Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Proyecto Final

Sistema de Identidad Soberana basada en Blockchain sobre Dispositivos Móviles

Autor del proyecto: Javier Garzo Pérez

Director del proyecto: Violeta Monasterio Bazán

Zaragoza, 11 de septiembre de 2020

universidad SANJORGE
GRUPO SANVALERO

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma

Fecha

11 de septiembre de 2020

Dedicatoria y Agradecimiento

En primer lugar, me gustaría agradecer a mi familia, que me han apoyado todos estos años y me han brindado una educación y estudios de calidad a lo largo de mi vida. Gracias a vosotros he conseguido saber el valor del esfuerzo, que me ha permitido llegar hasta aquí.

A Violeta Montasterio por ayudarme a que este TFG llegara a buen puerto.

A Miguel Angel Barea y Rodrigo Casamayor por aportarme los conocimientos necesarios que me han permitido realizar el proyecto y conocer mejor la tecnología Blockchain.

A todos los profesores de la universidad San Jorge y a todos los amigos que allí he hecho. Gracias a vosotros han sido unos de los mejores años de mi vida, que me han permitido desarrollarme como persona.

Gracias a todos vosotros por hacer esto posible.

Tabla de contenido

Tabla de figuras

Tabla de tablas

Resumen

Con el crecimiento del uso de internet el usuario ha podido ver como sus datos personales se han ido exponiendo cada vez más a múltiples terceros, tanto con las redes sociales como con la creación de una cuenta de cualquier tipo.

El incremento de la importancia que se da a la privacidad de nuestros datos ha venido acompañado de un concepto nuevo llamado identidad soberana autogestionada. Este concepto se basa en que nadie tiene el control de nuestros datos excepto nosotros y en la que nosotros somos responsables de ceder estos datos, los cuales no deberían ser almacenados por un tercero. Con el nacimiento de la tecnología Blockchain ha sido posible la creación de identidades protegidas en la propia red *blockchain* y con un móvil, elemento que todos tenemos en la actualidad, poder ceder estos datos bajo nuestra autorización.

El presente proyecto tiene como objetivo principal analizar las herramientas para la realización de aplicaciones basadas en identidad soberana autogestionada gracias al uso de una red *blockchain* y evaluar la madurez de estas mismas. Como objetivo secundario, se ha realizado un piloto que permita demostrar si es posible crear aplicaciones que soporten la identidad soberana autogestionada. Esto ha separado el proyecto en una fase de investigación y en otra de desarrollo.

Los resultados permiten ver el estado actual de la tecnología Blockchain y el piloto generado, que ha permitido analizar las herramientas *blockchain*. Esta información obtenida ha permitido evaluar la viabilidad de las aplicaciones basadas en identidad soberana autogestionada en la actualidad, además de generar una visión de futuro de estas.

Abstract

The growth of the internet use has caused the realization of its users regarding their personal data and how it has exponentially been exposed to other third party sites, such is the example of social networks or the creation of accounts of any kind.

The increasing importance of the privacy of our data has been accompanied by a new concept, called self-sovereign identity. This concept is based on the fact that no one has control of our data except us and that we are responsible of the transfer of this data, which should not be stored by a third party. The birth of Blockchain technology has made it possible to create protected identities in the blockchain network itself and with a mobile phone, an element that everybody has access nowadays, is able to transfer this data under our authorization.

This project is aimed at analyzing the tools for the development of applications based on self-sovereign identity, thanks to the use of a blockchain network, and to evaluate the quality of said development tools. As a secondary objective, a pilot has been carried out to demonstrate whether it is possible to create applications that can support self-sovereign identity. This has forced to split the project into two different phases: A research phase and a development phase.

The results of this study let us to observe the current state of Blockchain technology and the pilot generated, which has allowed the analysis of blockchain tools. The information obtained has made it possible to evaluate the viability of applications based on self-sovereign identity today, in addition to setting a path for the future of these.

1. Introducción

El interés sobre la tecnología Blockchain no ha hecho más que aumentar desde los últimos años, haciendo que pasara de una tecnología totalmente desconocida y nueva, a verse toda clase de proyectos donde se aplica su uso. El ascenso de esta tecnología se debe a que ha llevado la descentralización e inmutabilidad a un nivel nunca visto anteriormente, debido a las restricciones tecnológicas previas a la tecnología Blockchain. Por eso ha permitido dar soluciones a problemas que hace años no tenían una solución existente o completa. Uno de estos problemas, es la identidad soberana autogestionada (SSI, acrónimo en inglés de *Self-Sovereign Identity*), que hasta el nacimiento de la tecnología Blockchain era imposible de llevar a cabo. Buscando solucionar este antiguo problema nace la idea del proyecto, que consiste en investigar el estado de las herramientas necesarias para la creación de la SSI, así como evaluar si la tecnología Blockchain está lo suficiente madura para llevar esto a implementaciones reales.

El proyecto desde su origen forma parte de una colaboración con una empresa externa, Inycom [1], por lo cual se ha contado con la orientación de un tutor asignado por esta compañía con conocimientos sobre Blockchain y SSI. Pese a ser un proyecto con una compañía, no se presenta como producto de la empresa o la extensión de un producto actual de esta, ya que la idea es que el proyecto sea carácter exploratorio para ver la madurez de la tecnología Blockchain y las herramientas relacionadas con esta.

Es en el año 2008 cuando nace la tecnología Blockchain de la mano de Satoshi Nakamoto, con la creación del Bitcoin, una moneda digital de carácter descentralizado y alojada en la red *blockchain*. Principalmente la tecnología Blockchain ha ido de la mano de las criptodivisas a lo largo de su evolución, hasta ciertamente hace poco, cuando se inicia su uso por parte de las empresas en redes *blockchain* privadas, ya que estos usan sus propios equipos para realizar las pruebas de trabajo, que hacen funcionar la red *blockchain*. Pero siempre manteniendo el carácter de inmutabilidad y en la mayoría de los casos de descentralización, donde no existe una entidad que tenga el control total de los datos.

En cuanto a la identidad soberana autogestionada (SSI), desde la creación y la popularización de internet ha existido el problema de no poder identificarte como una persona real y de la poca privacidad de los datos personales. Con el surgimiento de la tecnología Blockchain han nacido proyectos como Uport, basado en Ethereum, o *frameworks* como Hyperledger Indy, destinados a solucionar estos problemas o prestar herramientas para crear un sistema que permita solventarlos y mantener los datos del usuario fuera de las manos de un tercero.

Por estas razones se ideó el proyecto con un objetivo claro, evaluar las herramientas disponibles en la actualidad que nos ofrece la tecnología Blockchain, para conseguir implementar sistemas basados en SSI y hacer un ejemplo para demostrarlo. Esto llevará a que el proyecto se descomponga en un proceso de investigación del estado actual de la tecnología Blockchain, así como ver intentos previos de SSI y encontrar un caso de uso para poder desarrollarlo y analizar las herramientas usadas.

Así como se indica en los objetivos, el proyecto se inició con esa idea principal en mente, y por lo tanto su primer punto fue una investigación total de los dos temas principales, Blockchain e identidad soberana autogestionada. El proyecto tuvo carácter exploratorio en su totalidad, y se fueron tomando decisiones que ayudaron de darle su forma final durante el desarrollo del mismo.

En cuanto a la estructura de la memoria, la descripción del proyecto comenzará por un estado del arte, el cual contiene la explicación básica de los conceptos principales de la tecnología Blockchain y del SSI, por si no se es conocedor de estos temas previos a este proyecto. Debido a las limitaciones de espacio no se trata de explicaciones en profundidad, por lo que tener conocimientos previos ayudaría a una mayor comprensión de todo el proyecto. El texto incluye referencias para ahondar en los temas presentados. En este apartado también se hablará brevemente del sector de las bibliotecas, que se utilizará como escenario de uso para el piloto, describiendo qué nivel tecnológico llevan en la actualidad, ya que la mayoría ya cuenta con una implementación tecnológica básica.

A continuación, vendrían los objetivos del proyecto, los cuales están reescritos para concordar con la idea que se generó durante las primeras reuniones. Esto se debe a que los objetivos eran muy genéricos, por lo que fue necesario especificarlos y desarrollarlos más en profundidad.

El siguiente capítulo describirá la metodología, una parte en la que se podrá ver la forma de trabajar que se ha seguido durante el proyecto, donde se podrá observar claramente el carácter exploratorio del proyecto durante su desarrollo, así como que no se eligió ninguna metodología inicialmente, pero la forma de trabajar se podría asociar con metodologías ágiles.

A continuación, se tratará el punto de análisis, la parte de proyecto previa a la fase de desarrollo. En este punto se tratan temas como la parte de investigación previa, así como que herramientas se usarán para el desarrollo, haciendo hincapié sobre todo en la relacionada con la red *blockchain*. De igual modo se describirán los casos de usos pensados y por qué fue elegido el caso de las bibliotecas para ser desarrollado en el proyecto.

El punto número seis mostrará las diferentes fases de desarrollo durante el proyecto. Las partes del desarrollo dividirán en las interacciones realizadas, donde se podrán ver las tareas, desarrollo y resultados de cada una. Este punto nos permite ver en más detalle cómo se ha conseguido llegar al resultado final del piloto.

Seguido por un punto de menor tamaño, el económico, donde podrá verse el coste de este tipo de proyectos de investigación si fuera realizado por una empresa real, con trabajadores bajo contrato.

El penúltimo punto corresponderá a los resultados, el cual tratará los resultados finales del análisis realizado con ambas tecnologías, así como los elementos generados a la finalización del proyecto. También tendrá un apartado destinado a ver las desviaciones con respecto la idea inicial.

Para terminar, se podrá observar las conclusiones que se han obtenido a lo largo del proyecto, además de una opinión personal del proyecto, el estado de las tecnologías ligadas a este y si es posible una implementación real del proyecto en la actualidad.

2. Estado del arte

Es fundamental entender en qué momento y cómo surge el nacimiento de tecnología Blockchain, de una forma general, y de la identidad soberana autogestionada. Además es necesario comprender los problemas a los que intentan dar una solución, para poder alcanzar la correcta compresión del proyecto y de los recursos que puede llegar a ofrecernos. Más aún si tenemos en cuenta que al ser unas tecnologías que, aunque han nacido recientemente, no han venido seguidas de ninguna estandarización.

2.1. Identidad soberana autogestionada (SSI).

Desde hace décadas, persiste un problema con la gestión de la información que se almacena sobre una persona, todavía latente y en aumento con la aparición de Internet. Últimamente, con el incremento del uso de las redes sociales, la actividad digital de los usuarios se ha favorecido. Este incremento ha provocado indirectamente la aparición de cuentas falsas, debido a la posibilidad de la creación de diferentes cuentas para los diferentes servicios que se deseen usar, quitando así autenticidad a muchas de estas identidades generadas en Internet y generando desconfianza alrededor de ellas. Asimismo, se ha roto la confidencialidad del usuario, en casos donde los datos de este son recolectados por la compañía que ofrece el servicio gratuito y son usados o vendidos sin ningún tipo de permiso por parte del usuario. Estas prácticas poco éticas producen preocupación en lo que respecta a la privacidad de los usuarios de internet, que no quieren que sus datos sean sustraídos o vendidos.

Estos dos problemas han generado un concepto llamado identidad soberana autogestionada. El SSI es una forma de identidad digital construida sobre la idea de que el usuario tiene control absoluto sobre sus datos y que estos no deben ser manejados o almacenados por un tercero, manteniendo así su privacidad.

SSI plantea que sea el usuario quién pueda decidir qué datos ceder de su identidad a un tercero y cuáles no, conservando esos datos y teniendo la capacidad de reusarlos en diferentes servicios, reduciendo la posibilidad cometer fallos en la introducción de los datos al ser menos las veces que deben ser introducidos de nuevo, siendo siempre los mismos. Pese a ello, sigue existiendo el problema de dónde se almacenan esos datos, ya que siempre se ha necesitado una entidad que los almacene. Esto se ha resuelto con el uso de tecnología Blockchain, que nos permite la creación de una base de datos abierta y descentralizada.

Hasta aquí, la privacidad del usuario está asegurada. Pero, todavía falta resolver el problema sobre la fiabilidad sobre los datos. La solución pasa por confiar en un tercero que actúa como entidad de confianza, permitiendo verificar que los datos sean reales, pudiendo ser el gobierno

para acreditar tus datos del DNI o una compañía para acreditar los años que se han trabajado en ella.

Lo más parecido en la actualidad sería el uso de tu cuenta de Google o Facebook para identificarte ante un tercero. Este servicio permitiría establecer una identidad en una web sin que esta llegara a tener tus datos, pero sabría que son verídicos, es decir que el usuario de Google existe y la cuenta es real, sin necesidad de almacenar los datos en sus servidores, ya que estos serían verificados por el api de Google que los mantendría almacenados en su servidor. Cogiendo este ejemplo previo y cambiando a los servidores de Facebook o Google por la red *blockchain*, tendríamos el mismo resultado salvo con las cualidades de descentralización y privacidad total que se buscan en el SSI.

2.2. Blockchain

2.2.1. Historia

Aunque la tecnología Blockchain es reciente y actualmente se encuentra en desarrollo, el origen de las primeras ideas que permitieron el desarrollo de esta ya se daban en 1991. Cuando dos científicos de investigación Stuart Haber y W. Scott Stornetta [2] diseñaron una solución computacional para aplicar un sello de tiempo a documentos para que no pudieran ser modificados. El sistema usó una cadena de bloques con seguridad criptográfica para almacenar los documentos a los que se les había aplicado el sellado de tiempo, sin embargo, esta era de una forma centralizada por lo que no es lo que entenderíamos por Blockchain en la actualidad. Un año después, en 1992, Dave Bayer junto a los dos autores anteriores incorporan los árboles de Merkles [3] al diseño, lo que mejoró la eficiencia, permitiendo certificar múltiples documentos dentro de un bloque.

Otro punto importante previo a la creación de la Blockchain fueron las pruebas de trabajo, que se dio por parte de Hal Finnel en 2004, creando el sistema RPoW (*reusable Proof of Work*) [4]. El sistema permitía intercambiar tokens de una persona a otra, aunque esta idea no era descentralizada, ya que existía un servidor principal que mantenía un registro del estado de todos los *tokens*. Esto permitía a los usuarios verificar con exactitud y a tiempo real los *tokens* del resto de usuarios, evitando así el problema de doble gasto (gastar más de una vez los mismos *tokens*).

No fue hasta finales de 2008 cuando se creó el primer uso de la tecnología Blockchain, el Bitcoin, con la publicación por Satoshi Nakamoto [5] de un libro blanco que introducía un sistema de *tokens* descentralizado. Basado en las ideas anteriores, aunque esta vez no se usara un sistema de protección con un servidor confiable como era el caso de RPoW. En el Bitcoin la protección contra doble gasto se resolvió usando un protocolo descentralizado *peer-to-peer*, es decir, sin

ningún servidor central, donde las verificaciones eran hechas por los mineros, que realizaban las pruebas de trabajo para confirmar los bloques, que almacenan las transacciones realizadas.

Fue el 3 de enero de 2009, cuando el primer bloque de bitcoin fue minado por Satoshi Nakamoto.

Otro punto importante en la historia de esta tecnología se da cuando en 2013 Vitalik Buterin comenzó el desarrollo de una nueva plataforma de computación distribuida basada en la red *blockchain*, el Ethereum [6]. Esta permitía ejecutar scripts, llamados *smart contracts* (pequeñas instrucciones almacenadas en la red *blockchain* que permiten autoejecutar acciones en la propia red *blockchain*). Este nuevo uso de la red *blockchain* permitió la creación de *DApps* (aplicaciones descentralizadas), que permitían ejecutarse y almacenar sus datos usando la red de Ethereum.

Este no ha sido el único caso de desarrollo de la tecnología Blockchain, ya que, al ser una tecnología abierta, se han producido en los últimos años toda clase de avances en diferentes campos con el uso de esta tecnología, con grandes diferencias de unos proyectos respecto de otros. Algunos proyectos se centran en hacer la red *blockchain* más eficiente, otros que la hacen aún más privada y otros también enfocados en cualquier tipo de actividad, no solo en la ejecución de *DApps* o las criptodivisas.

2.2.2. Funcionamiento

Para explicar el funcionamiento de la tecnología Blockchain usaremos un ejemplo relacionado con SSI. Aunque hay que tener en cuenta que el uso más extendido que se le ha dado en la actualidad a la tecnología Blockchain ha sido las criptomonedas. En ambos casos la red *blockchain* se comportaría de forma muy similar.

Primero de todo empecemos explicando que la red *blockchain* es un sistema descentralizado por lo que no existe una entidad que almacene todas las transacciones como ocurriría en un banco, el caso de *Server-based* de la Figura 1. El modelo que sigue una red *blockchain* sería un *peer-to-peer*, caso de P2P de la Figura 1, es decir un modelo en que existen unos nodos con la misma autoridad y las transacciones se almacenan y verifican por todos ellos.

Esto nos lleva a que todas las operaciones son almacenadas por todos los nodos de la red *blockchain*, estos nodos serian ordenadores que se conectan a la red *blockchain* y se comunican entre si difundiendo información sobre transacciones y bloques.

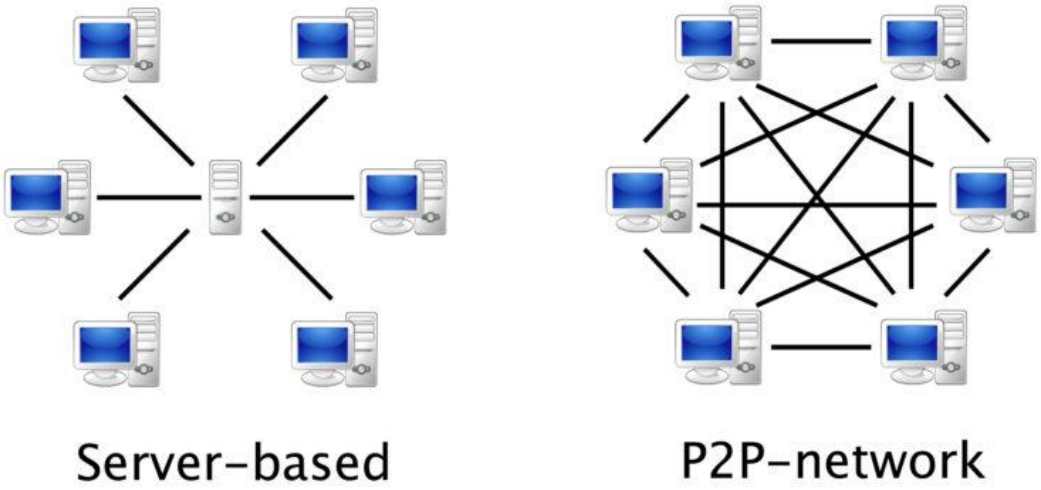

Figura 1 - Sever VS P2P Fuente: https://como-funciona.com/peer-to-peer

Después de entender cómo se distribuye la red *blockchain*, usaremos un ejemplo para entender conceptos de *proof of work* (pruebas de trabajo) y de *blockchain* (cadena de bloques).

El ejemplo está basado en el concepto de SSI, el cual consiste en la existencia de una *app* que funciona como una tarjeta sanitaria basada en SSI. Esta tarjeta sanitaria virtual será llamada IS (identidad sanitaria). El escenario está formado por tres actores:

- Bob: El usuario que posee la *app que* usa como una tarjeta sanitaria, que usará cuando necesita ir a cualquier centro médico.
- El gobierno: Este es el *trust anchor* (el encargado de dar las identidades, en este caso el IS). El IS solo puede ser distribuido por el gobierno en caso contrario será invalido.
- Centro médico: Este solo funciona como comprobador, el cual se conecta a la red *blockchain* para obtener el IS de los usuarios.

El ejemplo disponible en la Figura 2 presenta el funcionamiento de la red *blockchain* del sistema de IS en un conjunto de pasos:

1. Bob realiza la presentación de sus datos al gobierno para recibir la IS.
2. El gobierno confirma todos los datos de Bob y los introduce en un bloque para enviarlo a la red *blockchain*.
3. El bloque se transmite a todos los nodos de la red *blockchain* (los nodos podrían ser ordenadores del propio gobierno).
4. Estos nodos confirman la validez del nuevo bloque (los datos son correctos y el usuario creador es un *trust anchor*).

5. Bob va a un centro médico y le solicitan su IS, por lo que genera con su *app* un *hash* que contiene la dirección de su IS en la red *blockchain*. La ventaja de este sistema es que permite que el *hash* solo dé acceso a los parámetros del IS que Bob permita.
6. El centro médico usa el *hash* proporcionado por Bob para usar los datos visibles (los decididos por Bob) con la seguridad de que estos son datos reales.

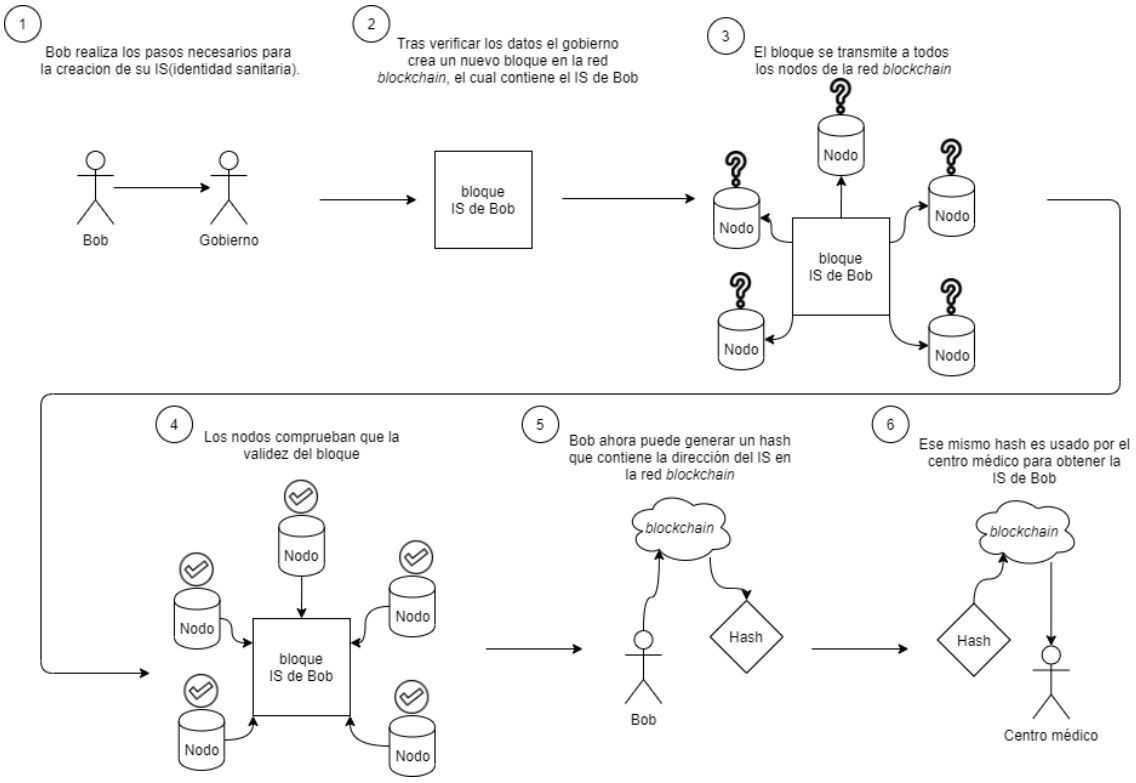

Figura 2 - Ejemplo funcionamiento red blockchain y SSI.

