitada. Este se autentifica en este proveedor de confianza, introduciendo su nombre de usuario y contraseña propios de dicha plataforma necesarios para tal fin.
3. El proveedor genera un código de autorización que es enviado a la aplicación solicitante.
4. La aplicación devuelve al proveedor el secreto calculado producto del reto y el método junto al código de autorización y la URL de *callback* que identifica la relación aplicación - proveedor.
5. El proveedor comprueba que el reto es correcto mediante el método para resolver dicho reto criptográfico.
6. En caso de serlo, provee un token de acceso y de identificación a la aplicación, con el cual esta es autorizada a tratar los datos para los que se le ha concedido el permiso del modo en el que el usuario lo ha aceptado y no otro. Estos tokens además suelen tener un uso temporal limitado, así como estar completamente restringidos para el propósito para el que han sido generados.
7. La aplicación ahora puede solicitar al proveedor la información mediante dicho token.
8. El proveedor comprueba la información del token, y siendo correcta, proporciona los accesos correspondientes a la aplicación.

Como puede verse, este protocolo de comunicación permite la autorización de una aplicación sobre unos permisos o datos concretos del usuario ubicados en otra entidad. Sin embargo, si necesitamos además validar que el usuario es quien dice ser, añadimos una capa de identificación. Esta nueva capa de autenticación es lo que conforma OpenID Connect. Para ello, se definen sobre el protocolo de OAuth2.0, datos concretos, o *scopes* que identifiquen de algún modo a este usuario. Por ejemplo, su correo electrónico o perfil de usuario. La información correspondiente a estos datos se incluye dentro de un nuevo token *id_token*, el cual se envía en el flujo de datos anterior junto al token de autorización.

3.2.2. OpenID4VC

Del protocolo anterior surge OpenID4VC, el cual adapta las posibilidades de este al modelo de Identidad Descentralizada.

Este protocolo parte de la base de Self-Issued OpenID, protocolo que permite realizar las funciones de OpenID Connect convirtiendo al propio usuario en su propio servicio de OpenID Provider. De este modo, el usuario presenta su información mediante una credencial estableciendo la comunicación de manera directa a los proveedores de servicios, firmándola y certificándola por sí mismo, sin un proveedor ajeno que actúe de intermediario y ceda los datos que posea sobre el usuario.[7]

Sin embargo, este protocolo no permite por sí solo el envío por parte del usuario de credenciales firmadas y certificadas por otras entidades. Esta capa fue añadida por OpenID4VC, donde el rol de las entidades emisoras es establecido.

OpenID4VC se construye sobre el concepto de Decentralized Identifiers (DIDs), donde tenemos una serie de credenciales almacenadas de manera descentralizada y emitidas por alguna entidad a los usuarios poseedoras

de las mismas, de tal modo que su validez es comprobable por los proveedores de servicios mediante el uso de firmas digitales.

Este protocolo permite básicamente tres tipos de acciones entre los actores que componen el modelo de Credenciales Verificables.[8]

- **Emisión de Credencial.** Un usuario solicita a una entidad reconocida y registrada la emisión de una Credencial Verificable. Esta entidad se la envía y registra dicha transacción de modo que otras entidades verificadoras puedan corroborar qué ocurrió y en qué momento lo hizo.
- **Presentación de Credencial.** El usuario se pone en contacto con una entidad verificadora, la cual le solicita identificarse ante ella. Este le envía la Credencial Verificable como una Presentación Verificable. La entidad verificadora es capaz de comprobar que la entidad emisora de dicha información es una entidad corroborada y de confianza. Así, acepta dicha información. Además, los usuarios pueden presentar esta información eligiendo qué datos quieren mostrar y cuáles no, evitando así revelar más información de la estrictamente necesaria para el acceso a dicho servicio.
- **Presentación mediante Pseudónimo.** Para mantener la privacidad del usuario en el momento de la presentación ante una entidad, el protocolo permite el uso de pseudónimos, permitiendo al usuario autenticarse ante la entidad sin tener que revelar su identidad.

Además, el protocolo no se cierra a que las wallets que maneja el usuario le notifiquen sobre la autenticidad de los proveedores de servicios y verificadores ante los que se presenta. De este modo, el usuario también puede autenticar la identidad de una entidad a la que proporciona sus datos. En caso de que esta no esté registrada en un sistema que permita su identificación ante el usuario, se avisará a este de los posibles riesgos, pero tampoco se le impedirá completamente la presentación de su información a una entidad verificadora no autenticada.[9]

Este protocolo de comunicación no está limitado a un modelo de datos para credenciales verificables concreto, por lo que admite la flexibilidad de los numerosos modelos existentes[10] para la generación de credenciales verificables.

De otro modo, si se cierra el protocolo de comunicación a un modelo específico para la transferencia de credenciales, pueden suceder problemas de incompatibilidad, sobre todo en el momento actual del Estado del Arte, donde múltiples comunidades se encuentran desarrollando diferentes modelos y protocolos con el fin de resolver diferentes problemas que se presentan en una tecnología tan nueva.

Además, permitiendo la flexibilidad en el modelo de credenciales verificables, no ya solo a nivel de protocolo, sino también de los distintos actores involucrados, el modelo se asegura de que la responsabilidad para dar el formato adecuado o transformar de uno a otro los diferentes modelos existentes no recaiga en una única entidad. Sino, los desarrolladores de wallets podrían esperar que fuesen los issuers los que se adaptasen a cada wallet en el mercado, de las cuales hay una enorme diversidad; y los issuers pudieran esperar que han de ser las wallets las que deben adaptarse, admitiendo múltiples formatos, incluso aquellos no estándar, dando problemas a la hora de desarrollarlas, elevando muy probablemente sus costes.[9]

Es por su sencillez, su facilidad de integrar a partir de OpenIDConnect, y por la admisión de múltiples formatos que en el ARF "*The Common Union Toolbox for a Coordinated Approach Towards a European Digital Identity Framework - The European Digital Identity Wallet Architecture and Reference Framework*"[**arf**], se establece el uso de este protocolo de manera obligatoria para credenciales que necesiten un nivel de confianza alto, y se recomienda para el resto (página 36). Cabe destacar que esta referencia establece el uso del modelo de W3C para la generación de Credenciales Verificables para ciertos DIDs de los que se requiera un nivel de confianza alto (como por ejemplo un pasaporte), y lo recomienda para el resto (como puede ser un certificado de una academia online no oficial). También, al mismo nivel, refiere al uso del modelo de la ISO/IEC 18013-5:2021 para licencias de conducir.[11]

3.2.3. DIDComm

DIDComm es un estándar para la creación de código y diseño de patrones y protocolos para el ecosistema de DIDs. Su objetivo actualmente es establecer la comunicación entre entidades mediante el uso de DIDs, es decir, que la confianza en dicha transmisión y ecosistema dependa de dichos identificadores y no de propiedades de la capa de transporte.

Se realizó teniendo en mente la comunicación entre las entidades involucradas en dicho sistema, sea cual fuese el modelo de DID que empleen. De esta manera permite la flexibilidad e intercomunicación entre diferentes estructuras y modelos de DIDs del mismo modo que OpenID4VC lo hace.

Para ello, DIDComm crea un canal de autenticación por el cual los actores del modelo de DIDs interactúan con seguridad garantizada y que permite la prueba de autenticación de los DIDs presentados para que las entidades se identifiquen y verifiquen adecuadamente entre sí.

Como puede verse, el protocolo permite desde el principio la autenticación en ambos sentidos, es decir, no solamente del usuario al proveedor de servicios, sino también a la inversa. De este modo el usuario sabrá en todo momento a quién le está presentando los datos que haya seleccionado.

Inicialmente este protocolo nace en el contexto de Hyperledger como parte del proyecto para el framework Aries,3.4.1 pero, sin embargo, este proyecto es separado de Hyperledger Indy con el fin de poder desarrollarlo en un contexto donde las especificaciones tecnológicas no supongan una delimitación para el mismo. De esta manera se busca iterar y crear el código base de una manera más agnóstica a la blockchain en la que se quiera implementar, abriendo el abanico a tecnologías de cadenas de bloques diferentes ajenas a Hyperledger.

El siguiente paso, la versión 2 del protocolo, resultó de la independencia con respecto también al proyecto Aries. Esto fue debido a que para lograr una mayor interoperabilidad a nivel de la capa de comunicación, las especificaciones de DIDComm tenían que ser escritas sin basarse en previas implementaciones de Aries. Así, DIDComm se independiza del contexto de Hyperledger y termina funcionando como un estándar abierto de la DIF (*Decentralized Identity Foundation*) para comunicación en entornos de identidad descentralizada.

Entre las características que esta versión DIDComm v2 posee se encuentran las siguientes:

- Desarrollado como estándar de DIF por un grupo de trabajo específico.
- En desarrollo de forma abierta con desarrolladores que no necesariamente provienen de un entorno de trabajo exclusivo de Aries y/o Hyperledger Indy.
- Uso de estándares JOSE (*Javascript Object Signing and Encryption*) para el cifrado de las transferencias entre entidades.
- Soporte para diversos modelos de DIDs.
- Conexión para el handshake streamlined, es decir, el canal establecido entre ambas partes prioriza la velocidad antes que la dedicación como en la versión 1 del protocolo DIDComm.

El desarrollo de esta tecnología para la comunicación de servicios de identidad descentralizada sigue en crecimiento, y entre los objetivos buscados para permitir esto se encuentran:

- Autenticación por ambas partes. De este modo, el proveedor de servicios identifica al usuario y además, el usuario sabrá a quién va a enviar la información relativa a sus credenciales.
- Soporte para nuevas primitivas para transporte de información de manera segura, como JWM (*JSON Web Message*), el cual busca expandir los estándares JOSE.
- Funcionamiento mediante Bluetooth o códigos QR para realizar identificaciones offline de los usuarios. Esto puede ser útil en entornos en los que se requiera la presencialidad del individuo.

Además, el protocolo permite el desarrollo de subprotocolos, por lo que un protocolo DIDComm se puede personalizar para intercambios de cierto tipo de DIDs o nuevos modelos, manteniendo la uniformidad del DIDComm genérico para permitir la compatibilidad entre comunicaciones. Esto permite flexibilidad a la hora de hacer uso de este protocolo.

Cabe destacar que DIDComm no está únicamente orientado a modelos de credenciales, sino también se puede emplear para mensajería, interacciones con sistemas IOT mediante identificación segura de los dispositivos o coordinación de pagos.[5]

3.3. Modelos de datos

Uno de los ámbitos donde más modelos se están creando es a la hora de crear un esquema para los datos representados como Identificadores Descentralizados. Para ello, muchas comunidades optan por la posibilidad de crear una credencial, es decir, una afirmación sobre una entidad específica; y hacerla verificable, es decir, que otras entidades miembro de su modelo puedan comprobar que esa información ha sido emitida por una entidad capacitada y autorizada para ello. Así, se le otorga validez intrínseca al dato.

En la siguiente tabla pueden verse algunos de los diversos formatos creados únicamente para el concepto de credenciales verificables. Cabe destacar que todos ellos pueden emplearse con numerosos algoritmos criptográficos para firma, verificación de entidades, manejo de claves o revocación, creando un gran número de combinaciones para modelos posibles. [10]

Formatos para Credenciales Verificables	**Estandarización**
AnonCred	Community Spec
LDP-VC	W3C
MDOC	ISO
JWT	IETF
JWP	DIF
JWT-VC	W3C
SD-JWT	IETF (OAuth WG)
SD-JWT-VC	IETF (OAuth WG)
CWT	IETF
ACDC (KERI)	IETF (con intención de cambiase a Blockchain WG)
CESR/CESR Proof	IETF (con intención de cambiarse a Blockchain WG)
ICAO DTC	ICAO
Gordian Envelope	IETF y W3C
Idemix attribute-based credential	Privacy by Design Foundation

Todos estos formatos han sido creados independientemente entre sí, provocando una enorme diversidad en representación de los datos, no solo a nivel de atributos, sino también de tecnologías empleadas para ello.

En este apartado vamos a centrarnos en las Credenciales Verificables de la W3C debido a su amplia adopción en territorio europeo [12] , y como contrapartida, se mostrará un modelo completamente diferente basado en la estandarización de Ethereum, enfocado en una mayor descentralización de la información y eliminando ciertas dependencias de algunas entidades a la hora de la revocación. Se definirá así el concepto de SoulBound Tokens, y se verán también protocolos previos a este que sirvieron para su creación, como los *Non-Fungible Tokens* o *Proof of Attendance Protocol*.

3.3.1. Credenciales Verificables W3C

Las Credenciales Verificables que aquí se definen consisten en un modelo de datos actualmente determinado por el W3C. Estas consisten en la definición de un esquema digital para la emisión y presentación de información a entidades, donde la firma digital y el sello temporal de las mismas prima, dándoles valor ante dichas entidades verificadoras. Dentro de su modelo podemos encontrar tres roles diferentes. [13]

- **Issuer.** Entidades que son capaces de generar una credencial firmada, y cuya firma queda registrada en una red descentralizada con el fin de ser comprobable en cualquier momento por otras entidades, independientemente del estado de la entidad emisora.
- **Holder.** Usuarios del sistema. Son los dueños de las Credenciales Verificables, y pueden solicitar su generación a Issuers y mostrarlas a entidades que se las soliciten. Son los únicos dueños de los datos plasmados en las credenciales.
- **Verifier.** Son las entidades encargadas de verificar la validez de una credencial presentada por un Issuer. Para ello comprueban que la firma adjunta en las mismas pertenezca a la de un Issuer registrado con la potestad para emitir dichas credenciales.

Las características principales de las Credenciales Verificables son las siguientes.

- **Únicas.** Cada credencial consiste en un token único registrado mediante una transacción en la tecnología blockchain a utilizar en el momento de su emisión. Esto permite que cada Credencial Verificable sea totalmente diferenciable del resto y permita la gestión individualizada de cada una, así como la agrupación de las mismas en Presentaciones Verificables en caso de ser necesario.
- **Asociadas a una cartera digital.** Una vez emitidas, estas credenciales quedan almacenadas en el dispositivo personal del usuario, en concreto en su cartera digital, desde la cual puede interactuar con ellas. Así podrá visualizarlas y presentarlas, tanto de forma individual como agrupadas, y de forma completa como generando valores a partir de las mismas que permitan mostrar la información de manera parcial para preservar la privacidad del usuario dueño de las mismas.
- **Emitidas por una entidad con cierto nivel de confianza.** Las entidades capaces de emitir estas credenciales han de ser registradas y valoradas previamente por una entidad superior, lo cual centraliza en parte el modelo pero proporciona un nivel de confianza a la hora de comprobar las emisiones de credenciales. Estas entidades emisoras son registradas en una blockchain a la cual las entidades verificadoras tengan acceso de lectura, para que a la hora de que comprueben su firma digital, puedan hacerlo fácilmente comprobándola en la blockchain.
- **Firmadas de forma implícita.** Además de estar relacionadas mediante al transacción de emisión con la entidad que las ha generado, estas contienen un campo de datos el cual corresponde a la firma digital de quien las haya emitido, así como otro campo para especificar el algoritmo de firma empleado. Esta información se incluye implícita con el fin de que cada vez que se presente ante una entidad verificadora o un proveedor de servicios, estos puedan atestiguar que la firma incluida es la de una emisora de confianza.
- **Con sello temporal explícito y de caducidad.** Dos de los campos de las Credenciales Verificables son la marca temporal del momento de emisión de la misma y, en caso de existir, una fecha a partir de la cual esta deja de ser válida. Este dato, de estar en una credencial, podría ser comprobado por las entidades verificadoras en el momento de la comprobación, desechándola si está caducada.
- **Información privada.** La información de la credencial no es almacenada en la blockchain en texto plano, de tal modo que aunque la tecnología empleada para ello sea pública, los datos como tal no son

almacenados en la misma sino o bien de forma local en el dispositivo del usuario o en algún servidor ajeno, centralizado, o en una blockchain privada que permita el acceso únicamente a ciertas entidades. El modelo admite diversas formas de presentación, entre las cuales se incluyen modelos criptográficos *ZK-Proof*, es decir, pruebas de conocimiento cero. Como se especificaba antes, el usuario será capaz además de presentar información deducida de las mismas en lugar de los datos completos. Un ejemplo puede incluir la demostración de la mayoría de edad a partir de la fecha de nacimiento, presentando únicamente la verdad de ese valor en lugar de la fecha completa (generando el mensaje demostración a presentar a partir de la comprobación sobre la Credencial Verificable de la fecha de nacimiento, en lugar de mostrar día, mes y año).

- **Presentables ante entidades verificadoras que admitan su modelo.** Las Credenciables Verificables pueden presentarse ante entidades verificadoras, tanto de forma individual como agrupadas en Presentaciones Verificables, las cuales incluyen tanto las firmas individuales de cada credencial, como la firma de la propia presentación. Esta acción siempre se lleva a cabo con el usuario como única entidad con control sobre sus propios datos, eligiendo siempre qué datos desea compartir con la entidad verificadora. Esto permite, por ejemplo, que el usuario demuestre que está en posesión de cierta información sin necesidad de mostrarla, sea mediante el uso de Pruebas de Conocimiento Cero (*ZK-Proof*), o la generación de mensajes con información parcial sobre sus Credenciales Verificables .
- **Admisión de distintos tipos de firmas digitales, siempre y cuando se especifique en los metadatos el algoritmo empleado.** Uno de los campos de las Credenciales Verificables es la especificación del algoritmo empleado para la firma de la misma en el momento de la emisión. De este modo se logra bastante flexibilidad a la hora de emplear algoritmos de firma digital, pudiendo convivir en el ecosistema entidades que hagan uso de diferentes métodos.

Además, el modelo permite la presentación de estas credenciales de manera agrupada en una única Presentación Verificable. Esta estructura permite firmar el conjunto de todas, manteniendo a su vez las firmas individuales para una comprobación distinguida, y además permitiendo añadir información que pudiera ser necesaria respecto al conjunto.

Cabe destacar que dentro de la European Blockchain Service Infrastructure (EBSI), el modelo de datos para credenciales verificables es el aquí descrito, de la W3C.[14] De este modo se logra la estandarización de la manera de presentar datos en Internet

3.3.2. SoulBound Tokens

Los SoulBound Tokens o SBTs son un estándar de Ethereum, ERC-4973, el cual hereda del estándar ERC-721 (Non-Fungible Tokens), el cual se mencionará más adelante. Por ello, es exclusivo de las cadenas que funcionen mediante la Ethereum Virtual Machine.

Estos tokens poseen la característica de que, una vez vinculados a una cuenta, no son transferibles excepto a la cuenta NullAddress mediante la técnica de *burning*. Esta acción podrá ser ejecutada únicamente por el emisor del SBT y/o por el holder, estableciéndose este rol en el momento de la creación del mismo. Los SBT poseen, además, las siguientes características, muchas de ellas heredadas del estándar ERC-721.

- **Únicos.** Cada token es rastreable independientemente del resto, lo cual permite la correcta diferenciación de información, así como la presentación de cada dato de forma individualizada. Esto permite el aislamiento de la información, restringiendo cada credencial a un único token.
- **Pertenecientes a una cuenta.** Cada token está asociado única y exclusivamente a una dirección en la red, sea una Externally Owned Account (EOA) o no, por lo que puede asociarse también a Smart Contracts. De este modo es posible identificar a su propietario en una relación uno a uno. Al permitir

su asociación a contratos, se pueden asignar características a estos de manera pública, pudiéndose así demostrar la fiabilidad de los mismos o algún tipo de categorización.

