Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes

Implementación de un sistema de transporte de datos entre servicios y clientes basado en la cadena de bloques

Por

Br. Víctor Manuel Rojas

Tutor: Ph D. José Luis Paredes

Enero, 2019

Implementación de un sistema de transporte de datos entre servicios y clientes basado en la cadena de bloques

Br. Víctor Manuel Rojas

Proyecto de Grado — Sistemas Computacionales, 60 páginas

Resumen: En la era digital, cuando la seguridad está en boca de todos y cada persona del planeta es un usuario en algún sistema, resguardar los millones de datos que se generan cada segundo es de vital importancia cuando todos están conectados. Pero, ¿cómo se está seguro que los datos están resguardados? Durante años se ha hecho hincapié en la seguridad, pero es ahora en esta época contemporánea cuando la privacidad se ha vuelto un tema de interés. Es por esto, que todos los sistemas que tienen contacto con el usuario han visto sus medidas de privacidad en aumento, en pro de darle al usuario la sensación de seguridad que tanto necesita. En tal sentido surge *blockchain*, un sistema sobre el cual almacenar información de forma segura y pública donde cada persona pueda acceder a los datos, pero que no sean de utilidad para más nadie que el usuario que los creó. Por lo tanto, la investigación surge con el propósito de profundizar sobre contratos inteligentes y mejorar la estructura de un proyecto de la cadena de bloques, basado en grafos acíclicos dirigidos para mejorar la seguridad, escalabilidad y la trazabilidad de los datos.

Palabras clave: cadena de bloques, criptografía, contratos inteligentes, seguridad, capas, internet de las cosas.

Índice general

Índice de tablas

Índice de figuras

Índice de algoritmos

Capítulo 1

Introducción

En 2018, Satoshi Nakamoto surge con una idea que no solo crearía una serie de divisas con alcance global, sino que emerge como un sistema en el cual se puede basar el futuro del manejo de la información. La cadena de bloques o *blockchain*, término en inglés, presentada en 2018 con *bitcoin*, trae consigo un conglomerado de protocolos que permiten el manejo de la información como elementos inmutables, que viene perfecto para manejar transacciones de una forma que en el pasado solo se había teorizado. Sin embargo, *bitcoin* no tuvo tanto auge de un día para otro, el crecimiento como divisa permitió que la tecnología obtuviera un aumento de interés por parte de centros de investigación hasta ser uno de los focos de los mismos.

Con esto en mente, este proyecto de grado tiene como finalidad usar una tecnología como la cadena de bloques pero con un punto de vista a la seguridad en ambientes de uso cotidiano, donde la mala decisión de un servicio puede generar problemas en el día a día del ser humano. Es por esto, que se propone crear un sistema que basado en una cadena de bloques permita enviar datos desde múltiples clientes a múltiples servicios, y que estos sean capaces de responder basado en un lógica concreta.

De igual forma, dotar a la red existente de un dinamismo en los datos, con el uso de los contratos inteligentes. Donde mediante un conjunto de reglas poder definir procesos automáticos que la cadena de bloques ejecutará para realizar cambios en la red.

Para esto, se pretende usar una metodología que permita un aprendizaje a medida que transcurre la investigación y el desarrollo, en consecuencia, una planificación

incremental surge como la mejor solución al problema a tratar.

1.1. Antecedentes

En la actualidad existen cientos de proyectos basados en la cadena de bloques para dar confianza, uno de ello es Benchoufi y Ravaud (2017) que propone un proyecto que tiene como objetivo la correcta implementación de *blockchain* en el sector médico, donde exponen las condiciones para implementar el protocolo en un ambiente para el que no fue diseñado y los cambios que aplicarle para su correcto funcionamiento. De igual forma, Nugent y cols. (2016) con una idea similar a la anterior, demuestran como adaptar los contratos inteligentes a ambientes médicos donde, usando *ethereum*, logran implementar con éxito una estructura que incluye sistemas tradicionales de la medicina como la *CDMS*, del inglés *clinical data management systems*, el cual es el objetivo de todos los sistemas basado en la cadena de bloques, el de integrar el protocolo con sistemas tradicionales.

En otro orden de ideas y apostando un poco más, Pop y cols. (2018) propone todo un sistema de energía basado en la cadena de bloques, donde gracias a contratos inteligentes se logre gestionar de forma automática cada uno de los servicios energéticos de toda una ciudad. Basado en los contratos inteligentes de *ethereum* logran ejemplificar las ciudades inteligentes unificando conceptos como el de Internet de las cosas (*Iot*) y *blockchain*.

Los aportes vistos anteriormente, ejemplifican como implementar *blockchain* como *ethereum* en otros sistemas. Sin embargo, Kosba y cols. (2016) propone su sistema *hawk*, el presenta una versión privada de los contratos inteligentes, con miras a que no sea necesario la existencia de un programador experimentado para realizarlos.

Uno de los campos de investigación con más énfasis en la actualidad, es la implementación de la cadena de bloques en el Internet de las cosas (IoT), donde la importancia de no comprometer dispositivos como los presentes en estos ambientes es fundamental. Es por esto, que la posibilidad de agregar algún tipo de sistema inteligente al proceso de verificación de los miembros de la red, puede generar una mejor evolución que los presentados en algoritmos tradicionales. En este orden de ideas, Outchakoucht

y cols. (2017) proponen una inteligencia artificial basada en entrenamiento reforzado para gestionar la seguridad de la red. Donde, apoyándose de ésta, posicionan a los miembros de la red en función de su confianza, usando uno de los pilares de protocolos de consenso como el de participación; que consiste en permitir validar bloques dentro de una cadena basándose en la confianza de los miembros de la red.