Como se observa en el ejemplo, al generar una nuevo IS el sistema genera un nuevo bloque en la red *blockchain* (en una red real un bloque no estaría formado por una sola identidad, sino por múltiples). Cada bloque generado tiene su propia huella virtual, que está formada por su contenido y la huella virtual del anterior bloque. Este enlace que se produce de los nuevos bloques al anterior forma una cadena de bloques (*blockchain*), término que da nombre a la tecnología. Este caso se puede observar en la Figura 3, donde se ve el enlace que se da entre los bloques del ejemplo.

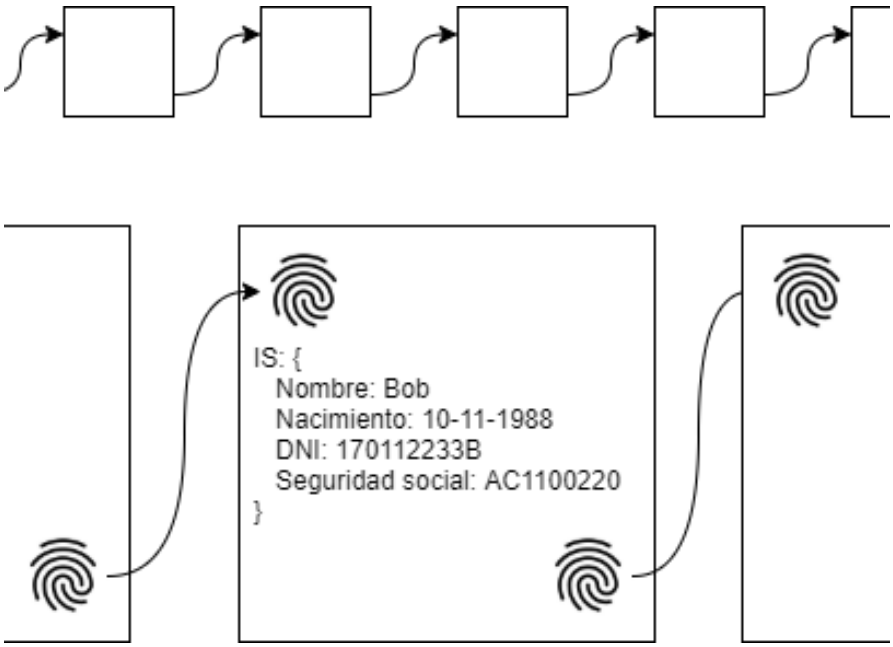

Figura 3 – Funcionamiento de la cadena de bloques, basada en el ejemplo de la Figura 2.

La generación de estas huellas digitales requiere de un conjunto de procesos matemáticos, que permiten realizar el sellado del bloque (realizar huella digital del bloque y no permitir más identidades dentro del este). Este conjunto de operaciones se puede dividir en dos posibles casos según el tipo de red que sea:

- *Proof of Work* (*PoW*) [7]: En estas redes los nodos compiten entre sí para sellar el bloque (normalmente compiten por una recompensa en criptomonedas que se lleva el "ganador"). La competición suele consistir en resolver un problema matemático relacionado con el bloque que conlleva mucho "trabajo" por parte de los nodos, de ahí el nombre *PoW*. Típicamente usada por las criptomonedas (Ejemplos: Bitcoin o Ethereum).
- *Proof of Stack* (*PoS*) [8]: En estas redes los nodos que son capaces de realizar el sellado se establecen por propias reglas de gobernanza en la red *blockchain*. De esta forma no hay competición como las anteriores (ni habitualmente recompensa). Estas son redes son más rápidas y eficientes que las anteriores. Este tipo de red es común en redes blockchain privadas.

Si alguien quisiera cambiar alguna identidad creada en el pasado, necesitaría recalcular todas las huellas digitales de los bloques desde el que deseara modificar, algo computacionalmente muy costoso en el caso de las redes *PoW*. Además, necesitaría controlar el 51% de los nodos para que el cambio se estableciera en la red *blockchain* en ambos tipos de redes. Estas cualidades hacen que la red *blockchain* sea considerada inmutable.

2.2.3. UPort

Uport nace en 2015 como un proyecto de SSI basado en Ethereum, como una *app* móvil [9], con la que la gente tiene el control de su identidad sin ningún tipo de intermediario, donde los usuarios pueden registrar su identidad, verificar y enviar datos relacionas con su identidad.

Uport funciona gracias a los testimonios, que son estructuras de datos firmadas digitalmente por una entidad donde se confirma algo sobre ese usuario, por ejemplo, Andrés (usuario) ha trabajado en Microsoft (entidad firmante).

Estos testimonios conforman la totalidad de la identidad de un usuario, por lo que el usuario tiene el poder absoluto sobre sus testimonios, dado que es él quien puede mostrarlos a quien desee sin que estos puedan ser censurados o modificados, ya que están controlados por el usuario y almacenados en la red de Ethereum. Todo esto permite que la identidad no esté repartida y duplicada en múltiples servicios como podría ser en la vida real, un ejemplo típico serían los datos de la seguridad social y los datos del banco, ambos muy parecidos pero almacenado en distintas entidades.

Por lo tanto, Uport consigue que los datos sean controlados por el usuario y que este sea quien se ocupa de proporcionarlos a quien desee. También ayuda a los proveedores de identidad, debido a que no necesitan almacenar los datos de los usuarios, por lo que no quedan expuestos a ser comprometidos o extraídos de una base de datos.

Uport era una posible herramienta para implementar una solución SSI basada en una red *blockchain*, pero se decidió no utilizar porque habría quedado un trabajo demasiado simple y pese a la dificultad añadida que poseía la siguiente herramienta *blockchain* que veremos, esta tenía unas innovaciones que hicieron que fuera mejor opción.

Figura 4 - Logo Uport. Fuente: https://www.uport.me/

2.2.4. Hyperledger INDY

Hay que señalar que Hyperledger no es una compañía, red *blockchain* o una criptodivisa. Hyperledger es un gran proyecto de carácter abierto y público para desarrollo de la tecnología

Blockchain a nivel industrial y empresarial. En su página web, hyperledger.org, se definen como (traducción propia al español): "Hyperledger es una comunidad de código abierto centrada en el desarrollo de un conjunto de *frameworks* estables, herramientas y librerías de nivel empresarial.

Sirve como hogar neutral para varios *ledger frameworks* distribuidos incluyendo Hyperledger Fabric, Sawtooth, Indy, así como herramientas como Hyperledger Capiler y librerías como Hyperledger Ursa."

Hyperledger fue fundada en diciembre de 2015 por The Linux Foundation, la fundación Linux, que es la fundación tecnológica sin ánimo de lucro famosa por el desarrollo del SO Linux. Hyperledger funciona gracias a que hay equipos y compañías que dirigen recursos al desarrollo de sus proyectos. Aunque en principio no eran muchos miembros, en la actualidad suman más de 250, muchos de ellos destacables tecnológicamente como IBM, Huawei, Nokia, Intel o Samsung, otros también son empresas financieras importantes como American Express o BBVA. Por consiguiente, cuenta con miembros muy importantes en su directiva, formada por 10 ejecutivos con décadas de experiencia en software libre, conocimientos en Blockchain o ingenieros de IBM.

Esta iniciativa está construyendo todo tipo de proyectos relacionada con la tecnología Blockchain a excepción de criptomonedas, ya que, aunque sean el principal uso que se le da a esta tecnología, la dirección de esta compañía está más orientada al desarrollo para su uso industrial o empresarial más que un sustituto para las monedas actuales. El total de diferentes proyectos que se han desarrollado están presentes en Figura 5.

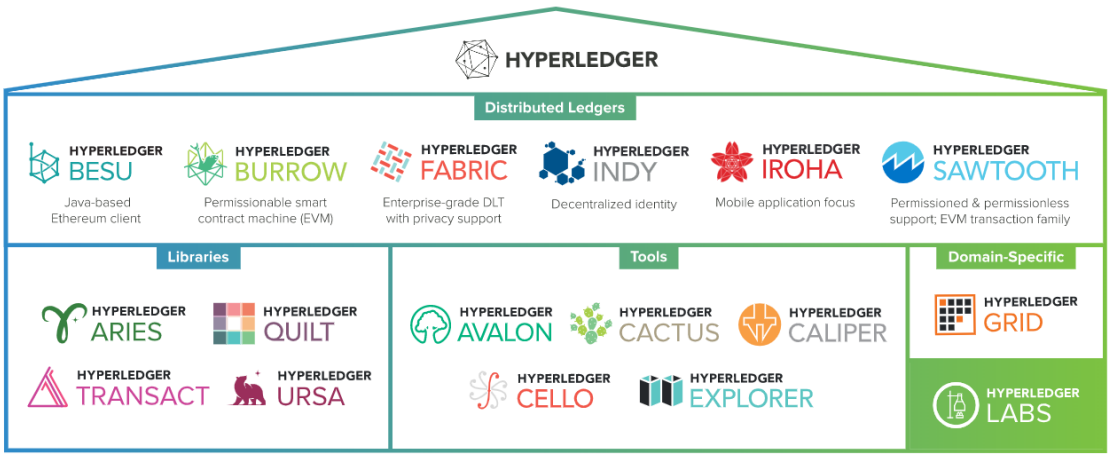

Figura 5 - Usos Hyperledger. Fuente: https://www.hyperledger.org/use

Principalmente Hyperledger destaca por los *distributed ledger*, los cuales son una base de datos distribuida sin ninguna autoridad central, ejecutándose sobre una red *blockchain*. Estos *ledger*

podrían confundirse con una red *blockchain*, sin embargo, estos son el puente con el que interactúa el desarrollador para crear aplicaciones sobre una red *blockchain*. Estos *ledgers* se desarrollan de forma independiente los unos de los otros por lo que difieren entre si según la orientación que tengan, ya que algunos buscan ser genéricos y modulares mientras otros buscan características más específicas. Para comprender un poco en que se centra cada *ledger* a continuación tendríamos un resumen de cada uno:

- Besu [10]: Es un cliente para Ethereum que permite una solución amigable destinada a empresas que deseen crear redes que requieren acceso para poder entrar tanto de tipo pública o privada.
- Burrow [11]: Una distribución completa de red *blockchain*, la cual funciona a través del uso de *smart contracts*. Este *ledger* se centra en la simplicidad, velocidad y comodidad del desarrollador.
- Fabric [12]: Siendo el más popular y liderado por IBM, es una solución para el desarrollo de aplicaciones *blockchain* usando una arquitectura modular, haciendo que soporte una gran cantidad de casos de uso en la industria.
- Iroha [13]: Plataforma que permite una red *blockchain* distribuida modular y fácil de usar con características propias como sus algoritmos de consenso, servicios de pedidos, permisos de roles y soporte multi-firma.
- Sawtooth [14]: Ofrece una arquitectura flexible y modular que permite separar el sistema central del dominio de la aplicación, haciendo que los *smart contract* puedan especificar reglas comerciales sin conocer el diseño del sistema central.
- Indy [15]: Es el elegido para este proyecto porque proporciona herramientas, bibliotecas y componentes reutilizables para proporcionar identidades digitales. El uso correcto de este *ledger* permite el desarrollo de la identidad soberana autogestionada. Este permite usarse tanto de forma individual como de forma interoperable con otras redes *blockchain* permitiendo implementar SSI en otros proyectos de *blockchain*.

La decisión de seleccionar Hyperledger Indy ha venido principalmente por su potencial de desarrollar unas herramientas que permitan usar las redes *blockchain* de forma más eficiente, privadas y sin el uso de criptomonedas (redes tipo *PoS*). El uso de criptomonedas en aplicaciones *blockchain* se ha basado en recompensar a los usuarios con la propia criptomoneda, a cambio estos mantienen la red *blockchain*, lo que es contrario a lo deseado (una red eficiente con pocos nodos), es decir una red *blockchain* privada. Otro punto que también ayudó a seleccionar esta tecnología fue que Hyperledger Indy sigue actualizándose y mejorándose por lo que es una tecnología más reciente y que puede aportar novedades interesantes no muy comunes en el mundo de la tecnología Blockchain.

2.3. Sistema de bibliotecas actuales

En la actualidad el sistema de bibliotecas públicas permite la creación de un carné para acceder a la totalidad de sus servicios.

Se ha observado que en los últimos años las comunidades autónomas han agrupado el carné de socio de las bibliotecas públicas de su comunidad, como en el caso de Aragón en 2018 [16]. Aunque se han producido avances en los últimos años, el problema de la necesidad múltiples carné para diferentes bibliotecas sigue existiendo por lo que sería positivo aportar una solución que además de solventar el problema incluyera SSI.

Por otra parte, de forma global las bibliotecas y librerías también están actualizándose, aplicando una digitalización de su administración para optimizar su funcionamiento. Una web que recoge información sobre esta actualización constante de las bibliotecas es librarytechnology.org [17], que nos da información de como muchas bibliotecas están optando por servicios en la nube y aplicación web para mejorar la administración y reducir costes. Esta actualización ha supuesto a la aparición de software destinado a bibliotecas como es Ex Libris Aleph, Ex Libris Voyager, SirsiDynix Horizon. Sin embargo, este software es centralizado, por lo que queda un nicho en el caso de la identidad soberana autogestionada.

3. Objetivos

Los ejemplos de productos comerciales basados en SSI son escasos. Por esa razón, la empresa Inycom decidió explorar este nicho y por ello propuso el presente PFG. El objetivo principal del proyecto es evaluar las herramientas actuales, así como la tecnología Blockchain para uso en la identidad soberana autogestionada y construir un piloto basado en un caso de uso sencillo como demostración de la viabilidad de estas.

Debido a que los objetivos eran bastante genéricos cuando se redactó la propuesta de proyecto de la empresa, disponible en el anexo página 75, los tres objetivos iniciales se rescribieron al inicio de su realización, para hacerlos más aclaratorios, así como para subdividirlos en unos subobjetivos:

I. Evaluar las herramientas actuales que permitan usar la tecnología Blockchain para el uso de la identidad soberana autogestionada.
   - Investigar y entender la tecnología Blockchain y sus usos principales.
   - Investigar sobre las herramientas o *frameworks* para combinar SSI con una red *blockchain*.
   - Analizar qué posibles herramientas se usarán para la realización del proyecto en el proceso de desarrollo.

II. Definición de unos casos de uso posibles y selección de uno de estos, para poder desarrollarlo en el siguiente punto.
   - Pensar en posibles casos donde implementar SSI basado en Blockchain.
   - Elegir un caso de uso y terminar de definirlo de forma completa.

III. Diseñar y desarrollar el *API REST*, interfaces y aplicaciones para el manejo e integración del caso de uso especificado en el punto anterior y evaluar el resultado.
   - Desarrollar la red *blockchain*.
   - Desarrollar una *API REST*, así como todas las llamadas a esta.
   - Desarrollar una *app* como sistema de interacción para usuario.
   - Evaluar el resultado obtenido con las herramientas usadas.

4. Metodología

Si bien la metodología inicial suele estar clara al principio de empezar el proyecto, en este caso no se definió ninguna en especial en relación con el desarrollo del proyecto por parte de la empresa, sino que esta fue cogiendo forma y como veremos más adelante se podrá ver características propias de metodologías ágiles o incluso asociarla con algunas de las metodologías más populares.

Fue en estas primeras reuniones donde se aclaró el rumbo que llevaría el proyecto, sabiendo los objetivos genéricos que se querían, pero sin saber de forma específica como se llevarían a cabo. Dejando el carácter exploratorio claro y dando libertad al alumno para elegir tanto el caso de uso como qué herramientas usaría.

4.1. Administración del proyecto

En cuanto hablamos de la administración del proyecto hay que entender que existe una dualidad en este, debido a que existieron dos diferentes partes con sus respectivos tutores, una destinada a la memoria y otra en relación con la empresa, esta última será la que se trate en más profundidad en el tema de metodología.

4.1.1. Elaboración de la memoria

La realización de la memoria fue supervisada por la tutora académica que designó la universidad para el desarrollo correcto de la memoria. Se ha seguido el modelo de revisiones periódicas, desarrolladas mediante email y reuniones. Todas las reuniones se registraron como se estableció en el manual (disponibles en el anexo, página).

4.1.2. Gestión del proyecto con la empresa

La gestión destinada a la investigación y desarrollo del proyecto fue acordada con el tutor de la empresa, Miguel Ángel Barea Lázaro. Esta gestión del proyecto es a la que nos referiremos en su totalidad a continuación.

La forma de administrar el proyecto en relación con la empresa se fue definiendo en las primeras reuniones, donde también se acordó que estas reuniones se celebrarían cada dos semanas y en las cuales se tratarían los siguientes puntos:

- Cómo había ido el progreso de los puntos acordados para esa reunión.
- La asignación de nuevas tareas para la siguiente reunión o extender el tiempo de alguna de las actuales en caso de que fuera necesario.

- Acabada la reunión el alumno debía de guardar un registro de esta guardando los puntos más importantes en una plantilla (Tabla 1) para así tener registro del camino que seguía el proyecto. El conjunto de estas reuniones está disponible en el anexo, pagina 76.

Fecha: (dd/mm/yyyy)	Próxima reunión: 2 semanas
Temas tratados: • •	
Objetivos para siguiente reunión: • •	

Tabla 1 - Tabla genérica iteración proyecto.

Fue después de las primeras reuniones, en las que primaba el carácter de investigación, cuando se realizó lo más parecido que sería al *backlog* del proyecto, dado por el desarrollo total de los objetivos terminó de cerrarse cuando se encontró el caso de uso elegido para el proyecto como veremos en el punto cinco.

La forma de administrar el proyecto se basó siempre en la flexibilidad sin dejar nada cerrado desde un primer instante, lo que llevó a que en ciertos momentos se realizaran pequeños ajustes a partes ya realizadas del proyecto, como fue en el caso de la API. Esto también da respuesta a por qué no se creó un *backlog* más especificado, unas iteraciones ya divididas a lo largo del proyecto o un diagrama de Gantt que hubiera estimado la duración de cada parte.

La planificación inicial se determinó con el tutor de la empresa, después de las primeras reuniones. Se decidió que los primeros meses serían destinados a que el alumno realizara una investigación sobre la tecnología Blockchain, así como herramientas o proyectos *blockchain* relacionados con SSI. Después se debería desarrollar un caso de uso para realizar una API con la herramienta *blockchain* elegida.

4.2. Parecidos Extreme Programming

Como se ha explicado en la gestión del proyecto con la empresa, la metodología seguida comparte características propias de metodologías ágiles. Si la comparamos con las más populares, es decir

Scrum o XP (Tabla 2), las diferencias entre ellas nos hacen situar más cerca de Extreme Programming la metodología seguida que de Scrum.

Scrum	**Extreme Programming**
Es una metodología de desarrollo ágil basada en la administración del proyecto.	Es una metodología de desarrollo que está más centrada en la programación o creación del producto.
Los *sprints* tienen una duración de 2 - 4 semanas	Las iteraciones tienen una duración de 1 - 2 semanas
Cada miembro de del equipo trabaja de forma individual.	Los miembros deben seguir unas buenas prácticas, por ejemplo, programan en parejas.
Trata de seguir el orden de prioridades que marca el *Product Owner* en el *sprint backlog*, pero pueden cambiarse el orden si es para mejorar el desarrollo del proyecto.	El equipo de desarrollo sigue estrictamente el orden de prioridades definido al inicio por el cliente.
Una vez terminado un *sprint* las tareas del *backlog* terminadas y que cuenten con la conformidad del *Product Owner* ya no se retocaran.	Las tareas que se van terminando pueden ser modificadas durante el trascurso del proyecto, aunque funcionaran correctamente de acuerdo con lo que se estableció.

Tabla 2 - Scrum VS Extreme Programming

Como se ha aclarado dentro de la introducción del punto de metodología, en ningún momento el proyecto se centró propiamente en el desarrollo de la metodología del proyecto, sino que se centró en el desarrollo de este, conociendo más con la filosofía de XP, donde la metodología se centra en el desarrollo del producto.

En cuanto la duración de las iteraciones si bien son de dos semanas, pudiendo ser abarcadas tanto por Scrum como XP, serían más propias de XP, ya que este aboga por iteraciones cortas como ha sido el caso del proyecto.

Si bien no se ha seguido otras prácticas de XP como son las buenas prácticas, dado que al ser un equipo de una única persona muchas de ellas se consideraban innecesarias o imposibles, como podría ser el caso de *pair programming*.

En cuanto el orden que siguió el proyecto ha sido estrictamente el orden en el que se establecieron los objetivos, como se indica en XP, pero en consecuencia hay que indicar que esto ha llevado a que puntos, ya terminados, hayan tenido que ser modificados para adaptarse de mejor forma a puntos en los que se trabajaba en otros momentos (otra característica que no habría cabido dentro de la filosofía de Scrum). Un ejemplo de esto serían unos pequeños cambios sufridos en la *API REST*, durante el desarrollo de la *app*, puesto que durante el desarrollo de ésta se podía ver que la mejora de estos ahorraba tiempo y arreglaba problemas que surgían con la integración del *API REST* y la *app*.

En conclusión, aunque no se eligió ninguna metodología inicial, la forma en la que se ha administrado el proyecto a lo largo de éste, podría consolidarse como una versión modificada y adaptada al proyecto de la metodología Extreme Programming.

5. Análisis

Como se ha explicado a lo largo de los puntos anteriores, el proyecto no tenía una idea cerrada del todo. Este proyecto en estaba orientado a evaluar la madurez de las herramientas y la propia tecnología Blockchain. Si bien es cierto que desde un principio se sabía que se quería realizar el proyecto en relación con la tecnología Blockchain y el SSI, el proyecto no tenía el caso de uso definido. Además, en el grado de Ingeniería Informática no se cursa una materia relacionada con la tecnología Blockchain, por lo que los conocimientos del alumno sobre esta eran muy superficiales.

Estos problemas ya estaban contemplados a la hora de realizar la propuesta del proyecto, por lo que los dos primeros objetivos de este consisten en investigar y analizar la tecnología Blockchain, así como sus herramientas, y el concepto de SSI para solventar los problemas indicados en el párrafo anterior.

Los siguientes puntos describen la investigación inicial, el análisis realizado sobre las herramientas *blockchain* y los casos de uso que considerados. Esta fase de análisis empezó en la primera reunión con el tutor de empresa destinadas al proyecto con la empresa (disponible en el anexo, página 76) hasta la 4, donde ya se cierra el caso de uso elegido y se establecen objetivos relacionados con la fase de desarrollo para la próxima reunión.

5.1. Investigación

Como se ha visto en el apartado de metodología, los primeros meses se dedicaron a realizar la labor de investigar sobre tecnología Blockchain y SSI.

La fase de investigación inicial consistió en realizar una búsqueda por separado de los dos términos. Por una parte, se empezó por conocer el de SSI y los problemas que buscaba resolver. Por la otra parte, fue necesario entender el funcionamiento de la tecnología Blockchain, investigar sobre su surgimiento reciente y que propiedades había traído consigo misma, que no estaban presentes previa a esta. Hay que destacar que la parte de investigación de la tecnología Blockchain no incluía saber programar una red *blockchain*, esta parte estaba orientada a entender los conceptos de esta tecnología.

Después de conocer los conceptos se inició el proceso de buscar ejemplos que combinaran Blockchain y SSI, donde se encontraron dos proyectos interesantes que nos darían dos caminos diferentes para llevar a cabo la implantación de una red *blockchain* con SSI.

- Uport: Ya explicado en el apartado 2.2.3. Este es un proyecto ya desarrollado que nos da un ejemplo de que se puede obtener combinando las dos tecnologías, así como

también nos da la posibilidad de extenderlo a nuestro propio proyecto. Uport nos da dos posibilidades de cómo actuar con respecto al desarrollo del proyecto:

1. Basado en Uport: Consistiría en basar el proyecto sobre Uport, ya que este ofrece herramientas para extender la creación de testimonios a ámbitos personalizados.
2. *Smart contracts*: Siguiendo el proceso que hace funcionar a Uport, es decir los *smart contracts* que se ejecutan en la red de Ethereum. Este camino consistiría en basar el proyecto en una tecnología que permita desarrollar aplicaciones en una red *blockchain* de forma genérica (Ethereum, Hyperledger Fabric o similares) y enfocarla a la creación de SSI.