- **Metadatos inscritos en la cadena.** Los metadatos de los SBTs quedan inscritos en la cadena, de tal forma que estos quedan asegurados de forma completamente descentralizada. Estos metadatos pueden contener la dirección del emisor, fechas de validez del token, tipo de token a modo de categorización, así como cualquier tipo de atributos que puedan ir contenidos en él (nombre de la entidad emisora, nombre del propietario, tipo de certificación...). También la dirección que apunte al contenido del propio token en caso de poseer algún tipo de media, como imagen o vídeo.
- **Intransferibles.** A la hora de emitir estos tokens se establece entre el emisor y el receptor quién posee el poder de quemarlos, es decir, de enviarlos a la cuenta Null Address. Este rol puede quedar asignado a la entidad emisora, al holder y propietario del token, o bien establecer que se debe llegar a un consenso entre ambas entidades. Una vez el token es emitido y alcanza la dirección del usuario, este no es transferible a ninguna otra cuenta (a parte de la Null Address por quien se estableció en el consenso anterior), estableciendo una relación segura e inconfundible entre ambos cara a entidades verificadoras o cualquier entidad que pueda y quiera consultar la información en la cadena.
- **Sello Temporal implícito**. Al hacerse uso de la tecnología Blockchain, estos tokens poseen de forma implícita una marca temporal, la cual corresponde con la de la transacción en la que se realiza la emisión del SBT a la wallet del propietario.
- **La información inscrita queda completamente pública.** El diseño original de los SBTs supone mostrar de forma completamente pública todos los datos contenidos en ellos. Esto implica que no es necesaria una presentación de la credencial como tal, pues las entidades verificadoras pueden realizar esta comprobación en cualquier momento en la cadena. Si bien esto trae desventajas sobre la privacidad con respecto al modelo de Credenciales Verificables de la W3C, posee a su vez otras ventajas. Algunas de ellas pueden ser la fácil identificación por parte de entidades que buscan realizar un *airdrop* de un determinado token y buscan usuarios que puedan estar interesados en su modelo de negocio. Para ello, pueden realizar dicho airdrop a cuentas que posean SBTs de características determinadas, como por ejemplo que demuestren la asistencia a congresos sobre materias relacionadas con su idea.

3.3.3. Non-Fungible Tokens

Los Non-Fungible Tokens, conocidos por sus siglas como *NFTs*, son un tipo de estándar de token adoptado en múltiples cadenas, como Ethereum (estándar ERC-721), Bitcoin (Ordinal Incriptions), Binance (BEP-721) o Cardano (CIP 25 - Media NFT Metadata) entre otras.

Las características que aúnan a todos estos estándares (con algunas excepciones para los Ordinals en Bitcoin, de los cual se hablará más adelante en este mismo apartado) son las siguientes.

- **No fungibles.** Es decir, no son intercambiables uno a uno entre sí. Cada token NFT posee su propio valor y es distinguible del resto de NFTs dentro de la blockchain utilizada para su almacenamiento. De este modo se logra tener activos digitales únicos e independientes del valor de la divisa propia de la cadena y del resto de tokens. Cabe destacar que existe una variación de este estándar, los tokens Semi-fungibles (SFTs, en Ethereum son el estándar ERC-1155), los cuales si bien son distinguibles entre sí, permiten la creación de varios tokens de un mismo activo digital. Entre ellos conservarían el mismo valor, pero no con otros SFTs con atributos y metadatos distintos.
- **Pertenecientes a una cuenta.** Los NFTs están asociados a una cuenta del mismo modo que una divisa o token fungible pudiera estarlo. Esta cuenta corresponde con una dirección en la red blockchain elegida, pudiendo ser una address de un usuario, de otro Smart Contract, o la Null Address en caso de que el NFT en cuestión se haya enviado a quemar (proceso de burning). Quien posea acceso a la firma de

transacciones mediante dicha cuenta es el propietario del token, pudiéndose enviar, listar por un precio en mercados secundarios, o identificarse con él en páginas que lo permitan para realizar múltiples operaciones posibles. Si el NFT cambia de cuenta, el dueño anterior perderá todos los permisos y utilidades que el token pudiera ofrecerle.

- **Metadatos inscritos en la cadena.** Los metadatos del token consisten en atributos que pueda tener asociados, URL a multimedia asociada (sea un servidor centralizado o IPFS), o número del token dentro de una colección, entre otros. Estos se almacenan en la blockchain de forma inmutable y descentralizada. Cabe destacar que la inmensa mayoría de NFTs poseen una URL a otro servicio de almacenamiento donde se encuentra una imagen, video u otra multimedia. Esto es debido a que la subida de archivos pesados a las blockchains que soportan este tipo de estándar es extremadamente costoso en poder de computación y por lo tanto en pago de tasas.
- **Transferibles.** Los NFTs son transferibles, es decir, no están vinculados de manera permanente a la cuenta que los recibe. Esto ha implicado la creación de mercados en torno a ellos, así como la posibilidad de deshacerse de un token de estas características en cualquier momento. Cabe destacar que esta característica choca con el concepto establecido de que un DID debe poseer la propiedad de permanencia y no poder ser reasignado a nuevas entidades tras su emisión inicial.
- **Sello Temporal implícito**. Del mismo modo que los SBTs, en el momento de su emisión queda escrita una marca temporal, correspondiente a su transferencia, en la blockchain. Cabe destacar que cada vez que este token sea transferido, se generarían nuevas marcas temporales para cada movimiento, estableciendo el momento inicial en el que un usuario diferente pasa a ser propietario.

Cabe destacar que los Ordinals (NFTs acuñados en la cadena de Bitcoin) poseen algunas características diferentes.

Los Ordinals nacen exclusivamente en la cadena de Bitcoin como un modo de seguir el rastro a satoshis, los cuales si bien son fungibles, es posible realizar una cierta diferenciación mediante la técnica First In First Out, enumerándolos. De este modo, es posible individualizarlos sin alterar su funcionamiento original; una unidad de medida inferior al Bitcoin, del mismo modo que los céntimos son a los euros. Esta forma de trackear los satoshis permite, junto a la actualización de 2021 Tatproof, almacenar información arbitraria en las transacciones y bloques de Bitcoin, con el mismo nivel de seguridad y descentralización que la propia red. Así nace el concepto de las inscripciones o Inscriptions, las cuales consisten en integrar datos arbitrarios en una transacción de la red Bitcoin. Si bien existe a día de hoy una limitación de 400 MB, este espacio de almacenamiento es suficiente para guardar imágenes o incluso pequeños vídeos. La descentralización de este paradigma lo hace muy seguro frente a los anteriores. Sin embargo, desde que su uso ha comenzado a extenderse, la red de Bitcoin a comenzado a verse cada vez más y más saturada. Esto hace que las transacciones sean extremadamente lentas. Por poner un ejemplo, se comenzó un minteo de un ordinal *Honey Badger* a una tasa de 11,070 sat, el 23 de marzo de 2023 a las 16:40. La transacción terminó el 25 de marzo a las 18:22. La extrema lentitud de este sistema va en contra del requisito de escalabilidad. Más todavía teniendo en cuenta que si inscribimos una certificación como un ordinal, esta competiría en la pool con todos los NFTs que pudieran estar inscribiéndose a la vez. Cabe destacar también que el sistema todavía no ha sido optimizado, teniendo que, por ejemplo, hacer uso de software específico y de una wallet sin BTC. Esto es debido a que el envío de BTC podría enviar con ello los ordinals, pues la wallet interpretaría todos los satoshis en ella como iguales, fungibles, y a la hora de enviar una cantidad fija de satoshis para hacer un pago, cogería dicha cantidad de satoshis de la wallet independientemente de la inscripción que pudieran tener contenida. La dificultad de uso y responsabilidad que esto puede causar en el usuario promedio, es otro motivo en contra de la adaptación de este modelo de datos para las certificaciones académicas.

Proof of Attendance Protocol

También conocidos por sus siglas, *POAP*, consisten en tokens no fungibles (NFTs), siguiendo el estándar ERC-721, por lo tanto propios de Ethereum, con la adición de que son acuñados por los Smart Contracts de POAP, los cuales están gobernados actualmente por POAP Inc. Comparten todas las características, por lo tanto, de un NFT, aunque estos son generados de un modo concreto.

A diferencia de los NFT que abundan en el mercado, predominando aquellos relacionados con coleccionables, los POAP se crearon con el fin de conmemorar un evento. Este puede ser una fiesta o celebración, una conferencia o congreso, o el lanzamiento de un producto. La idea consiste en que únicamente los asistentes o participantes puedan reclamar el POAP asociado a dicho suceso. Así por ejemplo los asistentes a un congreso internacional en concreto podrían recibir una contraseña que se solicite posteriormente para la acuñación del POAP, o bien acceder a un enlace privado mediante un código QR durante el trascurso del evento desde donde poder acuñar el token en cuestión.

Para la gestión de los mismos existe una aplicación móvil, en la cual es necesario establecer una dirección de Ethereum principal. Tras ello, los usuarios pueden reclamar los token registrados en los Smart Contrats de POAP mediante diferentes técnicas, como contraseña, código QR o código. En el momento en el que esta acción se lleva a cabo, el usuario realmente está acuñando un NFT, con la diferencia de que esta acción se realiza en la blockchain Gnosis, una blockchain basada en la Ethereum Virtual Machine. De este modo no hace falta pagar tasas de gas y la creación de los tokens es gratuita.

Si bien al tratarse de NFTs son transferibles, su valor no reside en la colección o compra y venta, sino en algo más sentimental. Esto no implica que los usuarios no puedan venderlos en mercados de NFTs como Opensea o Blur.

3.4. Frameworks

Para facilitar la implementación de todas las capas definidas por Trust over IP existen diversos *frameworks* que facilitan la implementación y ayudan a mantener las soluciones uniformes entre sí. De este modo, desarrollar aplicaciones finales se vuelve más sencillo, pues estos frameworks ya proporcionan las conexiones entre las diferentes capas del modelo ToIP, así como una serie de reglas para mantener la uniformidad entre los diferentes sistemas desarrollados haciendo uso del mismo.

3.4.1. Hyperledger Aries

Hyperledger Aries forma parte del conjunto de proyectos creados en Hyperledger, siendo aceptado en marzo de 2019. Este consiste en una infraestructura para la creación, transmisión y almacenamiento de Credenciales Verificables, todo ello de manera descentralizada. Cabe destacar que para su funcionamiento se complementa con Hyperledger Ursa,[15] la librería creada por Hyperledger para la gestión de algoritmos criptográficos.

Aries permite la creación de *agentes*, entidades que representan a los diferentes actores existentes en el modelo de identidad descentralizada, de manera individual. Poseen sus propias claves criptográficas privadas. Su interacción se produce sobre el protocolo para comunicaciones de DID-Comm,3.2.3 y opera en la Blockchain de Hyperledger-Indy, por lo que está orientado a un enfoque permisionado, como son los productos de Hyperledger.3.5

Sus características principales son las siguientes:

- Posee una capa para la interfaz blockchain, mediante la cual es capaz de realizar diversas interacciones con la cadena de bloques mediante transacciones.
- Incluye un elemento de almacenamiento criptográfico para guardar tanto claves como Credenciales Ve-

rificables, o cualquier otro elemento identificador y/o privado que pudiera requerirse para la generación de los actores del modelo.

- Comunicación peer-to-peer mediante DIDComm, permitiendo además el intercambio de mensajes sin uso del *ledger* en caso de que sea necesario mediante protocolos de transporte.
- De la mano del uso de DIDComm, hereda la capacidad para generar y trabajar con credenciales en múltiples formatos. Combinado con Hyperledger Ursa es posible realizar intercambios de Credenciales Verificables que empleen Zero Knowledge Proof.
- Una serie de protocolos de alto nivel y un subconjunto de los mismos, denominados "*Aries Interop Profiles*", los cuales permiten la implementación de manera independiente e interoperable de agentes de Aries.
- Diversos frameworks ya implementados orientados a casos de uso y aplicación de tecnologías específicas, como la wallet digital o la versión en la *nube* del agente de Aries.
- Un agente para pruebas que permite establecer pruebas de interoperabilidad continua entre agentes y frameworks dedicados al manejo y creación de agentes.

Como podemos ver, los fines de este framework son la creación de entornos DID de manera permisionada, y con un enfoque más centrado en la empresa privada debido a ello; así como la posibilidad de crear diversas entidades

3.4.2. AlastriaID

Alastria consiste en un ecosistema, también enfocado en blockchains permisionadas para entornos empresariales, donde una de sus soluciones principales es la Identidad Descentralizada.

Para ello, Alastria cuenta con su propio conjunto de contratos inteligentes desarrollados en Solidity. Estos pueden desplegarse en la red de Quorum y su funcionamiento es completo. Actualmente, se encuentra en desarrollo la implementación de este modelo también en las redes de Besu Red-b y de Hyperledger Fabric Red-H. Cada una de estas redes posee sus propias características:

- **Quorum.** La red Quorum es una blockchain empresarial, cuyo foco de atención son las aplicaciones privadas. Fue desarrollada por J.P. Morgan, y posteriormente cedida a Consensys (empresa líder también tras Metamask, Truffle, Infura y Diligence). Esto añade una capa de protección a las transacciones, asegurándose de que no haya actores indeseados con acceso a la información que se esté tratando en ellas. Emplea el protocolo de Ethereum, por lo que permite la implementación de Smart Contracts y de los estándares propios de Ethereum. También añade algunas modificaciones, como el uso de su propio protocolo de consenso, *QuorumChain*, donde solo una parte de los nodos de la red ha de estar de acuerdo para aprobar una operación en la misma.
- **Besu.** La red Besu es otra red blockchain orientada a fines empresariales, también desarrollada por ConsenSys, con compatibilidad con sus otras soluciones. Besu también emplea el protocolo Ethereum para su implementación, facilitando la creación de aplicaciones mediante sus contratos inteligentes. Como Quorum, también ofrece la posibilidad de crear transacciones privadas, aunque difiere en el funcionamiento del algoritmo de consenso. Besu emplea tanto *Proof of Work*, como *Proof of Authority* y también IBFT2 (Istanbul Byzantine Fault Tolerance 2.0), un algoritmo innovador que mejora con respecto a su anterior versión en una mayor flexibilidad, un menor número de nodos necesarios para alcanzar el consenso.

- **Fabric.** Fabric es una plataforma blockchain parte del proyecto HYperledger, desarrollado por la Fundación Linux. Posee un enfoque modular, además de modelos de permisionado flexibles, canales privados y diversos algoritmos de consenso, como Practical Byzantine Fault Tolerance (PBFT), diseñado para tolerar fallas bizantinas; Solo, donde se centraliza el consenso en un único nodo, centralizando el algoritmo; o incluso el uso de Kafka, una plataforma de transmisión de datos distribuida para la propagación de eventos y transacciones entre nodos.

Como puede verse, el framework de Alastria puede implementarse en diversas blockchains, pero todas ellas con una característica en común, la cual es que todas ellas son redes permisionadas.

3.4.3. EBSI Verifiable Credentials Framework

La *European Blockchain Service Infrastructure*, también conocida como EBSI, es una iniciativa dentro del marco europeo para crear un entorno basado en tecnologías blockchain donde los países miembro sean los participantes. Para ello, cuentan con su propia red blockchain permisionada formada por diferentes nodos repartidos por la Unión Europea. Debido a esta territorialidad tan definida, esta infraestructura se lleva a cabo teniendo en cuenta el marco legal europeo, tales como los diferentes estándares y reglamentos que regulan en sus países.

Dentro de la EBSI podemos encontrar tres tecnologías clave.

- **Verifiable Credentials.** El modelo de Credenciales Verificables admitido por la Unión Europea dentro del marco de la EBSI es el de la W3C y el de la ISO/IEC 18013-5:2021 para licencias de conducir.[11] Además, la comunicación entre actores del modelo para la creación, emisión y presentación de las Credenciales Verificables es OpenID4VC, permitiendo las interacciones básicas entre los agentes, y además interoperabilidad entre los modelos de datos de las credenciales. Esto es debido a que los modelos de la W3C y de la ISO solo son obligatorios para credenciales que necesiten un nivel alto de privacidad, tales como documentos de identidad o pasaportes; pero otras credenciales, si bien es recomendado, no tienen por qué hacerlo. De este modo, con OpenID4VC, el sistema se asegura que de todos modos exista la intercompatibilidad con otros posibles modelos.

- **Digital Wallet.** La especificación para la construcción de wallets digitales es también definida con el fin de permitir la interoperabilidad entre las distintas soluciones y *DApps* (Decentralized Applications). En ellas se almacena la información de los usuarios, es decir, los datos asociados a sus Credenciales Verificables, sobre las cuales solamente ellos poseen absoluto control. Para su correcta implementación, se define el framework *The Common Union Toolbox for a Coordinated Approach Towards a European Digital Identity Framework*[12] junto con *eIDAS electronic IDentification, Authentication and trust Services*, la plataforma que regula en Europa las identificaciones electrónicas, y que además, tras la incorporación de las tecnologías blockchain para desarrollar soluciones de Identidad Digital, se actualizó a *eIDAS2*, donde se amparan todas las especificaciones para el correcto desarrollo haciendo uso de redes descentralizadas.

- **Blockchain Ledger.** EBSI cuenta con su propia red blockchain permisionada, a la cual solamente pueden acceder nodos y participantes que sean países miembros de la Unión Europea. De este modo, la red se asegura y facilita el cumplimiento de normativa como el Reglamento General de Protección de Datos, el cual exige que los datos no salgan de los límites del territorio europeo o países con normativas similares en lo referente a la protección de datos aceptados por la Comisión Europea, tal y como se define en su artículo número 45.[16] En ella se almacenan los datos de los usuarios correctamente anonimizados con el fin de poder comprobar que el contenido de las credenciales es correcto y que han sido emitidas por una entidad emisora con las capacidades para ello y verificadad por el modelo.

De este modo, la EBSI se encarga de ofrecer un entorno de desarrollo para aquellas soluciones que quieran operar dentro del marco europeo, cumpliendo su regulación para el tratamiento de los datos, y asegurándose de mantener la interoperabilidad entre todas las posibes soluciones y aplicaciones construidas sobre su ecosistema.

3.5. Blockchains e Infraestructuras

Para el despliegue del modelo será necesaria una red descentralizada. Una de las tecnologías más populares que hace uso de la descentralización son las tecnologías blockchain. Estas son inmutables, albergan la información en múltiples nodos, participantes que conforman la red, los cuales se comunican entre sí para mantener siempre la cadena de mayor longitud, y están diseñadas a prueba de *ataques bizantinos*, por lo que en el caso de que un nodo cambie su información, este será comunicado siempre por el resto de nodos, la mayoría, la cadena correcta. Así se evita la propagación de cadenas modificadas maliciosamente siempre y cuando no se tenga bajo control el *ataques bizantinos* de los nodos de la cadena.

Se destacan a continuación los tres estándares principales que una red blockchain podría tener.