Otra de las características a tener en cuenta es una arquitectura para la red que permita mantener la seguridad, y más cuando son entornos donde los datos son de vital importancia debido a que se gestionan procesos de industrias enteras. Para esto, una propuesta hecha por Yuan y Wang (2016) es la de generar un desarrollo basado en capas donde cada de una de estas lleva un proceso. La primera, la denominan física, donde se gestiona el proceso de comunicación con los dispositivos en el campo, ésta se encargaría de gestionar los medios por los cuales se envían los datos. Luego siguen con la capa de datos, que se centraría en el manejo en general de la cadena de bloques, donde se atacarán temas como el del protocolo de encriptación o cómo generar los *hash*. La siguiente capa, habla sobre los procesos que usan los miembros de la red para la comunicación de mensajes internos, por esto lleva el nombre de capa de red, luego presentan la capa de consenso donde el objetivo es definir si se usará prueba de trabajo o algún tipo basada en confianza como la prueba de participación distribuida. En la siguiente capa, se tratará los contratos inteligentes donde se pretende dar dinamismo a los procesos de la cadena de bloques, por ultimo, la capa de aplicación donde se toca temas como los de la creación de una interfaz de programación, aplicaciones o la adaptación para permitir que otros procesos usen la cadena de bloques.

Estas propuestas muestran teóricamente como gestionar de una manera más avanzada la seguridad en una cadena de bloques, y más aún como adaptarla para ambientes del uso cotidiano, donde tendrán que trabajar en conjunto con sistemas tradicionales, por lo que son un aporte de la comunidad a tener en cuenta.

Para finalizar, el trabajo realizado por Guerrero (2020), tiene como objetivo dotar al protocolo *blockchain* de escalabilidad, a través de la inclusion de grafos acíclicos dirigidos (DAGs), eliminando limites teóricos, como los presentados en *bitcoin*, con esto en mente, la propuesta también adapta la red, para dejar atrás el protocolo de prueba de trabajo, por uno basado en transacciones como prueba de participación. Guerrero

(2020), logra adaptar una red *blockdag* funcional, permitiendo este desarrollo, ya que esta investigación es una continuación de la expuesta por este autor.

1.2. Planteamiento del problema

Durante años, la seguridad ha sido unos de los ejes en todos los centros de investigación del mundo. Desde el inicio de la computación como producto de consumo masivo, creció un interés por conocer lo que cada persona efectúa en su entorno, es por esto, que se han desarrollado infinidad de protocolos de seguridad para prevenir vulnerabilidades.

Con el avance de la tecnología surgieron los ambientes distribuidos que también trajeron problemas de seguridad a solventar. Con esto en mente, Nakamoto (2008) surge con una brillante idea: *bitcoin* una criptomoneda que usa el protocolo de la cadena de bloques o también conocida por su término en inglés como *blockchain*, que aprovecha el ambiente distribuido para validar los datos y generar inmutabilidad. Con el pasar del tiempo, la idea evolucionó y surgieron nuevas tecnologías basadas en la de Nakamoto, como Wood (2014), la cual además de mejorar escalabilidad y rapidez, tenían evoluciones más significativas como los contratos inteligentes, que otorgaron un dinamismo que *bitcoin* en 2008 no tenía en mente. Esto provocó que la cadena de bloques captara un interés mayor para generar productos que funcionaran sobre ellas, pero trajeron algunas vulnerabilidades que actualmente están en investigación.

La cadena de bloques es una herramienta que viene a revolucionar el ambiente tecnológico, pero como toda nueva tecnología necesita un tiempo para madurar y solventar el problema para el que fue hecha, el cual es generar seguridad en todo tipo de transacciones o manejo de información.

En multitud de sistemas, existe una transferencia de datos entre los clientes y los servicios, en este proceso siempre se ha buscado que sea seguro, pero más aún, auditable. Un servicio puede gestionar el control de sistemas de alta relevancia, un error en la decisión del servicio causado por errores en los datos, o vulnerabilidades en el sistema puede causar inmensurables pérdidas. De ahí que se piense en la idea de cubrir con mayor esfuerzo estas vulnerabilidades con un protocolo como *blockchain*.

Debido a las carencias en seguridad de los sistemas actuales y la incapacidad de auditar de *blockchain*, se plantea un sistema basado en este último pero, con un conjunto de alteraciones en el protocolo para brindar rastreabilidad, que permita de igual forma cubrir aplicaciones mediante el acceso a una API.

1.3. Justificación

La cadena de bloques surgió con el propósito de resolver un problema de seguridad en ambientes distribuidos, fue una revolución como tecnología, pero como toda primera idea tiene sus limitaciones. En 2015 surgió *ethereum* con el propósito de resolver muchos problemas que traía consigo *bitcoin*, sin embargo, siguieron los avances y actualmente existe una nueva red de aplicaciones que necesitan requerimientos especiales que la cadena de bloques en sus protocolos no tiene en cuenta. *Bitcoin* por un lado no permite rastrear los orígenes de las transacciones, esto concede anonimato y seguridad al consumidor, sin embargo en otros ambientes esto no es deseado, cuando se habla de entornos donde la veracidad es necesaria, poder auditar en todo momento la información y los eventos que están sucediendo es una característica vital, más aún sin perder seguridad. Es por esto que existe una matriz de investigación fundamentada en generar plataformas de transporte de datos rastreables basados en la cadena de bloques.

Blockchain es de esas tecnologías que ha generado una matriz de opinión a nivel mundial, gracias a que *bitcoin* ha logrado subir en las tasas de cotización, permitiendo así que la investigación y el desarrollo en la tecnología aumentara de manera sustancial. De igual forma, su desarrollo más que probado en varios ámbitos le da una madurez a la tecnología que pocas han tenido, teniendo en cuenta el poco tiempo que tiene de haber salido al mercado como producto. Sin embargo, la idea de usar los miembros para generar seguridad ha tenido vulnerabilidades, el ataque del 51 % es uno de los más conocidos, donde se adquiere el control de la red para validar transacciones externas, esto ha causado grandes pérdidas de activos, lo que demuestra que no es un protocolo perfecto.

Y es que más allá del mercado de criptodivisas que se ha creado alrededor de la

tecnología, ésta abre una nueva era en el desarrollo de software que todavía no se ha explorado, que permite llevar un paso adelante el desarrollo de aplicaciones disruptivas que sobre la cadena de bloques brinden una seguridad basada en confianza.

Es por esto, que se plantea implementar un sistema que cuente con todas las bondades de la cadena de bloques, y que además, esté pensado para ambientes en los cuales la verificación de los datos sea el objetivo principal, así como la seguridad de los mismos y la flexibilidad que para cualquier servicio sin importar su arquitectura sea capaz de entrar en la red.

1.4. Objetivos

1.4.1. General

Implementación de un sistema de transporte de datos entre servicios y clientes basado en la cadena de bloques.