- Hyperledger Indy: Explicado en el apartado 2.2.4, es una herramienta para el desarrollo de identidades digitales en una red *blockchain*. Esta opción consistiría en elegir este Indy sobre el resto *ledgers* disponibles en Hyperledger, debido a que es específico para el problema que buscamos solventar, y usarlo para desarrollar la parte de la red *blockchain* del proyecto.

La opción seleccionada fue la de usar Hyperledger Indy sobre las otras opciones, ya que la opción más innovadora y es una herramienta pensada para este tipo de casos. Además de que la opción de usar Uport dejaría el proyecto excesivamente simplificado y usar los *smart contracts* o Hyperledger Fabric no tiene sentido si se dispone de herramientas más específicas como es el caso de Indy.

Una vez seleccionada la herramienta a usar tocaba conocerla para poder extenderla a una REST API que sería usada en el proyecto. Por lo que el primer contacto con el código de Indy (en adelante llamado Indy-sdk, nombre del proyecto en GitHub) fue conocer su documentación [18] y comenzar la instalación, para la cual se usó este articulo [19]. El lenguaje que se usaría para ejecutar el código del proyecto sería Node.js de los disponibles (Java, Python o Node.js), ya que Node.js era una opción conocida y permitía el desarrollo de la API REST con express.js (un *framework* popular para el desarrollo de API REST, con más de 12 millones de descargas en npm [20]).

Después de terminar el proceso de instalación y ejecutar el ejemplo de Alice [21] en Node.js, empezó una fase estudio y comprensión del código hasta el final de estos dos primeros objetivos y el comienzo de la fase de desarrollo del proyecto, por lo que esta fase convivió con la fase de búsqueda de casos de uso. Esta fase supondría entender el código y acciones realizadas en el ejemplo de Alice, donde existen un grupo de entidades que ligan un conjunto de credenciales a Alice como su identificación (DNI), un trabajo o un préstamo (parecido al ejemplo de Bob en el punto 2.2.2).

Una vez terminada la parte de investigación inicial y sabiendo qué herramienta se iba a usar, comenzó la parte de pensar posibles casos de uso, en los cuales se podría usar la tecnología Blockchain para conseguir un ejemplo de SSI.

La idea era pensar un grupo de casos de uso para hablar con el tutor y seleccionar uno que desarrollar para poder enseñar de que es capaz Hyperledger Indy, así como también ver en qué medida es posible llevar el SSI a la realidad. Se pensaron en los siguientes casos para implementar:

- Sistema de titulaciones internacionales: El sistema estaba pensado como una forma de verificar títulos académicos cuando se desconocía su veracidad (por ejemplo, cuando una persona de un país del extranjero posee un título que dice que es Ingeniero Informático en una Universidad con la cual tendrías dificultades para contactar y confirmar este título). El sistema permitiría a las universidades solicitar ser entidades distribuidoras y así poder almacenar la titulación de sus alumnos en una red *blockchain* para su rápida y fácil verificación.

- Reserva de espacios en la universidad: La idea consistía en un sistema que servía para reservar espacios en la universidad basado en SSI, donde el usuario podía ver y realizar reservas leyendo el QR de la sala que deseaba reservar. Por lo que las horas de las aulas se enlazaban a unas identidades soberana autogestionadas. El proyecto requería hacer un sistema de SSI para los usuarios con la posibilidad de reserva de aulas.

- Sistema de carné de biblioteca virtual: Este consistía en ofrecer una solución a las bibliotecas que no tenían un sistema en común para prestar libros con un mismo carné entre ellas. El sistema se centraba en usar una red *blockchain* para almacenar los datos de identidad básicos del usuario, una vez una biblioteca creaba y verificaba su cuenta del usuario. Con estos datos verificados en la red de *blockchain* se podría ir a otra biblioteca y crear la cuenta sin necesidad de verificar de nuevo todos los datos.

El caso del sistema de titulación internacional aun siendo un caso muy básico y fácil de implementar, fue descartado por dos motivos principalmente. Por un lado, teníamos el problema de que era muy parecido a la idea de Uport por lo que no se estaba haciendo nada innovador. Por el otro lado teníamos la dificultad de que el sistema tenía que ser integrado por todas las universidades algo muy poco realista.

El caso de las reservas de espacios también se descartó, en vista que el proyecto estaba más centrado en el sistema de reservas que el propio sistema SSI. Asimismo, el sistema no era en principio muy realista ya que a la universidad no necesitaría la creación de SSI en un sistema donde los datos de los alumnos son únicamente administrados por ellos.

El caso seleccionado fue el del sistema de carné de la biblioteca virtual. Aunque presentara el inconveniente de que los datos del usuario seguían almacenándose en las bibliotecas, la parte del sistema de la red *blockchain*, que permitía extender cuentas a otras bibliotecas, respetaba la filosofía del SSI donde el usuario tiene el control de sus datos y se sabe que estos son válidos. Otro punto a favor era la sencillez de este ejemplo el cual permitía que fuera una implementación más realista que podría marcar un inicio de SSI en algún sector para después extenderse a otros.

Debido a que era la opción más conveniente de implementar en términos de realismo y que estaba bien enfocada en cuanto crear un sistema SSI, la opción del sistema de carné de biblioteca virtual fue escogida. Por lo que fue terminado de definir y se decidió que el sistema debería ofrecer la siguiente funcionalidad:

- Sistema de registro en la red *blockchain*: Cuando un usuario realizara un registro habitual en cualquiera de las bibliotecas del sistema este enlazaría la cuenta a una *app* del móvil, que le serviría posteriormente para realizar las demás acciones.
- Sistema de identificación: El usuario podría identificarse en la biblioteca usando la propia *app*, lo que le permitiría sustituir los múltiples carnés de cada biblioteca por únicamente una *app* en el móvil.
- Extender la cuenta: Los datos de la red *blockchain* son usados para cuando el mismo usuario fuera a otra biblioteca, el cual ya tenía la identidad verificada en la red *blockchain*, este pudiera extender esta identidad a la cuenta de la nueva biblioteca sin tener que volver a realizar el proceso de verificar todos los datos.
- Sistema de multas globales: También se pensó en aportar la funcionalidad de que el usuario confirmara la reserva de los libros, para así poder saber si el usuario debía libros como dato compartido por todas las bibliotecas y de este modo evitar el mismo problema para las otras bibliotecas. Asimismo, este sistema permite evitar que el usuario niegue haber reservado un libro, dado que el usuario confirmaba en un sistema inmutable, como es la *blockchain*, la reserva del libro.

6. Implementación

Una vez se terminó la fase de investigación, comenzó la fase de desarrollo. Esta fase que consistió en el desarrollo del código relacionado con el Indy-sdk y el *API REST*, así como el desarrollo de los sistemas de interacción usuario-biblioteca.

En los siguientes puntos se podrán ver las cuatro grandes iteraciones en las que se han agrupado esta fase de desarrollo, así como las reuniones (disponibles en el anexo, página 76) que correspondieron a cada iteración. Esta fase de desarrollo se inició con la reunión numero 4 cuando se finalizó la fase de análisis.

6.1. Desarrollo de sistema de identidades

Esta es la primera de todas las iteraciones relacionada con el desarrollo del proyecto, iniciada después de la fase de investigación inicial. Esta iteración consistió en el desarrollo del *API REST* inicial con las acciones de identidad básicas. Al ser la primera y partir desde cero fue la más extensa de todas, la cual se inició en la reunión número 4 cuando se acordó empezar a modificar el código de ejemplo de Indy-sdk y empezar a desarrollar la *API REST*. La iteración acabó al final de la reunión numero 8 cuando se presentaron los resultados al tutor y este estuvo de acuerdo con los resultados.

6.1.1. Tareas

La tarea principal para esta iteración consistió en desarrollar el *API REST* inicial y que este fuera capaz de realizar todas las acciones relacionadas con el sistema de identidades. Al terminar la iteración API REST debería poder realizar las siguientes acciones:

- Crear nuevas bibliotecas (usuarios *trust anchor*), las cuales serán capaces de crear usuarios y añadirles una identificación.
- Crear usuarios con una identidad, la cual permitía identificarse en la biblioteca que los crea.
- Extender el usuario a otra biblioteca, generando una cuenta en esta y permitiendo identificarse en ella.
- Abrir y cerrar la *wallets* (cuenta del usuario en el *ledger*), así el usuario tendría control sobre su cuenta. Solo las *wallets* abiertas pueden realizar acciones en el *leldger*.

También se realizaron otras tareas además de la principal:

- La modificación del código de ejemplo inicial haciendo una adaptación al proyecto.

- Investigación sobre maneras de mostrar las acciones que se hacían en la *blockchain* fuera de la propia consola.

6.1.2. Desarrollo

El desarrollo se inició modificando el código de ejemplo de Alice, ya que al empezar de cero era la mejor forma de hacer funcionar un código relacionado con la idea del proyecto. Esta forma de iniciar permitió un desarrollo más fluido que si se hubiera empezado en la propia *API REST* a programar, ya que ejecutar el código entero facilitaba realizar el *debug* y causaba menos problemas.

Al mismo tiempo que se comenzó a modificar el código de ejemplo, se empezó la creación del *API REST*, aunque de manera muy sencilla, permitiendo una llamada de ejemplo para ver que funcionaba de forma correcta.

Esta fase inicial también incluyó la investigación de formas de ver las acciones en el *ledger* sin el uso del comando *console.log* en cada acción en el código. Una opción muy interesante era el concepto de *node monitor,* el cual era un nodo en el *ledger pool* (conjunto de nodos que forman la red *blockchain* del *ledger*) que permitía ver todas las acciones que se realizaban en esta. Sin embargo, un poco de investigación reveló que Indy no lo tenía implementado, cosa que no sucedía en otros *ledgers* de Hyperledger.

Una vez se progresó bastante en la modificación del código de ejemplo y con el API REST en funcionamiento se procedió a empezar a integrar el primero en el segundo. Esta fase empieza con el inicio de la configuración del *pool ledger* y la creación de las credenciales (definir la estructura que tendría la identidad en el *ledger*), esto debería ejecutarse siempre que se iniciase la *API REST*.

Luego se continuaría con la integración, solo que a partir de este punto todas las acciones deberían ir en sus respectivas llamas *API*. La siguiente parte que se añadió fue la creación de las bibliotecas, que eran los usuarios *trust anchor*. Esto permitiría avanzar a la siguiente fase, ya que es necesario los usuarios *trust anchor* para la creación de los usuarios que reciben la identidad (los usuarios de la biblioteca).

El punto de crear los usuarios de la biblioteca era bastante similar al anterior, salvo porque esta vez había que otorgarles una credencial(identidad) al usuario en su creación y que la biblioteca necesitaba abrir su *wallet*. Se estableció que la identidad estuviera formada por el nombre, apellido, DNI, fecha de creación de la identidad. En cuanto el inconveniente de abrir el *wallet* se

crearon unas llamadas del *API REST* sencillas que permitían abrir y cerrar las *wallets,* para este tipo de operaciones que necesitaban una *wallet* abierta.

Todos estos puntos eran fáciles de extraer del código modificado, con la dificultad de que debía funcionar como llamadas a una *API REST* y no de una sola ejecución. La cosa cambia cuando se plantea la fase de identificación y extensión de la cuenta, ambas muy similares, estas requerían recuperar los datos del usuario del *ledger*. La extensión de la cuenta del usuario consistió en realizar el *onboarding* (enlace entre cuenta biblioteca y usuario), como se realiza en el registro, y devolver la identidad del usuario para que la nueva biblioteca realizara una cuenta en su base de datos para este usuario. En cuanto a la fase de identificación consiste en proporcionar el DNI del usuario. Al principio esta fase estaba pensada con que el usuario generaría un QR en su móvil con su DNI bastaría, pero se reflexionó y decidió recoger el DNI del propio *ledger* (usando la *wallet* del usuario), lo que haría la operación totalmente imposible de falsificar.

Esto finalizaría la parte del desarrollo de este punto y como se ve en los párrafos anteriores, este consistió principalmente en el desarrollo del código de Indy y su implementación en el API REST.

6.1.3. Resultado

El resultado principal fue el estimado. Con esta iteración terminada la *API REST* contaba con las siguientes llamadas (Tabla 3):

Operación API REST (origen)	**URL llamada**	**Parámetros de entrada (URL para los GET y Body para los otros tipos)**	**Parámetros de salida**
POST (Servidor central)	/createTA	{nameTA: string stewardWName: string stewardWkey: string stewardDid: string}	{DidNameForUsers: string walletConfig: string walletCredentials: string did: string userIdentitySchemaID: string}
POST (cualquiera)	/openWallet	{walletName: string walletKey: string}	{walletID: number}
POST (cualquiera)	/closeWallet	{walletID: number}	{walletClosed: boolean (siempre true)}
POST (usuario)	/user	{trustAnchorWallet: number trustAnchorDid: string TAwalletUserIdentityCredDefId: string dniUser: string	{walletConfig: string walletCredentials: string userIdentity: { first_name: string last_name: string

		firstName: string lastName: string}	DNI: string creationDate: string}}
GET (biblioteca)	/user	{userWallet: number TADid: string}	{DNI: string}
PUT (biblioteca)	/user	{TAWalletConfig: string TAWalletCredentials: string TADid: string userWallet: number}	{first_name: string last_name: string DNI: string creationDate: string}

Tabla 3 - Llamadas API REST después de la primera iteración.

En cuanto a estas llamadas permitían realizar los siguientes posibles casos de uso:

- Creación de cuenta del usuario (Figura 6): El usuario llega a una biblioteca, la cual tiene este sistema de identidades implantado, y solicita crear una cuenta. Además de crear una cuenta en la base de datos de la biblioteca, se crea también una en la red *blockchain*.
- Identificación usuario (Figura 7): El usuario puede solicitar identificarse con la *app* que mantiene ligada su identidad, si tiene una cuenta en la biblioteca este podrá acceder con la *app*.
- Extensión de cuenta del usuario a una nueva biblioteca (Figura 8): En el caso de que el usuario no tenga cuenta y quiere extender su cuenta a una nueva biblioteca. Consiste en entregar los datos del usuario a la biblioteca para que esta le genere una cuenta en su base de datos.

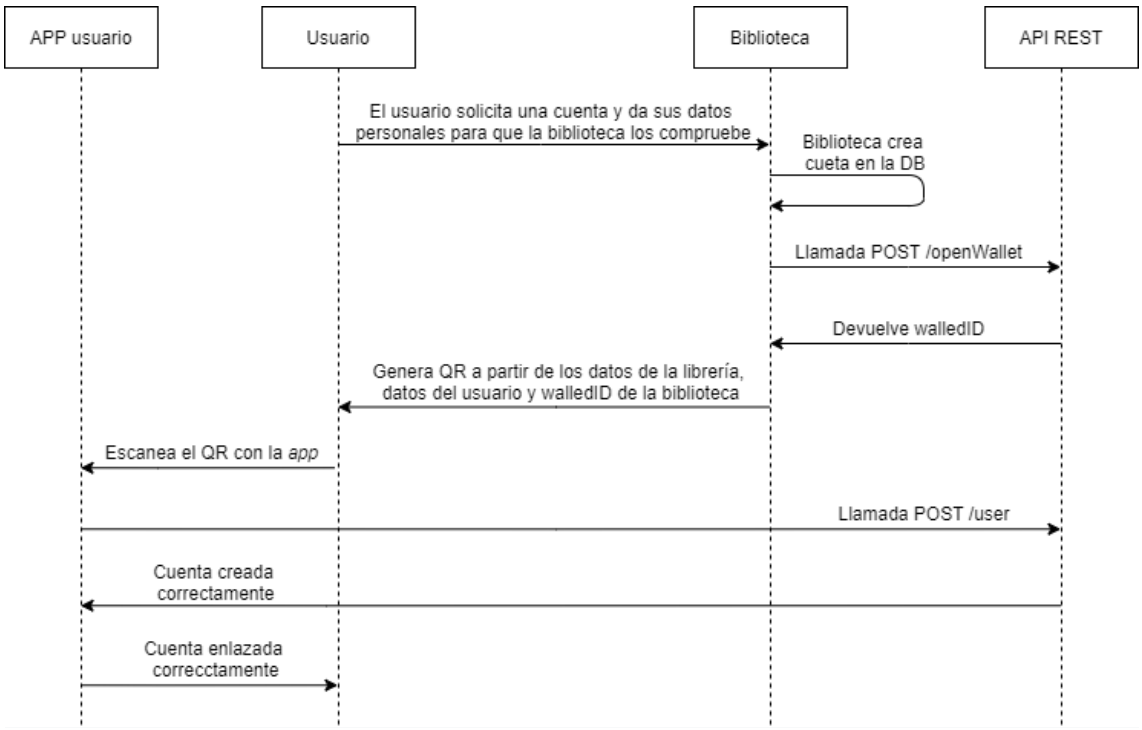

Figura 6 - Diagrama creación cuenta usuario.

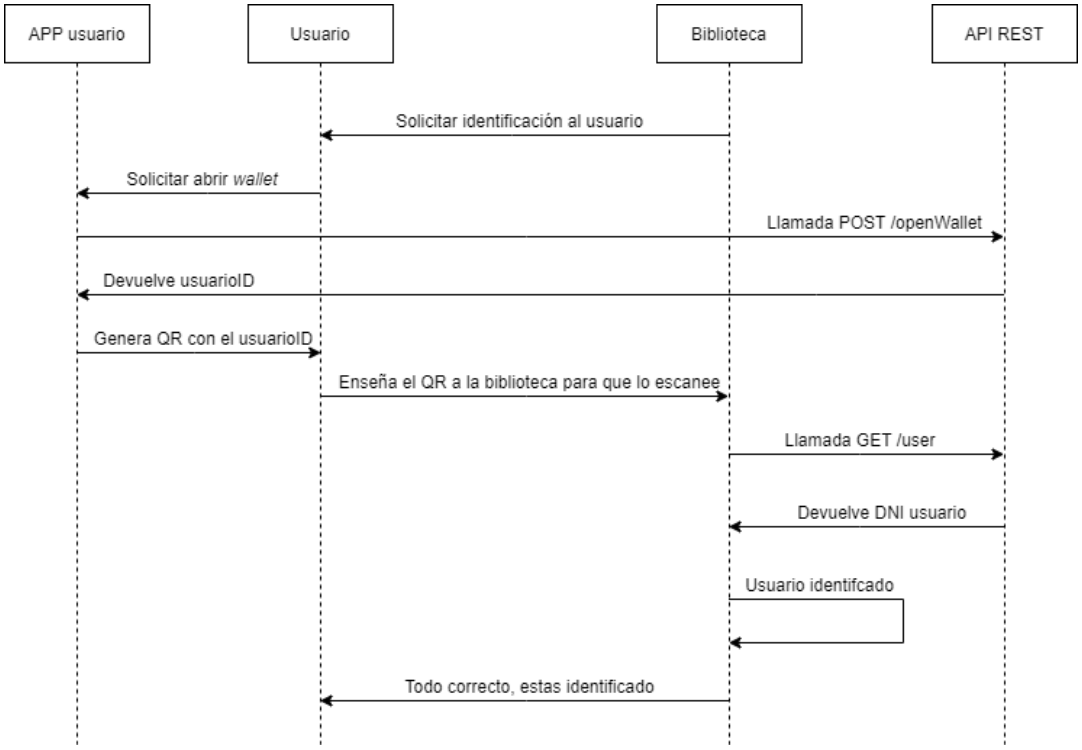

Figura 7 - Diagrama identificación usuario.

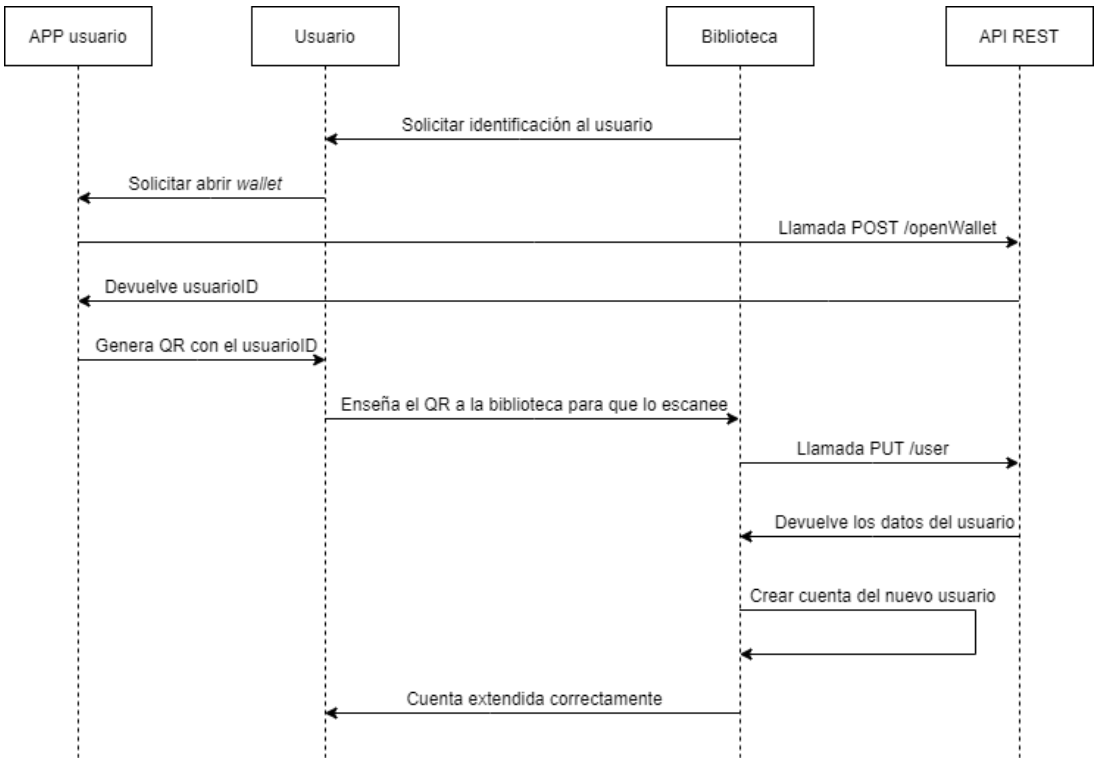

Figura 8 - Diagrama extensión de cuenta de usuario a nueva biblioteca.

En cuanto a las tareas secundarias, la de modificar el código de ejemplo de Alice se reliazó de forma correcta y ayudó al desarrollo del resultado principal. Por otra parte, la investigación para mostrar los procesos que se realizan en el *ledger* de forma externa, no acabó como se esperaba ya que al no existir un *node monitor* para Indy, se decidió mantener el uso de la consola. Esto no significa que el resultado de la investigación fuera fallido, sino que la tarea se realizó de forma correcta, debido a que el motivo era investigar una solución. Al terminar la investigación se estimó que al no poder implementar un *node monitor* no se debería realizar un medio externo para ver las operaciones del *ledger*.

6.2. Desarrollo sistema de multas globales

Esta fue la segunda iteración desarrollada en el proyecto, después de terminar el sistema de identidades. Esta iteración siguió trabajando en la zona del *API REST* por lo que los resultados fueron parecidos a los de la primera iteración (nuevas llamadas *API REST* y casos de uso posibles). Aunque los resultados fueron parecidos, el código que se utilizó fue más avanzado que el del sistema de identidades. Esta iteración se inició al terminar la reunión número 8, cuando se marca como objetivo el inicio del sistema de multas. La iteración se finalizó en la reunión número 11,

donde después de corregir una serie de fallos el sistema de multas globales funciona correctamente y se da por cerrada la parte del Indy-sdk.