- **Bitcoin.** Bitcoin nace como un protocolo para realizar transferencias monetarias dentro de la red del mismo nombre empleando la criptomoneda BTC. Su función principal es la de envíos y recibos de dicho activo digital por la red entre distintas cuentas, si bien últimamente los usos y la cantidad de información que se puede almacenar en una transacción han sido incrementados. En parte esto se debe a la actualización presentada en 2018 por Greg Maxwell, y finalmente implementada el 12 de junio de 2021. La intención inicial de esta actualización era incrementar el tamaño de los scripts a ejecutar en Bitcoin y de mejorar la privacidad de las transacciones que ejecutasen información más compleja, exponiendo únicamente la última instrucción del script en la cadena y no realizando una diferenciación entre estas operaciones y las transacciones habituales de BTC. Sin embargo, el espacio incrementado para el almacenamiento de información arbitraria en las operaciones OP del protocolo (OP_FALSE, OP_IF, OP_PUSH), permite la subida de información en dichos argumentos hasta un máximo de 400 KB. Esto no tardó en dar lugar a un protocolo que permitiera el almacenamiento de multimedia en las transacciones, dando lugar a Ordinals. Ordinals permite inscribir información en los satoshis (unidad inferior al Bitcoin en la cadena, del mismo modo que un céntimo es a un euro). Si bien estas unidades de información son fungibles, es posible diferenciarlas por su fecha de emisión, desde el primer satoshi *sat 0 = nvtdijuwxlp*, hasta el último que todavía está por minar *sat 2,099,999,997,689,999 = a*. De esta manera, es posible disponer wallets que sean capaces de gestionar el envío de satoshis diferenciando por su número de inscripción. De otro modo, al realizar un envío en BTC no se realizaría diferenciación entre ellos y podría enviarse por error un satoshi que contenga una inscripción.

  De este modo se logra un protocolo similar a los de los NFT de Ethereum, donde es posible guardar, de forma completamente descentralizada en la red, información arbitraria. Esta es igual de inmutable y descentralizada que la propia red de Bitcoin. Esto, si bien añade una capa de seguridad, también implica que absolutamente toda la información incluida en una inscripción mediante Ordinals está a la vista de cualquier entidad que quiera visualizarla a través del ledger de la red de Bitcoin.

  Sin embargo, el minado de estas transacciones, al poseer una mayor cantidad de información, actualmente tarde una cantidad de tiempo enorme, por lo que la escalabilidad del modelo no sería factible. Tampoco, por el momento, la usabilidad, pues el protocolo se encuentra todavía en fases muy iniciales y no existe suficiente investigación al respecto.

  Además, la red de Bitcoin sigue empleando a día de hoy *Proof-of-work*, protocolo de minado que consume cantidades ingentes de energía y que es bastante contaminante.

- **Ethereum.** El protocolo Ethereum nace con la idea de emplear la tecnología blockchain no solamente para la realización de transacciones de divisas, sino también la ejecución de manera descentralizada de

aplicaciones (*DApps*). Para ello emplea la tecnología de la Ethereum Virtual Machine, máquina virtual que permite la ejecución de Smart Contracts en los diferentes nodos de la red. Estos Smart Contracts contienen código arbitrario, por lo que es posible crear diversos tipos de tokens o aplicaciones con ellos.

Así nacieron los estándares ERC-721, el cual sirve para el desarrollo de NFTs, y ERC-4973, para SBTs. Dichos estándares son implementables en cualquier cadena que tenga soporte para la *EVM*, no solo la Mainnet. Esto, como veremos, abrirá el abanico de posibilidades, al poder emplear redes de Layer 2 como Polygon o redes alternativas como Gnosis, donde los POAP (Proof-Of-Attendance-Protocol) son almacenados. La programación haciendo uso del estándar Ethereum, sin embargo, no se limita únicamente a estos estándares a la hora de desarrollar un modelo de Identidad Descentralizada. Dado que la programación de Smart Contracts es Turing-completa, se pueden implementar también otros modelos, como el de la W3C, por ejemplo.

Otro estándar clave del protocolo de Ethereum, y también transportable a otras cadenas, es el ERC-4337. Este estándar permite llevar Externally Owned Accounts (EOA), las cuales son las cuentas usadas habitualmente por los usuarios, gestionadas por Metamask u otras carteras digitales similares, a ser gestionadas por Smart Contracts. De este modo se logra un nuevo nivel de personalización de cuentas, pudiendo implementarse algoritmos post-cuánticos para su protección, algoritmos criptográficos ZK-Proof, recuperación de la clave privada asociada a la cuenta, multifirma, firma de varias transacciones a la vez en una sola operación o envío de los activos digitales a otra dirección en caso de pérdida de la cuenta original, entre otros.

Todas estas nuevas características hacen el manejo de las cuentas autocustodiadas mucho más usable cara al público no técnico. De este modo, la adopción del modelo a nivel global es mucho más posible.

En cuanto a la escalabilidad, es cierto que Ethereum, si bien es capaz de procesar más transacciones por minuto que Bitcoin, sigue sin ser un número suficiente para una adopción y uso masivo. Sin embargo, es posible hacer uso de redes de Layer 2 o incluso la creación de redes privadas o de consorcio que implementen la EVM.

- **Hyperledger Indy.** Hyperledger engloba toda una comunidad de desarrolladores dedicados a la elaboración de frameworks, herramientas y librerías como soluciones blockchain. Está especialmente orientada a empresas privadas, ofreciendo como servicio la trazabilidad de accesos y verificación de los usuarios mediante firmas digitales, así como la eliminación de actores intermedios que puedan convertirse en puntos únicos de fallos u objetivos de ataques cibernéticos. Aun así, esto no implica que su uso sea exclusivo para empresas. Pero, por ello que, al no tenerse tanto como objetivo la privacidad e inmutabilidad como tal, sino el control dentro de la propia empresa, sus soluciones son redes privadas y de consorcio. [17]

  Esto conlleva una pérdida en la privacidad de los datos, así como en la inmutabilidad y seguridad de los mismos, pues existe una o varias entidades que gobiernan sobre el funcionamiento de la red y que pueden alterar el contenido de la misma, así como tener control sobre todo tipo de participantes del modelo. Una opción por consorcio es, desde luego, más privada, al mantener cierto nivel de descentralización al tener el control varias entidades.

  Dentro de las soluciones de Hyperledger encontramos **Hyperledger Indy**, orientado al desarrollo de Identidad Descentralizada. Esta tecnología descentralizada está modelada de acuerdo al estándar de la W3C; las Credenciales Verificables, modelo integrado en la European Blockchain Service Infrastructure, blockchain utilizada por la Unión Europea y que unifica el modelo de la Identidad Digital en ella. Una diferencia con respecto al modelo puro, es que en esta red es posible implementar identificadores que emparejen entidades que componen el modelo con el fin de relacionarlas uno a uno. De este modo es posible tener una mejor visualización de cómo interaccionan los componentes del ecosistema entre sí.

Cabe destacar que el uso de una red pública de las anteriores mencionadas poseerá mayor tráfico de red, y por lo tanto congestión y mayor lentitud a la hora de integrar la información en un bloque. Esto es debido a

que las transacciones propias del modelo deberían convivir con el resto de transacciones que se pudieran llevar a cabo simultáneamente en la red, es decir, transferencia de otro tipo de tokens fungibles y no fungibles, así como mintings que se puedan estar llevando a cabo en ese momento, conversiones entre ERC-20s en cadenas basadas en *EVM*, *burnings*, *staking* de tokens... Esto puede, además, dar lugar a picos impredecibles en las tarifas de gas a pagar, por ejemplo durante el minteo de una colección de 10k NFTs muy demandada, donde el valor de *Gwei* puede alcanzar valores atípicos y esto pueda conllevar el fallo de transacciones enviadas con valores inferiores.

4. Comparativa

Como se puede ver, existen diversos protocolos para cada una de las capas definidas. Esto a día de hoy supone una dificultad a la hora de definir un estándar global, pues pueden darse ciertas incompatibilidades entre modelos. Es por ello que varios grupos encargados del diseño de diferentes tecnologías buscan la colaboración entre sí o facilitar la implementación de diferentes modelos. Por esto, la flexibilidad a la hora de establecer comunicación entre entidades descentralizadas y DIDs es fundamental.

Sin embargo, es cierto que algunas de estas iniciativas poseen claras ventajas sobre las otras y que, por lo tanto, pueden destacar en ciertos entornos donde se exige el cumplimiento de ciertas condiciones, legislación o regulaciones. Por ejemplo, en el territorio europeo es obligatorio el cumplimiento del Reglamento General de Protección de Datos (RGPD o GDPR). Esta regulación afecta a todos los DIDs que almacenen datos personales, así como a diversas entidades del ecosistema descentralizado que los traten. En este caso, debido a que el RGPD fue diseñado teniendo en mente los sistemas centralizados donde es posible identificar tanto a controladores de datos como a procesadores de datos, han surgido ciertas problemáticas para la correcta implementación de DIDs como tecnología para tratamiento de datos en el territorio europeo. Aun así, como se mencionó antes, algunas tecnologías existentes en cada capa del modelo se adaptan mejor a estas restricciones o facilitan su implementación.

4.1. Modelos de Credenciales Verificables y SoulBound Tokens

Estos modelos son definidos para la representación de DIDs, si bien poseen características muy diferentes entre sí. Esto es debido a que ambos esquemas nacen en ecosistemas diferentes.

Por un lado, las Credenciales Verificables son diseñadas bajo el cumplimiento de ciertos estándares definidos, tales como el framework JOSE para la encriptación de la información o el uso de Java Web Tokens para su estructuración y envío. Sin embargo, los SBTs nacen dentro del ecosistema de la blockchain de Ethereum inicialmente como un Ethereum Improvement Proposal (EIP) que tras su aceptación en la comunidad se convirtió en un Ethereum Request for Comments definido, el 5484 [18]. Es por ello que, obviamente, este estándar está estrechamente ligado al protocolo Ethereum.

4.1.1. Descentralización de la información

Los datos representados en estos modelos son almacenados de maneras distintas cuando se realiza de forma implícita en los mismos. Una Credencial Verificable almacena los datos en formato *JSON* y *JWT*, y estos son enviados dentro del canal de comunicación especificado, pero su almacenamiento no tiene por qué ser completamente descentralizado. Es cierto que las claves criptográficas empleadas para la verificación de las entidades emisoras pueden hacer uso de *Descentralized Ledger Technologies* para la correcta comprobación de de las transacciones por parte de diferentes entidades verificadoras y proveedores de servicios miembros del ecosistema, pero los datos pueden ser almacenados de manera ajena a las tecnologías blockchain, pudiendo hacerlo de forma local, en redes *peer-to-peer* o simplemente en diversos servidores a modo de copias de seguridad.

Así pues, la descentralización fundamental en estos modelos implica la descentralización de las claves públicas de los *Issuers*, a disponibilidad de quien quiera comprobar un DID en formato de Credencial Verificable, el cual contiene una firma digital de dicho Issuer. De este modo se eliminan las autoridades centrales de registro o CA para la comprobación de emisiones de credenciales, tal y como se define en las características necesarias que un DID ha de poseer.

De forma contraria, un SoulBound Token consiste en un Smart Contract bajo el estándar ERC-5484, es decir, un contrato inteligente almacenado y ejecutado en blockchains con que funcionan mediante el uso de la Ethereum Virtual Machine. Esto implica que los SoulBound Token solo pueden ser modelados y ejecutados

en redes blockchain bajo el protocolo de Ethereum, sea la red principal u otras derivadas, sea cual sea la capa (*layer*) o nivel de permisionado. También conlleva que la información registrada en los atributos de un SoulBound Token queda inscrita dentro de la red de Ethereum en el bloque donde se lleva a cabo la transacción al momento de crearlo. Por ello, los datos incluidos en un DID representado bajo esta estructura quedan tan descentralizados como la red de Ethereum donde se haya subido la información.

Como puede verse, la descentralización de la información en el caso de los SoulBound Tokens es obligatoria en su forma más pura, mientras que para las Credenciales Verificables esto no tiene por qué ser así.

4.1.2. Reglamento General de Protección de Datos

Del punto anterior, se destaca que cabe la posibilidad de en lugar de almacenar el dato como tal dentro del SoulBound Token, es posible introducir un cifrado del mismo y almacenar la información a la que este refiere en un sistema diferente, más centralizado. También es aplicable a DIDs representados como Credenciales Verificables cuya información es almacenada en tecnologías blockchain.

Esta opción surge de la necesidad de permitir un borrado o modificación de los datos una vez han sido inscritos en una blockchain, pues cualquier información subida a la misma se convierte en inmutable.

La inmutabilidad de la blockchain se debe a que dentro de la información almacenada en cada bloque se refiere a aquella en el bloque anterior, creando así una cadena donde para alterar la información albergada en un bloque concreto, habría que alterar la referencia del subsiguiente y así la de todos al variar la información contenida en todos ellos. Es por ello que a estas blockchains se las denomina probabilísticas, pues a un mayor número de bloques producidos tras el que contiene la transacción con la información dada, más bloques de datos se deberían alterar para preservar la lógica de la cadena.

Esto se logra mediante el uso de ***hashes***. Un cifrado *hash* consiste en un algoritmo criptográfico al que dada una entrada de datos de una longitud arbitraria, obtenemos un cifrado de una longitud fija, el *hash*. Este resultado varía completamente aunque la diferencia entre las entradas sea de solamente un bit. Además, la obtención del dato original a partir del *hash* obtenido es imposible, por lo que se dice que es un cifrado unidireccional. Esta es la información contenida en cada bloque, un resumen *hash* de los datos albergados en el bloque inmediatamente anterior n-1 a este, lo cual incluye también el resumen *hash* del bloque n-2, y así sucesivamente. Es por ello que la alteración de cualquier información en el interior de un bloque requeriría la modificación de todos los subsiguientes con el fin de mantener la consistencia de la cadena.

Debido a esto, ejercer el derecho a la rectificación o el derecho a supresión de los datos contemplados en el Reglamento General de Protección de Datos (artículos 16 y 17 respectivamente) [19] se vuelve una tarea, de primeras, técnicamente imposible de llevar a cabo en tecnologías blockchain, pues supondría la modificación de todo el sistema de almacenamiento, lo cual no es posible por el propio funcionamiento de la blockchain.

Es por ello que aquellas credenciales cuya información se almacene directamente *on-chain* no cumplen esta regulación por su diseño. Para solventar este problema, una de las propuestas mayormente aceptadas es el cifrado *hash* de esta información previamente a ser subida a la cadena de bloques.[20]

Sin embargo, no cualquier técnica es considerada como adecuada para la anonimización de datos por el Reglamento General de Protección de Datos, pues ha de cumplir con ciertas características para ello.

- **No ha de ser posible obtener el dato original únicamente a partir del dato cifrado.** Esta característica es intrínseca al cifrado *hash*, el cual es unidireccional y no es posible por lo tanto obtener el mensaje original a partir del resumen *hash*.
- **No ha de ser posible obtener el dato cifrado únicamente a partir del dato original.** Aquí sin embargo nos encontramos con el problema de que para obtener un resumen *hash* tan solo hace falta el dato original, al cual se aplica el algoritmo concreto, el cual es público, sin ningún elemento más.

De este modo, se añade a este cifrado *hash* el uso de una clave, la cual influye en el proceso de cifrado, haciendo que el resultado obtenido aplicándola sea diferente del que se consigue sin ella. Con esto obtenemos un resultado *hash* que no es posible obtener únicamente con el dato original, pues haría falta tener en posesión también la clave empleada.

De esta manera, además de lograr la correcta anonimización exigida, en el caso de querer eliminar el dato de la cadena, se puede realizar de forma equivalente un borrado de la clave. Consecuentemente, obtener la información almacenada en la cadena a partir del dato original sería imposible, y por lo tanto contrastar esa cadena de caracteres con cualquier valor no sería factible, quedando como una secuencia arbitraria escrita en la blockchain de la cual no es posible extraer ninguna información. En caso de modificación, se crearía un nuevo token bajo el mismo protocolo tras realizar este borrado, con su nueva clave y su nuevo resumen *hash*.

De este modo, aunque se haga uso de una estructura para los DIDs completamente descentralizada, cabe la posibilidad de seguir cumpliendo con el Reglamento General de Protección de Datos, aunque para ello haya que almacenar una clave empleando alguna otra metodología diferente a la blockchain.

4.1.3. Revocación de la información por parte de distintas entidades

Una de las características que un DID debe poseer es la posibilidad de ser revocado. Es especificado en su definición que, además, esto sea comprobable a partir de la información contenida en el mismo.

De nuevo, nos encontramos con cierta discrepancia entre las Credenciales Verificables y los SoulBound Tokens. Las primeras solamente pueden ser revocadas por la entidad que las haya emitido. Con ello se mantiene cierto nivel de centralización y dependencia de los *Issuers* que no termina de otorgar un control completo al usuario sobre sus propios datos. De todos modos, para que ocurra una revocación esta debiera estar justificada o en caso contrario la entidad emisora podría perder su derecho a la emisión de credenciales en el modelo al cometer una infracción realizando un borrado sin una acreditación adecuada.

Por otro lado, los SoulBound Tokens permiten establecer quién poseerá el control de la revocación en el momento de la emisión del mismo gracias al valor BurnAuth, comprobable además mediante la función *burnAuth(uint256 tokenId)*, la cual devuelve dicho valor establecido dado un identificador de token SoulBound Token. Este valor BurnAuth puede ser uno de los siguientes:

- **IssuerOnly.** El token tan solo podrá ser revocado por la entidad emisora que lo haya generado y enviado al usuario.
- **OwnerOnly.** El token tan solo podrá ser revocado por el usuario propietario de dicha información.
- **Both.** El token podrá ser revocado cuando ambas entidades, emisora y usuario, acepten que el token pueda ser quemado.
- **Neither.** El token no podrá ser revocado bajo ninguna circunstancia, quedando fijado a la billetera digital del usuario para siempre.

La revocación en este caso no consiste tanto en un cambio de estado, sino en que el token pasa de estar asociado a la cartera digital del usuario y es enviado a la NullAddress mediante el proceso de *burning*. Esto se hace mediante una transferencia, por lo que es visible para cualquier participante de la cadena y por lo tanto del modelo. Se puede realizar la comprobación mediante el uso de emisión de eventos o simplemente de manera manual, viendo cómo el token abandonó la *address* para la que fue emitido originalmente y fue enviado a la NullAddress. [18]

4.2. Protocolo de comunicación. OpenID4VC y DIDComm

En cuanto a protocolos para la comunicación de DIDs, debido a la existencia de diversos modelos para la representación de los mismos, siendo las Credenciales Verificables de la W3C o los SoulBound Tokens solamente dos de los muchísimos existentes, se produce la necesidad de que las comunicaciones de los mismos sean flexibles y admitan una gran diversidad de esquemas. Solamente en modelos para Credenciales Verificables podemos encontrar numerosos formatos, como pudo verse en el apartado para los Modelos de Datos.3.3

Ambos protocolos cubiertos aquí, OpenID4VC y DIDComm, poseen la característica de ser flexibles a la hora de transmitir los datos, no cerrándose a un modelo o estructura específico para Credenciales Verificables. Esto facilita tanto la creación de carteras digitales o *wallets*, pues no se han de adaptar a un único modelo o encargarse de transformar los datos cuando los reciben para mantener una uniformidad acorde al protocolo de comunicación; y también facilita la integración de nuevas entidades emisoras, las cuales pueden emplear el modelo que deseen sin preocuparse de que el resto de componentes del modelo no lo acepten o deban crear algún tipo de conversor previamente. Esto hace la entrada de entidades emisoras al modelo descentralizado más amigable al no tener que adaptar su modelo de datos previo a un esquema específico.