1.4.2. Específicos

- Estudiar y seleccionar el protocolo de software sobre el cual fundamentar el desarrollo.
- Generar una arquitectura que permita realizar envíos seguros de información usando la cadena de bloques.
- Adaptar la arquitectura propuesta para realizar auditorías de los eventos en la cadena de bloques.
- Implementar la arquitectura de seguridad a la cadena de bloques basada en grafos acíclicos dirigidos.
- Desarrollar una demostración de rendimiento para verificar que el desempeño de la cadena de bloques no se haya visto afectado.
- Realizar pruebas de seguridad para confirmar el desempeño de la plataforma.

1.5. Metodología

Se utilizará una metodología de desarrollo incremental, el cual consiste en una serie de iteraciones con el propósito de tener un producto cada vez más robusto con el apoyo de los conocimientos de cada iteración pasada, donde el proceso consiste en:

- Etapa de inicialización
- Etapa de iteración
- Lista de control de proyecto

La razón por la cual se eligió ésta metodología, es debido a que muchos de los focos de investigación actualmente se encuentran en modelos teóricos, por lo cual, con cada iteración se van a probar y comprobar su funcionalidad para el objetivo general.

1.6. Estructura del documento

En esta sección se tratara la forma en que el proyecto esta estructurado, así como también el tema a tratar en cada uno de los capítulos, por lo tanto, el proyecto de grado se estructura de la siguiente manera. El capítulo 2 presenta las bases teóricas necesarias para entender todos los avances logrados en el proyecto. Se da un repaso al funcionamiento del protocolo *blockchain*, así como a los usos realizados por diferentes empresas a nivel mundial, se destacan las mas conocidas como lo son *bitcoin* y *ethereum*.

En el capítulo 3 se presenta todos las modificaciones realizadas a la red para otorgar seguridad, de igual forma, la inclusión de contratos inteligentes para obtener dinamismo en la red. También se diseña la arquitectura basado en el diseño en capa, y las modificaciones al protocolo de comunicación de *blockdag*

El capítulo 4 se realizan un conjuntos de pruebas para la comprobación de que los objetivos fueron cumplidos, así como también que esta no ha perdido rendimiento.

Finalmente, en el capítulo 5 se presentan las conclusiones y recomendaciones del trabajo realizado.

Capítulo 2

Marco Teórico

En este capítulo se tratarán todos los conceptos necesarios para entender el desarrollo realizado, empezando por los términos más conocidos del área, para luego proseguir con el funcionamiento de las redes que marcaron tendencia. Terminado con los métodos de seguridad que comparten las diferentes *blockchain* y con los diferentes proyectos que han usado la tecnología para resolver un problema.

2.1. Cadena de bloques

Actualmente existe un mercado mundial de cambio de criptodivisas en el cual el eje central se llama cadena de bloques o *blockchain*, donde Swan (2015) define la tecnología como un protocolo de software que permite la transferencia segura de dinero, activos e información a través de Internet, sin la necesidad de un intermediario externo, como bancos u otras instituciones financieras. Por lo que si se presta atención no solo lo refiere a activos, sino también a información, el cual es unos de los elementos que más se han generado en el siglo XXI y donde más esfuerzos en investigación se han hecho a través de los años. Por esto es que *blockchain* adquiere un valor como tecnología a la par de la Internet.

Y es que el sistema está hecho para perdurar o por lo menos así lo establece Swan (2015), donde además de exponer todas las virtudes de la cadena de bloques, nos alenta que el futuro de la información será basada en *blockchain* o en alguna tecnología

proveniente de esta.

2.1.1. Bloque

Cada bloque posee una serie de información característica que lo diferencia de otros. Mediante un proceso de conversión a *hash* se obtiene un identificador único que ante la más mínima diferencia en los datos del bloque, causará la modificación total de *hash*. Con esto en mente, cuando un nuevo bloque es insertado, entre los datos que contiene, está presente el *hash* del bloque anterior, lo cual crea una conexión entre los datos del anterior bloque y el siguiente. Al modificar los datos del bloque anterior su *hash* cambiará, ocasionando que el del siguiente también, lo que forma una relación de dependencia directa entre todos los bloques; ahora bien, este proceso permitiría que si se conoce el *hash* del bloque génesis o primer bloque, se podría cambiar la cadena entera. Para esto se hace uso del ambiente distribuido donde cada acción de la red tiene que estar validada por la misma, es decir, que debe poseer la mayoría de la red si se quiere realizar un ataque.

Esto elimina problemas como el del doble gasto donde Swan (2018) explica que la cadena de bloques usa un sistema de firmado digital que permite hacer un rastreo de todos los usuarios que han poseído la divisa hasta el poseedor actual, lo cual permite en todo momento evitar el problema antes mencionado, causado por la clonación de divisas fraudulentas.

En otras palabras, la cadena de bloques es una base de datos que puede ser compartida por una gran cantidad de usuarios en forma directa. Que permite almacenar información de forma inmutable y ordenada; de esta forma Retamal y cols. (2017) define el protocolo que cambió el mundo de la información.

2.1.2. Participantes

El núcleo detrás de la cadena de bloques decae en sus miembros, debido a que todos los protocolos que presenta la cadena de bloques están basado en ellos.

Dependiendo de la red que se este tratando, los participantes pueden tener más o menos roles. En el caso de *bitcoin* Nakamoto (2008), describe 3 tipos de usuarios:

los emisores, que son aquellos que sólo generan transacciones; los de conexión, los cuales se encargan de transmitir las transacciones y por último, los mineros que son los encargados de validar transacciones.

Por otro lado *ethereum* presenta un sistema donde la mayoría de nodos son participantes completos, estos tienen que tener toda la copia de la red, siendo cientos de Gb de memoria, además de tener la tarea de generar y validar transacciones. También existen los participantes simples, que se asemejan a los creadores de transacciones de *bitcoin*. Sin embargo, en la actualidad casi todos los miembro de *ethereum* son completos.

Cada vez más redes usan sus participantes, para generar seguridad. La prueba de trabajo rinde muy buenos frutos, pero son los miembros los que en el futuro otorgarán seguridad a la red. Al final, las personas que usan el sistema son las más interesadas en que la red funcione de forma óptima y segura.