6.2.1. Tareas

La tarea consistió en extender estado del *API REST* conseguido en la primera iteración y que este fuera capaz de conseguir una forma de detectar si el usuario poseía libros sin devolver. Este sistema debía ser global y no ser dependiente de cada biblioteca. En el caso de que un usuario no devolviera un libro todas las bibliotecas del sistema serian conscientes y no permitirían reservar nuevos libros a este usuario.

6.2.2. Desarrollo

El desarrollo empezó principalmente con la complicación de cómo implementar el sistema de multas para que fuera global. La solución fue decidida en la reunión número 8, que es cuando se inicia esta iteración, la cual era una solución sugerida por el tutor y que consistía en usar el sistema de eventos de la red *blockhain*. El sistema de eventos consistía en que la red era capaz de realizar acciones de forma automática. En este caso cuando un libro superara una fecha sin ser devuelto cambiar un *flag* ("multa") de la identidad del usuario a verdadero. Sin embargo, después de una investigación inicial, se descubrió que Hyperledger Indy no dispone de esta función, así como si lo hace otros *ledgers* de Hyperledger (Fabric, por ejemplo). La solución decidida por parte del alumno fue la de usar el sistema de revocaciones de Indy para simular reservas de los libros. Esto llevaría a usar un código de Indy-sdk que no se había usado antes, profundizando más en el propio *ledger*.

Por lo tanto, el primer paso fue similar a los realizados en la primera iteración, se creó en la configuración inicial una nueva credencial. Esta nueva credencial consistió en el libro reservado con un grupo de parámetros (nombre, id del libro, fecha reserva y fecha devolución). Esto conllevó realizar una serie de cambios necesarios en la configuración inicial (crear la credencial) y en la creación del *trust anchor* (este necesita inicializar las credenciales en su creación). También se desarrolló una función destinada al enlace de la credencial "libro" con la *wallet* del usuario (igual funcionamiento que la identidad).

Una vez se realizó la parte inicial de enlazar las nuevas identidades con el usuario, la siguiente acción fue la de refactorizarlo. Cuando se presentaron los avances en la reunión número 9, el tutor indicó que una red *blockchain* necesita concentrarse en ser lo más liviana en cuanto cantidad de datos almacenados, revelando que la solución implementada no era la correcta. Como el proyecto no pretende sustituir las bases de datos de las bibliotecas, este no necesita almacenar

todos los datos de los libros. Es por este motivo que la decisión tomada por el alumno fue la de reducir la estructura inicial de la identidad libro únicamente a la fecha de devolución. Esta sería la estructura más eficiente y nos permitiría saber si tiene o no libros devueltos.

Después de la refactorización que se realizó, se produjo el desarrollo del sistema de revocación. Este sistema consistía en cancelar el enlace realizado cuando asignamos una credencial "libro" al usuario. Esto se hace mediante una identificación que se genera al enlazar la credencial con el usuario. Esta identificación nos permite que el *trust anchor* que la crea (la biblioteca que presta el libro) pueda revocar esta identidad más tarde sin permiso del usuario, únicamente conociendo la identificación (la cual guardaría en su base de datos, en los datos de la reserva).

Una vez se acabó el sistema de revocación, se realizó la parte del sistema de verificación. Este consistía en comprobar que el usuario no tenga ninguna credencial "libro" cuya fecha de entrega sea inferior a la fecha actual. Si durante la comprobación encontráramos una credencial que cumpliera esa característica, sabríamos que el usuario tendría un libro no devuelto. Para entender un poco el fruncimiento esta la Figura 9 donde los cuadrados son libros. Según el color del libro este significa una cosa, los grises (libro devuelto), los rojos (no devuelto y fuera de fecha) y el verde (devuelto y dentro de fecha). Si el usuario devuelve los libros el sistema de comprobación detectara que su enlace ha sido cancelado. No obstante, si alguna credencial "libro" tiene una fecha inferior a la actual y no tiene su enlace cancelado, el sistema nos indicaría que el usuario estaría bajo multa. Esta solución se incorpora en el sistema de identificación del usuario, de manera que cada vez que este se identificara en una biblioteca del sistema se sabría si tiene todos los libros devueltos.

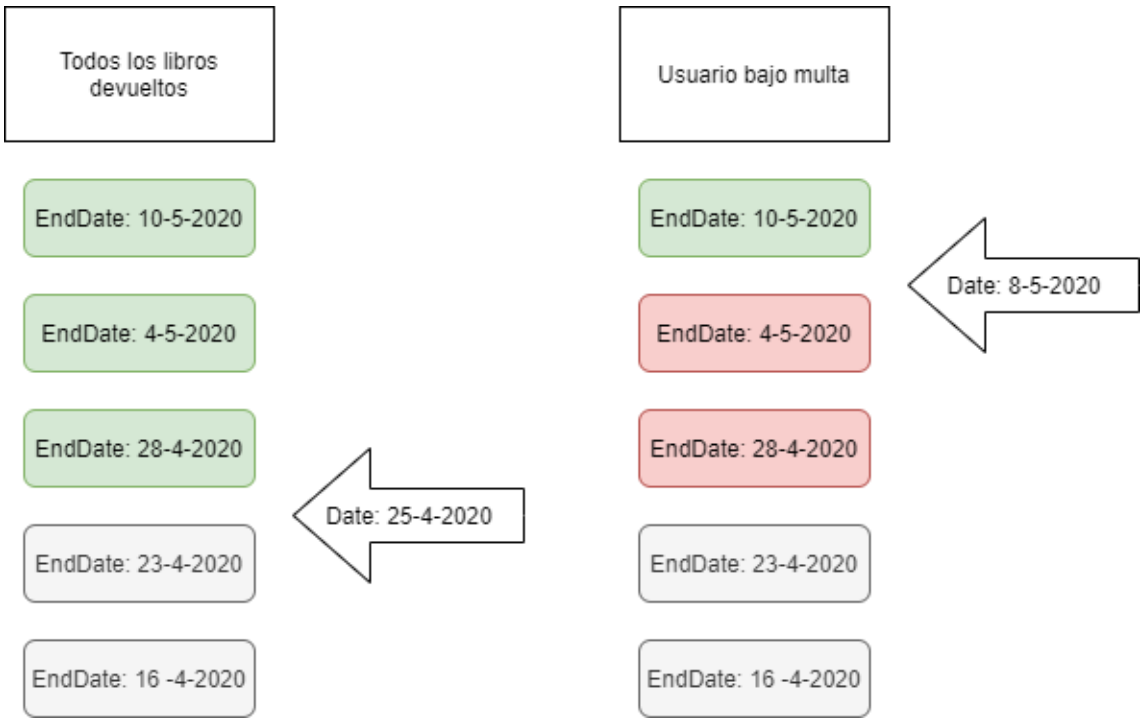

Figura 9 – Funcionamiento sistema de verificación del API REST.

Si bien el sistema contaba con todas las funciones en este punto, un fallo desconocido provocaba que el sistema de verificación siempre devolviera falso y por lo tanto no era capaz de hacer funcionar de forma correcta el sistema de verificación, poniendo en peligro todo el trabajo realizado en la iteración. Este fallo resultó ser un mal uso del *master-secret* del usuario, el cual se podría explicar como una firma secreta que poseía la *wallet* del usuario. Dicho problema resultó muy difícil de resolver, puesto que al partir de código que se ejecutaba de principio a fin este problema no surgía, ya que siempre se usaba el mismo *master-secret*. En cambio, en el código del API REST se generaba un nuevo *master-secret* cada vez que se realizaba una operación diferente, por ejemplo: enlazar una nueva credencial "libro" con un usuario o comprobar si un usuario estaba bajo multa. Este fallo no era detectado por Indy-sdk por lo que todo se ejecutaba de forma correcta y normal, con el problema de que al comprobar si el usuario estaba bajo multa, el nuevo *master-secret* generado no era capaz de comprobar la validez de los enlaces de las credenciales por lo que no sabían se estaba revocados o no.

La solución para solventar este problema consistió en generar un único *master-secret* por usuario, cuando se realizaba la creación de su cuenta. Cuando el usuario necesite realizar una acción y por lo tanto abrir su *wallet*, este enviará su *master-secret* al API REST. En caso de que se realizara una operación se podría usar este *master-secret* guardado y en el caso de que cerrara la *wallet*,

el *master-secret* sería borrado del *API REST*. Debido a eso se requirieron un par de cambios en las funciones de creación de usuario y abrir/cerrar *wallets*.

Este problema supuso un gran quebradero de cabeza, al que se dedicó bastante tiempo para su corrección, pero una vez terminado supuso el fin del desarrollo del *API REST* y el comienzo del sistema de interacción usuario-biblioteca.

6.2.3. Resultado

El resultado fue un sistema que detecta si el usuario tiene algún libro sin devolver de alguna de las bibliotecas. Este sistema permite saber al resto de bibliotecas si este usuario está bajo multa y en tal caso no prestarle ningún libro.

Estos resultados llevaron consigo cambios en las llamadas a la *API REST*, como se ve en la Tabla 4 (los cambios realizados están en verde):

Operación API REST (origen)	**URL llamada**	**Parámetros de entrada (URL para los GET y Body para los otros tipos)**	**Parámetros de salida**
POST (Servidor central)	/createTA	{nameTA: string stewardWName: string stewardWkey: string stewardDid: string}	DidNameForUsers: string walletConfig: string walletCredentials: string did: string CredebtialsOffers: { userIdentitySchemaID string endDateLoanSchemaID: string endDateLoanRevocationId: string}};
POST (cualquiera)	/openWallet	{walletName: string walletKey: string secretID: string}	{walletID: number}
POST (usuario)	/user	{trustAnchorWallet: number trustAnchorDid: string TAwalletUserIdentityCredDefId: string dniUser: string firstName: string lastName: string}	{walletConfig: string walletCredentials: string userSecretId: string userIdentity: { first_name: string last_name: string DNI: string creationDate: string}}
GET	/user	{userWallet: number	{DNI: string

(biblioteca)		TADid: string}	unreturnedBooksOutOfDate: boolean}
POST (biblioteca)	/bookLoan	{TAWalletConfig: string TAWalletCredentials: string TAdid: string didName: string userWallet: number credencialOfferId: string endDateLoanRevocationId: string endDate:string(dd-mm-yyyy)}	{revocationID: number}
DELETE (biblioteca)	/bookLoan	walletConfig: string walletCredentials: string TADid: string revRegfId: string revId: number	{revocationDone: boolean}

Tabla 4 - Llamadas API REST después de la segunda iteración (cambios realizados en verde).

Estas nuevas llamadas afectaron al caso de uso de la identificación, donde la llamada de la API REST también devolvería junto al DNI si el usuario tiene libros fuera de fecha sin devolver. Además, aportó los siguiente nuevos casos de uso:

- Usuario desea reservar un libro (Figura 10): Cuando el usuario está correctamente identificado, este puede solicitar que la biblioteca le preste un libro. Lo que enlazará una credencial "libro" al usuario y la biblioteca guardará el *revocationID* de este para cuando lo necesite para romper este enlace.
- Usuario devuelve un libro (Figura 11): Caso de uso cuando el usuario devuelve el libro que se le había prestado (no hace falta estar identificado, el libro funciona como identificador en la base de datos). La biblioteca usa el *revocationID* para romper el enlace que tenía la credencial con el usuario y confirma en su base de datos que el usuario ha devuelto el libro.

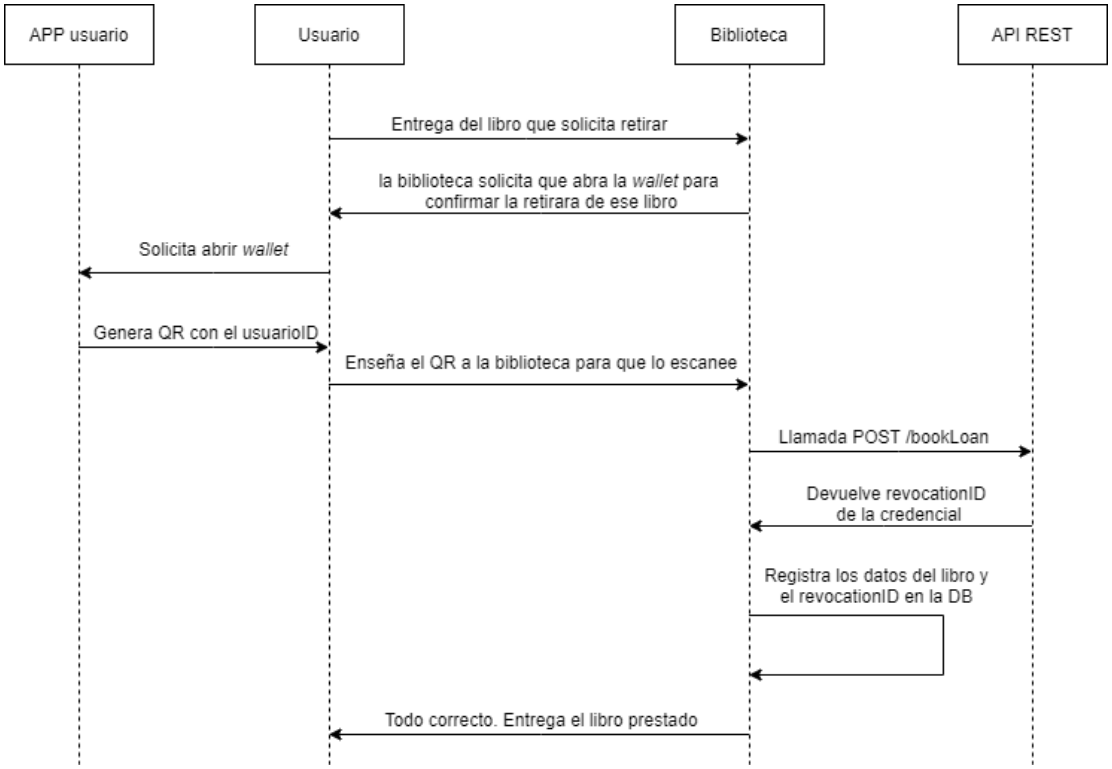

Figura 10 - Diagrama prestación de libros.

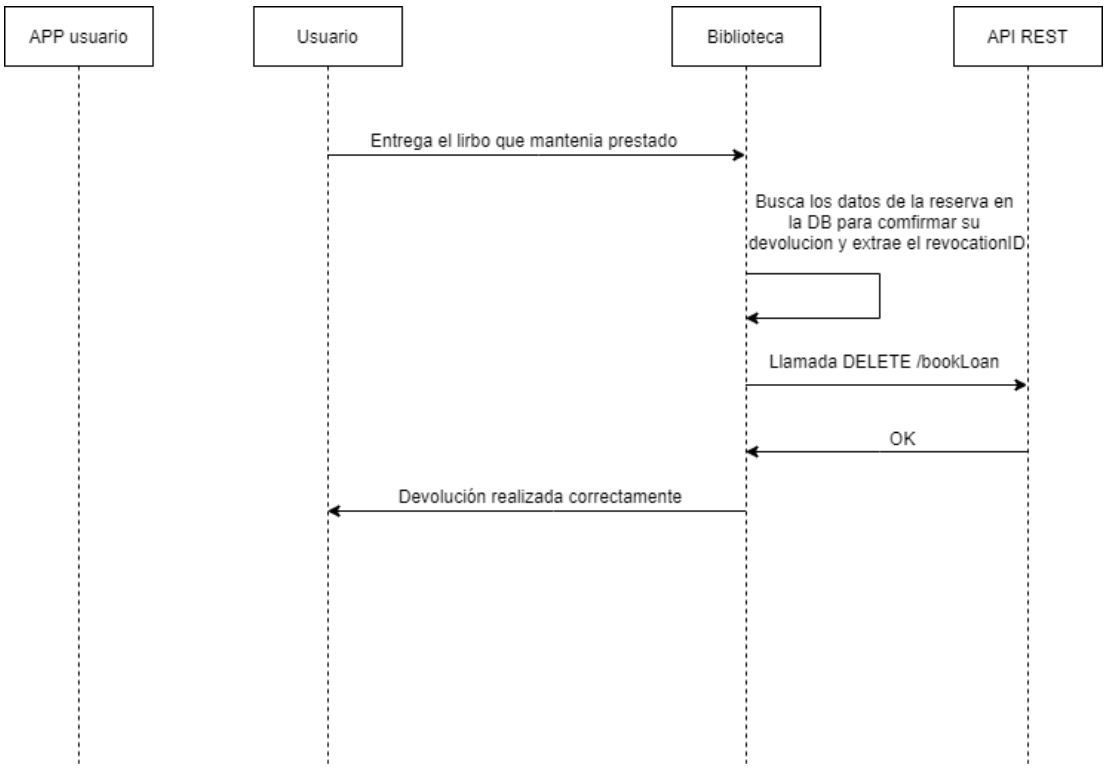

Figura 11 - Diagrama devolver libro prestado.

6.3. Desarrollo de la *app móvil*

Esta iteración fue la primera que se salió del desarrollo propio del Indy-sdk y el *API REST*. Esta se dedicó a desarrollar la parte interactiva del proyecto. En este caso la parte dedicada al usuario y la *app* con la que interactúa. Esta parte se basó en generar una *app* que implementara el uso de todo lo realizado en las iteraciones previas. Donde también se realizaron algunos cambios en la *API REST,* aunque fueron de pequeña índole. Esta iteración comenzó con la reunión número 11, donde se acuerda como objetivo el desarrollo de la *app* del usuario, y dura hasta la reunión número 13, que establece que la *app* esta correctamente terminada.

6.3.1. Tareas

La tarea que se definió para esta iteración fue la de desarrollar una *app*, que funcionaría como sistema de interacción con el sistema y las bibliotecas por parte del usuario. La *app* contaría con las siguientes funcionalidades:

- Sistema de enlace *app* y cuenta de usuario: Primera parte a desarrollar de la *app*, que consiste en enlazar una cuenta creada en la API REST con la propia *app* para que todas las acciones sean realizadas con esta cuenta enlazada.
- Sistema de escaneo de códigos QR: Esta funcionalidad está relacionada con la anterior y se usa como sistema de comunicación con la biblioteca. Si el usuario desea realizar una acción que requiere datos de la biblioteca escaneará el QR generado por esta última.
- Sistema de apertura de cuenta: Esta consiste en usar el sistema de abrir y cerrar *wallets* desarrollado en la *API REST*. Es la forma que tendrá el usuario de dar permiso para la realización de acciones con su cuenta.
- Sistema de generación de QR: Complementario del anterior, consiste en generar un QR que contienen la información de la cuenta abierta, para que la biblioteca realice acciones con esta.

6.3.2. Desarrollo

Para el desarrollo de la *app* se estableció el uso de Android Studio. Se seleccionó este *IDE* de forma conjunta en la reunión ya que era el que iba a permitir el desarrollo más rápido y flexible de la *app*. Puesto que el objetivo era cerrar el sistema de interacción del proyecto, esta era la opción más clara.

El desarrollo de *app* se inició con el sistema de enlace de la cuenta de usuario, que consistía en el uso del sistema de *API REST* para llamar a la creación de la cuenta y que se mantuviera guardada en dispositivo. Para realizar esta acción se realizaron dos actividades diferentes en la

app. Una consistía en el sistema de llamada a la API REST y guardado de datos permanentes en el dispositivo y la segunda consistía en un sistema de escaneo con cámara para códigos QR.

El primero de los dos en terminarse fue el desarrollo del sistema de escaneos de códigos QR, ya que este QR contenía un *json* con todos los datos necesarios para hacer la llamada de la *API RES*T que permitiría la creación de la cuenta del usuario. Para el desarrollo de esta actividad de Android se recurrió a una librería [22] que permitía la lectura de códigos QR para extraer los datos contenidos en este. Se verificaba que los datos eran los esperados y en tal caso, se devolvían estos datos a la actividad de enlace.

La actividad de enlace se terminó de desarrollar una vez se consiguieron extraer los datos del código QR de la actividad de lectura de códigos. Estos datos devueltos se usaban con otra librería [23] que permitía realizar llamadas a una *API* externa, en nuestro caso la *API REST* diseñada en las dos primeras iteraciones. Esto permitía que esta llamada generara una nueva cuenta en el *ledger* que sería usada por el usuario en el resto de las operaciones. Otro punto importante de la actividad de enlace de la cuenta de usuario era el mantener la permanencia de los datos. Para realizar este enlace se recurre a guardar el resultado de la llamada de la *API* hecha al crear la cuenta, ya que devuelve todos los datos relacionados con la cuenta. Estos datos se guardan usando el sistema nativo de "*SharedPreferences*" de Android Studio. El desarrollo de estas dos actividades permitía crear y enlazar una cuenta de usuario al móvil.

Después de esa actividad se desarrolló la principal (la que se ve al iniciar la *app,* si hay cuenta enlazada). Consistía en recuperar parte de los datos guardados en el dispositivo y mostrarlos, intentado simular la estructura de un carné. También debía contener unos botones que nos permitiera abrir la *wallet* para realizar diferentes acciones (identificarse, extender cuenta, confirmar prestación de un libro). Si bien la actividad principal estaba terminada, estos botones que llamarían a abrir la *wallet* del usuario revelaron un fallo de diseño. El fallo se producía debido a que la cuenta no se sabía qué operación permitía el usuario, por lo que se podía desarrollar cualquiera.

Desde la actividad principal se iniciaba una secundaria encargada de abrir la *wallet* con la operación que se deseaba ejecutar. Para abrir la *wallet* se usaba la misma librería encargada de realizar llamadas a una API que se usaba al enlazar la cuenta. Para mostrar los resultados de abrir esta *wallet* (*walletID*), estos eran devueltos por la pantalla a través de un código QR. Por lo que también se necesitó una tercera librería [24] para el desarrollo del generador de códigos QR. Esta actividad permitía cerrar la *wallet* en caso de querer cancelar la operación y debía mostrar el estado de la *wallet*, para una vez terminada la operación y cerrada la *wallet*, indicar el cierre de la operación y volver a la actividad anterior. Estas operaciones realizadas mediante la llamada

a la API mostraron la ausencia de una llamada a la API REST que permitiera obtener el estado de la *wallet* actual del usuario.

Los dos fallos de diseño encontraros provocaron que fuera necesario volver a realizar dos pequeños cambios en la *API REST.* Uno de estos cambios era el de incluir el tipo de operación deseada cuando se abría la *wallet* para que así el usuario tuviera control sobre qué operación se iba a realizar con su cuenta. El otro cambio se basó en la creación de una nueva llamada muy sencilla que únicamente consistía en comprobar si la *wallet* estaba abierta, permitiendo que la actividad que abría la *wallet* inspeccionara cada pocos segundos el estado de esta y así una vez cerrada, indicárselo al usuario y terminar de ejecutar esa actividad.

Una vez terminadas todas las actividades y con los cambios realizados en el *API REST*, el paso final consistió en terminar de darle un refinamiento a la estética de la *app*, ya que el estado que tenía era bastante técnico y poco estético.

6.3.3. Resultado

Primero hablaremos de los cambios que se generaron en las llamadas de la *API REST*:

Operación API REST (origen)	**URL llamada**	**Parámetros de entrada (URL para los GET y Body para los otros tipos)**	**Parámetros de salida**
POST (cualquiera)	/openWallet	{walletName: string walletKey: string operation: number secretID: string}	{walletID: number}
GET (cualquiera)	/walletOpen	{walletID: number}	{walletOpen: boolean}

Tabla 5 - Llamadas API REST después de la tercera iteración (cambios realizados en verde).