Sin embargo, las posibilidades que cada protocolo ofrece son diferentes. OpenID4VC está enfocado en ofrecer las comunicaciones necesarias para las interacciones básicas de un modelo de identidad descentralizada. DIDComm, por otro lado, se crea como un protocolo de comunicación de mensajes descentralizado y seguro, pero no se cierra únicamente a casos de uso de identidad descentralizada, sino que admite también otro tipo de comunicaciones basadas en modelos descentralizados.

4.2.1. Casos de uso admitidos

En OpenID4VC vimos que los casos de uso para los que ha sido diseñado son para la emisión y presentación de credenciales, y para el manejo de pseudónimos a la hora de que el usuario se comunique con una entidad.

Esta orientación cierra su uso a otras posibles comunicaciones descentralizadas, centrándose en el modelo de tres entidades, *Issuer*, *Verifier/Service Provider* y *Holder/User* (poseedor de la *Wallet*). Por un lado, su simplicidad facilita su implementación y también hace más sencillo la formación en ello. Por otro lado, la interoperabilidad mediante este protocolo queda reducida a casos de uso relacionados con el modelo de Identidad Digital.

Por otro lado, DIDComm permite la construcción de subprotocolos sobre su protocolo principal. Esta posibilidad admite la creación de comunicaciones más diversas, como mensajería, coordinación de pagos o interacciones más diversas con tecnologías IOT (*Internet Of Things*). Esto amplía las capacidades del protocolo pero a su vez hace más complejo su ecosistema. Esto no quita que las operaciones básicas incluidas en OpenID4VC permanezcan de modo similar y pueda prescindirse del resto de posibilidades para un caso de uso cerrado a identidad.

4.2.2. Admisión en diferentes comunidades

Ambos protocolos han sido apoyados por diferentes comunidades u organizaciones. DIDComm comenzó como un proyecto íntimamente ligado a Hyperledger y a Aries. A día de hoy forma parte de DIF Working Group. Es por ello que es apoyado por sus respectivas comunidades open-source y una gran mayoría de desarrolladores lo prefiere debido a ello.

Por otro lado, OpenID4VC ha sido seleccionado como protocolo online por *The European Digital Identity Wallet Architecture and Reference Framework* [12] para desarrollar su sistema de identidad descentralizada y cartera digital europea. Es claro que en este modelo priman las interacciones entre identidades descentralizadas con ese único fin, sin necesidad de añadir otro tipo de casos de uso. De este modo se mantiene la

sencillez del protocolo y la solución directa al problema.

4.3. Tecnología Blockchain a utilizar

De entre las principales tecnologías blockchain, Hyperledger y Ethereum son las preferidas para desarrollar casos de uso de identidad digital, la primera por estar especializada en ello y por tener un uso orientado a la empresa; la segunda por la flexibilidad de su protocolo, pudiendo programar en múltiples capas o *layers*, así como con distintos niveles de permisionado.

Por otro lado, Bitcoin queda actualmente descartada. Esto no significa que en un futuro pudiera llegar a considerarse debido a los cambios que están aconteciendo recientemente en su red y filosofía.

4.3.1. Descarte de Bitcoin por el momento

Bitcoin, como se ha mencionado previamente, se creó con el fin de procesar transacciones para pagos, relacionados directamente con casos de uso económicos y de reserva de valor. Es por ello que transacciones con una mayor cantidad de datos, como ha ocurrido con los Ordinals, han producido una congestión en la red que hace inviable la transferencia inmediata de activos. Es por ello que a día de hoy la construcción de un modelo complejo para el intercambio de información en Bitcoin no es viable.

Por poner un ejemplo, a inicios de mayo con la creciente popularidad de los tokens BRC-20, tokens fungibles para monedas digitales desarrollados sobre Bitcoin de modo análogo a los tokens ERC-20 en Ethereum; la *mempool* de Bitcoin tenía en torno a 425000 transacciones sin confirmar y una *tasa de sat* de en torno a 250 - 1000, cuando habitualmente esta era de 12 o 15 para alta prioridad.

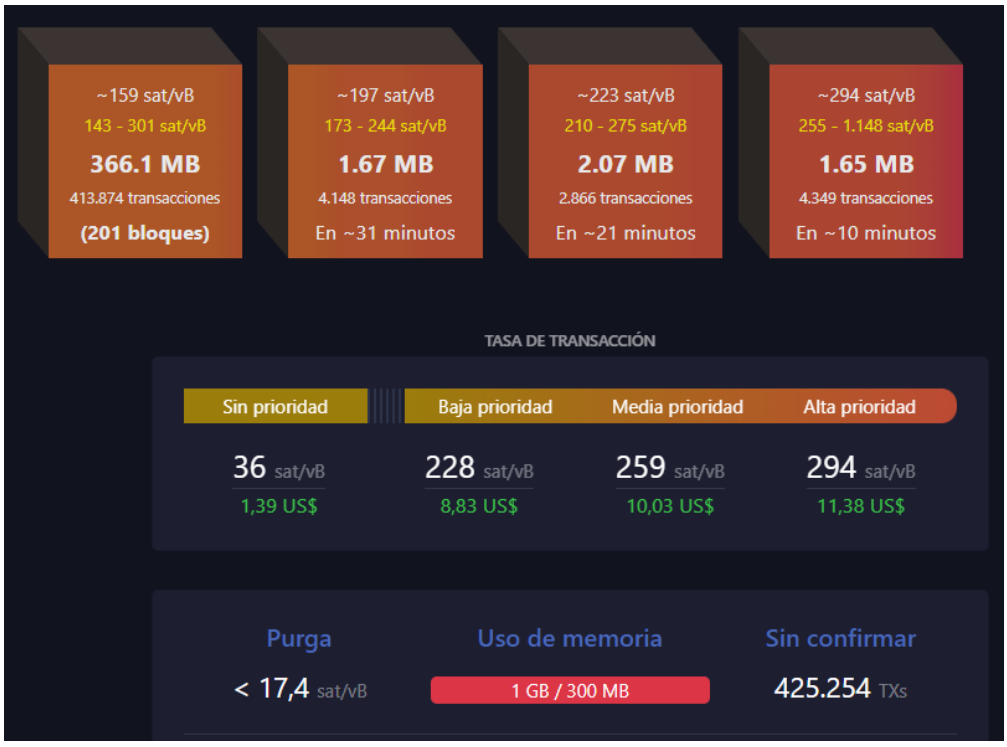

Figura 2: Mempool 9 de mayo de 2023

Cabe destacar sin embargo, que recientemente se está realizando el desarrollo e implementación sólida de una Bitcoin Virtual Machine. Esta funcionaría a modo similar de la Ethereum Virtual Machine y permitiría un nivel de abstracción mayor, así como la construcción de tokens, artefactos digitales o entidades para casos de uso más complejos, sin tener el impacto actual en la congestión de la red. Está por ver cómo evoluciona y si se solventa el problema de la congestión de la red al incrementarse la cantidad de datos enviados con cada transacción en la misma.

4.3.2. Redes Permisionadas

Dentro de las tecnologías blockchain contamos con tres tipos de las mismas, pudiendo crearse redes de los tres tipos bajo el protocolo de Ethereum y estando Hyperledger más enfocada a redes de consorcio y privadas.

- **Públicas.** En una red pública, si bien la descentralización puede llegar a ser mucho mayor debido a la mayor cantidad de nodos que esta pueda tener, así como contar con una accesibilidad mayor, nos encontramos con que toda la información se mantiene en un sistema público. Esto podría solventarse en parte mediante el uso de un modelo en parte centralizado, con el almacenamiento de claves en servidores centrales y la subida de los datos cifrados de algún modo para mantener el anonimato de los usuarios. Sin embargo, requerir el uso de un sistema centralizado aquí se contradice con la propia decisión de diseño de usar una red más descentralizada.
- **Privadas.** Las redes privadas, a diferencia de las públicas, permiten un mayor control en el almacenamiento de los datos, así como en su procesamiento. Este tipo de redes están gobernadas por una organización, la cual de algún modo admite al resto de participantes y gobierna sobre la información que se almacena en la misma. La inmutabilidad y seguridad, así como la privacidad con respecto a dicha organización, son menores que en una red pública completamente descentralizada. Además, al limitar el acceso a las mismas, estas cuentan con un número de nodos muchísimo menor al de las redes públicas. Esto, por otro lado, facilita los ataques del 51 % a la red, así como la posibilidad de un dominio o control por parte de algunos nodos sobre el funcionamiento de la misma, dando lugar al efecto Matthew.
- **Consorcio.** Similares a las redes privadas, solo que en lugar de contar con una única organización con control sobre el resto de participantes y sobre la información, son varias organizaciones las que poseen ese control. Esto implica un consenso mínimamente descentralizado antes de tomar decisiones que puedan afectar al funcionamiento de la cadena, información o sus participantes. Si bien no son tan descentralizadas como una red pública, al menos el control de la red no recae únicamente en una única organización. Un ejemplo interesante es la Enterprise Ethereum Alliance, la cual, haciendo uso de la Ethereum Virtual Machine busca proporcionar al ecosistema una solución blockchain para empresas privadas. Otra red similar es Hyperledger Indy, enfocada, también a la empresa privada.

Una de las problemáticas que surgen en el territorio europeo con las redes públicas es el hecho de que los datos queden expuestos a cualquier individuo que simplemente compruebe el histórico de la blockchain. Esto colisiona con la regulación del Reglamento General de Protección de Datos, pues cualquier dato subido a la blockchain debe estar restringido para su visualización a únicamente aquellas entidades a las que el usuario haya dado permiso para ello. Otro problema surge de que los validadores de la red (mineros, stakers...) puedan situarse en cualquier lugar del mundo. Esto conlleva a que, dado que los validadores son considerados procesadores de datos por el Reglamento General de Protección de Datos, estos podrían situarse fácilmente fuera del territorio eurpeo, colisionando de nuevo con el Reglamento General de Protección de Datos.

Es por ello que el uso de redes privadas o de consorcio, son una mejor aproximación para los casos de uso de Identidad Descentralizada. Tanto Ethereum como Hyperledger cuentan con opciones de dichos niveles de permisionado.

4.3.3. Convivencia con otro tipo de Artefactos Digitales

Una diferencia fundamental entre Ethereum e Hyperledger es que la primera no está únicamente cerrada a los casos de uso de Identidad Descentralizada. Esto permite la convivencia de DIDs en un ecosistema más rico, con otro tipo de tokens y artefactos digitales en la misma cadena. Hyperledger por el contrario no cuenta con tal posibilidad.

Por un lado esta convivencia facilita la gestión de todos los activos digitales bajo el uso de una misma cartera digital, aunque por otro lado puede hacer más difícil una adopción masiva de la tecnología al ser algo más complejo de cara al usuario, el cual deberá aprender a diferenciar cada tipo de activo de algún modo para tener conocimiento sobre qué tipo de token está empleando en qué momento. No es lo mismo un activo transferible que uno que no lo es, y a día de hoy las firmas de transacciones on-chain no son del todo amigables para un usuario promedio que pudiera confundir un token con otro y transferir para siempre algo que no debiera por desconocimiento. Bajo este punto de vista Hyperledger es más amigable.

Cabe destacar que además se están desarrollando ya opciones de interconectividad entre blockchains, como LayerZero, Cosmos o Hyperledger Cactus. Todas estas tecnologías permiten la transferencia de activos entre cadenas de blockchains diferentes, incluso bajo diferentes protocolos. Si bien es una tecnología muy nueva, en caso de que una implementación haga uso de Hyperledger no cierra del todo la puerta a una convivencia con otro tipo de activos mediante *bridges* haciendo uso de las mencionadas tecnologías. Estas además funcionan de modo descentralizado, sin hacer uso de una blockchain o servicio centralizado intermediario que facilite la conexión.

5. Demostrador

Para demostrar el funcionamiento del modelo se procede a realizar una demostración de cómo funciona el mismo. Para ello se crea un entorno ficticio en el cual tendremos los siguientes elementos:

- Una entidad académica emisora de certificados. Esta ha sido previamente registrada para poder comprobar sus firmas digitales, y posee la capacidad de proporcionar Credenciales Verificables a los usuarios que las soliciten, firmadas por ella.

  Además, también ejerce el rol de entidad verificadora, pudiendo solicitar y recibir Presentaciones Verificables llevando a cabo el proceso de modo independiente de su rol como emisora.

- Un usuario que solicitará la creación de la credencial a la entidad académica. Además, podrá realizar comprobaciones sobre el aquellas Credenciales Verificables que posee y generar Presentaciones Verificables para proporcionárselas a las entidades verificadoras que se las soliciten.

- Una credencial que contenga datos de ámbito académico propios de una titulación obtenida por la entidad emisora, y referente al usuario del ecosistema. Esta credencial será emitida y firmada por la entidad académica como una Credencial Verificable, la cual el usuario almacenará y de la cual podrá generar Presentaciones Verificables para proporcionar de forma completa o parcial los datos contenidos en la misma a los Proveedores de Servicios que lo requieran con el fin de identificarse ante ellos.

Para llevar a cabo esta demostración se hará uso del framework de Aries.3.4.1 A lo largo de la demostración, se realizarán las siguientes acciones principales.

1. Se establece el canal de comunicación entre el usuario y la entidad académica emisora. Para ello, ambos actores deben abrir dicho canal y confirmar la comunicación con el otro, enviando una invitación de conexión desde una gente a otro, y el receptor aceptándola para poder iniciarla. Esto es posible debido al uso de DIDComm, protocolo de transmisión de información no cerrado al uso exclusivo de envío y recepción de Credenciales Verificables. Este canal permite a los actores enviarse mensajes y comunicarse de manera segura mediante dicho protocolo.

2. Registro por parte del agente emisor de credenciales en el ledger al uso, el cual es Hyperledger Indy, para que los proveedores de servicios puedan comprobar su identidad como válida en el momento de recepción de una credencial emitida por él. De este modo, la veracidad, confianza y autenticación de la misma, quedan cubiertas.

3. Tras ello, la entidad emisora podrá iniciar el proceso de creación de una credencial verificable. Para ello, especificará los atributos que el certificado ha de contener, y esta será emitida.

4. Una vez la credencial es generada, es enviada al usuario. Este la recibe y la almacena en su cartera digital, desde donde tendrá acceso a la misma a partir de ese momento y hasta su fecha de expiración, de tenerla.

5. La entidad emisora, además, recibirá a la vez un comprobante que muestra cómo, efectivamente, el usuario ha recibido la credencial de manera correcta.

6. Tras ello, la entidad emisora pasará a realizar el rol de una entidad verificadora y enviará una solicitud de envío de credencial al usuario. La razón o motivo que justifique el requerimiento del certificado irá incluido en dicha petición.

7. El usuario, tras recibir la notificación, enviará al verificador la credencial que corresponda con la petición recibida.

8. Finalmente, el verificador recibe la credencial y procede a comprobar que, efectivamente, la firma que verifica la autenticidad de la credencial fue registrada y es válida en el ledger de Hyperledger Indy.

De este modo, la demostración abarca el establecimiento de la comunicación de manera segura, el registro de la entidad emisora en la blockchain, la creación de una credencial verificable y su correspondiente envío al usuario y almacenamiento en su cartera digital, y la presentación y validación de la misma en el momento de presentarla ante una entidad que así lo requiera.

5.1. Hyperledger Aries Cloud Agent Python

Realizaremos ahora todo lo anterior paso por paso haciendo uso de Hyperledger Aries Cloud Agent Python (ACA-Py).[21] Este consiste en una base para la creación de Credenciales Verificables y ecosistemas basados en las mismas. Para ello, hace uso tanto de Hyperledger Aries como framework para la creación de las credenciales, como de DIDComm como protocolo de comunicación entre los diferentes actores del ecosistema. ACA-Py provee todo esto de manera ya implementada para que los usuarios del mismo no tengan que crear desde cero e interconectar entre sí cada capa de las definidas por Trust Over IP.

La peculiaridad de ACA-Py reside en que este es ejecutado en un servidor, a diferencia de otros protocolos que necesitan de dicha integración mediante una cartera digital en un teléfono móvil. De esta manera, ACA-Py permite la integración y desarrollo de ecosistemas basados en Credenciales Verificables cuyo funcionamiento sea en entornos de ejecución no móviles. Un ejemplo de esto puede ser en el entorno empresarial, donde las carteras digitales estén integradas en la propia infraestructura privada de la misma.

Dentro de las herramientas provistas por ACA-Py podemos encontrar Aries OpenAPI Demo.[22] Este demostrador nos permitirá ejecutar paso a paso el procedimiento descrito anteriormente desde el punto de vista de un controlador, el cual interactúa con diferentes *API endpoints* donde se exponen instancias de ACA-Py. Así, podremos ver el funcionamiento de todo este ecosistema, pudiendo ver cada respuesta y mensaje que se envíen los diferentes actores, así como cada elemento que forme parte del ecosistema.

Para llevar a cabo la demostración se emplearán dos agentes y una Credencial Verificable.

- **Faber.** Faber es el nombre del agente que realizará primero el rol de entidad emisora (*Issuer*), y tras generar una Credencial Verificable pasará a hacer el rol de entidad verificadora (*Verifier o Service Provider*). Como *Issuer* se registrará en el ledger de Indy, generará una Credencial Verificable académica y se la enviará a un usuario, Alice. Como *Verifier*, solicitará el envío de una Presentación Verificable de carácter académico y, tras recibirla, comprobará en el ledger de Indy la validez de la misma.
- **Alice.** Alice es el nombre del agente que realizará el rol de usuario (*Holder*). Se le hará entrega de una Credencial Verificable académica desde Faber. Tras ello, se le requerirá la presentación de la misma y Alice la enviará desde su cartera digital, ejecutada en el servidor de ACA-Py.
- **Certificado Académico**. El certificado académico será representado como una Credencial Verificable en el momento de su emisión y envío desde Faber hasta Alice. Se almacenará ahí en su cartera digital y en el momento de que una entidad verificadora lo requiera, será enviado como Presentación Verificable. Contendrá una prueba de que ha sido generado por el *Issuer* Faber, haciendo posible la autenticación y validación de su contenido por parte de las entidades proveedoras de servicios en el ecosistema.

Todas las transacciones quedarán registradas en el ledger público de Indy, pues es necesario una blockchain donde se almacenen las pruebas de validación y autenticación para el correcto funcionamiento del modelo de Credenciales Verificables aquí empleado.

5.2. Demostración

Para realizar la demostración vamos a hacer uso de Docker, lo cual es una plataforma para el despliegue de contenedores, es decir, archivos que contienen las dependencias y configuraciones necesarias para ejecutar una aplicación de manera aislada. Para facilitar el proceso y hacerlo de una manera más accesible sin necesidad de instalar ningún software, vamos a emplear Play with Docker. Esta página nos permite ejecutar Docker en nuestro navegador, permitiéndonos crear una sesión con duración de 4 horas. Será necesario registrarse para comenzar a crear instancias de docker donde podamos ejecutar y probar las diferentes acciones como si las estuviésemos ejecutando en un entorno completo.

Para ello se utilizará este ledger público operado por el *BC Government's VON Team*, un equipo del gobierno de la provincia de Columbia Británica (Canadá) que se dedica a la implementación de *Verifiable Organizations Network*, o red de organizaciones verificables, que mediante el uso de tecnología blockchain permite la creación y emisión de Credenciales Verificables. En este caso, la blockchain subyacente será la de Hyperledger Indy.