Si se observa la Fig. 2.1, se puede ver que cada transacción es enviada desde un participante de la red, de igual forma sucede con su almacenamiento, cada uno de los miembros puede almacenar desde una parte hasta todas los datos de las transacciones, es decir, que los miembros de la red brindan un sistema de respaldo que cualquier sistema tradicional soñaría; en el caso de la Fig. 2.1, todos los miembros hacen el papel de unos miembros completos, al crear transacciones y enviar transacciones.

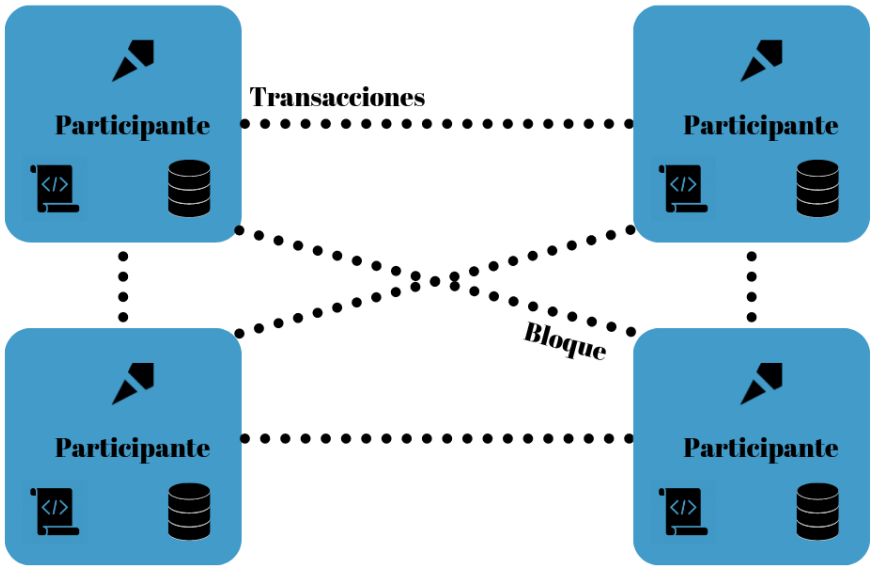

Figura 2.1: Rol de participantes. Fuente Preukschat (2017).

2.2. Redes que marcaron tendencia

2.2.1. Bitcoin

En Nakamoto (2008) define *bitcoin* como una versión de envío punto a punto de dinero electrónico, que permitió que los pagos en línea se envíen de una parte a otra sin pasar por una institución financiera. Con el propósito de generar esta idea nueva, se desprenden un conjunto de nuevos conceptos y protocolos que no solo sirven para crear *bitcoin* sino que embarcaron un nuevo paradigma de la programación de la seguridad para sistemas informáticos.

En *bitcoin*, la información añadida a la *blockchain* es pública y puede ser consultada en cualquier momento por cualquier usuario de la red. La información solo puede ser añadida a la cadena de bloques si existe un acuerdo entre la mayoría de las partes. Transcurrido un cierto tiempo, se puede asumir que la información agregada en un bloque ya no podrá ser modificada o que es inmutable. La creación de nuevos bloques es realizada por nodos denominados “mineros”; estos son nodos de la red que participan en el proceso de escritura de datos en la *blockchain* a cambio de una recompensa

económica. La validez de la escritura de un bloque por parte de un minero es revisada y acordada tácitamente por el resto de participantes. El proceso que permite alcanzar un consenso con garantías entre los mineros de la *blockchain* para el orden de escritura de bloques es la denominada prueba de trabajo o *Proof-of-work* (PoW). En concreto, para que un bloque sea aceptado, el minero tiene que ser el primero en completar una PoW para el siguiente bloque de la *blockchain*.

El PoW es un rompecabezas matemático de dificultad ajustable. En particular, la PoW consiste en encontrar un parámetro (*nonce)* que consiga que al hacer el *hash* sobre todo el bloque (incluido el *nonce*) se obtenga un valor inferior a la dificultad actual establecida por la red. Dicho de otra forma, se trata de encontrar un *nonce* que consiga un valor *hash* del bloque con un determinado número de ceros al inicio. Debido a las características de la función de *hash*, no es posible calcular estos valores analíticamente, es decir, para obtener un bloque válido, el minero debe recurrir a la fuerza bruta: probar valores del parámetro *nonce* hasta hallar uno válido. El proceso de probar valores o fuerza bruta es un proceso computacionalmente costoso, de ahí que este mecanismo se conozca como «prueba de trabajo».

La PoW hace que la creación de bloques con la intención de subvertir el consenso, tenga un coste alto para el atacante. Por otra parte, la dificultad de este rompecabezas criptográfico es fácilmente ajustable: se puede incrementar la dificultad aumentando el número de ceros necesarios para completar la PoW o decrementarla reduciendo dicho número de ceros. En particular, en *bitcoin* la dificultad se reajusta cada 2016 bloques (que equivalen a catorce días), con tal de que la creación de nuevos bloques tenga una frecuencia aproximada de un bloque cada diez minutos.

2.2.2. Ethereum

En Wood (2014) promueve un paso más allá en la cadena de bloques en la cual explica que *ethereum* es una criptomoneda que mejora ciertos aspectos en términos del rendimiento y permite la especificación de condiciones de pago complejas en las transacciones mediante el uso de contratos inteligentes. Las principales mejoras respecto a *bitcoin*, son cambios en el protocolo de red que permiten bajar el tiempo entre bloques de 10 minutos a 14 segundos, haciendo posible que se realicen más transacciones por

unidad de tiempo.

Al hablar de contratos inteligentes, Cámara Albuixech (2018) los define como un programa informático que ejecuta acuerdos establecidos entre dos o más partes haciendo que ciertas acciones sucedan como resultado de que se cumplan una serie de condiciones específicas. Es decir, cuando se da una condición programada con anterioridad, el contrato inteligente ejecuta automáticamente la cláusula correspondiente.

Vileriño (2017) también expresa que *ethereum* puede ejecutar transacciones más complejas que *bitcoin* mediante contratos inteligentes, su ejecución está acotada por el dinero total a disposición en las cuentas. Cada instrucción ejecutada, *byte* transferido o almacenado en memoria, tiene un costo, denominado *gas*. Cuando se crea una transacción, se define el valor de los campos *GasPrice*, y *GasLimit*.