El resultado final fue una *app* (Figura 12) que permitía la interacción del usuario con el sistema, la cual tenía las siguientes actividades:

- Actividad cuenta sin asociar: Es la actividad que se inicia en el caso de no tener una cuenta enlazada a la *app*. Esta actividad te explica que necesitas crear una cuenta en cualquier biblioteca del sistema. Dispone de un botón que permite moverse al lector de códigos QR. Cuando vuelve del lector de códigos ejecuta la llamada a la API REST que crea la cuenta y procede a su enlace con la *app.*

- Escáner códigos QR: Consiste en un lector de códigos que permite escanear un código QR, que contiene un *json* generado por la biblioteca. Una vez realizado el correcto escaneo los datos se transmiten a la actividad de cuenta sin asociar, que los usará para crear la cuenta en el *ledger*.
- Principal: La actividad que se ejecuta por defecto cuando tienes una enlazada la cual muestra algunos datos de tu cuenta y permite llamar a la actividad abrir *wallet* con diferentes tipos de operación.
- Abrir *wallet*: Esta actividad es la encargada de abrir la *wallet* y comprobar su estado. Permite al usuario cerrarla en caso de cancelar la operación. También indica cuando la operación se ha realizado correctamente, volviendo a la actividad principal.

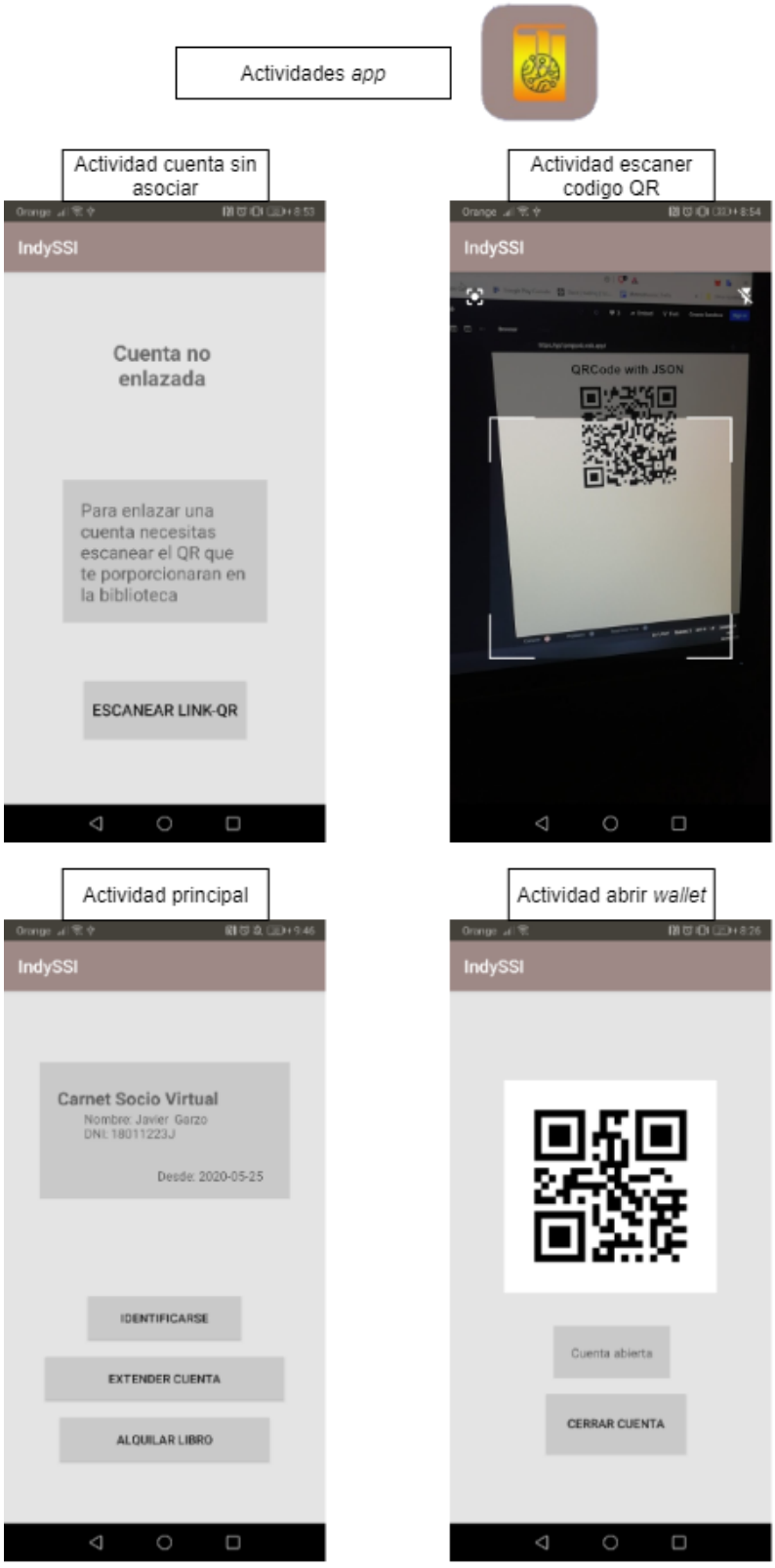

Figura 12 - Actividades de la app.

6.4. Desarrollar la web para la biblioteca

La cuarta y última iteración permitió cerrar el ciclo de la interacción usuario-biblioteca. Esta parte se centraba en crear una web bastante sencilla, que permitía representar las acciones que realizaría la biblioteca. Debido a que desarrollar toda la parte de la biblioteca era muy costoso y tampoco era la parte central del proyecto, la web se desarrollaría de forma bastante simplificada. Esta iteración se inició con la reunión número 13 en la cual se cerró la parte correspondiente a la *app* y se estableció como objetivo el desarrollo de esta web. El proceso de desarrollo de la misma duró hasta la reunión número 14, donde se acordó que el funcionamiento de la web era el correcto y por lo tanto la finalización de la fase de desarrollo del proyecto.

6.4.1. Tareas

El objetivo para esta iteración se fundamentó en desarrollar una web de manera sencilla que permitiera representar las acciones de la biblioteca mediante las siguientes funciones:

- Escáner de código QR: Usado para leer los códigos QR generados por el usuario. Permite usar los códigos QR proporcionados por el usuario para diferentes operaciones (identificar usuario, extender usuario y confirma reserva de libro).
- Generador de código QR: Implementar un generador de QR a partir de un *json*. Este se usará para pasar de los datos de un formulario a un QR que permite enlazar la *app* del usuario.
- Realizar llamadas API: La forma de interaccionar con la API REST creada en las dos primeras iteraciones. Esta parte es necesaria para realizar todas las operaciones relacionadas con la red *blockchain*.
- Simulación DB: Una forma de simular y hacer visibles los datos de la biblioteca que corresponderían a su base de datos.

6.4.2. Desarrollo

La creación de la web se inició con el desarrollo general de la interfaz de esta, previo a la realización todas las funcionalidades destacadas en las tareas. Esta interfaz consistía en dos columnas verticales, una destinada a simular la web que usuaria la biblioteca y la otra destinada simular la base de datos que tendría la biblioteca. Para ser más específicos, la columna de la biblioteca contendría lo que serían dos tablas de la base datos simuladas (usuarios y libros prestados). La columna con la que interactúa el bibliotecario contendría el lector y generador de códigos QR, así como una interfaz sencilla con un par de botones para elegir las acciones que deseara realizar. Este tipo de interfaz nos permitiría tener una visualización total de lo que ocurre en la parte de la biblioteca.

El siguiente paso después de definir la interfaz básica fue desarrollar todo lo necesario para realizar la primera operación que podría hacer una biblioteca: crear la cuenta de un usuario y enlazarla con su móvil. Por lo que sería necesario el desarrollo del generador QR, la base de datos simulada y las llamadas a la API REST.

El desarrollo de generador QR consistió en el uso de una librería [25] que permitía generar un QR. Este QR contendría un *json*, que poseía los datos necesarios para generar y enlazar la cuenta de la red *blockchain* en la *app* del usuario. Este QR se generaría cuando la biblioteca creara el usuario en su base de datos simulada, siendo analizado por el usuario y así terminando el proceso de generación de un nuevo usuario.

Respecto a la base de datos simulada, contaba con dos tablas; una destinada a los usuarios creados en esa biblioteca o extendidos a la misma y otra usada para almacenar los préstamos de libros que se hacen en la biblioteca. La base de datos simuladas permitía insertar datos en las tablas, guardando estos datos en un *array* de *Javascript* y mostrándolos en las tablas *Html* de la página web. También se crearon otras funciones destinadas a eliminar los libros de la tabla de los libros reservados y extraer el *revocationID* de estos libros (usado en el sistema de multas).

Respecto a las llamadas a las *API REST* se realizaron dos funciones que permitían realizar llamadas, la primera destinada a llamadas tipo GET y la segunda destinada llamadas de tipo POST, PUT, DELETE. Esta diferencia en dos funciones radicaba en que las llamadas de tipo GET usaban los parámetros contenidos en su URL para interactuar con el API REST y los otros tipos de llamadas usaban el *body* para transmitir los datos necesarios en la llamada. Finalizado este último proceso se podía realizar llamadas a la API REST para realizar todas las posibles operaciones (abrir *wallet* biblioteca, identificar usuario, extender usuario o reservar/devolver libros).

La siguiente parte consistió en agrupar estos tres elementos desarrollados. Se realizaron algunos cambios en los mismos, así como en la interfaz. Concluir la operación de creación y enlace de usuario permitió confirmar el correcto funcionamiento de las tres partes desarrolladas, casi en su totalidad, ya que algunas funcionalidades se probarían más adelante con el desarrollo de otras operaciones.

A continuación, se procedió a desarrollar la última funcionalidad pendiente, el escáner de códigos QR. Este posibilitaba escanear el código QR proporcionado por el usuario que indicaría el id de su *wallet* abierta para poder realizar operaciones. Para el desarrollo de esta se recurrió a una librería [26] que permitía usar la *webcam* del ordenador para reconocer y leer un código QR. El

lector de QR permitiría ver en la web lo que ve la cámara, en la sección de escáner QR, haciendo así más fácil leer el código QR del usuario.

Este último paso posibilitó terminar las funcionalidades previstas en la tarea de la iteración. Sin embargo, quedaba por acabar las operaciones relacionadas con el escáner del QR generado por el usuario (identificación, extender usuario y prestar/devolver libros). Estas operaciones llevaron a la extensión de la interfaz de la web y algunos arreglos en el código de las funcionalidades previamente desarrolladas. Aunque eran varias operaciones, la dificultad de realizarlas se basó principalmente en la primera, ya que eran todas muy parecidas entre si (todas se basan en leer el QR del usuario, realizar la llamada a la *API* y mostrar un *feedback* por pantalla).

La web se pudo dar por acabada en la última reunión realizada (la numero 14), dado que todas las funcionalidades estaban realizadas y la web soportaba todas las operaciones. Esto cerraba el piloto del proyecto con la finalización de la interacción destinada a la biblioteca.

6.4.3. Resultado

El resultado que se obtuvo en esta última iteración fue una web, como se puede ver en la Figura 13, la cual servía para finalizar un piloto del caso de uso seleccionado. Esta web era muy básica puesto que simplemente se creó para cerrar la interacción usuario-biblioteca. Hay que indicar que la web no mantiene los datos persistentes, debido a su sencillez.

La *web* está formada como se explica en el apartado de desarrollo por dos columnas. La primera columna, "web de la biblioteca", será la interfaz con la que interactuara el bibliotecario y también es donde se generaran los códigos QR (Figura 14) o se verá la vista de la *webcam* usada por el escáner de códigos QR (Figura 15). En la segunda columna, "Simulación de la base de datos", se podrán observar los datos que corresponderían a la base de datos de la biblioteca (tabla de usuarios y libros prestados).

Respecto a las características obtenidas en la web:

- Escáner de código QR: Como se muestra en la Figura 15 podemos ver un ejemplo de cómo sería el escaneo de un código QR de la *app* del usuario.
- Generador de código QR: Se pude ver el resultado en la Figura 14 donde se genera el QR al crear un nuevo usuario.
- Realizar llamadas API: Encargadas de realizar la interacción con la red *blockchain*, no se observa en las figuras. No obstante, sin su correcto funcionamiento no sería posible realizar las operaciones en la web.

- Simulación DB: Una base de datos que se inicia vacía como se muestra en la Figura 13 y puede almacenar datos, quedando el resultado de la Figura 14 y Figura 15.
- Desarrollo web: Esta funcionalidad va intrínseca en las propias tareas, pero es conveniente indicarla. Hace referencia a el diseño propio de la web, los estilos aplicados y todos los botones que permiten navegar al bibliotecario dentro de esta.

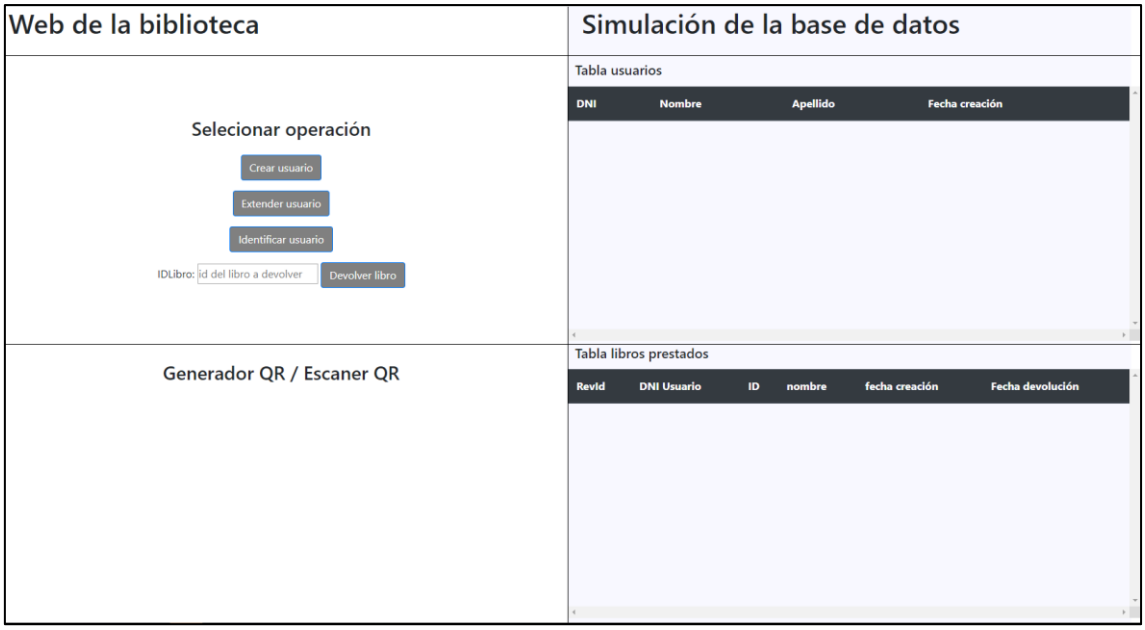

Figura 13 - Pagina web de la biblioteca, estado de inicio.

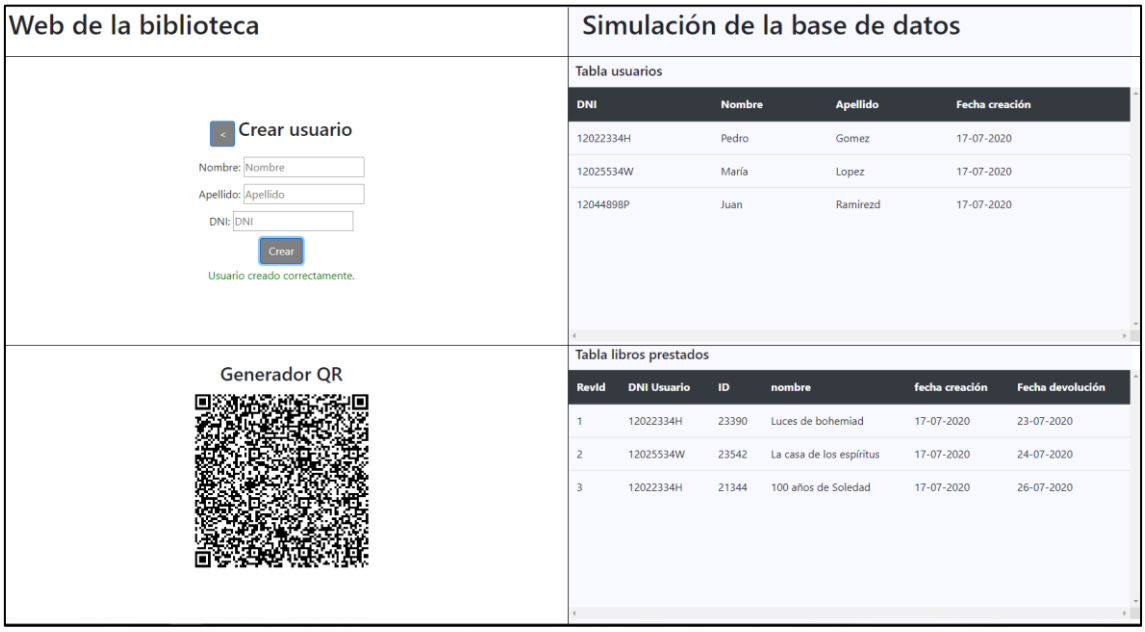

Figura 14 - Pagina web de la biblioteca, ejemplo de QR generado al generar un usuario.

Figura 15 - Pagina web de la biblioteca, ejemplo escáner de QR generado por la app del usuario.

7. Estudio económico

Esta sección agrupa todo lo relacionado con el coste económico del proyecto, entendiendo como escenario el coste total que habría supuesto en el caso de ser desarrollado por una empresa, en este caso la que colaboró con el proyecto (Inycom).

La sección se centra en el apartado de costes del proyecto, debido a que es una investigación. Este se centra en el análisis y exploración del estado actual de las herramientas *blockchain* para el desarrollo de aplicaciones SSI. Aunque después, los conocimientos adquiridos durante el desarrollo de este podrían usarse para el desarrollo de aplicaciones orientadas a nicho con una competencia casi nula. Esta posible monetización sería cargo de la empresa y ajena a este proyecto, el cual se centra en la investigación y desarrollo de un piloto.

7.1. Coste recursos humanos

El primer aspecto a tener en cuenta sería el coste en recursos humanos. Calcularemos los costes por parte del personal que trabajó en el proyecto en función de todas las horas realizadas, como se observa en la Tabla 6. Se ha creído conveniente no incluir el tiempo invertido en realizar la memoria del proyecto, ya que en un proyecto empresarial el tiempo y tipo de documento habrían sido diferentes. En cuanto al coste por parte del alumno en desarrollar el proyecto, se ha calculado como el coste de un programador junior [27] (Tabla 7), en este también se incluirían tareas como el diseño del software o la organización del proyecto. También se ha tenido en cuenta que las reuniones realizadas conllevaban recursos humanos extras, en este caso un sueldo extra de un programado senior [28] (Tabla 7). Los costes humanos se han resumido en la Tabla 8.

Tarea	Horas de coste aproximadas
Reuniones realizadas	24
Fase de análisis	39
Fase de desarrollo	165
Total	228

Tabla 6 - Tabla de horas de trabajo del proyecto.

Se han calculado que las horas realizadas por un trabajador son de 1750, según la media de los convenios colectivos de España [29]. Estas horas son usadas a la hora de extraer el sueldo / hora a partir del sueldo / año en la Tabla 7.

Empleo	Sueldo / año	Sueldo / hora	Coste empresa / hora (30% SS)
Programador junior	18242€/m	10,42€/h	13,55€/h
Programador senior	30757€/m	17,76€/h	22,85€/h

Tabla 7 - Salarios programador en España.

Tarea	Horas	Coste empresa/hora	Total
Desarrollo del proyecto	228	13,55€/h	3089,40€
Coste extra de reuniones	24	22,85€/h	548,40€
		Total	3637,80€

Tabla 8 - Costes humanos del proyecto.

7.2. Coste software

Respecto a los gastos de software, estos han sido nulos, dado que todas las herramientas usadas eran gratuitas (Android Studio y Indy-sdk). En cuanto a la aplicación web estaba ejecutándose en el propio ordenador, lo que supuso también cero gastos de mantenimiento de un servidor. Por lo que respeta a al SO de Linux donde se realizó la parte del desarrollo de Indy-sdk fue realizado mediante un *dual-boot* por lo que no se necesitó un servidor externo, una posible solución.

7.3. Coste hardware

Se ha calculado el coste mensual de su uso de los diferentes dispositivos para valorar cuánto dinero se ha gastado de forma indirecta en relación con el *hardware*. Para ello se ha aproximado que las 228 horas serian igual a un mes y medio de desarrollo en un trabajo de jornada completa. Todo recogido en la Tabla 9, el coste mensual del *hardware* usado. Se ha considerado que solo fue necesario un único ordenador con *dual-boot* para el desarrollo, aunque se necesiten dos para la demo.

Hardware	**Coste compra**	**Tiempo de vida**	**Coste de un mes y medio de uso**
Ordenador y periféricos	700€	5 años	17,50€
Móvil Android gama media	250€	4 años	7,80€
		Total	25,30€

Tabla 9 - Costes del proyecto relacionados con el hardware.

7.4. Costes totales

Referente a los costes totales estarían basados en la suma de los costes previos, resumido en la Tabla 10. Como se puede observar la gran mayoría de costes han tenido relación con los recursos humanos, lo común en proyectos de software informático. También hay que indicar que no se han tenido en cuenta los costes indirectos (alquiler oficinas, internet, etc). Puesto que al ser una empresa grande donde se comparte oficinas, internet y servicios entre todos sus trabajadores hace difícil estimar cuando porcentaje del coste total seria en relación con este proyecto.

Tipo de coste	**Coste monetario**
Recursos humanos	3637,80€
Software	0€
Hardware	25,30€
Total	3663,10€

Tabla 10 - Coste totales del proyecto.

8. Resultados

En los siguientes puntos, veremos el resultado obtenido durante el desarrollo del proyecto. Hay que recordar que el objetivo principal del proyecto era el de analizar la madurez de las herramientas *blockchain* para el desarrollo de aplicaciones basadas en SSI. Por tanto, los resultados se podrían dividir en dos grupos:

- Conocimientos y evaluación sobre las herramientas *blockchain*: Abarca los conocimientos obtenidos durante la fase de análisis y el uso de Indy-sdk en la fase de desarrollo. Estos nos permiten analizar el estado de las herramientas *blockchain*.
- El piloto y su funcionamiento: El resultado obtenido al terminar el desarrollo del caso de uso elegido nos sirve como demostración de que es posible en la actualidad la creación de aplicaciones basadas en SSI.

Además de los resultados finales de las tareas indicadas en los objetivos, también se incluirá un análisis sobre las diferencias finales conforme la planificación inicial, principalmente relacionadas con la distribución del trabajo.

8.1. Análisis herramientas *blockchain*

Este punto tratará de un análisis del estado actual de la tecnología Blockchain y sus herramientas para poder realizar aplicaciones basadas en SSI. Sin embargo, la conclusión de si se debería desarrollar estas aplicaciones o qué futuro le espera al SSI y a la tecnología Blockchain será visto en el apartado de conclusión.

En la fase de análisis se encontraron multitud de herramientas flexibles, que permitían producir todo tipo de aplicaciones para la tecnología Blockchain, tanto para redes públicas (los *smart contracts*) o privadas (Hyperledger Fabric). Aunque no se profundizó mucho en estos elementos, quedó claro que el desarrollo de todo tipo de aplicaciones en una red *blockchain* es posible. Pero las ventajas e inconvenientes de esta tecnología se pueden ver más claramente cuando exploramos más en profundidad el uso de la misma y en nuestro caso cuando se desarrolla un pequeño piloto.

El proyecto se orientó más específicamente en usar el *ledger* que ofrecía Hyperledger destinado al SSI. Se pudo analizar en profundidad la idea de Hyperledger Indy, así como su parte más practica (Indy-sdk). Los puntos fuertes para destacar durante el uso del Indy-sdk fueron:

- Nuevas posibilidades tecnológicas: Este punto más referido a toda la tecnología Blockchain, pero con el uso de Indy-sdk y SSI se ve hecho realidad. Proyectos e ideas

que eran imposibles hace unos años se han abierto hueco gracias a la posibilidad de inmutabilidad y descentralización que nos ofrece la tecnología Blockchain.