5.2.1. Creando las Instancias de Docker

Para poder comenzar a trabajar con la demo que aporta ACA-py, vamos a iniciar una sesión en Play with Docker. Dentro de la misma crearemos dos instancias de manera muy similar. La primera será para iniciar la ejecución del agente de Faber, y la segunda para el mismo fin con el agente de Alice.

Para crear una nueva instancia en la sesión clickamos el botón "*Add New Instance*". Esto nos mostrará en la página una pantalla de comandos, así como una serie de parámetros que nos podrán de ser uso durante la ejecución de comandos en la misma, como direcciones *IP* mediante las que podamos acceder a algún despliegue realizado desde nuestros contenedores de Docker.

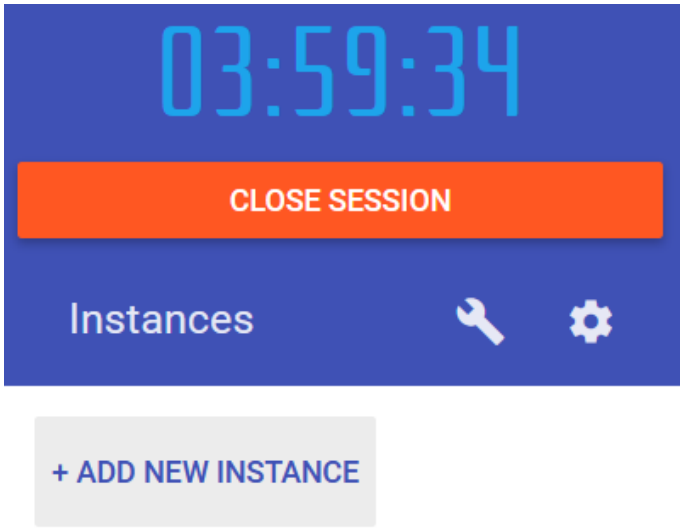

Figura 3: Añadiendo una nueva instancia en Play with Docker

Una vez nos aparece la pantalla de comandos, lo primero que vamos a hacer es clonar el *repositorio* de *github* mediante el comando `git clone` y especificando la URL del repositorio.

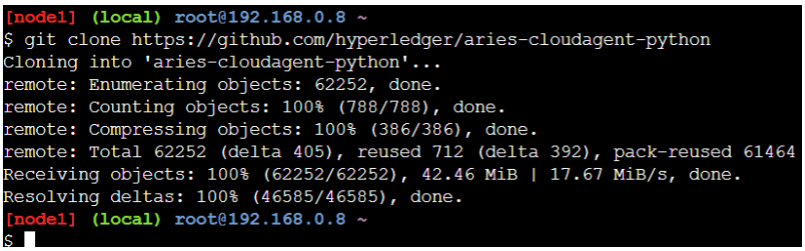

Figura 4: Comando para clonar el repositorio

Una vez el repositorio es clonado en la carpeta de mismo nombre que el proyecto, nos cambiamos de directorio a este y desde ahí ejecutamos el comando
`LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo faber --events --no-auto --bg`

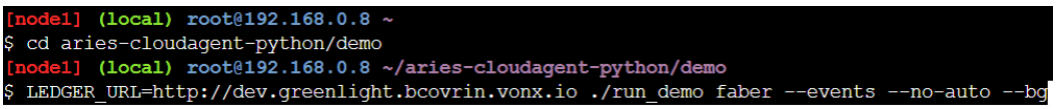

Figura 5: Ejecutando el agente de Faber

Este comando especifica cómo levantar el agente de Faber.

- `LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io` - La variable`LEDGER_URL` informa al agente qué ledger debe utilizar, en este caso el especificado anteriormente de Hyperledger Indy en *BC Government's VON Team*.
- `./run_demo faber` - Mediante esta especificación ejecutamos el agente de Faber.
- `--events` - Esta opción indica que queremos que el controlador muestre en el registro de la terminal los eventos webhook, es decir, aquellos que avisan de cambios producidos en la aplicación, en este caso por ACA-Py.
- `--no-auto` Esta opción indica a ACA-Py que ciertos eventos, como el establecimiento de la conexión, no se realicen de manera automática, pues vamos a realizarlo nosotros de manera manual en el rol del controlador para comprender el procedimiento del protocolo.
- `--bg` Con ello indicamos que el contenedor de Docker se ejecute en segundo plano, para así, en caso de pulsar accidentalmente Çtrl + C", el proceso no se detenga.

Una vez tenemos el agente ejecutándose, podemos consultar la dirección IP mediante la cual nos podemos conectar a Faber y ejecutar sus diversos métodos. Esta dirección es otorgada en el campo `DOCKERHOST` en el momento en el que el comando anterior termina de ejecutarse.

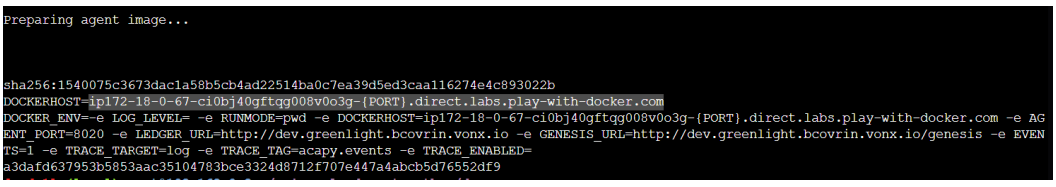

Figura 6: Dirección IP de Faber en el campo DOCKERHOST

Sin embargo, el puerto no viene especificado, sino que aparece como un campo *PORT* a ser sustituido. Para poder visualizarlo, podemos ejecutar `docker logs -f faber` para visualizar mediante docker los registros que se van produciendo en la instancia de Faber. Además, el seguimiento se hace en tiempo real, mostrando por pantalla los cambios que van aconteciendo al momento. Esto se especifica mediante la opción `-f`, *follow*.

```
[node1] (local) root@192.168.0.8 ~/aries-cloudagent-python/demo
$ docker logs -f faber
Starting [faber] agent with args [--port 8020 --no-auto]
Initializing demo agent faber with AIP 20 and credential type indy
```

Figura 7: Seguimiento de los eventos acontecidos en Faber y su puerto: 8021.

Como podemos ver, el puerto en el que Faber se está ejecutando es el 8021. Este es el que debemos sustituir en `PORT` en la URL anterior para poder acceder al controlador del mismo.

Procedemos ahora de manera análoga para el agente Alice. Creamos una nueva instancia desde *Play with Docker* para no interrumpir la ejecución del agente Faber ni la terminal desde la que visualizamos sus eventos.

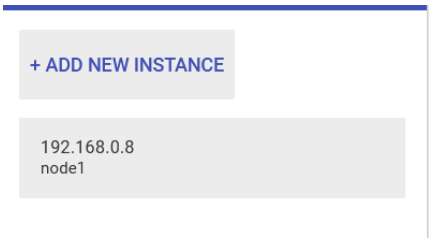

Figura 8: Creación de otra instancia para la ejecución del agente Alice.

Una vez creada, repetimos la clonación del repositorio *https://github.com/hyperledger/aries-cloudagent-python/*. Tras ello, también cambiamos de directorio al de la demo y de manera análoga ejecutamos el agente de Alice mediante `LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo alice --events --no-auto --bg`.

```
[node2] (local) root@192.168.0.7 ~
$ cd aries-cloudagent-python/demo
[node2] (local) root@192.168.0.7 ~/aries-cloudagent-python/demo
$ LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo alice --events --no-auto --bg
```

Figura 9: Ejecución del agente Alice.

Tras ello, del mismo modo que con Faber, accedemos a la URL de su controlador, la cual también viene especificada en el campo de `DOCKERHOST` en el momento en el que la ejecución del agente termina de ser procesada.

```
Preparing agent image...
sha256:8371adf06219e1c0f4118efe0966a82b8b5f59eb3a5e8cab30e70ff41d07071c
DOCKERHOST=ip172-19-0-66-ci1fgdggftqg009dhu40-{PORT}.direct.labs.play-with-docker.com
DOCKER_ENV=-e LOG_LEVEL= -e RUNMODE=pwd -e DOCKERHOST=ip172-19-0-66-ci1fgdggftqg009dhu40-{PORT}.direct.labs
.play-with-docker.com -e AGENT_PORT=8030 -e LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io -e GENESIS_URL
=http://dev.greenlight.bcovrin.vonx.io/genesis -e EVENTS=1 -e TRACE_TARGET=log -e TRACE_TAG=acapy.events -e
 TRACE_ENABLED=
7cf3a2c2273b537d762b483033ca8fc970757026bd0fdc1a724b1165a820b6bf
```

Figura 10: Dirección IP de Alice en el campo DOCKERHOST.

El puerto, de nuevo, podemos consultarlo al igual que todos los posibles cambios que ocurran en su agente, mediante el comando análogo `docker logs -f alice`. Como podemos ver, esta vez el puerto es el 8031.

```
[node2] (local) root@192.168.0.17 ~/aries-cloudagent-python/demo
$ docker logs -f alice
Starting [alice] agent with args [--port 8030 --no-auto]
Initializing demo agent alice with AIP 20 and credential type None
```

Figura 11: Seguimiento de los eventos acontecidos en Alice y su puerto: 8031.

Las direcciones IP de ambos agentes nos llevan a páginas como la siguiente, desde las cuales podemos ejecutar fácilmente los diferentes métodos que cada uno de los agentes posee.

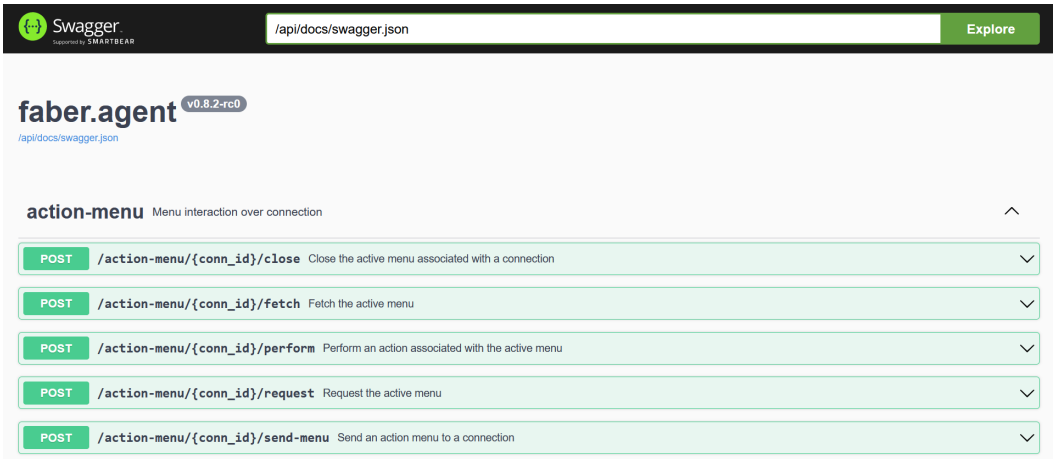

Figura 12: Interfaz desde la cual se puede interactuar con los agentes.

Desde aquí llevaremos a cabo todas las interacciones con los agentes de ACA-Py para establecer primero la comunicación entre ellos, enviar un mensaje para probar que funciona correctamente; y posteriormente generar una credencial verificable y enviársela al agente usuario. Tras almacenarla, el agente verificador le solicitará el envío de una presentación verificable, y el usuario enviará la credencial en dicho formato para intentar verificarse, primero con una credencial incorrecta y tras ello con una correcta.

5.2.2. Establecimiento de la Conexión

Para establecer la comunicación entre Faber, la entidad emisora; y Alice, la entidad usuaria; vamos a hacer uso de DIDComm. Como se especificó previamente, este protocolo permite la comunicación de manera segura entre los actores, pero no necesita del uso de una blockchain para ello. Por ello, el establecimiento de la conexión y el envío de mensajes no harán uso, por el momento, de Hyperledger Indy.

Para establecer la conexión pues, Faber ha de crear una invitación para abrir el canal de comunicación con otro agente, en este caso Alice. Para ello, mediante la interfaz dispuesta anteriormente para la interacción entre agentes, seleccionamos el método de petición HTTP `POST /connections/create-invitation`. La información a enviar será un JSON vacío .

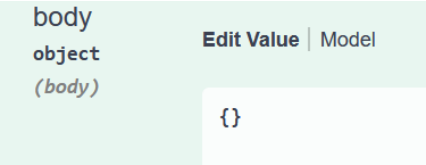

Figura 13: JSON vacío para enviar la invitación.

Además, para facilitar el seguimiento de la apertura del canal de comunicación, emplearemos un alias *TFM-UOC*, el cual podemos incluir como campo en el método de POST.

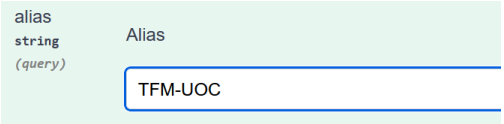

Figura 14: Uso de alias para facilitar el seguimiento.

Como respuesta, recibimos un identificador de conexión único, el cual emplearemos posteriormente

Figura 15: Identificador único de conexión (c5d...736).

De la respuesta obtenida, copiamos todo el contenido de `invitation`, pues lo emplearemos ahora desde el agente de Alice con el método `POST /connections/receive-invitation`

Tras ejecutarlo, obtenemos la siguiente respuesta, donde efectivamente vemos que tenemos una conexión iniciada en estado de `invitation`.

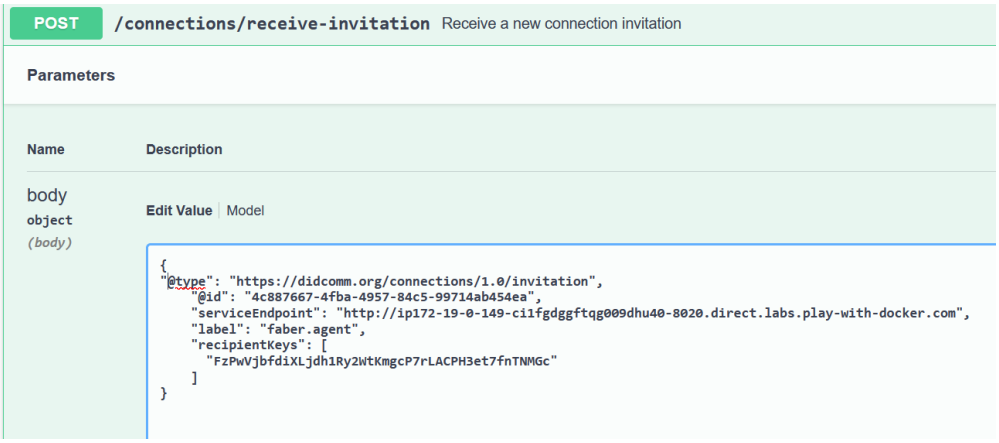

Figura 16: Contenido del campo invitation enviándose como cuerpo de la petición.

Figura 17: Conexión en estado de invitation

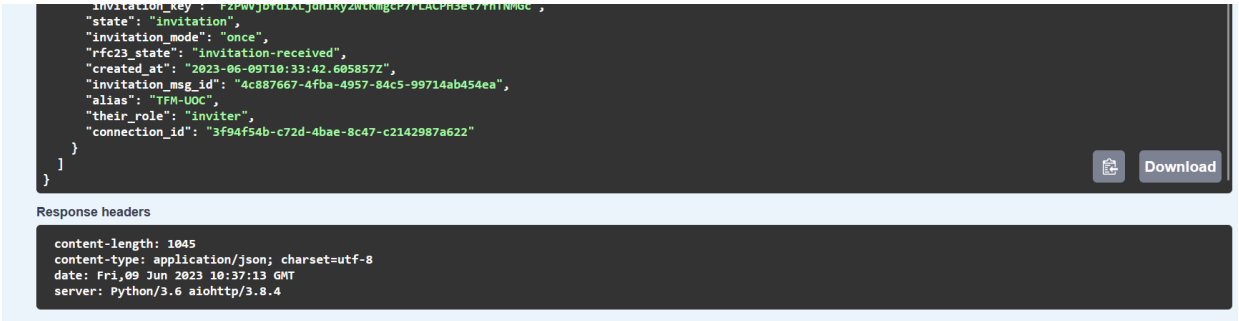

Figura 18: Conexión en estado de invitation

Entre los campos que vemos aquí cabe destacar el alias, que como escribimos antes, nos facilitará el seguimiento de la conexión. También vemos que el rol del agente emisor es el de `inviter`. Este rol cambiará

dependiendo de las acciones que esté realizando el agente de Faber, pues será tanto emisor de credencial como verificador a lo largo de todo el demostrador.

Con esto todavía no hemos establecido la conexión. Hace falta que Alice además de recibir la invitación la acepte. Para esto existe el método `POST /connections/{conn_id}/accept-invitation`, donde el campo çonn_id"se refiere a que hará falta introducir un identificador de conexión, correspondiente a aquella que queramos aceptar.

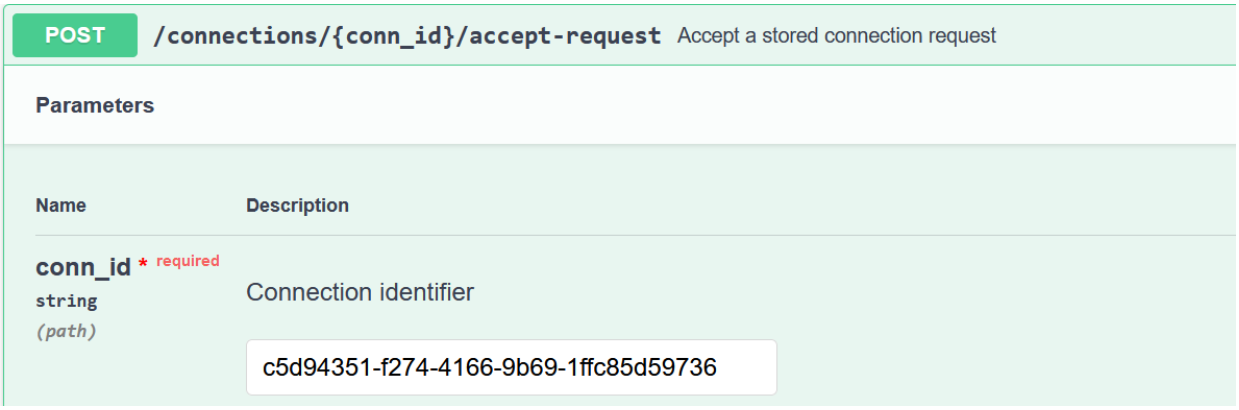

Figura 19: Introducimos el identificador obtenido en el momento de la creación de la invitación

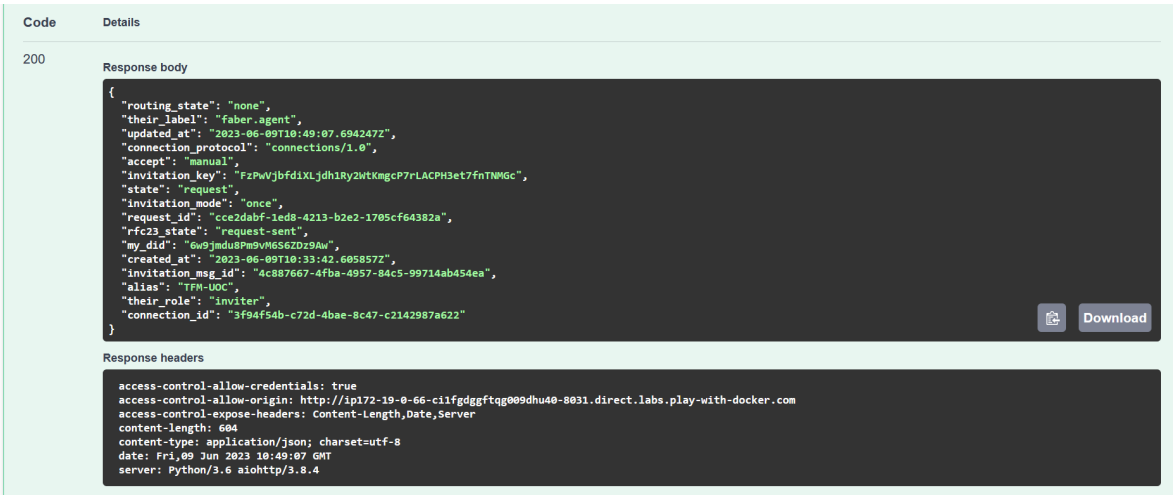

Figura 20: Respuesta recibida

En la respuesta podemos ver un valor de `connection_id` diferente al anterior, el cual en este caso empieza por `3f94...`. Este será el valor de conexión a utilizar en el siguiente paso cuando Faber acepte en su lado establecer la comunicación con Alice.