GasLimit determina cuánto puede ejecutar esa transacción como máximo. *GasPrice* hace referencia a la conversión entre *Ether*, la moneda de *ethereum*, y el *gas* usado por la transacción. Por otro lado, análogamente al concepto de direcciones en *bitcoin*, *ethereum* introduce la noción de cuentas, cuyo estado se actualiza luego de la ejecución de las transacciones de cada bloque.

Por otro lado, Swan (2018) al hablar de *ethereum* lo define como una plataforma descentralizada. Diseñado para ejecutar contratos inteligentes. *Ethereum* es una red pública de *blockchain* para que cualquiera puede descargar una billetera y usarla. No hay un punto único de control o fracaso, y además resistente a la censura. *Ethereum* es una plataforma de aplicación *blockchain*, una superposición a Internet que es otra capa y plataforma sobre la que se pueden construir aplicaciones distribuidas.

Como explica Swan (2018), el elemento central de la plataforma *ethereum* es la máquina virtual *ethereum* (Del inglés *Ethereum Virtual Machine*), que sirve como modelo de ejecución para contratos inteligentes. El contrato en primera instancia no puede ser ejecutado por el EVM, para esto es pre-compilado a un lenguaje de más bajo nivel, que es pasado al EVM para su ejecución. Cada uno de las participantes de la red cuenta con la maquina virtual para ejecutar contratos y hacer validaciones.

2.3. Protocolos de seguridad presentes en la cadena de bloques

Uno de los primeros conceptos para el proceso de validación de identidad es el explicado por Llumiugsi y cols. (2018), él lo define de la siguiente manera: La clave pública utiliza mecanismos criptográficos que se acoplan a un par de claves asimétricas: una clave pública y una clave privada. El cifrado de clave pública utiliza ese par de claves para el cifrado y el descifrado. La clave pública como su nombre lo indica se hace pública y se distribuye amplia y libremente. La clave privada nunca se distribuye y debe mantenerse en secreto.

Otro de los protocolos que permiten verificar seguridad de forma rápida es el explicado por Vileriño (2017) un Árbol de Merkle el cual es un árbol binario de *hashes* criptográficos, en donde cada nodo contiene el *hash* de la concatenación de sus dos hijos. En la Fig. 2.2 se puede ver cómo se forma la raíz de este árbol de *hashes*, usando todas las transacciones que se encuentran en el cuerpo del bloque.

El árbol de *hash* de Merkle permite almacenar diversas piezas de información independiente (en el caso de *bitcoin* son transacciones económicas) en las hojas de una estructura en árbol. Para formar el árbol, se hace un *hash* de la información contenida en cada nodo hoja. Cada nivel superior del árbol se concatenan diversos valores *hash* del nivel inferior (dos valores si el árbol es binario) y se le aplica la función *hash* a esta concatenación tal como se ve en la Fig. 2.2. Repitiendo este proceso se llega a un nivel donde hay sólo un nodo, denominado la raíz del árbol. La ventaja de esta estructura en árbol es que se puede consultar la presencia en dicho árbol de los datos de un cierto nodo/hoja de forma autenticada y sin tener que disponer de toda la información que almacena el árbol. En particular, se puede consultar de forma autenticada cualquier contenido del árbol con una cantidad de valores *hash* proporcional al logaritmo del número de nodos del árbol. Esto es para validar un contenido únicamente hay que proporcionar los nodos adyacentes en cada nivel y el nodo raíz (que tiene contribución de todos los datos almacenados en las hojas) autenticado. Entonces, para validar un contenido se calcula el valor raíz a partir de los nodos adyacentes proporcionados y se comprueba que coincide con el valor raíz autenticado. La estructura es segura porque

no se puede generar un conjunto de nodos adyacentes a voluntad que dé como resultado el valor del nodo raíz autenticado.

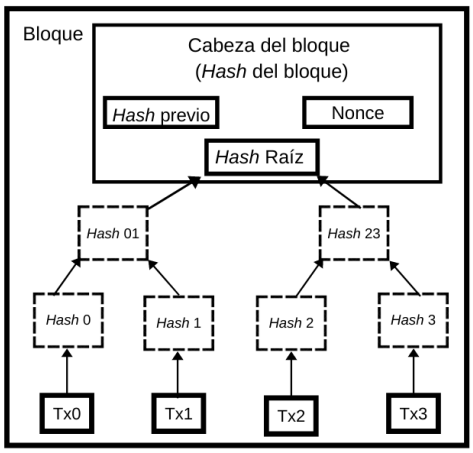

Figura 2.2: Estructura del árbol del Merkle. Fuente Nakamoto (2008).

2.4. Uso de la cadena de bloques en aplicaciones de terceros

Según un estudio realizado por David Schatsky (2018), en GitHub, existían al momento 86034 proyectos que utilizan *blockchain* pero solo un 8 % se encontraban activos. Varios de estos proyectos corresponden a la creación de plataformas que permitan desarrollar aplicaciones utilizando *blockchain*.

IBM es una de las organizaciones que se ha dedicado a realizar acuerdos con diferentes compañías e implementar casos de uso que utilicen la tecnología *blockchain*. Ejemplo de ello es su asociación con Walmart, empresa con la cual se encuentran desarrollando un prototipo que permita rastrear el origen de los alimentos que se venden en la cadena de supermercados de forma transparente, con el objetivo de garantizar la seguridad de los mismos a los consumidores (Galvin, 2017). Otro ejemplo, es el resultado de su asociación con Hejia, con quienes desarrollaron una red basada

en *blockchain* entre proveedores del mercado farmacéutico, sistema que se encuentra actualmente en producción. Además, IBM cuenta con un acuerdo con la *Food and Drug Administration (FDA)* con quienes planifican comenzar a realizar un proyecto para garantizar el intercambio seguro de registros de salud (Engelhardt, 2017) y con *Energy Blockchain Labs* con el objetivo de utilizar *blockchain* para hacer más eficiente la generación de insumos de carbono reduciendo su impacto en el medio ambiente (Andoni y cols., 2019).