- Herramientas de calidad y *open source*: El concepto que está siguiendo el desarrollo de la tecnología Blockchain actual es el de generar herramientas *open source*, por lo tanto gratuitas y de código visible, y en su mayoría con una calidad suficiente para el desarrollo correcto de aplicaciones. El ejemplo más cercano a este proyecto ha sido el conjunto de *ledgers* ofrecidos por Hyperledger, más en específico Hyperledger Indy.
- Múltiples lenguajes: Si bien esto no se dará en todas las herramientas, en Indy-sdk se tenía la posibilidad de elegir entre diferentes lenguajes de programación (Java, Node.js o Python), incluyendo los ejemplos en los propios lenguajes [30]. Esta ventaja hace que los desarrolladores puedan elegir un lenguaje conocido facilitando el primer contacto. En el caso de este proyecto, este hecho también permitió el uso de express.js (librería API REST) con el propio Indy-sdk.
- Constante desarrollo: Visible en el propio GitHub del Indy-sdk [18]. Se ve como la herramienta está en constante desarrollo, permitiendo que esta queda más refinada y adquiera más cualidades con el tiempo.

Pese a existir estas ventajas, el desarrollo del proyecto dejó claro un conjunto de problemas que siguen latentes en muchas herramientas de esta tecnología:

- Herramientas de nicho: El reducido uso de las herramientas *blockchain* hace que en muchos casos sea difícil encontrar soluciones a problemas que encuentras durante el desarrollo. Esto se traduce en que muchas veces, al buscar el error que sucede no exista algún *post* de alguien a quien ya le ha sucedido o que cuando realices una pregunta en foros populares como podría ser "stack overflow" no encuentres respuesta (me sucedió durante el desarrollo del proyecto [31]). Si bien esto no se aplicará a absolutamente todas las herramientas, por ejemplo Hyperledger Fabric [32] tiene 5125 preguntas realizas en "stack overflow" frente a las 75 de Hyperledger Indy [33].
- El problema del "TODO": Este problema hace referencia al famoso comentario que se refiere a código pendiente de hacer ("TODO" en inglés). Es el doble filo que existe en los proyectos *open source*, así como se han indicado sus ventajas arribas, ser *open source* también hace común este inconveniente. En el caso de Indy-sdk algunos ejemplos contenían funciones con este problema, aunque se podrían solventar y completar con el resto de los ejemplos, esto podía llevar a errores difíciles de solucionar.

8.2. Los tres elementos del piloto

Durante el desarrollo del caso de uso seleccionado, una solución para agrupar carnés de diferentes bibliotecas basado en SSI, ha generado tres diferentes elementos:

- *API REST* y red *blockchain*: Correspondiente al trabajo realizado durante las dos primeras iteraciones del proyecto, las cuales habrán ocupado un 75% de la fase de desarrollo. Esta API REST permite recibir llamadas y ejecutar toda la lógica de Indy-sdk, que posibilita realizar y registrar todas las acciones realizadas en la red *blockchain*. El conjunto de operaciones que soporta este API REST son: Abrir y cerrar *wallets*, crear/identificar/extender usuarios en las bibliotecas y el sistema de multas (confirmar prestamos/devolución de libros y comprobar si el usuario tiene multas).
- *App* móvil: El resultado de la tercera iteración, es una *app* que sirve como interfaz para interactuar con la *API REST* y la biblioteca, destinada al usuario. Sirve para crear una *wallet* y enlazarla a la *app*, así como abrir esta *wallet* para permitir operaciones a la biblioteca. Esta *app* cuenta con funciones como: escanear códigos QR, generar códigos QR, realizar llamadas a la *API REST* y guardar los datos de la cuenta del usuario de forma persistente en la *app*.
- Web biblioteca: Esta web simple es el resultado de la cuarta iteración. Basada en ser lo más sencilla posible y permitir simular el programa administrativo usado por la biblioteca. Sirve para realizar todas las operaciones de la API REST originadas desde la biblioteca e interactuar con la *app* del usuario. El conjunto de funcionalidades que posee es: generador y escáner de códigos QR, realizar llamadas a la *API REST* y simular una base de datos no persistente.

El conjunto de estos elementos da forma a un piloto que permite realizar una pequeña demostración de que es posible un sistema basado en SSI, donde los datos del usuario estén bajo su control.

En cuanto a unos resultados más específicos de cada elemento, ya han sido tratados en el apartado de implementación (página 27) cada uno en la subsección "resultado" correspondiente. La relación entre estos elementos será tratada a continuación de forma visual.

8.3. Resultados visuales del piloto

En este punto se tratará el resultado final del piloto, el referido a la interacción de los tres elementos desarrollados (*API REST*, *app* usuario y web biblioteca). Veremos el estado de cada parte en cada acción realizada.

8.3.1. Creación de usuario

La creación del usuario empieza en la página web en la sección propia, donde deberemos poner los datos del usuario, viéndose como la Figura 16. En cuanto se pulsa el botón de "crear" se genera el usuario en la base de datos simulada de la web, se abre la *wallet* de la biblioteca (primeras líneas de la Figura 19) y se genera el QR que será usado por el usuario (Figura 17). Los pasos realizados por el móvil, visibles en la Figura 18, van en orden:

1. Primero se parte de la actividad de enlace, usando el botón de enlazar se inicial la actividad de escáner de códigos QR.
2. En la segunda actividad, el escáner de códigos QR, se escanea el QR generado por la biblioteca (Figura 17). Una vez analizado se termina esta activad y se vuelve a la anterior. Devolviendo el contenido del QR analizado (un *json*).
3. Esta vez la actividad de enlace detecta que se le ha devuelto un *json* de la actividad del escáner, por lo que cambia su interfaz y empieza interactuar con la *API REST* para generar la cuenta del usuario y asignarle una identidad (visible en la Figura 19 a partir de la cuarta línea). Por último, se guardan los datos de esta cuenta en el móvil, terminando de enlazarla con la *app*.
4. Como resultado se abre la actividad principal, que será el punto de entrada a partir de ahora cuando el usuario abra la *app*.

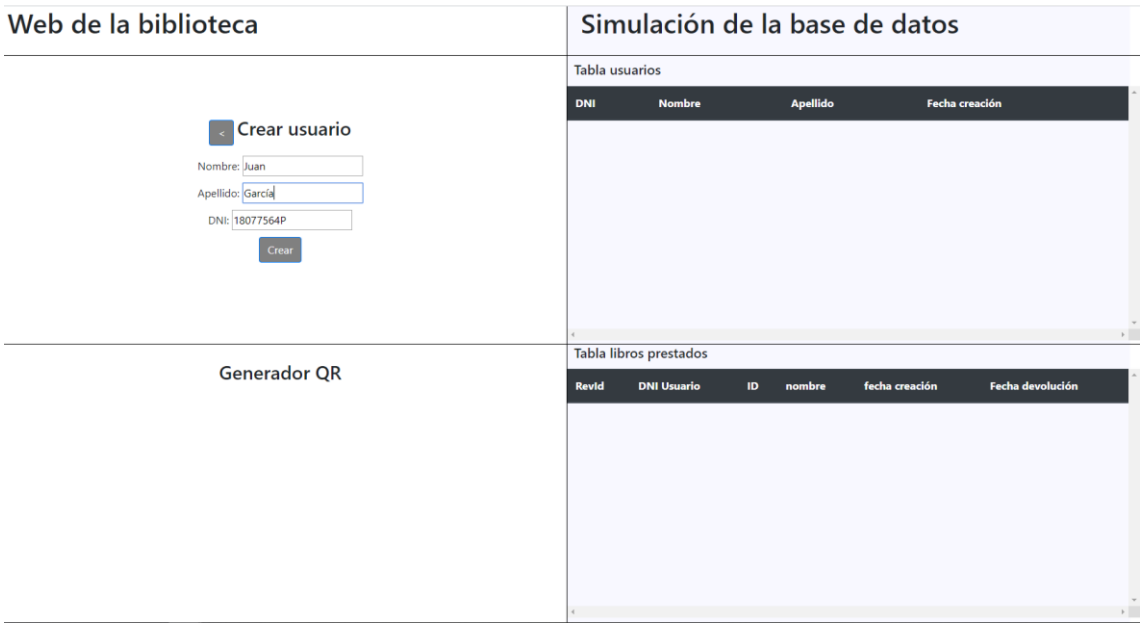

Figura 16- Creación de usuario, web con los parámetros insertados.

Figura 17 - Creación de usuario, usuario creado.

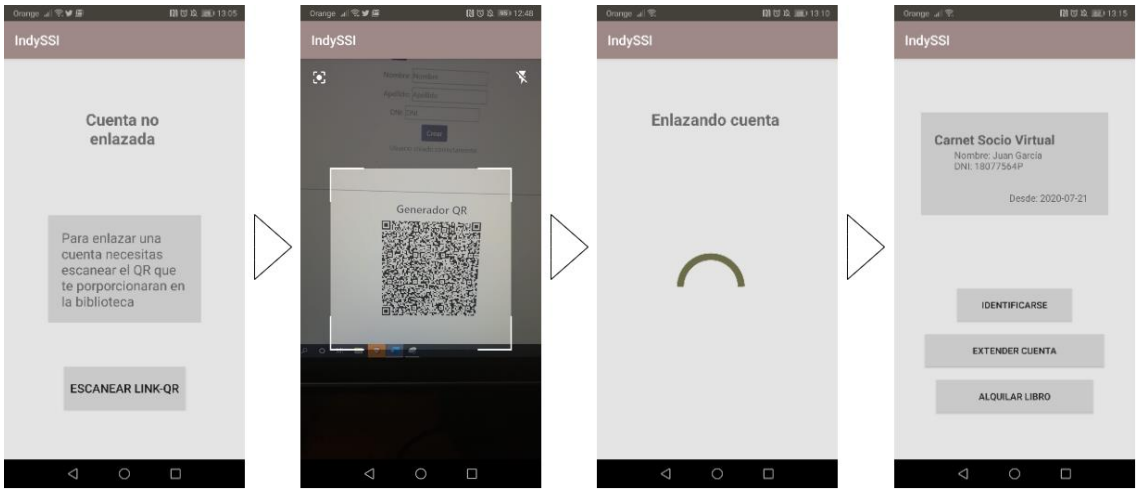

Figura 18 - Creación de usuario, pasos realizados en el móvil.

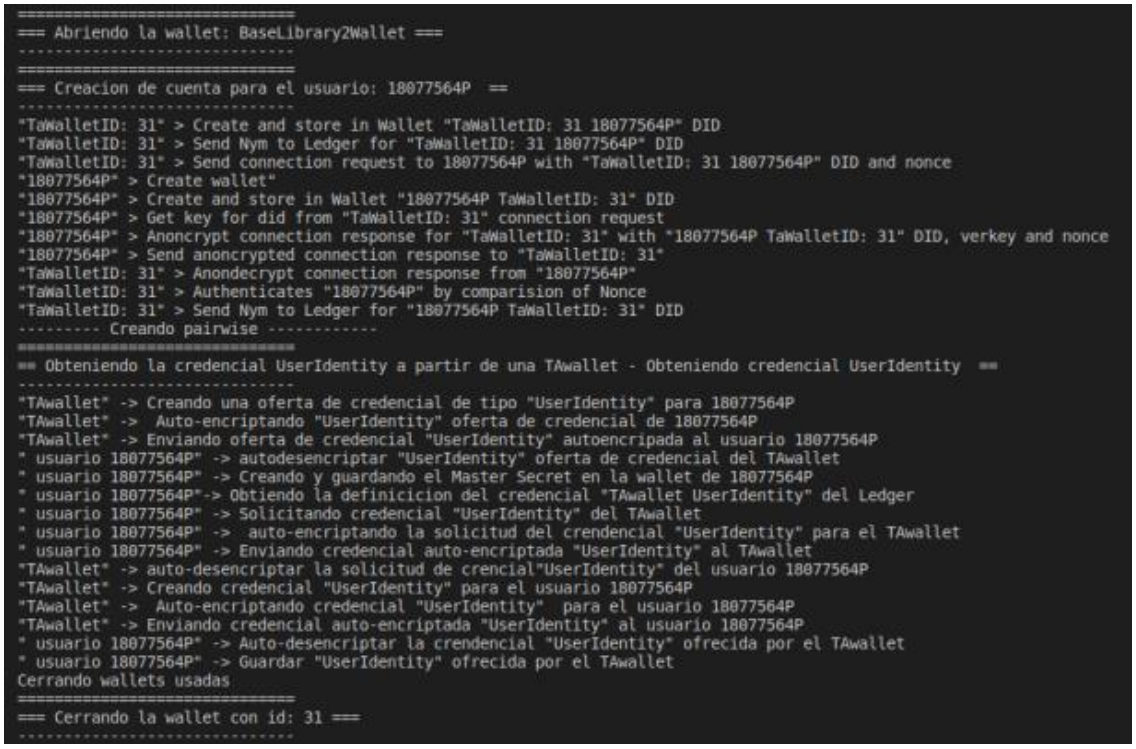

Figura 19 - Creación del usuario, log API REST.

8.3.2. Identificación del usuario

Una vez el usuario solicita ser identificado, el bibliotecario solo debe elegir esta acción en el menú de la web y se cambia la interfaz e inicializar el escáner QR (Figura 20). Por lo que el usuario debe abrir la *wallet*, para que la biblioteca escanee el QR (*walletID* usuario) que se genera (Figura 22). Esto permite extraer del *ledger* el DNI y si tiene multas con la seguridad de que estos son verídicos (visible debajo del texto "reserva de libros", Figura 21). En cuanto las acciones que ocurren en el *API REST* están recogidas en la Figura 23.

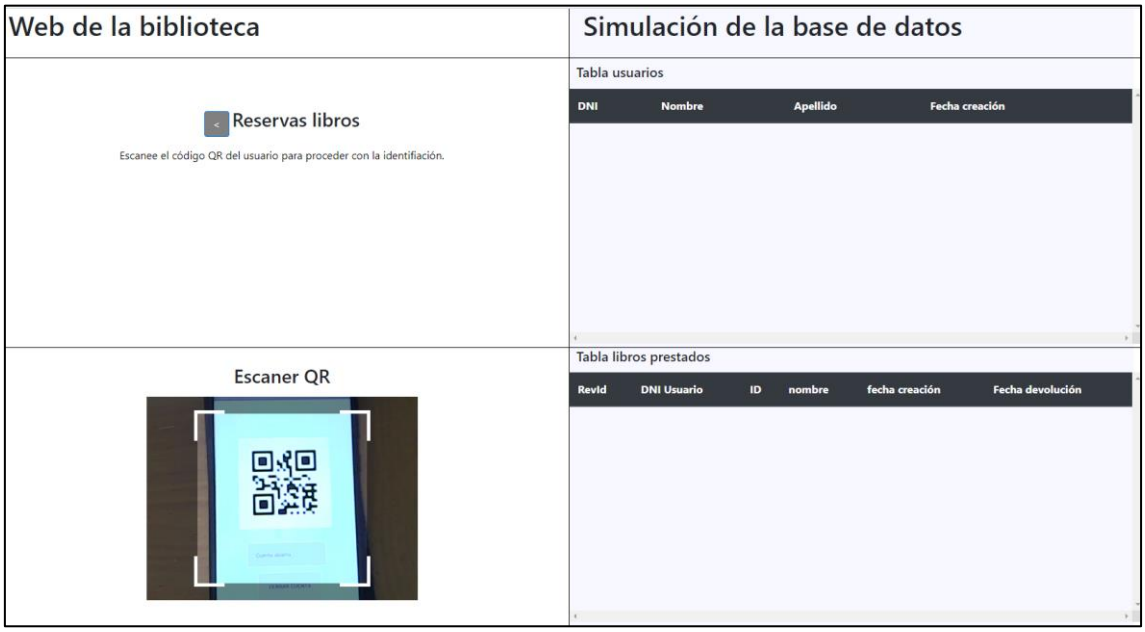

Figura 20 - Identificación de usuario, escáner de código QR.

Figura 21 - Identificación de usuario, usuario correctamente identificado.

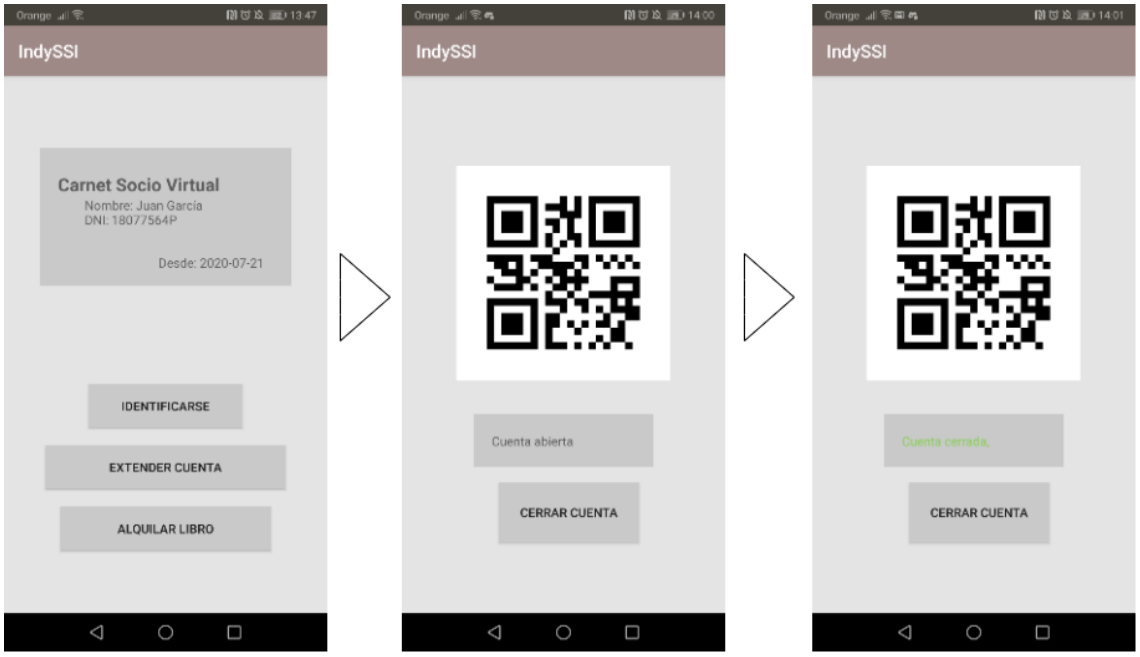

Figura 22 - Abriendo wallet en la app del usuario (Actividad principal, QR generado y wallet cerrada correctamente).

```
==============================
=== Abriendo la wallet: 18077564PWallet ===
------------------------------
==============================
============== Obtiendo identidad de la wallet: 8  ================
------------------------------
Obtiendo datos del usuario del ledger
Comprobando si el usuario esta bajo multa
==============================
== Comprobando multas de la wallet con id: 8 ==
------------------------------
Obteniendo fecha actual
Generando esquema de prueba
Realizando busqueda de las credenciales de la wallet que coincidan con el esquema de prueba
Ningún libro reservado por el usuario: 8
==============================
=== Cerrando la wallet con id: 8 ===
------------------------------
```

Figura 23 - Identificación del usuario, log API REST.

8.3.3. Extensión de cuenta de usuario

En el momento que un usuario quiere extender su cuenta a otra biblioteca, el bibliotecario selecciona en la web esta acción lo que le permitirá escanear el QR de la *app* del usuario (Figura 24). El proceso que sigue el usuario para abrir la *wallet* y dar permiso a la extensión del usuario, es el mismo que en la identificación del usuario solo que seleccionando otra operación (Figura 22). Cuando el código es escaneado este realiza dos cosas, genera el usuario en la base de datos simulada de la web (Figura 25) y realiza un enlace (*onboarding*) entre la *wallet* del usuario y la *wallet* de la nueva biblioteca (visible en las operaciones de la *API REST*, Figura 26)

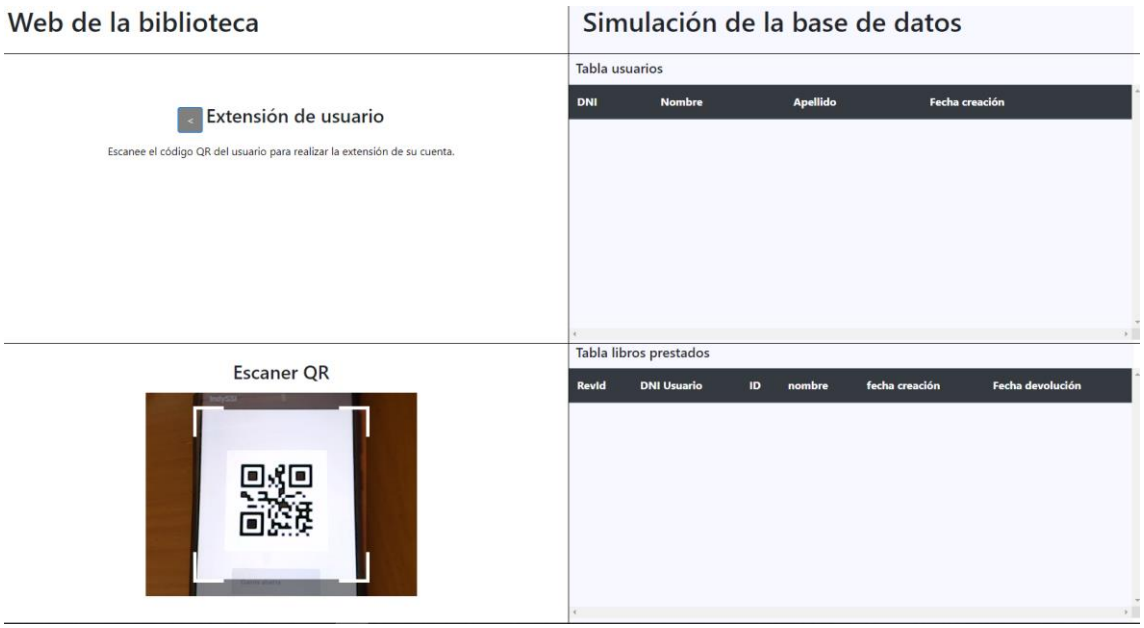

Figura 24 - Extensión de usuario, escáner QR.