Tras la ejecución de este paso podemos ver en tiempo real desde la terminal de Faber en la página de *Play with Docker* como este ha recibido un evento. Este corresponde a la aceptación de la invitación por parte de Alice. Aquí también podemos ver tanto el alias, como el identificador de dicha conexión.

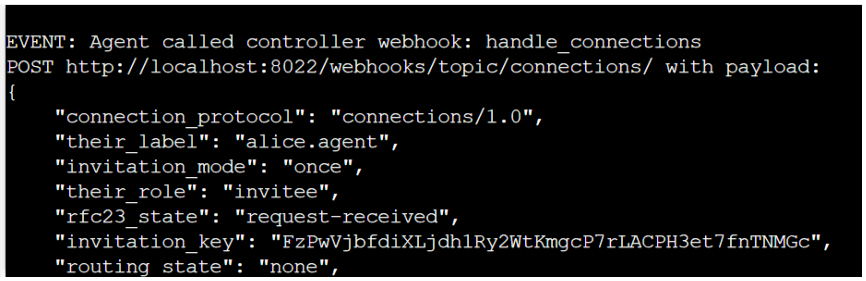

Figura 21: Evento recibido en Faber

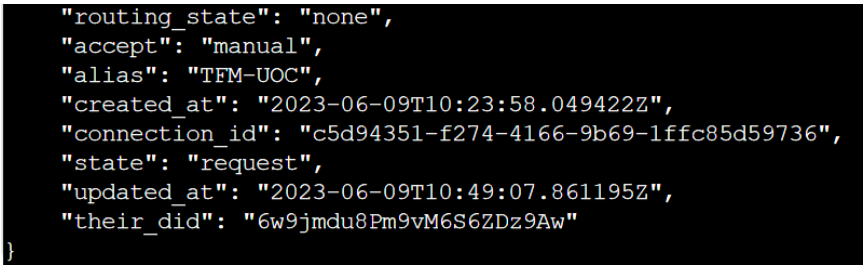

Figura 22: Evento recibido en Faber

Faber ahora completa el establecimiento de la comunicación con Alice aceptando su confirmación al otro lado de la conexión. Para ello hace uso del método `POST /connections/conn_id/accept-request`, el cual hace uso del identificador que obtuvimos de Alice para terminar de establecer la comunicación entre los agentes.

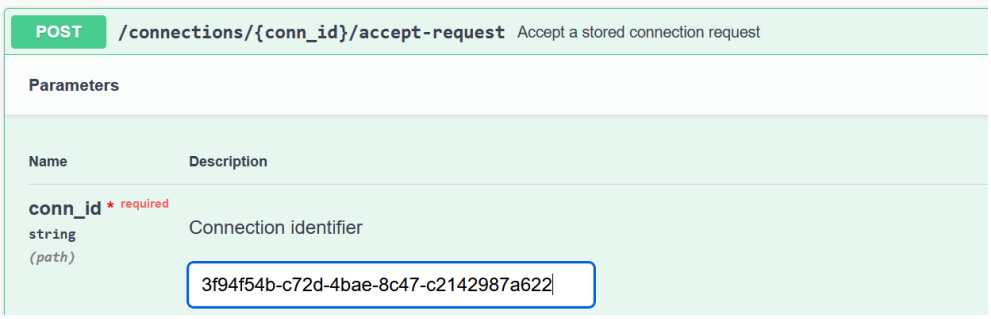

Figura 23: Especificamos el identificador de Alice en Faber

Una vez ejecutado, podemos comprobar desde ambos agentes mediante el método `GET /connections` que el estado de la conexión es `active`.

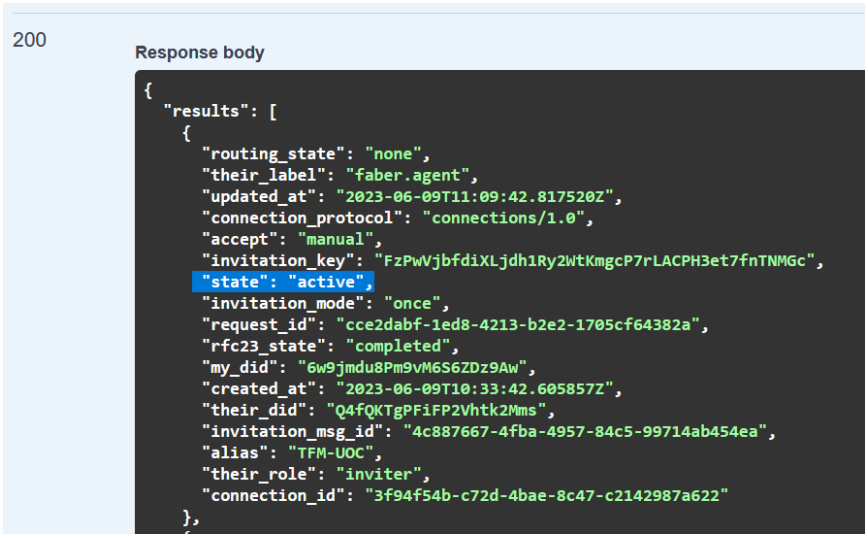

Figura 24: Estado de la conexión activo

Además, podemos observar en la terminal de Alice la recepción de un evento, correspondiente a la aceptación por parte de Faber de la conexión y de que, por lo tanto, la comunicación está establecida y abierta entre ambos.

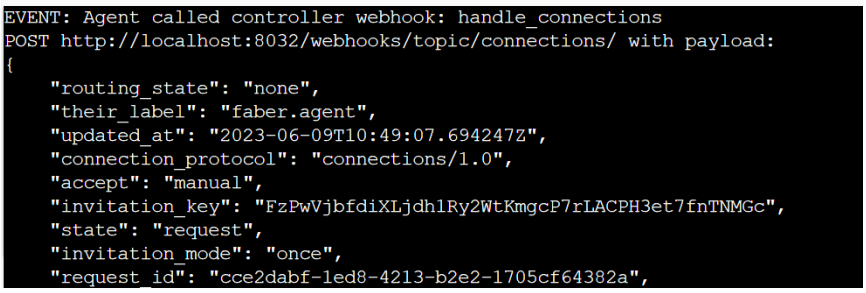

Figura 25: Evento recibido en Alice

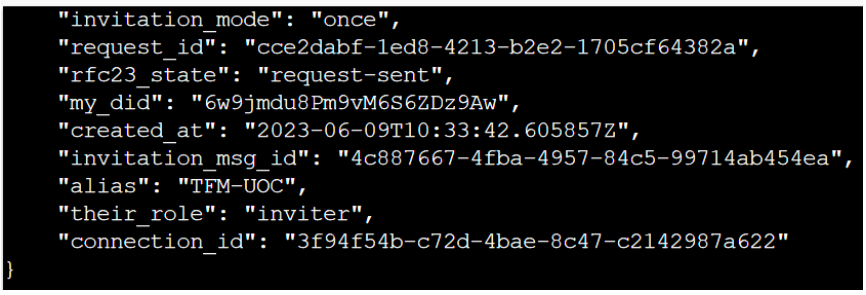

Figura 26: Evento recibido en Alice

5.2.3. Envío de un mensaje

Ahora que tenemos la conexión creada entre ambos agentes, vamos a probarla enviando un mensaje desde Alice hasta Faber. Para ello, desde el agente de Alice haremos uso del método `POST /connections/conn_id/send-message`. Aquí especificamos tanto el identificador de conexión por parte de Alice, como el contenido del mensaje que queremos enviar, en nuestro caso *Hello UOC*.

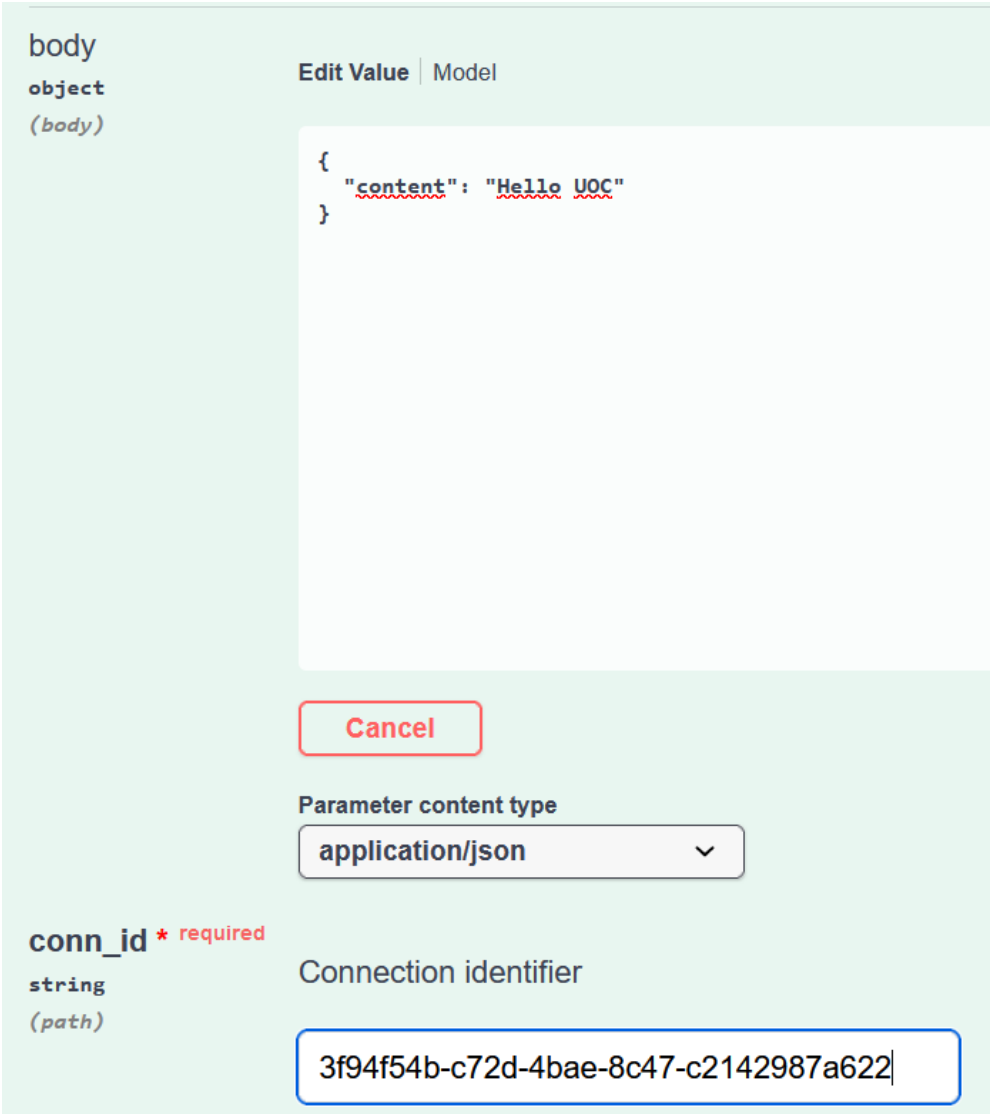

Figura 27: Preparación de envío de mensaje de Alice a Faber

Tras ello, en la terminal de Faber vemos la ejecución de un evento junto al mensaje que Alice ha enviado. A su vez, en la terminal de Alice vemos un evento de que efectivamente, Faber recibió el mensaje.

```
EVENT: Agent called controller webhook: handle_basicmessages
POST http://localhost:8022/webhooks/topic/basicmessages/ with payload:
{
Faber      | Received message: Hello UOC
    (1) Issue Credential
```

Figura 28: Recepción del mensaje en la terminal de Faber

```
POST http://localhost:8032/webhooks/topic/basicmessages/ with payload:
{
    "connection_id": "3f94f54b-c72d-4bae-8c47-c2142987a622",
    "message_id": "2bdc94cf-20a9-4825-8cd3-2ad3998cc6e1",
    "content": "faber.agent received your message",
    "state": "received",
    "sent_time": "2023-06-09T11:20:45.019890Z"
}

Alice      | Received message: faber.agent received your message
```

Figura 29: Confirmación de la recepción en la terminal de Alice

5.3. Emisión de una Credencial Verificable

Para que Faber pueda comenzar a emitir credenciales, primero se tiene que registrar en la cadena de bloques que se emplee en el modelo, en este caso Hyperledger Indy. De este modo, su identificador, asociado a su clave pública, queda registrado para la futura comprobación de las entidades proveedoras de servicios en el momento en el que que necesiten corroborar la autenticidad de una Presentación Verificable asociada a una Credencial Verificable emitida por Faber. No solo eso, sino que también se registra el tipo de esquemas de Credenciales Verificables que ese *issuer* puede emitir y el formato que estas van a tener. De este modo nos aseguramos que una entidad emisora genere únicamente credenciales para las cuales está cualificada emitir.

5.3.1. Preparación de Faber para generar una credencial

En el demostrador de ACA-Py que estamos utilizando esta serie de acciones se realizan en el mismo momento en el que comenzamos a ejecutar el agente. Para poder visualizar en la blockchain dicho registro, necesitamos el identificador único de nuestra entidad emisora. Este lo podemos obtener haciendo uso del método `GET /wallet/did/public`, el cual nos devolverá dicho valor, la clave pública asociada y un valor *booleano* que nos indica que efectivamente se trata de los elementos públicos identificadores del *issuer* Faber.

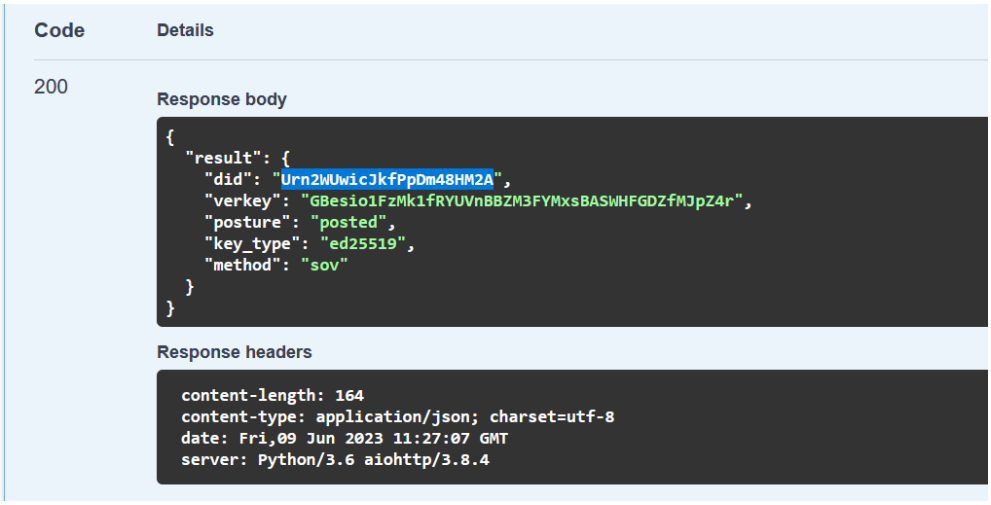

Figura 30: Respuesta del método GET con el identificador de Faber

Ahora con este identificador podemos ir a la página del ledger público de Indy en *BC Government's VON Team*, y buscarlo desde la página de domain. En su buscador pegamos el identificador que nos devolvió el método anterior para filtrar únicamente aquellas transacciones *on-chain* relacionadas con Faber.

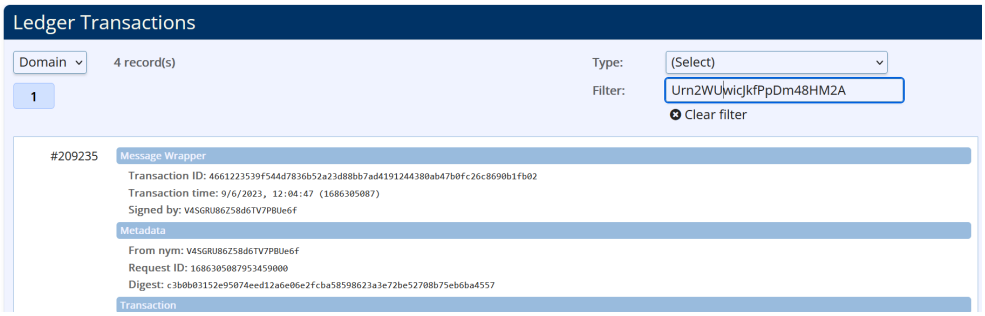

Figura 31: Resultado de la búsqueda on-chain del DID de Faber.

Como podemos ver, hay un total de 4 transacciones en la cadena. Estas equivalen a las siguientes acciones llevadas.

- El registro inicial del DID del issuer Faber.
- El registro del DID público que actúa como endpoint para el resto de agentes.
- El registro del esquema.
- El registro del formato para las Credenciales Verificables.

Figura 32: El registro inicial del DID del issuer Faber.

Figura 33: El registro del DID público que actúa como endpoint para el resto de agentes.

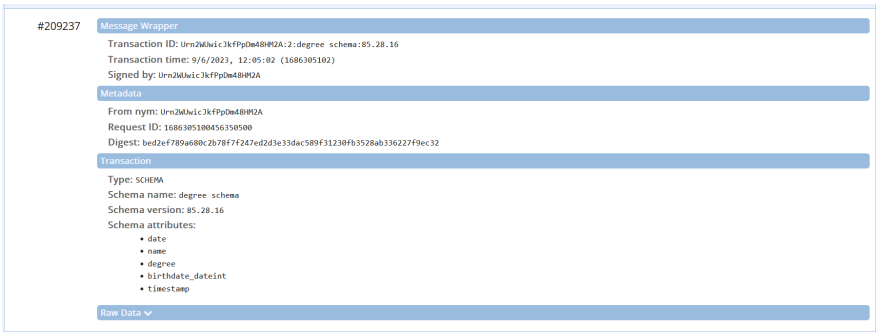

Figura 34: El registro del esquema.

Figura 35: El registro del formato para las Credenciales Verificables.

5.3.2. Envío de la Credencial Verificable

También podemos obtener los identificadores de las definiciones en la cadena de bloques desde la interfaz del agente de Faber tanto para el esquema `GET /schemas/created` y como para el formato de las Credenciales Verificables mediante `GET /credential-definitions/created`.

Estos valores serán necesarios para la correcta emisión de credenciales, pues se han de especificar en los atributos del fichero JSON como valores para su posterior comprobación.