Las instituciones financieras también han demostrado especial interés por la tecnología en cuestión. A modo de ejemplo, J.P Morgan lanzó en Octubre de 2017 una red basada en *blockchain* para el procesamiento de pagos. El sistema fue construido utilizando Quorum, plataforma desarrollada por J.P Morgan utilizando Wood (2014). Otra de las implementaciones existentes relativas a este rubro money transfer (2019) y Amazon.com (2019) que permiten realizar transferencias de criptomonedas, pagos de poco valor monetario como alternativa a las tarjetas de crédito y pagos de regalías respectivamente. Se profundiza sobre estos casos de aplicación en siguientes secciones así como también se analizan implementaciones relativas a la filantropía (donaciones), la industria del entretenimiento (juegos y apuestas) y sistemas de validación (identidad, voto electrónico y diplomas universitarios), entre otros.

Todos estos son casos de uso de *blockchain* que darán un futuro más seguro a el manejo de procesos, desde el control general de la vida del ciudadano hasta la gestión de seguridad empresarial de toda una cadena de productos, *blockchain* da la llaves para generar seguridad basado en confianza por la tecnología.

Capítulo 3

Implementación de la arquitectura de seguridad

Este capítulo tiene como objetivo plantear los algoritmos y flujos de procesos, necesarios para generar seguridad y dinamismo en la red. En primer lugar, una explicación general de la estructura *blockdag* desarrollada por Guerrero (2020), dando detalles sobre como las transacciones se conforman, y en general como las red es capaz de mantener seguridad. Siguiendo con el capítulo, se presentan los contratos inteligentes como métodos generadores de lógicas complejas dentro de la red, siendo este un mecanismo usado por el protocolo *blockchain* para generar cambios a la red de forma programada y sin perder seguridad. Siguiendo con el desarrollo, se tratará el sistema de comunicación entre participantes, siendo de vital importancia para una red que se basa en el intercambio de información entre miembros. Para finalizar se tratará el cambio de la arquitectura, basándose en un modelo de capas visto en Yuan y Wang (2016).

3.1. Arquitectura blockdag

La red *blockdag*, presenta una forma de manejar las transacciones, de manera que puedan ser usadas para distintos objetivos, este desarrollo no tiene pensado modificar ese formato, por lo que la estructura es la siguiente:

- Autor
- Destino
- Contenido
- Fecha de creación

Esta estructura permite el manejo de información, sin importar el contenido de la misma, la red usa el autor y destino para procesos de validación y de la entrega de los datos.

Cada bloque representa un nodo dentro del DAG, estos están relacionandos con otros bloques formando la red. Cada uno de estos bloques presentan las siguiente estructura:

- Indice
- Transacciones
- Marca de tiempo
- *Hash*
- *Hash* anterior
- Minero

El ***Hash*** y ***Hash* anterior**, permiten generar esa dependencia entre bloques que forma el DAG, durante este desarrollo, se harán ciertas modificaciones con el fin de cumplir los objetivos propuestos.

La red *blockdag*, propuesta por Guerrero (2020), utiliza las transacciones como prueba de participación para la validación de los bloques; otorga a los participantes de la red la posibilidad de elegir que procesos de la misma son válidos y cuáles no, así como también en que miembros confiar. En general este protocolo para la validación de los datos, da aun más poder a los participantes.

Como se puede observar en la Fig. 3.1, la red *blockdag* toma forma, permitiendo crecer sin limites teóricos, cada uno de los bloques forman un nodo dentro del grafo,

dotando así, a la red de una agilidad en la búsqueda de información y en los procesos de validación.

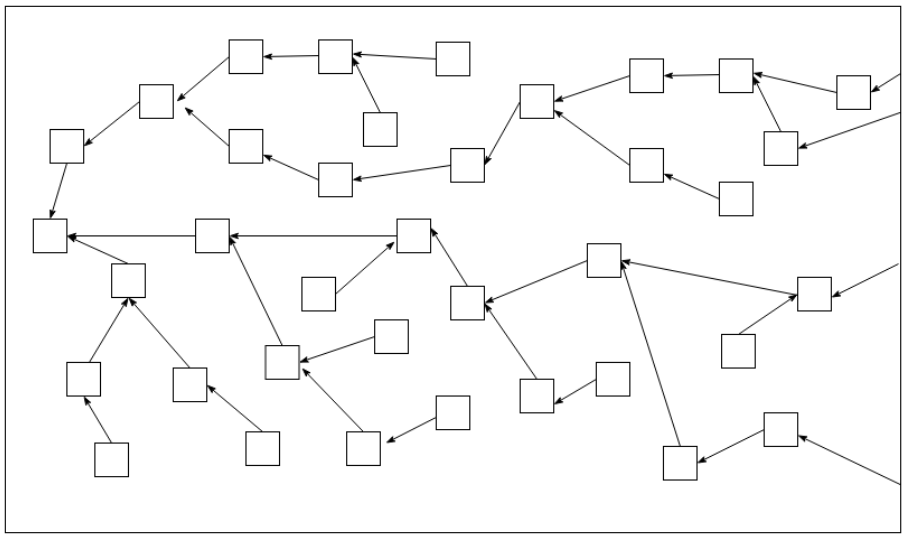

Figura 3.1: Generacion de la red *blockdag.*(tomado de Guerrero (2020))

3.2. Contratos inteligentes

Un contrato inteligente es un *script* o fragmento de código que facilita, verifica o ejecuta los requisitos de un contrato sin la necesidad de un documento de contrato o un tercero. Permite el intercambio de dinero, activos, acciones o cualquier otro objeto de valor de manera transparente sin la intervención de un tercero, como lo requieren los contratos tradicionales que necesitan de un abogado, bancos, registros o instituciones gubernamentales, es por esta razón que usando la tecnología de contratos inteligentes se logran automatizar procesos.

En el caso de *bitcoin* usa *Script*, el cual no es un lenguaje completo de turing, pero permite generar dentro de la red cierto dinamismo. Ahora bien, comparado con *ethereum* éste usa *solidity* el cual es un lenguaje completo de turing, lo que le permite a *ethereum* tener más dinamismo en cada uno de sus contratos, sin embargo, esto trae consigo un problema y es que *solidity* al ser un lenguaje completo, existe la posibilidad de vulnerabilidades, que en caso de *Script* no son posibles.