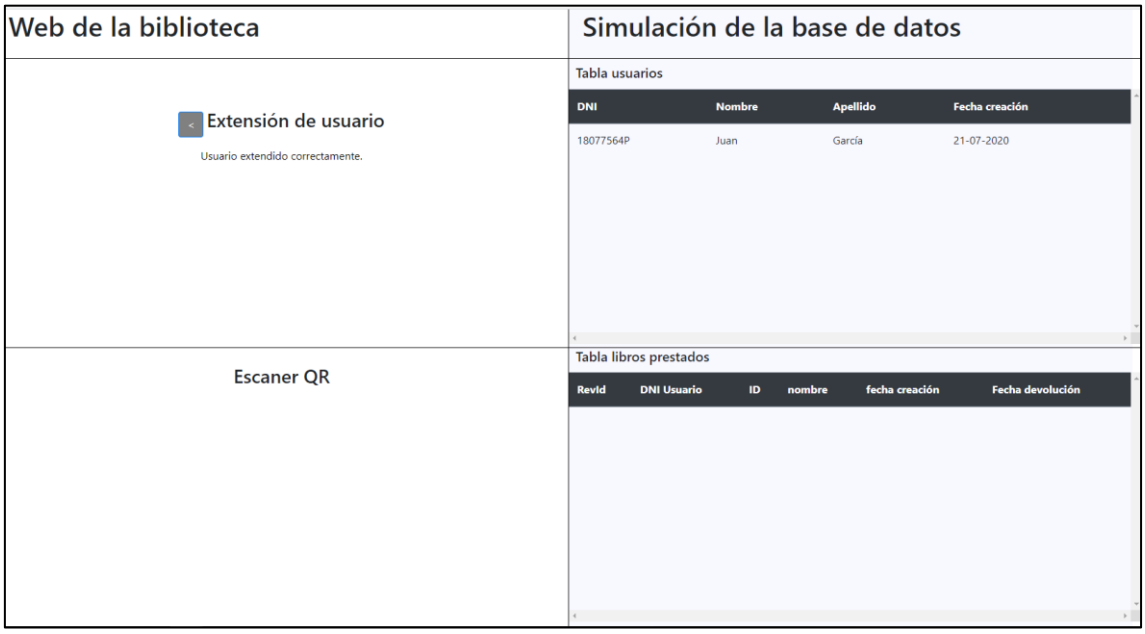

Figura 25 - Extensión de usuario, usuario extendido.

```
==============================
=== Abriendo la wallet: 18077564PWallet ===
------------------------------
==============================
=== Extendiendo usario con wallet: 8 ea la biblioteca con did: LsMDFNBN7oH1qQJ3Es7qux ===
------------------------------
Realizando el enlace entre la wallet del usuario y la biblioteca
"TaWalletID: 9" > Create and store in Wallet "TaWalletID: 9 userWallet 8" DID
"TaWalletID: 9" > Send Nym to Ledger for "TaWalletID: 9 userWallet 8" DID
==============================
=== Comprobando si la wallet con id: 8  esta abierta ===
------------------------------
"TaWalletID: 9" > Send connection request to userWallet 8 with "TaWalletID: 9 userWallet 8" DID and nonce
"userWallet 8" > Create and store in Wallet "userWallet 8 TaWalletID: 9" DID
"userWallet 8" > Get key for did from "TaWalletID: 9" connection request
"userWallet 8" > Anoncrypt connection response for "TaWalletID: 9" with "userWallet 8 TaWalletID: 9" DID, verkey and nonce
"userWallet 8" > Send anoncrypted connection response to "TaWalletID: 9"
"TaWalletID: 9" > Anondecrypt connection response from "userWallet 8"
"TaWalletID: 9" > Authenticates "userWallet 8" by comparision of Nonce
"TaWalletID: 9" > Send Nym to Ledger for "userWallet 8 TaWalletID: 9" DID
==============================
=== Comprobando si la wallet con id: 8  esta abierta ===
------------------------------
--------- Creando pairwise ------------
Obteniendo datos del usuario del ledger, para replicar en la DB de la biblioteca
Cerrando wallets usadas
==============================
=== Cerrando la wallet con id: 8 ===
------------------------------
```

Figura 26 - Extensión de usuario, log API REST.

8.3.4. Préstamos de libros

Para realizar un préstamo de un libro es necesario que el usuario esté identificado previamente, entonces en la web del bibliotecario se podría introducir los datos del libro. Una vez se pulsa el botón de "crear" se inicia el escáner de QR (Figura 27). El siguiente proceso es el de abrir la *wallet* para confirma la reserva de un libro (Figura 22). Una vez se escanea el código del usuario el resultado en la web es la de un nuevo libro en la base de datos simulada y un *feedback* positivo en el texto de debajo de "Reserva libros" (Figura 28). En cuanto las acciones el API REST, se produce una creación de una nueva credencial "*bookLoan*" y se enlaza con el usuario correspondiente (Figura 29).

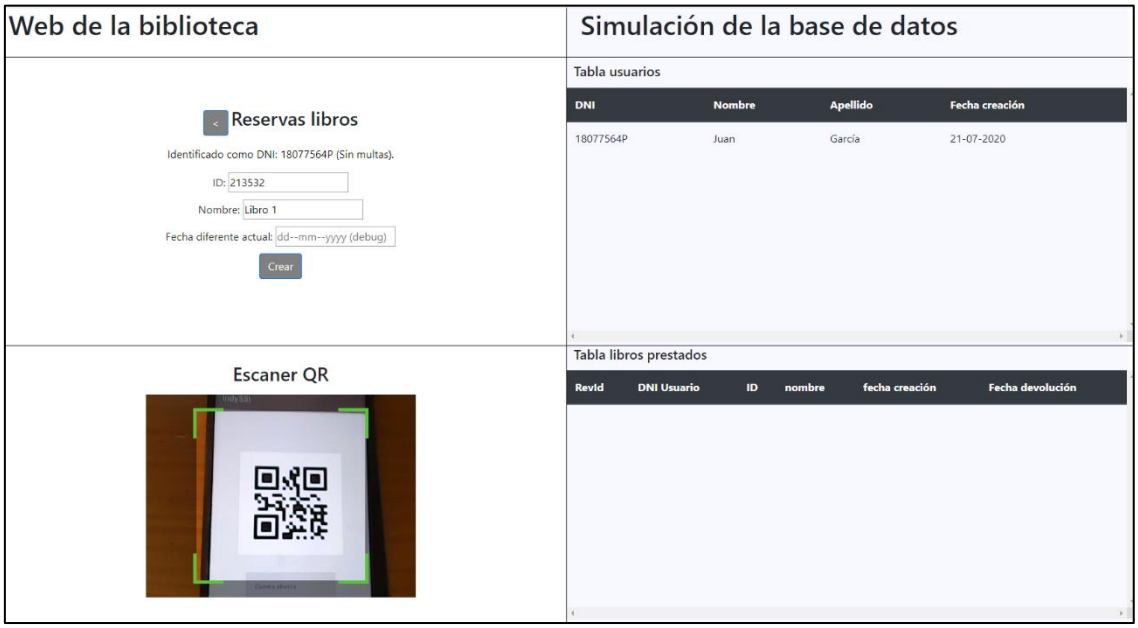

Figura 27 - Préstamo de libro, escáner QR.

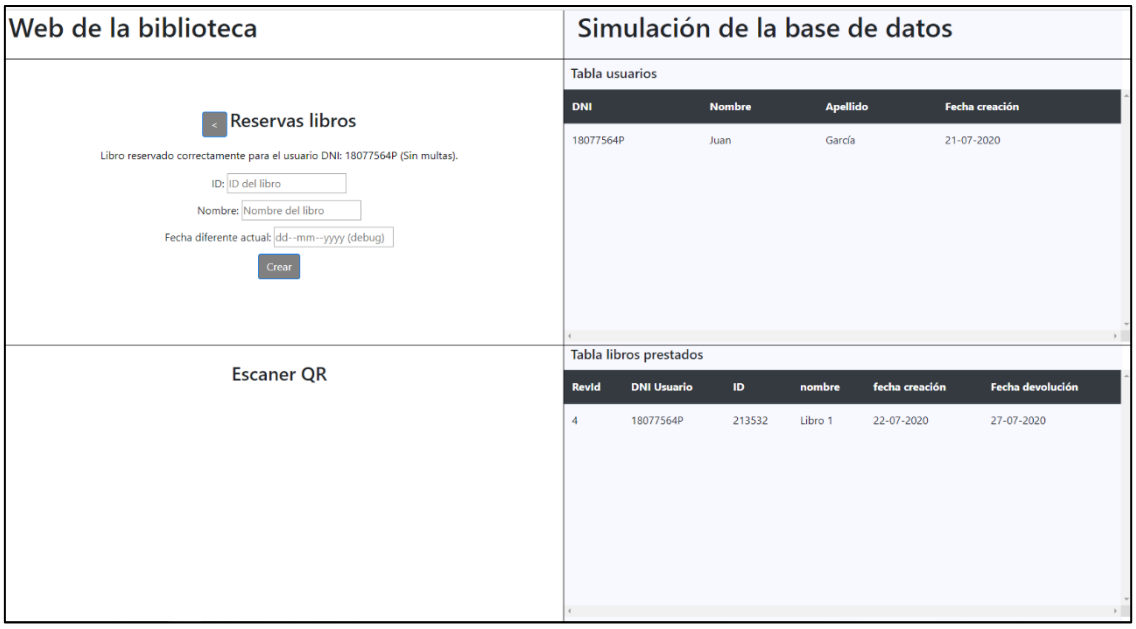

Figura 28 -Préstamo de libro, libro prestado correctamente.

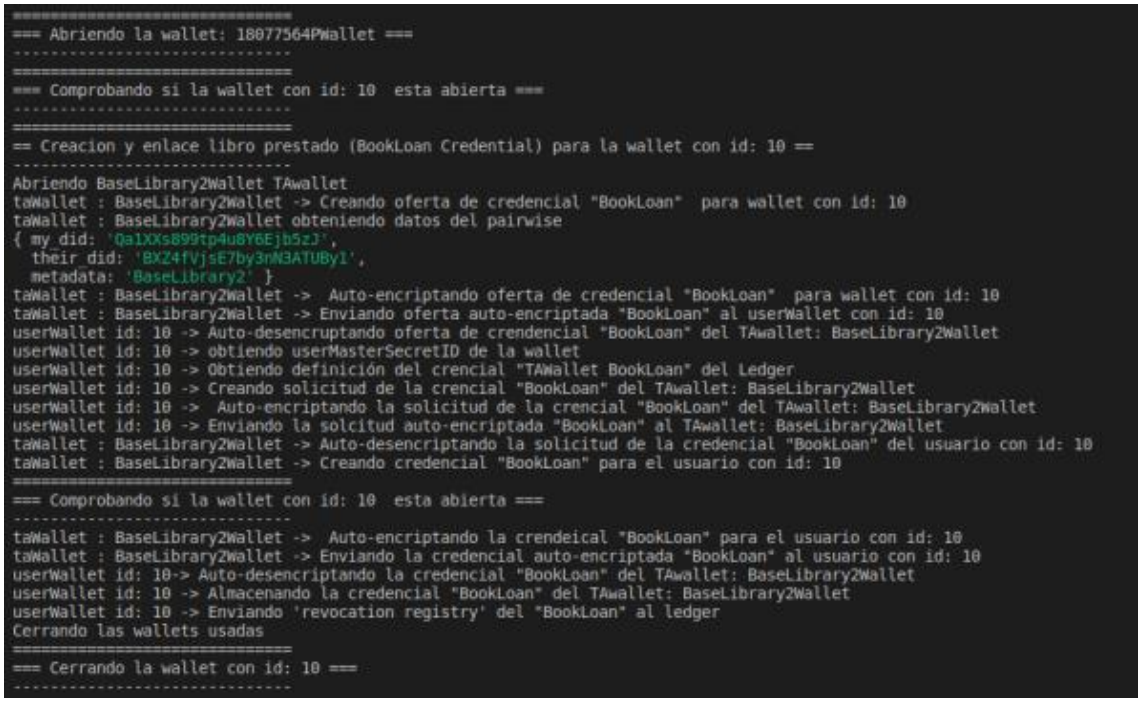

Figura 29 - Préstamo de libro, log API REST.

8.3.5. Devolver un libro

La devolución de un libro se hace desde el menú principal de la web y no es necesario el permiso de ningún usuario. Solo hay que introducir el Id del libro que se ha devuelto (Figura 30) y presionar "devolver libro". Esto recuperará el "revocationID" del libro y será usado para romper el enlace que tiene con la *wallet* del usuario (Figura 32), evitando así el sistema de multas. El resultado es un *alert* que confirma que todo salió bien y la eliminación del libro de la base de datos de libro prestados (Figura 31).

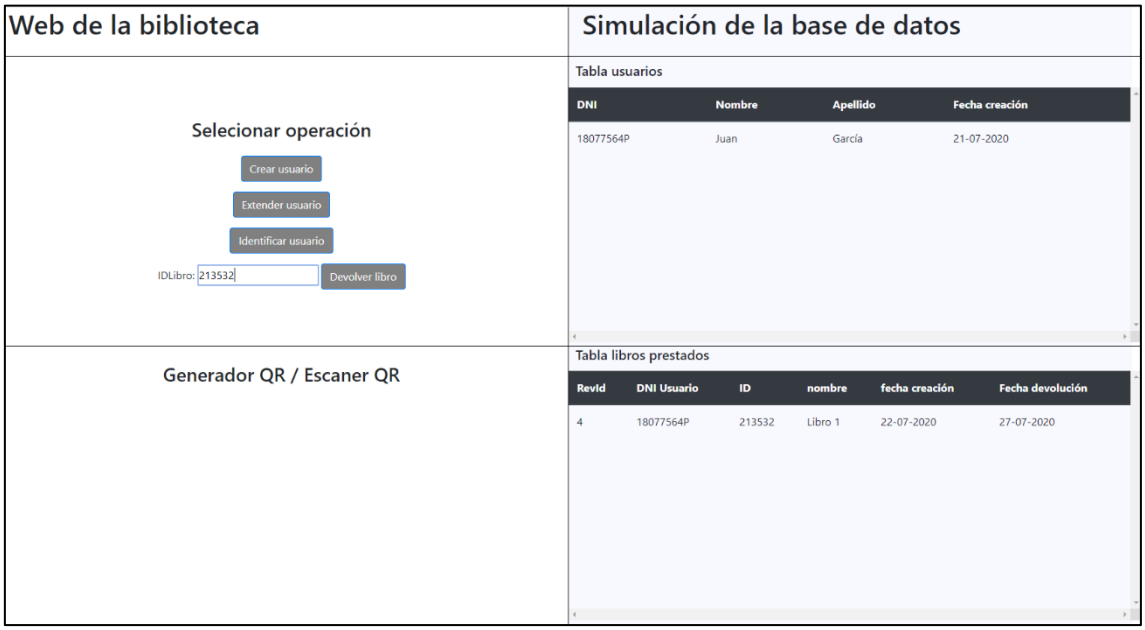

Figura 30 - Devolver un libro, menú principal con input introducidos.

Figura 31 - Devolver un libro, devolución realizada correctamente.

```
==============================
== Rompiendo el enlace entre un libro reservador y la wallet del usuario (revocate BookLoan credential) ==
------------------------------
Abriendo wallet de la biblioteca
Biblioteca revoca la credencial
Biblioteca envia el estado del 'revocation registry delta' al ledger
Cerrando wallet de la biblioteca
```

Figura 32 - Devolver un libro, log API REST.

8.4. Desviación respecto la planificación inicial

Este punto se centra en analizar si el resultado final es el que se esperaba en la planificación inicial. Hay que recordar que la planificación inicial se basó en realizar en orden los objetivos extendidos (apartado "Objetivos", página 17), cuando estos se reescribieron para hacerlos más específicos.

A la hora de comprobar los resultados obtenidos, se cumplieron las tres tareas principales incluyendo sus subjetivos:

1. Se realizó un análisis inicial de la tecnología Blockchain y sus herramientas, como se indicaba en la primera tarea. El desarrollo inicial está disponible en el punto de análisis (página 23) y las ventajas e inconveniente de estas herramientas se han podido ver en este mismo punto de resultados.
2. Se pensaron unos casos de uso y se eligió uno de ellos para ser desarrollado, como se indica en la segunda tarea de los objetivos. Este proceso de reflexión y selección se recogió también en el punto de análisis antes mencionado.
3. Por último, se realizó la tarea de crear un piloto. Esta fue la única cuyo producto final difirió de la tarea, aunque no demasiado. La tarea contaba de cuatro elementos, sin embargo, se añadió un quinto:
    a. Desarrollar la red *blockchain*: Esta se realizó mediante el uso de la herramienta que se seleccionó, Hyperledger Indy, la cual se combinó con la siguiente tarea.
    b. Desarrollar una *API REST*, así como todas las llamadas a esta: Se realizó un *API REST* mediante express.js, con todas las llamadas necesarias para interactuar con la red *blockchain*.
    c. Desarrollar una *app* como sistema de interacción para usuario: Mediante Android Studio se realizó esta *app*, que funcionaba como interfaz para el usuario.
    d. Evaluar el resultado obtenido con las herramientas usadas: Como hemos hecho en el punto 8.1 de los resultados.
    e. Diseñar una web para completar la demo: Este es el punto que no estaba especificado en los objetivos desarrollados, pero fue necesario ya que servía para poder tener una demo cerrada e interactiva.

Como se ha visto, exceptuando el desarrollo de la web, el resto de los elementos desarrollados estaban presentes en los objetivos reescritos a el inicio del proyecto. Esta nueva tarea se dio debido a que al no saber el caso de uso que iba a implementarse, era imposible prever todos los elementos de este.

La otra característica que debería tratarse es el tiempo estimado en realizar cada tarea del proyecto. Si bien es cierto que no se diseñó un cronograma inicial que nos permita comparar con el resultado final, podríamos definir uno aproximado a la idea que se estimaba en ese momento. Esta aproximación se representa en Figura 33-A, la cual debería comparase con la Figura 33-B.

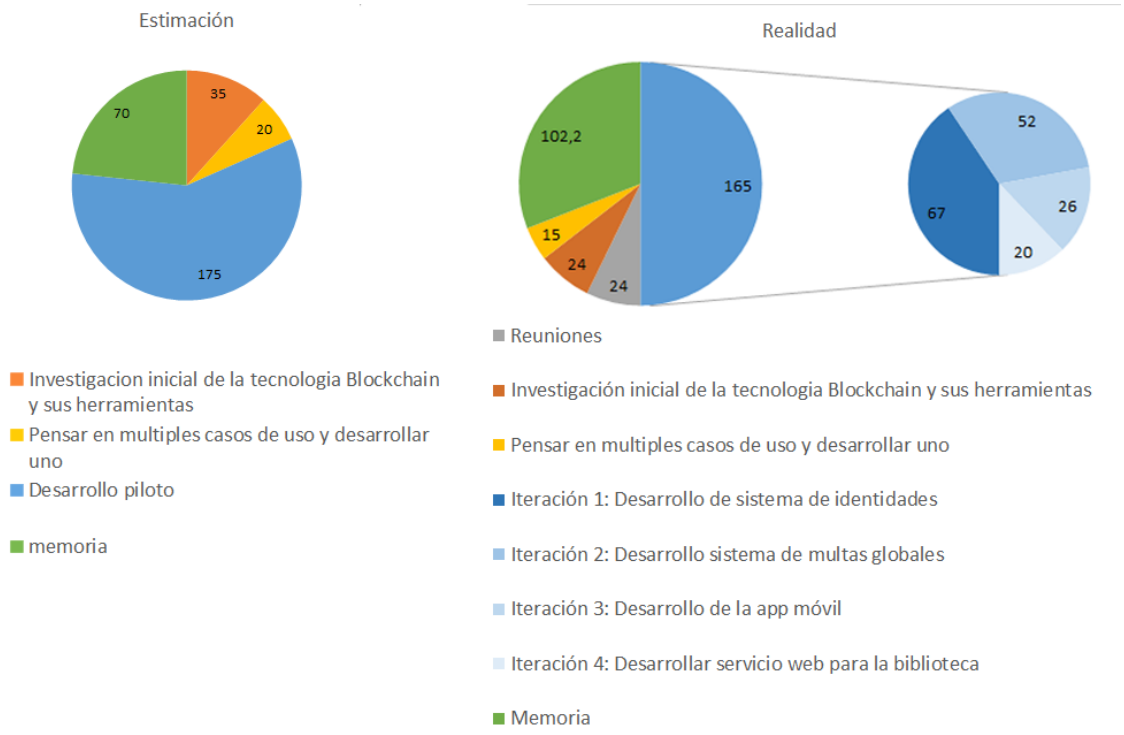

Figura 33 - A) Grafico de las horas estimadas al inicio del proyecto. B) Grafico de las horas reales tras la finalización del proyecto.

La comparación de estos diagramas revela algunos errores en cuanto la estimación de las horas. El primero de ellos fue el de no estimar las reuniones en el tiempo de desarrollo, ya que las reuniones tenían una duración inferior a una hora, pareciendo no tener importancia a la hora de organizar el proyecto. Sin embargo, estas ocuparon 24 horas dentro del proyecto. Otro fallo fue el coste estimado de las dos primeras tareas, siendo mayor al necesitado, debido principalmente a la dificultad de estimar el coste de una investigación o el pensar posibles casos de uso. Otra diferencia con las horas estimadas sucedió en relación con la memoria, dado que en un principio se pensaba que costaría entre 60-70 horas frente a las 102 que finalmente resultaron. Al final el proyecto costó 330 horas, lo cual se acerca a la estimación de las 300. En cuanto a sus diferentes partes, pese a ver algunas diferencias entre las horas reales y las estimadas, se puede decir que el proyecto resulto en gran medida lo esperado en un inicio.

9. Conclusión

El proyecto se inició bajo la idea principal de evaluar las herramientas *blockchain* actuales, así como evaluar la tecnología Blockchain para su uso en la identidad soberana autogestionada. Para demostrar las posibilidades de llevar esto a la realidad, se pensó en desarrollar un caso de uso sencillo, una solución que permite agrupar los carnés de las bibliotecas basado en SSI. Este piloto se ha realizado de forma correcta dejando claro que la tecnología Blockchain y sus herramientas son más que capaces de crear productos empresariales.

Sin embargo, durante el desarrollo del proyecto ha quedado claro que pese a existir un nicho explotable en el desarrollo de aplicaciones basadas en SSI, el coste de estas será superior a cualquier solución bajo los estándares actuales (un servidor central). Este problema hace que, aunque el SSI ofrezca ventajas a los usuarios, el sobrecoste provoque que el nicho no termine de ser la solución más rentable para las empresas.

El problema que se da con las aplicaciones basadas en SSI es similar a cualquier otro tipo de aplicación basada en la tecnología Blockchain. Estas soluciones innovadoras requieren de una formación específica, poseen herramientas en desarrollo que no se encuentran terminadas en su totalidad y su comunidad es escasa, lo que dificultad la solución de *bugs*.

Estos problemas, que ya eran previsibles previamente al inicio del proyecto, quedaron confirmados una vez terminó el desarrollo del mismo. Aun así, se distingue un futuro bastante prometedor para el desarrollo de estas aplicaciones basada en SSI así como en la red *blockchain*, permitiendo que con la popularización de estas herramientas y el incremento de su comunidad los problemas antes mencionados desaparezcan.

Por último, es necesario indicar que esto no significa que no se puedan desarrollar aplicaciones basadas en SSI, ya que tiene sus ventajas sobre otras soluciones. Si se dispone de un personal ya formado en la tecnología Blockchain, el desarrollo de una aplicación de este tipo es posible con un coste más adecuado a lo habitual y dando una mejora de visibilidad a la empresa en cuanto a su innovación frente a otras.

9.1. Opinión personal

La realización de este proyecto me ha aportado conocimientos sobre una tecnología que está en auge y cuya popularidad no para de crecer, permitiendo tener conocimientos que me serán muy útiles en el futuro. También me gustaría indicar que este proyecto me ha servido para confirmar que he adquirido los conocimientos suficientes a lo largo de mi formación para realizar proyectos

de cualquier tipo, como es el caso, donde se ha usado herramientas diferentes como Indy-sdk, express.js, Android Studio o realizado un desarrollo web.

10. Bibliografía

[1] Inycom, "pagina web Inycom." https://www.inycom.es/.

[2] S. Haber and W. Scott Stornetta, "How to time-stamp a digital document," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 537 LNCS, pp. 437–455, 1991, doi: 10.1007/3-540-38424-3_32.

[3] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the Efficiency and Reliability of Digital Time-Stamping," pp. 329–334, 1993, [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.4891&rep=rep1&type=pdf.

[4] H. Finney, "Reusable Proofs of Work (RPOW)," *Self-published*, 2004. http://web.archive.org/web/20071222072154/http://rpow.net/%0Ahttps://cryptome.org/rpow.htm.

[5] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. https://bitcoin.org/bitcoin.pdf.

[6] V. Buterin, "Ethereum white paper," 2015. https://web.archive.org/web/20150328054135/https://github.com/ethereum/wiki/wiki/White-Paper.