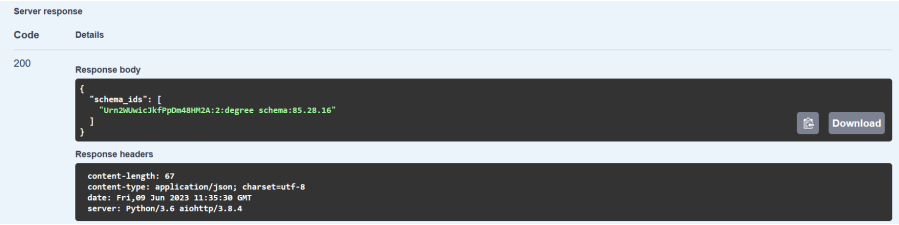

Figura 36: Respuesta del método GET /schemas/created

Para emitir ahora una credencial vamos a necesitar varios valores. Primero, obtenemos el identificador de la conexión connection_id mediante el valor `GET /connections`. Este valor le pondremos en el mismo campo del JSON que podemos encontrar en el método de `POST /issue-credential-2.0/send` Además, hace falta rellenar otros campos en ese mismo archivo antes de la ejecución del método. Estos son los siguientes

- `issuer_did.` En este campo ponemos el identificador de Faber que empleamos para realizar la búsqueda en el ledger.
- `schema_id`. Aquí ponemos el identificador del esquema que el método de GET /schemas/created nos devolvió.
- `cred_def_id.` Este atributo refiere al identificador de la definición del formato de las credenciales obtenido de GET /credential-definitions/created.
- `schema_version.` Es el campo referente a los números que podemos encontrar justo al final de schema_id, justo después de los dos puntos *:*.
- `schema_issuer_did`. Aquí también pondremos el identificador de Faber como issuer, el que empleamos para hacer la búsqueda en el ledger público de Indy.
- `schema_name`. En este caso el valor será *degree schema*, pues vamos a emitir una credencial académica asociada a una titulación.
- `auto_remove.` Este valor refiere a si, tras la emisión de la credencial, el issuer eliminará o no el registro de que dicha credencial fue emitida. Lo pondremos a true, y de este modo nos aseguramos de que Faber no almacene excesivos datos y pueda saturarse por ello. Cabe destacar que, en caso de que sea necesaria una prueba de la emisión de esta credencial, bien por motivos legales o de políticas de registro, se deberá realizar el almacenamiento de forma obligada, sea en el propio agente especificando este valor a *false*, o en otro sistema.
- `comment.` Ponemos el valor a *string* para que se pueda especificar un comentario en el momento de la emisión de la credencial sobre qué es su contenido u otras especificaciones necesarias para el usuario al que se emite.
- `trace.` Lo ponemos en *false*. Este valor se emplea para realizar *debugging* y observar el procedimiento de envío paso a paso, mediante una traza.

- Eliminamos además toda la parte que engloba *dif*, es decir, aquellos campos referentes a especificaciones técnicas de la conexión que no nos harán falta.

Además, incluimos los atributos exigidos en la definición del formato de la credencial académica, quedando el archivo JSON a enviar en el método POST de la siguiente forma. En este envío vamos a generar una credencial que posee un valor de `birthdate_dateint` superior al valor que el verificador exigirá y otro que cumpla la condición. De este modo veremos como una presentación verificable será rechazada y la otra aceptada. Una vez todos los valores han sido cambiados, ejecutamos el método `POST /issue-credential-2.0/send`. Para ello, el JSON debiera quedar de la siguiente manera.

```
{
  "auto_remove": true,
  "comment": "string",
  "connection_id": "c5d94351-f274-4166-9b69-1ffc85d59736",
  "credential_preview": {
    "@type": "issue-credential/2.0/credential-preview",
    "attributes": [
       {
        "name": "name",
        "value": "Alice Smith"
      },
      {
        "name": "timestamp",
        "value": "1234567890"
      },
      {
        "name": "date",
        "value": "2023-06-28"
      },
      {
        "name": "degree",
        "value": "Master en Ciberseguridad y Privacidad"
      },
      {
        "name": "birthdate_dateint",
        "value": "910266624"
      }
    ]
  },
  "filter": {
    "indy": {
      "cred_def_id": "Urn2WUwicJkfPpDm48HM2A:3:CL:209237:faber.agent.degree\_schema",
      "issuer_did": "Urn2WUwicJkfPpDm48HM2A",
      "schema_id": "Urn2WUwicJkfPpDm48HM2A:2:degree schema:85.28.16",
      "schema_issuer_did": "Urn2WUwicJkfPpDm48HM2A",
      "schema_name": "degree schema",
      "schema_version": "85.28.16"
    }
},
  "trace": false
}
```

En la terminal de Alice de Play with Docker podremos ver que ha recibido un evento del envío de la credencial, pero, como podemos comprobar desde su interfaz de control de métodos `GET /issue-credential-2.0/records`, todavía no ha sido almacenada, sino que se encuentra en estado `state = credential-received`.

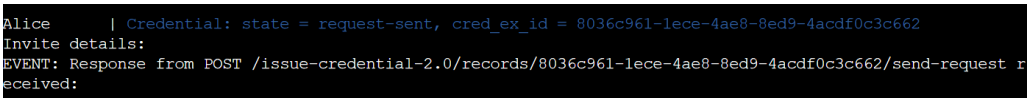

Figura 37: Vemos en la terminal de Alice la recepción de la credencial

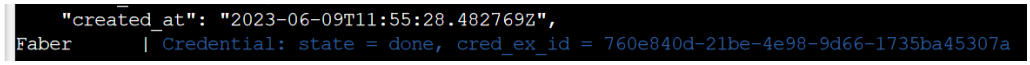

Figura 38: Vemos en la terminal de Faber que Alice recibió la credencial correctamente

Ahora hace falta que Alice almacene la credencial en su cartera digital. Para ello debemos aceptar la credencial recibida y realizar el guardado de forma manual mediante el método `POST /issue-credential-2.0/records/cred_ex_` y especificando el identificador de la credencial en el campo de `cred_ex_id`, el cual podemos ver en el método GET ejecutado anteriormente. El campo con los valores en formato JSON a enviar lo dejaremos vacío, pues el propio agente es capaz de generar un valor único para `credential_id` de forma aleatoria. Una vez tenemos ambos campos especificados, ejecutamos el método.

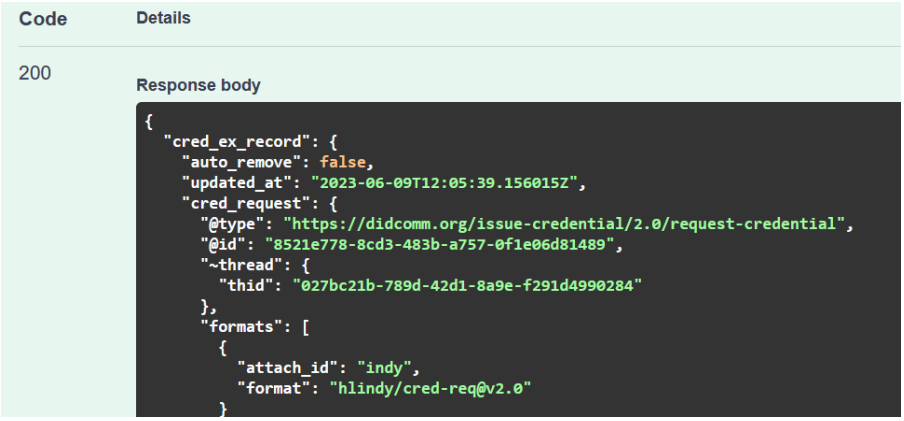

Figura 39: Respuesta tras ejecutar el método para almacenar la credencial recibida en la cartera digital de Alice

Tras ello, en la terminal de Faber capturaremos el evento de que Alice ha aceptado la credencial. Debido a que especificamos el valor de `auto_remove` a `true`, Faber, tras dicha notificación, elimina el registro del envío. Esto lo podemos comprobar mediante el método `GET /issue-credential-2.0/records`, el cual nos devolverá un elemento vacío.

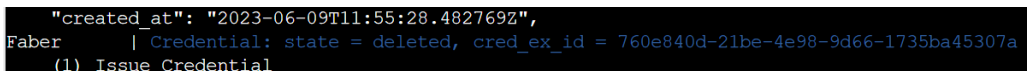

Figura 40: En la terminal de Faber vemos que eliminó el registro del envío de la credencial.

5.4. Solicitud de una Presentación Verificable

Ahora vamos a realizar la presentación de la credencial, donde veremos cómo la entidad verificadora Faber, que cambia de rol para poder realizar esta parte de la demostración, solicitará a Alice una credencial académica y esta le enviará la Presentación Verificable con la credencial almacenada en el paso anterior. La selección de qué credencial a enviar se hará de manera automática, si bien, al poseer Alice dos credenciales, una válida y otra no válida, ejecutaremos primero la demostración con la no válida, veremos cómo da error, la borraremos, y repetiremos la ejecución con la credencial correcta.

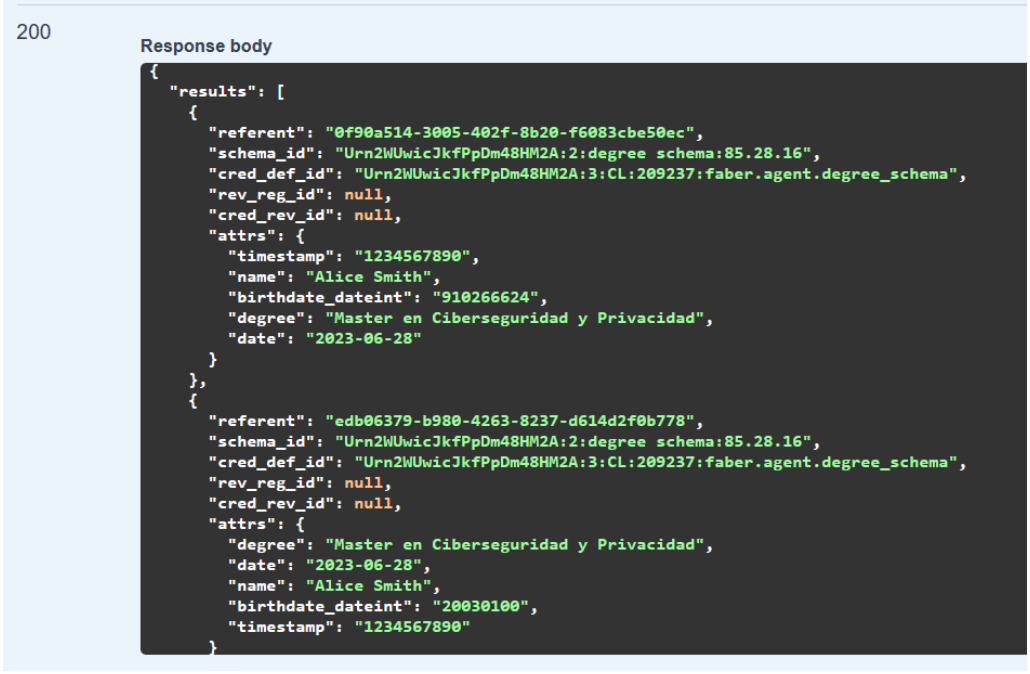

Figura 41: Contenido de la cartera digital de Alice.

Para que Faber envíe ahora a Alice la solicitud de presentación utilizaremos el siguiente método: `POST /present-proof-2.0/send-request` . En este método será necesario cambiar en el JSON a enviar con la petición el valor de `cred_def_id` con el valor de la credencial solicitada. Primero vamos a requerir la credencial incorrecta, y ahora veremos por qué esta no es válida. Además, hay que actualizar el valor `connection_id` con el de la conexión de Faber con Alice para emplear el medio de comunicación establecido en los primeros pasos. Además, en `comment` ponemos el valor que queramos y que consideremos necesario para que Alice comprenda a qué se debe la solicitud.

Una vez la presentación es enviada, con la credencial errónea esta fallará. Como podemos ver, en el momento en el que Alice envía su información en forma de Presentación Verificable, Faber comprueba que el valor de `birthdate_dateint` almacenado sea igual o inferior al de la credencial. En este caso no se cumple la premisa, por lo que los datos son descartados y la autenticación finaliza.

Figura 42: Condición de admisión.

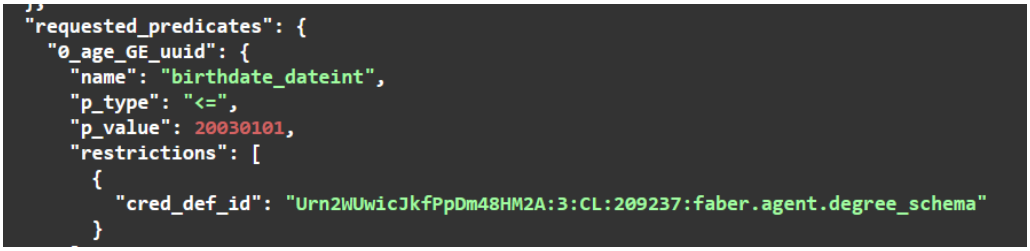

Figura 43: Presentación Verificable rechazada, pues no cumple la condición.

Vamos a borrar ahora la credencial anterior y volver a ejecutar este paso. Para borrar la credencial incorrecta, ejecutamos `DELETE /credential/credential_id`. Aquí, especificando el identificador de la credencial, logramos su borrado de la cartera de Alice.

Figura 44: Borrado de la credencial incorrecta.

Ahora, repetimos los pasos anteriores para volver a solicitar la presentación de la información. En este caso, todo ocurrirá de la forma esperada y podremos comprobar tanto en la terminal de Faber como en la de Alice que el envío se ha hecho de la forma correcta. En la terminal de Alice habremos capturado un evento por el envío de la credencial. En este evento podemos comprobar que el código en *base 64* se traduce por un JSON que contiene la información referente a su credencial.

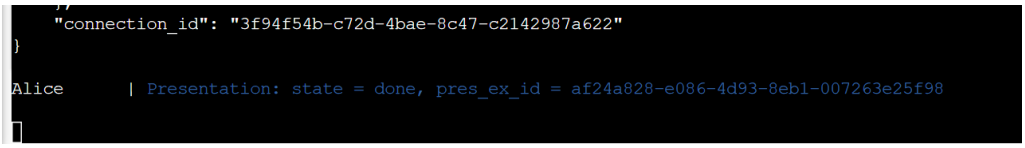

Figura 45: Terminal de Alice, donde vemos que la presentación se ha realizado correctamente.

Por otro lado, desde la terminal de Faber vemos que hemos capturado otro evento, complementario al anterior, el cual muestra que Faber ha recibido la información correspondiente a la petición. Como podemos ver, el estado de la petición ha cambiado a `state=done`, indicando que el procedimiento ha finalizado de manera exitosa.

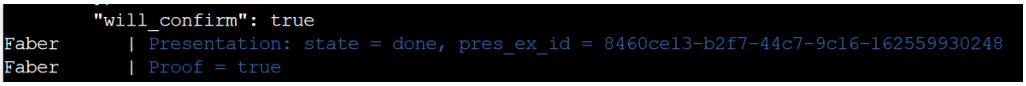

Figura 46: Terminal de Faber, donde vemos que la presentación se ha realizado correctamente.

Con este paso concluye el demostrador. Hemos podido comprobar cómo dos entidades se comunican emitiendo y verificando una serie de datos gracias a la infraestructura proveída por Hyperledger Indy para llevar un seguimiento de las transacciones occuridas, DIDComm para las comunicaciones e Hyperledger Aries para integrar todas las capas del modelo y permitir la interacción con el mismo.

6. Conclusión

El ámbito de la Identidad Digital se encuentra en un momento claramente de cambio, donde el foco de atención ha pasado a ser la descentralización de esta y, con ello, hacer al usuario el único dueño de sus datos. De este modo se otorga completo control al usuario sobre su información, eligiendo qué quiere compartir con qué entidades, evitando así que sus datos sean reenviados o comprados y vendidos por las empresas a terceros. Así, los usuarios volverían a ser dueños de su identidad, pudiendo controlar a quién desean proporcionar esos datos en todo momento. Sin embargo, todavía es demasiado pronto para poder hablar de un modelo estrictamente definido, por lo que están apareciendo diversas soluciones en torno al tema. No solo eso, sino que para el correcto funcionamiento del modelo hacen falta diversas tecnologías, categorizadas por capas según Trust Over IP. Como ha podido verse, el actual estado del arte muestra una enorme diversidad tecnológica en todas y cada una de dichas capas. Es por ello por lo que aquellas tecnologías que logren adaptar un mayor número de esquemas y modelos son las que están ganando más terreno, permitiendo la convivencia de una mayor diversidad de propuestas.

Como también hemos podido ver, aun así, existen claros beneficios y desventajas en el uso de ciertos protocolos sobre otros, por lo que la decisión de cuando implementar uno u otro dependerá en gran parte del propósito y entorno para el que se destine el modelo en cada caso. Mediante el demostrador hemos podido llevar la teoría a la práctica. Paso a paso hemos visto el funcionamiento de DIDComm para establecer la conexión entre dos agentes sin necesidad de una blockchain para el intercambio de información. Después, mediante Hyperledger Indy, hemos podido realizar el correcto intercambio de credenciales, comprobando el correcto y necesario registro de Faber como entidad emisora válida en el modelo, generando la Credencial Verificable para Alice, y finalmente solicitando la Presentación Verificable de la misma, probando tanto un caso erróneo como uno correcto. Todo ello ha sido posible gracias al framework de Hyperledger Aries, el cual facilita las comunicaciones entre capas.

Como puede verse, ya existen modelos que implementan la teoría para la Self Sovereign Identity. Sin embargo, todavía está por ver todos los demás avances en torno al uso de las nuevas tecnologías descentralizadas. Hará falta una adaptación por parte legal para el correcto tratamiento de los datos de los usuarios en entornos descentralizados, donde los roles de Controlador de Datos y Procesador de Datos, no están tan claros. Ya existen diversos estudios realizados para poder adaptar la regulación, sobre todo la europea referente al Reglamento General de Protección de Datos, al uso de estas tecnologías descentralizadas, garantizando el correcto tratamiento de la información de los usuarios. Algunos de ellos son *Blockchain and the GDPR: Solutions for a responsible use of the blockchain in the context of personal data*, del CNIL[23]; *Blockchain and the GDPR. A thematic report prepared by the European Union Blockchain Observatory and Forum*[20]; o *Blockchain and the General Data Protection Regulation. Can distributed ledgers be squared with European data protection law?*, del *EPRS - European Parliamentary Research Service*[24]; aunque la existencia de estos no implica que la European Data Protection Board haya emitido comunicados oficiales sobre ello, por lo que todavía son necesarios avances a nivel legal. No solo esto, sino que también habrá que clasificar adecuadamente a las nuevas entidades emergentes proveedoras de servicios de acuerdo a la ley de Servicios de la Sociedad de la Información,[25] especificando y delimitando sus responsabilidades en caso de que alguna impertinencia o comisión delictiva tuviera lugar en las infraestructuras descentralizadas de la blockchain.