Por esta razón se implementa la siguiente arquitectura:

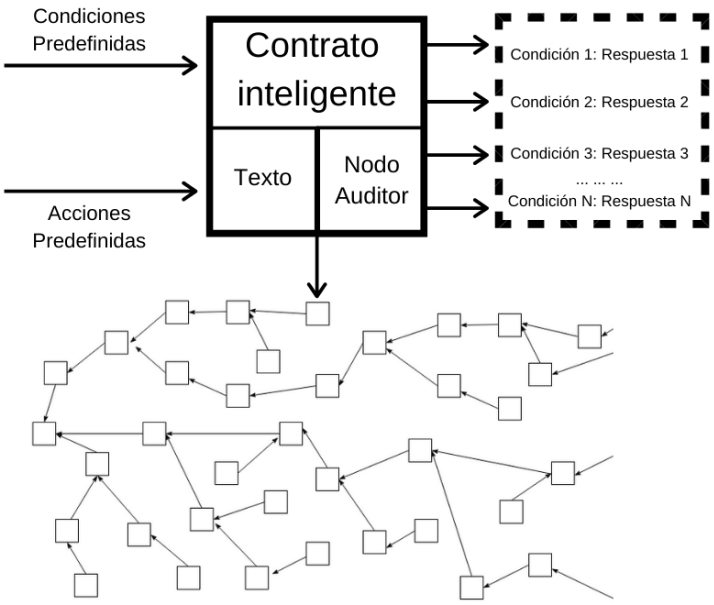

Figura 3.2: Esquema de un contrato inteligente dentro de la red.(adaptado de Guerrero (2020))

Como se puede observar en la Fig. 3.2, el grafo acíclico dirigido que usa la red le brinda la posibilidad a cada bloque de almacenar los contratos, al ser estos bloques distribuidos en participante, da la capacidad de recuperación de los datos a los participantes del sistema. Es necesario que todos los miembros de la red desaparezcan para perder información en la misma, lo que da una virtud que no tienen los sistemas jurídicos actuales; donde al ser sistemas centrales, la posibilidad de perder o pasar desapercibido una cláusula de un contrato es una realidad, lo cual es un problema que una base de datos distribuida y autoejecutable como la que se propone en la investigación soluciona.

Para generar un contrato que sea autoverificable, autoejecutable e imperdible se propone la siguiente estructura de los datos del contrato:

- Origen

- Destino
- Texto
- Título
- Lista de regulaciones
- Lista de acciones
- Participante Auditor
- *Hash*

El primer elemento es el **origen**, el cual es usado como referencia al participante que creo el contrato, para esto se usa la clave pública única que identifica al participante de los demás, de igual forma sucede con el destino que es la referencia al participante al cual es dirigido el contrato.

El **texto** es un parámetro variable que permite agregar información extra al contrato, ya sea una cadena de texto o alguna estructura más compleja. El **título** por otro lado es auto explicativo, es el título que lleva el contrato inteligente.

Ahora bien, la **lista de regulaciones** es un parámetro de gran importancia dentro del contrato inteligente, ya que éste lleva cada uno de los elementos que son necesarios que se cumplan para ejecutar las **acciones** del contrato. Teniendo en cuenta que para permitir que estas regulaciones se adapten a la mayor cantidad de problemas posibles se presenta la siguiente estructura:

- Nombre
- Valor
- Posibles Valores
- Instrucciones

El **nombre** es un identificador que permite diferenciar a la regulación de otra, por otro lado, **valor** es una condicional que posibilita verificar si el contrato se cumplió o no y necesita continuar con su monitoreo. De igual forma, **posibles valores** es un

parámetro opcional que permite realizar verificaciones sobre el parámetro **valor**. Es decir, el campo **valor** no puede tener un valor diferente a alguno de los encontrados en el campo **posibles valores**, finalizando con el campo **instrucciones**, el cual es un algoritmo en *python* que será ejecutado para evaluar si se cumple la regla, éste recibe como parámetro el **valor** y la red, con esto se busca limitar por seguridad el rango de acción del mismo.

Siguiendo con los parámetros del contrato, las acciones son los eventos que se tienen que cumplir una vez las regulaciones están satisfechas, su estructura es parecida a la de las regulaciones:

- Nombre
- Valor
- Acción

Siendo **acción** un método que se ejecutará recibiendo el **valor**, y que gestionará un proceso dentro de la red.

El campo **participante auditor** es una referencia al participante encargado de llevar el control y verificación del contrato. Por último, el parámetro ***hash*** del contrato no es más que el resultado del algoritmo de cifrado de curva elíptica, que permite identificar dentro de la red el contrato de forma única.

Como se puede observar en la Fig. 3.3 cada contrato que solicita entrar a la red, pasa por una verificación de cada uno de los parámetros de los que esta compuesto el contrato, si este es aceptado se comprueban las condiciones para aplicar las acciones.