[7] bit2me academy, "Proof of Work explicación," [Online]. Available: https://academy.bit2me.com/que-es-proof-of-work-pow/.

[8] bit2me academy, "Proof of Stake explicación," [Online]. Available: https://academy.bit2me.com/que-es-proof-of-stake-pos/.

[9] UPort, "uportlandia web." https://uportlandia.uport.me/.

[10] Hyperledger, "Hyperledger Besu." https://www.hyperledger.org/use/besu.

[11] Hyperledger, "Hyperledger Burrow." https://www.hyperledger.org/use/hyperledger-burrow.

[12] Hyperledger, "Hyperledger Fabric." https://www.hyperledger.org/use/fabric.

[13] Hyperledger, "Hyperledger Iroha." https://www.hyperledger.org/use/iroha.

[14] Hyperledger, "Hyperledger Sawtooth." https://www.hyperledger.org/use/sawtooth.

[15] Hyperledger, "Hyperledger Indy." https://www.hyperledger.org/use/hyperledger-indy.

[16] Heraldo de Aragón, "Aragón implantará un carné único para su red de bibliotecas," Zaragoza, Sep. 25, 2018.

[17] M. Breeding, "Library Perceptions 2020: Results of the 13th International Survey of Library Automation," *Libr. Technol. Guid.*, 2019, [Online]. Available: https://librarytechnology.org/perceptions/2019/.

[18] Hyperledger, "Github Hyperledger Indy-sdk," 2017. https://github.com/hyperledger/indy-sdk.

[19] L. Kumar, "Step by Step Guide to Set Up Hyperledger Indy with Node.js," 2019, [Online]. Available: https://medium.com/akeo-tech/step-by-step-guide-to-set-up-hyperledger-indy-64eeb524f558.

[20] "Express.js en npm," 2020. https://www.npmjs.com/package/express.

[21] Hyperledger, "Indy Walkthrough." https://github.com/hyperledger/indy-sdk/blob/master/docs/getting-started/indy-walkthrough.md.

[22] yuriy-budiyev, "code-scanner library." 2018, [Online]. Available: https://github.com/yuriy-budiyev/code-scanner.

[23] amitshekhariitbhu, "Fast Android Networking library." 2018, [Online]. Available: https://github.com/amitshekhariitbhu/Fast-Android-Networking.

[24] Androidmads, "QRGenerator library." 2020, [Online]. Available: https://github.com/androidmads/QRGenerator.

[25] davidshimjs, "qrcode.js GitHub web." https://github.com/davidshimjs/qrcodejs.

[26] mebjas, "html5-qrcode Github web," 2020. https://github.com/mebjas/html5-qrcode.

[27] indeed.com, "Salarios de programador junior en España," 2020. https://es.indeed.com/salaries/programador-junior-Salaries?period=yearly.

[28] indeed.com, "Salarios de programador sneior en España," 2020. https://es.indeed.com/salaries/programador-senior-Salaries?period=yearly.

[29] Statista and R. Fernández, "Número medio de horas de trabajo al año acordadas en convenios colectivos en España de 2006 a 2019," 2020.

https://es.statista.com/estadisticas/478441/promedio-de-horas-de-trabajo-al-ano-segun-convenios-colectivos-de-espana/.

[30] Hyperledger, "Indy-sdk códigos de ejemplo en GitHub," [Online]. Available: https://github.com/hyperledger/indy-sdk/tree/master/samples.

[31] J. Garzo, "Pregunta realizada en stack overflow," 2020. https://stackoverflow.com/questions/61816506/indy-update-credential-parameter.

[32] Stack overflow, "Preguntas realizadas en stack overflow con el tag 'hyperledger-fabric,'" 2020. https://stackoverflow.com/questions/tagged/hyperledger-fabric.

[33] Stack overflow, "Preguntas realizadas en stack overflow con el tag 'hyperledger-indy,'" 2020. https://stackoverflow.com/questions/tagged/hyperledger-indy.

11. Anexos

11.1. Anexo – Propuesta de proyecto

Nombre alumno: Javier Garzo

Titulación: Graduado en Ingeniería Informática/Graduado en Diseño y Desarrollo de Videojuegos

Curso académico: 2019-2020

1. TÍTULO DEL PROYECTO

Sistema de Identidad Soberana basada en Blockchain sobre Dispositivos Móviles

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

El proyecto consiste en el estudio, análisis y desarrollo de un sistema de Identidad Soberana con base tecnológica Blockchain, generación de DiD (Distributed Identities), y manejo preferente a través de dispositivos móviles.

3. OBJETIVOS DEL PROYECTO

Los objetivos del proyecto son:

i) Analizar la tecnología Blockchain más adecuada

ii) Definición y desarrollo de los casos de uso a implementar.

iii) Diseñar y desarrollar los APIs e interfaces básicos para el manejo e integración del sistema

4. METODOLOGÍA

La metodología se establecerá en las primeras fases del proyecto

5. PLANIFICACIÓN DE TAREAS

Las tareas quedan predefinidas de manera global en los objetivos. Serán fijadas de forma concreta durante el desarrollo del proyecto

6. OBSERVACIONES ADICIONALES

Para que la asignación del proyecto sea definitiva, el alumno deberá contar con la aprobación por parte de la empresa Inycom tras una entrevista personal.
En el supuesto de no ser asignada la propuesta al alumno, se le asignará desde la coordinación de la asignatura un tutor para desarrollar una nueva propuesta.

11.2. Anexo – Reuniones con el tutor de empresa

ID: 1	Fecha: 03/01/2020	Próxima reunión: 2 semanas
Temas tratados: • Hablar de la dirección que va a tomar la investigación inicial.		
Objetivos para siguiente reunión: • Investigar y comprender el funcionamiento de la blockchain y el SSI • Investigar sobre Uport e Hyperledger Indy.		

ID: 2	Fecha: 17/01/2020	Próxima reunión: 2 semanas
Temas tratados: • Hablar del alcance y objetivo del TFG. • Comentar investigar casos de uso para la siguiente reunión. • Comentar que elementos deberían estar incluidos en el estado del arte (Blockchain, Hyperledger, Uport). • Comentar sobre la instalación de Hyperledger Indy, en cuanto que SO elegir. Linux es el recomendado así que se eligió ese.		
Objetivos para siguiente reunión: • Buscar posibles casos de uso en el proyecto. • Implementar un *dual boot* Linux - windows. • Mirar la documentación del Hyperledger Indy.		

ID: 3	Fecha: 30/01/2020	Próxima reunión: 2 semanas
Temas tratados: • Hablar de los posibles casos de uso (Sistema identificación para bibliotecas, sistema de titulaciones internacionales y reserva de espacios para la universidad). Es elegido el de las bibliotecas. • Hablar del funcionamiento de Indy-sdk. • Dudas con respecto al código de ejemplo de indy-sdk. • Dudas en cuanto como se hará el proyecto., se estimó que el mejor resultado sería usar una *API REST* y que la *app* se conecte a esta.		
Objetivos para siguiente reunión: • Pensar en toda la posible utilidad del sistema de identificación para bibliotecas. • Terminar de entender todo el código presente de la documentación indy-sdk y el ejemplo de código existente (el que está en NodeJS). • Investigar sobre la configuración de los *nodes pool*. • Saber cómo se va a realizar el *API REST* para la próxima reunión.		

ID: 4	Fecha: 14/02/2020	Próxima reunión: 2 semanas
Temas tratados: • Hablar del caso de uso elegido y cerrarlo del todo. • Hablar de los progresos realizados, terminar de leer la documentación restante y el enfoque que va a llevar el *API REST*. • Hablar de cómo hacer más visible los procesos detrás de la *blockchain*, *node monitor* o *logs*.		
Objetivos para siguiente reunión: • Modificar el código de ejemplo para que se adapte al modelo del proyecto. • Empezar la realización del *API REST*, que pueda recibir una llamada y dar un resultado. • Investigar sobre un *node monitor* para la *node pool* de Indy.		

ID: 5	Fecha: 28/02/2020	Próxima reunión: 3 semanas
Temas tratados: • Hablar de dudas de los roles de los *nodes* y el *node monitor*. • Enseñar el código de ejemplo modificado, aun sin acabar. • Comentar el funcionamiento correcto del *API REST* de pruebas.		
Objetivos para siguiente reunión: • Terminar de modificar el código de ejemplo e integrarlo en la *REST API*.		

ID: 6	Fecha: 20/03/2020	Próxima reunión: 2 semanas
Temas tratados: • Hablar sobre el código de ejemplo ya terminado. • Enseñar código y ejemplo llamadas en el *API REST* (Iniciar la *node ledger*, crear TA). • Comentar el estado de la memoria (sobre todo algunas dudas técnicas de *blockchain* relacionadas con el estado del arte).		
Objetivos para siguiente reunión: • Seguir desarrollando la *REST API*. • Corregir algunos puntos del estado del arte de la memoria.		

ID: 7	Fecha: 06/04/2020	Próxima reunión: 2 semanas

Temas tratados:

- Enseñar código y ejemplo llamadas en el *API REST* (abrir/cerrar *wallets*, creación de usuario, ligar identidad, identificar usuario).
- Hablar de dudas temas de seguridad (dudas *keys* usuario, seguridad y privacidad).
- Hablar del origen de las llamadas del *API REST*, algunas deberían hacerse desde el usuario y otras desde la biblioteca (Para mantener los datos siempre en el control del usuario). Ahora mismo todas se hacían desde la biblioteca.

Objetivos para siguiente reunión:

- Terminar el REST API con la extensión de usuario.
- Modificar funcionamiento identificar usuario (el DNI se extrae del *ledger*, no de un QR que del usuario).
- Modificar las llamas para que su origen sea el correcto (el usuario o la biblioteca).

ID: 8	Fecha: 20/04/2020	Próxima reunión: 2 semanas

Temas tratados:

- Hablar del correcto funcionamiento de la extensión de usuario y la modificación de la función de identificación.
- Hablar sobre la refactorización de la dirección de las llamadas del *API REST*.
- Hablar de cómo desarrollar el sistema de multas globales y las dudas en relación con este.
- Comentar el estado de la memoria.

Objetivos para siguiente reunión:

- Crear la reserva de libros para saber si algún usuario no ha devuelto un libro.
- Investigación sobre sistema de eventos para multar usuarios.

ID: 9	Fecha: 4/05/2020	Próxima reunión: 2 semanas
Temas tratados: • Comentar dudas respecto reservas de libro. • Problema con el sistema de eventos para multar usuarios, Indy no lo permite. Sistema de revocación de Indy-sdk como solución. • Hablar del desarrollo del *API REST* (asignar libros a un usuario). • La eficiencia de la blockchain y no que debería guardar datos duplicados, así que las reservas de los libros deben tener los menos datos posibles.		
Objetivos para siguiente reunión: • Refactorización de la estructura de libros, esta debe ser lo más mínima posible para no sobrecargar la *blockchain*. • Tratar de solventar el problema del sistema de eventos con el sistema de revocación de Indy.		

ID: 10	Fecha:18/05/2020	Próxima reunión: 2 semanas
Temas tratados: • Hablar sobre la solución de la estructura de los libros (solo se almacena la fecha de devolución). • Comentar el desarrollo de sistema de revocación (revocar libros y verificar si tiene alguno sin devolver). Terminado si no fuera porque este presenta problemas en el sistema de verificación.		
Objetivos para siguiente reunión: • Solucionar el problema de la verificación del sistema de multas globales.		

ID: 11	Fecha:1/06/2020	Próxima reunión: 2 semanas
Temas tratados: • Explicar que el api esta completa, estado terminado, pero si es necesario se podrían añadir modificaciones. • Hablar sobre el problema del sistema de verificación (la master-secret del usuario siempre tiene que ser la misma). La solución fue guardar el master-secret cuando se abre la *wallet*. • Consultar duda privacidad empresarial del código (en principio se podría compartir)		
Objetivos para siguiente reunión: • Crear *app* móvil		

ID: 12	Fecha:16/06/2020	Próxima reunión: 2 semanas
Temas tratados: • Hablar de las modificaciones menores en la estructura del *json* que devuelve las llamadas al *API REST*, para hacerlas más fáciles de leer por la *app* y evitar enviar datos innecesarios. • Enseñar progreso de la *app* móvil (un 50% realizada)		
Objetivos para siguiente reunión: • Terminar *app* móvil • Realizar las modificaciones necesarias en el *API REST* para el correcto funcionamiento con la *app*.		

ID: 13	Fecha:30/06/2020	Próxima reunión: 2 semanas
Temas tratados: • Explicar que fue necesario una modificación en el sistema de abrir/cerrar *wallets*, para evitar que se realice otra operación diferente a la cedida por el usuario. • Enseñar el resultado final de la *app*, ya terminada.		
Objetivos para siguiente reunión: • Desarrollar una simulación de la biblioteca, cerrando el ciclo REST API, *app* usuario e interfaz biblioteca. Tener en cuenta que la idea es permitir una demo completa.		

ID: 14	Fecha:15/07/2020	
Temas tratados: • Presentar el resultado de la web de la biblioteca y su funcionamiento (interacción *API REST* y *app* usuario). Demostración de demo terminada. • Hablar de la importancia de una correcta explicación de que sucede detrás de la red *blockchain* para entender la profundidad del proyecto. Importante para la presentación del proyecto. • Comentar el envío de la parte de implementación para que el tutor dé el visto bueno.		
Objetivos para siguiente reunión:		

11.3. Anexo – Lista de actas con la tutora de la universidad

REUNIÓN:	**1ºReunión**

Fecha:	3/12/2019		
Hora comienzo:	15:30	**Hora finalización:**	15:55
Lugar:	Edificio rectorado, 2º planta, taller 11		
Elabora acta:	Javier Garzo		
Convocados:	Javier Garzo, Violeta Monasterio		

Orden del día / Acta

No.	Asunto	Acuerdo
1	Explicación de la idea principal y hablar de las primeras reuniones con la empresa	
2	Diferenciación de roles entre tutores de la universidad y de empresa.	
3	Ponerme en contacto una vez se cierre la idea.	001
4		
5		
6		
7		
8		
9		
10		

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Acordar otra reunión o retomar el contacto	Cierre idea	Javier Garzo
002			
003			
004			
005			
006			

REUNIÓN:	**2ºReunión**

Fecha:	22/5/2020		
Hora comienzo:	18:00	**Hora finalización:**	18:43
Lugar:	Reunión remota a través de Microsoft teams		
Elabora acta:	Javier Garzo		
Convocados:	Javier Garzo, Violeta Monasterio		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Hablar organización del proyecto y fechas de entrega de este	001
2	Hablar sobre el tema del consentimiento, también si la empresa da permiso	002
3	Tratar tema introducción	003
4	Tratar temas objetivos	003
5	Pequeña charla sobre la metodología	004
6	Hablar sobre el carácter exploratorio del proyecto y que este debe quedar claro a lo largo de la memoria del TFG	
7	Próxima reunión acordada para 05/06	
8		
9		
10		

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Fecha objetivo para la entrega de la versión final del proyecto en Julio	Finales de Julio	Javier Garzo
002	Preguntar sobre compartir el código con el tutor de la empresa	05/06	Javier Garzo
003	Realizar los puntos Introducción y Objetivos	05/06	Javier Garzo
004	Realizar borrador sobre metodología	05/06	Javier Garzo
005			
006			

REUNIÓN:	**3ºReunión**

Fecha:	5/06/2020		
Hora comienzo:	18:30	**Hora finalización:**	19:00
Lugar:	Reunión remota a través de Microsoft teams		
Elabora acta:	Javier Garzo		
Convocados:	Javier Garzo, Violeta Monasterio		

Orden del día / Acta

No.	Asunto	Acuerdo
1	Tratar correcciones generales de la memoria: Doble espacio (comprobar el interlineado en el guía de la PDU), intentar el uso de frases cortas.	001
2	Hablar de la forma de corregir por parte del alumno. Un cambio de formato (imprimir o cambiar a .pdf) u otra persona que lea la memoria como formas de mejor revisión.	
3	Tratar el punto de Introducción: Debería existir un párrafo tratando la colaboración con Inycom, eliminar el caso de uso seleccionado en la introducción y centrarse en su explicación en el punto de análisis.	002
4	Tratar el punto de estado del arte: Debería existir un ejemplo de SSI, el ejemplo del apartado de Blockchain debería cambiarse por uno relacionado con SSI y no criptomonedas como es actualmente.	002
5	Tratar el punto objetivo: Reescribir estos para una mejor comprensión y añadir en el anexo la propuesta de proyecto.	002
6	Confirmar el correcto estado de borrador de metodología, hacerlo para la próxima reunión.	003
7	Hablar del desarrollo del punto de análisis.	004
8	Establecer la próxima reunión para el 24/06.	
9		
10		

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Realizar las correcciones genéricas a lo largo de la memoria y mantenerlas durante el desarrollo de esta.	24/06	Javier Garzo
002	Realizar las correcciones puntuales dadas de cada punto.	24/06	Javier Garzo
003	Realizar el punto de la metodología como se a establecido en el borrado realizado para esta reunión.	24/06	Javier Garzo
004	Realizar el punto de análisis o en su defecto un borrador de este para la próxima reunión.	24/06	Javier Garzo
005			
006			

REUNIÓN:	**4ºReunión**

Fecha:	24/06/2020		
Hora comienzo:	19:30	**Hora finalización:**	20:20
Lugar:	Reunión remota a través de Microsoft teams		
Elabora acta:	Javier Garzo		
Convocados:	Javier Garzo, Violeta Monasterio		

Orden del día / Acta

No.	Asunto	Acuerdo
1	Tratar correcciones generales de la memoria: Usar puntos en párrafos largos, ser mas especifico en general para facilitar la comprensión del lector, revisar las formas verbales (algunos párrafos presentan el tiempo verbal incorrecto), incrementar el uso de referencias siempre que sea posible y poner los anglicismos en cursiva.	001
2	Hablar con respecto al objetivo principal del proyecto. El objetivo principal no está destinado a desarrollar una aplicación que combine SSI y blockchain, está más orientado al el análisis y evaluación de las herramientas blockchain para implementar SSI.	002
3	Hablar del apartado de Hyperledger en el estado del arte, este puede presentar más referencias, también los algunos párrafos necesitan más contenido y otros les sobra.	003
4	Hablar del punto nuevo de metodología: Hay que aclarar las diferencias entre tutor de la empresa y tutora de la universidad, también hay que hablar de lo segundo (actualmente solo se trata el primero).	004
5	Confirmar el correcto estado de borrador de Análisis, hacerlo para la próxima reunión.	005
6	Establecer la próxima reunión para el 03/07.	
7		
8		
9		
10		

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Realizar las correcciones genéricas a lo largo de la memoria y mantenerlas durante el desarrollo de esta.	03/07	Javier Garzo
002	El cambio de enfoque del objetivo principal hará que sea necesario modificar algunos párrafos a lo largo de la memoria.	03/07	Javier Garzo
003	Añadir todas las referencias a los *ledgers* y otras si es posible y modificar los párrafos tratados en la reunión.	03/07	Javier Garzo
004	Realizar las modificaciones necesarias para arreglar los puntos de metodología tratados en la reunión y añadir las referencias.	03/07	Javier Garzo
005	Realizar el punto de Análisis para la próxima reunión.	03/07	Javier Garzo
006			

REUNIÓN:	**5ºReunión**

Fecha:	03/07/2020		
Hora comienzo:	18:30	**Hora finalización:**	19:30
Lugar:	Reunión remota a través de Microsoft teams		
Elabora acta:	Javier Garzo		
Convocados:	Javier Garzo, Violeta Monasterio		

Orden del día / Acta

No.	Asunto	Acuerdo
1	Tratar correcciones generales de la memoria: Usar puntos en párrafos largos, espacios después antes de referencias, usar frases completas y no telegrama (Fallo común después enumeraciones).	001
2	Tratar temas objetivos, fallos menores y algunas modificaciones en los párrafos para mejorar la comprensión.	002
3	Hablar de la sección añadida de metodología (algunos párrafos sobraban).	002
4	Tratar temas objetivos, fallos menores y algunas modificaciones en los párrafos para mejorar la comprensión.	002
5	Hablar de las fechas y organización para el mes de Julio. Metodología y estudio económico para la próxima reunión	003
6	Establecer la próxima reunión para el 15/07.	
7		
8		
9		
10		

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Realizar las correcciones genéricas a lo largo de la memoria y mantenerlas durante el desarrollo de esta.	15/07	Javier Garzo
002	Realizar los cambios indicados en cada punto.	15/07	Javier Garzo
003	Realizar los apartados de metodología y estudio económico para la próxima reunión.	15/07	Javier Garzo
004			
005			
006			

REUNIÓN:	**6ºReunión**

Fecha:	10/07/2020		
Hora comienzo:	17:00	**Hora finalización:**	17:25
Lugar:	Reunión remota a través de Microsoft teams		
Elabora acta:	Javier Garzo		
Convocados:	Javier Garzo, Violeta Monasterio		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Hablar de los puntos realizados y responder dudas. Se destaca que la estructura es correcta.	
2	Responder comentarios metodología. Solución a una expresión incorrecta. Hablar respecto enseñar esta sección de memoria al tutor de la empresa.	001
3	Revisión de estudio económico. Repasar expresiones de un párrafo y hablar de incluir o excluir la documentación de la memoria).	002
4	Hablar de cambios menores. Algunos párrafos modificados y un ejemplo de una red *blockchain* que quedaba pendiente.	
5	Establecer próxima reunión para el 24/07, para cuando se debería acabar por completo la memoria.	003
6		
7		
8		
9		
10		

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Realizar la corrección de la expresión y presentar el punto al tutor de la empresa.	24/07	Javier Garzo
002	Repasar el párrafo indicado y tomar una decisión en cuanto incluir o no el coste de la documentación.	24/07	Javier Garzo
003	Acabar memoria para la próxima reunión, realizar resultados y conclusión.	24/07	Javier Garzo
004			
005			
006			

REUNIÓN:	**7ºReunión**

Fecha:	124/07/2020		
Hora comienzo:	17:00	**Hora finalización:**	17:35
Lugar:	Reunión remota a través de Microsoft teams		
Elabora acta:	Javier Garzo		
Convocados:	Javier Garzo, Violeta Monasterio		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	Hablar del problema del documento equivocado. El correo contenía otro documento diferente a la memoria y esta no ha podido ser revisada.	
2	Tratar un conjunto de dudas que surgieron durante el desarrollo del proyecto.	001
3	Tratar un conjunto de temas relacionados con la estructura del proyecto y algunos temas no relacionados con el correcto estado de la memoria (ejemplo: Como funciona el sistema de consentimiento para hacer público el proyecto).	002
4	Organizar las revisiones restantes para poder generar la versión final. Las revisiones serán enviadas la semana que viene.	003
5		
6		
7		
8		
9		
10		

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Realizar el desarrollo de la cronología, pendiente en el punto de resultados, y enviársela a la tutora.	27/07	Javier Garzo
002	Consultar documentos firmados para ver si es posible hacer público el proyecto o no.	31/07	Javier Garzo
003	Una vez recibido las correcciones realizar las modificaciones oportunas y crear la versión final.	31/07	Javier Garzo
004			
005			
006			

11.4. Anexo – Documentación adjunta

Como documentación complementaria se aporta los siguientes elementos:

- Autorización del TFG: Autorización dada por la tutora para la defensa del TFG.
- web biblioteca: Contiene la web usada por la biblioteca, compuesta por archivos Html, Javascript y Css. Además, contiene un readme.txt con una explicación mas extensa de su contenido.
- *App* usuario: Contiene todo lo relacionado con la *app* del usuario, el .apk y el proyecto de Android Studio, así como un readme.txt.
- Indy-sdk + API REST: Contiene el API REST creado durante el proyecto, así como un archivo de postman de llamadas con datos de ejemplo y un readme.txt que explica como ejecutar el API REST.