Otro ámbito donde deben realizarse claros avances es a nivel social. Será necesario educar tanto al sector público como privado del nuevo modelo para el intercambio de datos, adaptar sus infraestructuras y formar a profesionales en el ámbito, tanto a nivel legal como técnico.[26] También hay que educar a los usuarios, los cuales serán ahora responsables de su información. Por ello mismo, es necesario facilitar la usabilidad del modelo, proporcionando medidas para la recuperación de claves privadas, como las iniciativas en Ethereum de ERC 4337,[27] donde se permite la creación de carteras digitales personalizadas, por lo que sería posible implementar la recuperación habitual por envío a correo u otra cartera de los datos mediante una contraseña o similar; o la recuperación por convenio social,[**souls**] donde varias entidades poseen la información necesaria para confirmar la recuperación de la clave una cuenta, y solo cuando estas están de acuerdo a petición del

dueño de la misma, ponen a disposición su parte de la información para recuperar la clave.

Está claro que esta tecnología necesita de varios avances antes de poder ser establecida en su plenitud, aunque muchas de estas adaptaciones ya poseen de propuestas e iniciativas por los expertos en la materia. Este interés por lograr su implementación en la sociedad proviene del gran beneficio que proporcionaría a sus usuarios, haciéndolos plenamente dueños de sus datos y mejorando de manera sustancial su privacidad ante el actual mercado de datos existente en la nueva era de Internet.

7. Trabajo a Futuro

En esta sección se muestran aquellos puntos a tratar en el futuro del proyecto, debido a que escapan el alcance de este trabajo.

Uno de los apartados más interesantes que puede expandirse es la repercusión legal bajo el Reglamento General de Protección de Datos, pues no existe un consenso oficial respecto al uso de tecnologías descentralizadas donde no es tan fácil determinar el rol del Controlador de los Datos ni el de los Procesadores de Datos. Tampoco es trivial ejercer el derecho al olvido ni a la rectificación en una tecnología donde la información es inmutable. Todas estas problemáticas poseen una serie de posibles soluciones, las cuales sería muy interesante discutir y comparar.

Además, junto a ello, un análisis de las responsabilidades de la propia blockchain y sus participantes bajo el amparo de la Ley de Servicios de la Sociedad de la Información, donde de nuevo el borrado de información en caso de ser requerido, no sería trivial debido a las características intrínsecas de las redes blockchain.

Otro apartado que podría expandirse es el referente al Estado del Arte. Aquí se han seleccionado diversos protocolos para cada capa definida en el esquema de Trust Over IP, pero existen muchísimos más que podrían ser analizados y comparados. También sería interesante hacer una comparación a nivel técnico, empleando demostradores que hagan uso de otros frameworks, protocolos de comunicación, esquemas de datos para credenciales... O incluso propuestas más novedosas como modelos que hagan uso de SoulBound Tokens o incluso adaptaciones con NFTs o sobre la recientísima Bitcoin Virtual Machine.

Finalmente, cabe destacar que dado que esta tecnología se encuentra en un momento de evolución e investigación de nuevas propuestas, surgirán constantemente nuevas aproximaciones y protocolos innovadores, así como adaptaciones y análisis bajo los correspondientes reglamentos. Todo ello requerirá en el futuro de su correspondiente evaluación para seguir iterando tanto a nivel técnico como legal hasta llegar a un consenso que permita establecer y adoptar la Identidad Descentralizada a nivel global.

Anexo I. Glosario

JOSE

El estándar Javascript Object Signing and Encryption, o JOSE, se encarga de definir formatos y algoritmos para el tratamiento de objetos JSON en la web. Estos son los JSON Web Signature (JWS), que definen cómo firmar y verificar objetos JSON; JSON Web Tokens (JWT), que describen la estructura de datos y contenido que los objetos JSON han de tener; y los JSON Web Encryption (JWE), que especifica los métodos y algoritmos a usar para cifrar y descifrar objetos JSON.

ZK-Proof

Los *Zero Knowledge Proof* (ZKP) o Protocolos de Conocimiento Nulo/Cero, son algoritmos criptográficos que permiten que una parte demuestre a otra la posesión de un elemento sin la necesidad de que revele su contenido. Un ejemplo claro es, la demostración de la mayoría de edad sin tener que mostrar la fecha de nacimiento, sino simplemente un valor que diga si el usuario posee más o menos de 18 años.

Burning o quema

La técnica de burning dentro del contexto de las tecnologías blockchain, consiste en enviar un token a una dirección a la que nadie tiene acceso, comúnmente la NullAddress. De este modo, la recuperación de dicho objeto se vuelve imposible, anulando cualquiera fuese su previa utilidad en el ecosistema.

IPFS

InterPlanetary File System o IPFS para acortar, consiste en un protocolo y una red *peer to peer* para almacenar y compartir archivos multimedia. Dicho almacenamiento se realiza en los distintos nodos que componen la red de IPFS, pudiendo el usuario elegir entre almacenarlo en su nodo de forma local o en toda la red, enviándolo y replicándose este en todos los nodos que la componen para asegurar su perduración en el tiempo.

Gnosis

Gnosis es una DAO, es decir, una *Decentralized Autonomous Organization* donde los miembros votan para tomar decisiones dentro de la misma en lugar de poseer un único líder o conjunto de ellos. Esta, ha creado una serie de productos, entre los cuales está la cadena blockchain Gnosis Chain, creada mediante el protocolo de Ethereum pero añadiendo ciertas características diferenciadoras, como un pago de gas fees inferior.

Gas fees

Las gas fees o tarifas/tasas de gas son un pago necesario en la red de Ethereum por cada transacción realizada. La cantidad a pagar varía dependiendo de lo congestionada que se encuentre la red en ese momento, siendo a un mayor número de operaciones en marcha, y por lo tanto un mayor gasto computacional de la Ethereum Virtual Machine, una cantidad de gas mayor solicitada. Este elemento existe para hacer la red más segura, evitando de este modo los ataques de denegación de servicio que se pudieran hacer a los diferentes nodos que componen la red, pues en el proceso cargarían demasiado la red y la continuación de emisión de transacciones en la misma implicaría a los atacantes un mayor pago de gas fees por cada conexión realizada.

Framework

Un framework es un esquema de trabajo definido para la resolución de problemas similares. En software refiere a un conjunto de herramientas y técnicas que facilitan el desarrollo de código al no tener que escribirlo de manera repetitiva y manteniéndolo consistente entre los diversos proyectos que comparten el mismo framework.

Cloud Computing

El servicio en la nube o *cloud computing* es la entrega de recursos informáticos, como sistemas de almacenamiento, software o servidores, a través de internet en lugar de mantenerlos a nivel local. El acceso a estos se realiza de manera remota.

Proof of Work

Proof of Work (PoW) es un algoritmo de consenso empleado en algunas blockchain para validar las transacciones. Este es llevado a cabo por los mineros, los cuales resuelven problemas computacionales complejos para demostrar su participación a cambio de criptomonedas. Con ello se garantiza la seguridad de la red, pues para modificarla sería necesario un altísimo nivel de capacidad computacional.

Proof of Authority

Proof of Authority (PoA) es un algoritmo de consenso empleado en algunas blockchains para validar transacciones. Este se trata de un algoritmo más centralizado, donde una serie de nodos de confianza conocidos como autoridades, determinan la validez de una transacción y si esta debe o no ser agregada en la información del nuevo bloque.

Proof of Stake

Proof of Stake (PoS) es un algoritmo de consenso empleado en algunas blockchains para validar transacciones. Este está basado en la selección del nodo validador en función de la cantidad de criptomonedas que tenga bloqueadas para este fin, donde a un mayor número de monedas y tiempo en la red, mayor es la probabilidad de ser seleccionado y computar y validar la transacción a cambio de recibir criptomonedas.

ISO/IEC 18013-5:2021

Documento de la ISO (*International Standard Organization*) que establece especificaciones de interfaz para la implementación de una licencia de conducir en asociación con un dispositivo móvil.

DApp

Una DApp o Decentralized Application es una aplicación descentralizada que utiliza tecnología blockchain y contratos inteligentes para operar de manera transparente, segura y sin la necesidad de intermediarios; ejecutándose en la red distribuida de nodos que interactúan entre sí.

eIDAS

eIDAS (*Electronic Identification, Authentication and Trust Services*) es un reglamento establecido dentro del marco de la Unión Europea el cual define el marco legal para la identificación electrónica y los servicios de confianza en el ámbito digital. Fue adoptado en 2014 y se aplica en todos los países miembros de la UE. Con ello se busca facilitar y promover las transacciones electrónicas en dicho territorio al proporcionar un marco común bajo el que trabajar.

eIDAS2

eIDAS2 (*electronic IDentification, Authentication and trust Services 2*) es una nueva propuesta publicada el 3 de Junio de 2021 surgida de la necesidad de adaptar el reglamento de eIDAS a las nuevas necesidades que surgen entorno a la identidad digital. Con ello se busca ofrecer una Identidad Digital Europea común, la cual tiene como enfoque al usuario como único dueño de sus datos e información.

Ataque Bizantino

Un ataque bizantino es aquel que, en un sistema distribuido, un conjunto de nodos se comportan de manera maliciosa y comienzan a engalar al resto enviando información errónea. El objetivo del ataque es interrumpir la comunicación entre los nodos bienintencionados para comprometer la seguridad y la integridad de la red. Los algoritmos de consenso se construyen para mitigar todo lo posible este tipo de ataques.

Ataque del 51 %

El ataque del 51 % es un escenario en estructuras blockchain donde un solo individuo o un grupo coordinado controla más del 50 % del poder de cómputo de la red. Esto les permite ganar el control de la red y, con ello, la validación de transacciones y la creación de nuevos bloques, así como la capacidad para modificar transacciones pasadas alterando el histórico de la blockchain.

Ethereum Virtual Machine

La Ethereum Virtual Machine o EVM es una máquina virtual que se ejecuta en todos los nodos que conforman la red de Ethereum. Es un entorno de ejecución que permite a los desarrolladores crear y ejecutar contratos inteligentes en la blockchain de Ethereum de forma segura y aislada. El código aquí ejecutado es Turing completo, por lo que los *Smart Contracts* pueden realizar cálculos complejos y lógica programática.

Staking

El *staking*, es el proceso en el que los usuarios de una red blockchain, cuyo algoritmo de consenso es Proof of Stake, bloquean una cantidad de criptomonedas para convertirse en validadores y, dependiendo de la cantidad bloqueada, tener un mayor o menor número de posibilidades de obtener recompensas en forma de criptomoneda a cambio.

JSON

JSON o *JavaScript Object Notation* (Notación de Objetos de JavaScript) es un formato de datos ligero y legible por humanos empleado para representar información en la forma de pares clave-valor. Es ampliamente

utilizado en el intercambio de datos entre aplicaciones, además de ser compatible con diversos lenguajes de programación.

JWT

JWT o *JSON Web Token*, es un estándar para la creación de tokens de acceso basados en el formato JSON, empleados para autenticar y autorizar el acceso a recursos tanto en sistemas web como en aplicaciones.

Un JWT consta de tres partes. La primera es el encabezado o *header*, que especifica el tipo de token y el algoritmo de firma utilizadola; la segunda es la carga útil *payload*, que contiene la información a transportar; y la última es la firma o *signature*, que se emplea para verificar la autenticidad e integridad del token.

Peer-to-peer

Peer-to-peer, también cono cido como P2P de forma abreviada, es una arquitectura de red descentralizada en la cual los nodos participantes (conocidos como pares) juegan el rol de cliente y servidor. Estos se conectan directamente entre sí sin la necesidad de un servidor centralizado para realizar el intercambio de comunicaciones.

NullAddress

La NullAddress (la cual puede referir bien a la dirección 0x0000000000000000000000000000000000000000 o a la dirección 0x000000000000000000000000000000000000dead, ambas en Ethereum) es una dirección a la cual ningún usuario posee acceso. De este modo, cualquier token enviado a la misma es invalidado, pues no es posible recuperarlo o seguir utilizándolo de ningún modo. Cabe destacar que la dirección acabada en 0000, también es la dirección empleada por protocolo como la dirección desde la que supuestamente se origina un token cuando es creado por primera vez, es decir, minteado.

Mempool

La mempool o *memory pool* es la parte de ciertas blockchains donde se almacenan aquellas transacciones que han sido enviadas pero todavía no validadas por los nodos participantes. Incluye pues, la lista de transacciones pendientes que todavía no han sido agregadas a la cadena de bloques.

Tasa de satoshis

La tasa de satoshis, o *satoshis per byte (sat/byte)*, es la unidad utilizada para medir la tarifa de transacción en la red de Bitcoin. Un satoshi es la unidad más pequeña de Bitcoin, equivalente a 0.00000001 BTC, de manera similar que el céntimo es al euro. A mayor sea la tarifa adjunta a una transacción en esta red, mayor es el incentivo para los mineros y, por lo tanto, mayor prioridad tendrá en ser escogida y añadida a la cadena de bloques.

API Endpoint

Un API endpoint es un punto de acceso a una API (Interfaz de Programación de Aplicaciones) que permite a los desarrolladores interactuar con los recursos proporcionados por el servicio .

Debugging

El debugging es el proceso de identificar y corregir errores o fallos en el código de un programa o una aplicación, también conocidos como *bugs*.

Ledger

El ledger o libro mayor digital registra todas las transacciones y actividades realizadas en una red blockchain.

Este consiste en una base de datos descentralizada y distribuida, la cual se encuentra en varios de los nodos que conforman la red. Cada nodo tiene una copia completa del mismo, donde se contienen todas las transacciones de forma transparente e inmutable. De esta manera, todos los participantes de la red poseen la misma información consensuada de las acciones ocurridas en la misma.

Base64

Base64 es un esquema de codificación el cual permite representar datos binarios en formato de texto ASCII, código de caracteres basado en el alfabeto latino, tal como se usa en inglés. Consiste en convertir las secuencias de bytes en una cadena de caracteres compuesta por un conjunto de 64 caracteres predefinidos.

Github

GitHub es una plataforma de almacenamiento de código fuente que también sirve para la colaboración en proyectos de desarrollo de software. Permite a los desarrolladores almacenar, administrar y compartir su código, así como colaborar en proyectos de código abierto.

Repositorio

En GitHub, los proyectos se organizan en repositorios, los cuales son espacios donde se almacena el código fuente de un proyecto específico. Los repositorios pueden ser públicos, y por lo tanto, cualquiera puede ver y contribuir al código; o privados, donde solo los usuarios autorizados pueden acceder a ello y contribuir.

Dirección IP

Una dirección IP (*Internet Protocol*) es una etiqueta asignada a cada dispositivo conectado a la red de Internet, la cual permite la identificación y comunicación entre los mismos sobre la red.

Bibliografía

[1] DLT Labs. OIDC and Blockchain: Most Secured Identity System on Planet. [Online]. Mar. de 2023. URL: https://dltlabs.medium.com/oidc-blockchain-most-secured-identity-system-on-planet-e1bc6608fdb8 (visitado 21-08-2019).

[2] Comisión Europea. Identidad Digital Europea. [Online]. Mar. de 2023. URL: https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_es.

[3] Juan Caballero. Decentralized Identity FAQ. [Online]. URL: https://identity.foundation/faq/.

[4] Davie JM. et al. 0289: The Trust Over IP Stack. [Online]. Nov. de 2019. URL: https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0289-toip-stack.

[5] Leonardo Micheloni. SGVirtual 20.12 - OAuth 2 y OpenID Connect para mortales. [Online]. Feb. de 2021. URL: https://www.youtube.com/watch?v=nNVlewjKQEQ.

[6] auth0. Authorization Code Flow with Proof Key for Code Exchange, PKCE. [Online]. URL: https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow-with-proof-key-for-code-exchange-pkce.

[7] Self-Issued OpenID Provider v2. Standards Track. OpenID Connect, ene. de 2023. URL: https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

[8] Lodderstedt T Mathieu. SSI Orbit E50 - About Podcast Episode. [Online]. Mar. de 2023. URL: https://northernblock.io/podcasts/open-id-4-vc-openid-for-verifiable-credentials/.

[9] Lodderstedt T Mathieu. OpenID4VC: OpenID for Verifiable Credentials — SSI Orbit E50. [Online]. Mar. de 2023. URL: https://www.youtube.com/watch?v=5Azq0i9rNvw.

[10] Kudra A. et al. Credential Comparison Matrix. [Online]. URL: https://docs.google.com/spreadsheets/d/1Z4cYfjbbE-rABcfC-xab8miocKLomivYMUFibOh9BVo/edit#gid=422610777.

[11] Personal identification — ISO-compliant driving licence — Part 5: Mobile driving licence (mDL) application. Standard. International Organization for Standardization, sep. de 2021.

[12] The European Digital Identity Wallet ARF. The Common Union Toolbox for a Coordinated Approach Towards a Eu [Pdf]. 2023. URL: https://www.intesigroup.com/en/wp-content/uploads/sites/4/2023/02/ARF-v1.0.0-final.pdf.

[13] María Ruiz Molina. Self Sovereign Identity Spanish. [Online]. Ago. de 2021. URL: https://github.com/Mashenka-ad/self-sovereign-identity-spanish.

[14] EBSI European Blockchain. EBSI W3C Verifiable Credentials (VCs) and W3C Verifiable Presentations (VPs). [Online]. URL: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=555222155.

[15] The Linux Foundation - Hyperledger Foundation. Hyperledger Ursa. [Online]. 2022. URL: https://www.hyperledger.org/use/ursa.

[16] Council of European Union. Regulation (EU) 2016/679 - General Data Protection Regulation - Artículo 45. https://gdpr-info.eu/art-45-gdpr/. 2016.

[17] Hyperledger White Paper Working Group. An Introduction to Hyperledger. [Pdf]. Jul. de 2018. URL: https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf.

[18] ERC-5484: Consensual Soulbound Tokens. Ethereum Request for Comments. Ethereum, ago. de 2022. URL: https://eips.ethereum.org/EIPS/eip-5484.

[19] Council of European Union. Regulation (EU) 2016/679 - General Data Protection Regulation. https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679. 2016.

[20] Timsit K. Lyons T. Courcelas L. Blockchain and the GDPR, a thematic report prepared by The European Union Blo [Pdf]. 2018. URL: https://www.eublockchainforum.eu/sites/default/files/reports/20181016_report_gdpr.pdf.

[21] Hyperledger Foundation. Hyperledger - Aries Cloud Agent Python. [Online]. Jun. de 2019. URL: https://github.com/hyperledger/aries-cloudagent-python.

[22] lineko Hyperledger Foundation. Hyperledger - Aries Cloud Agent Python - Aries OpenAPI Demo. [Online]. Jun. de 2022. URL: https://github.com/hyperledger/aries-cloudagent-python/blob/main/demo/AriesOpenAPIDemo.md.

[23] Commission Nationale Informatique et Libertés. Solutions for a responsible use of the blockchain in the context of pe [Pdf]. Sep. de 2018. URL: https://www.cnil.fr/sites/default/files/atoms/files/blockchain_en.pdf.

[24] EPRS. Blockchain and the General Data Protection Regulation. Can distributed ledgers be squared with European [Pdf]. 2019. URL: https://www.europarl.europa.eu/RegData/etudes/STUD/2019/634445/EPRS_STU(2019)634445_EN.pdf.

[25] Jefatura del Estado. Ley 34/2002, de 11 de julio. https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758. 2002.

[26] Joo Yeon Park. Chang Soo Sung. "Understanding of blockchain-based identity management system adoption in the public sector". En: Journal of Enterprise Information Management 34.5 (2021). DOI: https://dx.doi.org/10.2139/ssrn.4105763.

[27] ERC-4337: Account Abstraction Using Alt Mempool [DRAFT]. Ethereum Request for Comments. Ethereum, ago. de 2022. URL: https://eips.ethereum.org/EIPS/eip-4337.