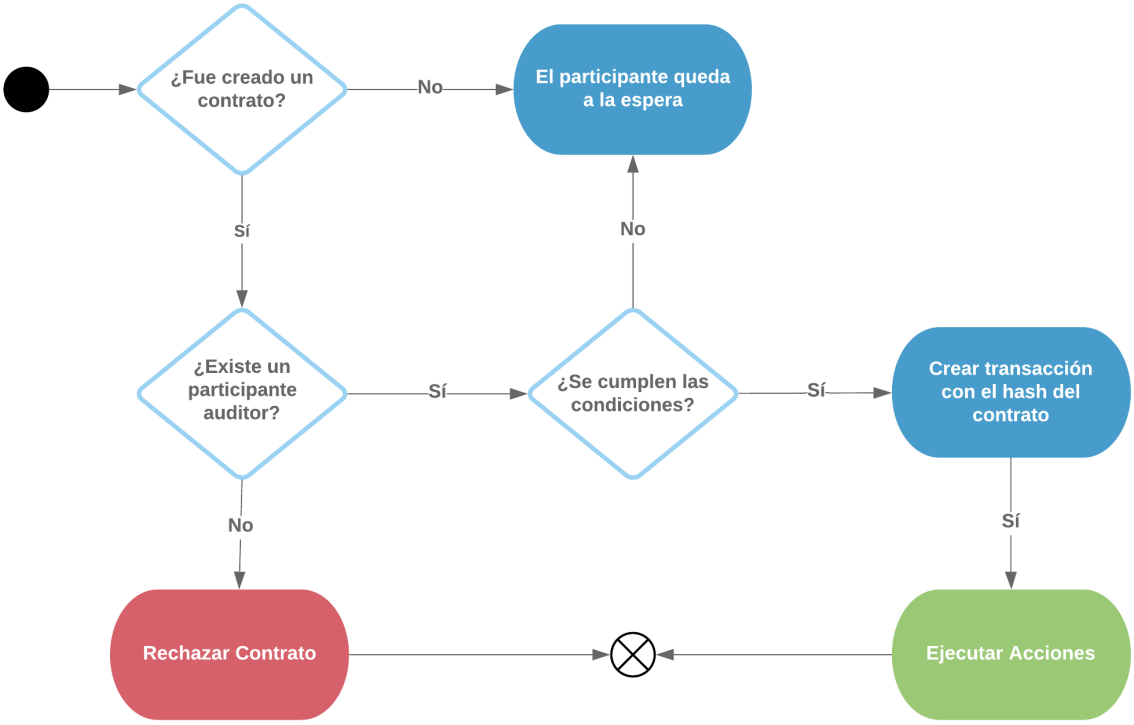

Figura 3.3: Protocolo de verificación del contrato inteligente.

La contribución del participante auditor, tiene cierta similitud con el verificador de transacciones, éste trabaja en segundo plano con la única tarea de corroborar que los contratos inteligentes se han cumplido o no. El auditor forma parte de la red de una forma diferente a la del verificador de transacciones, a diferencia de éste, no hay un protocolo de selección basado en su contribución, el auditor trabaja sobre el hecho de que se supone que desea una red segura, en la que él no pueda perder sus activos. Por esta razón, basado en el protocolo de transacciones como prueba de participación estos delegados son usados tanto para labores de verificación de transacciones, donde obtienen un premio como en la tarea de verificar contratos.

La forma en que el participante auditor interactúa en la verificación de un contrato, se puede ver reflejada en la Fig. 3.4 donde una vez el contrato es validado el participante auditor queda a la espera y en constante revisión de los mismos. Es decir, cada cierto tiempo el participante auditor verifica las regulaciones, como el contrato es asíncrono (se puede cumplir en cualquier momento), cada uno de los participantes auditores

realizan monitoreo de los datos para corroborar su finalización, de ser correcto se crea una transacción con el *hash* único del contrato que sirve como mensaje y constancia de que ese contrato ha concluido; indicando así, que no se necesita más verificación.

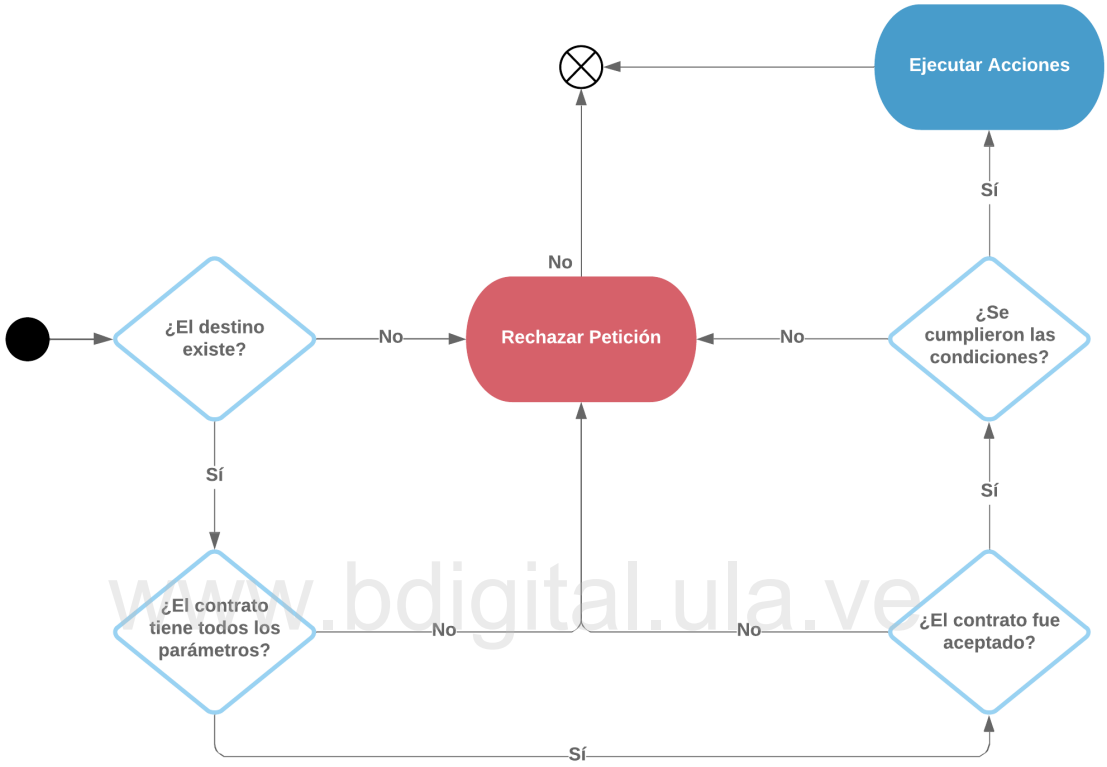

Figura 3.4: Protocolo de verificación del participante auditor.

La lógica computacional para llevar a cabo este proceso se puede observar en el Algoritmo 3.1, donde al iterar cada contrato que tenga como observado el participante actual, se busca la transacción que revela si el contrato en cuestión ha concluido. En caso de no existir, se procede a verificar regulaciones, las cuales en caso de cumplirse, continua en dos pasos, en primer lugar se crea la transacción que establece que el

contrato esta cumplido para luego aplicar las acciones.

Algoritmo 3.1: Algoritmo de verificación de contratos.

```
1  def revisarContrato(grafo, blockdag, participante, participantes):
2      arcos ← grafo.obtenerArcos();
3      para arco en arcos hacer
4          para contracto en arco.obtenerContratos() hacer
5              transaccion ← blockdag.buscarTransaccion(contrato.hash);
6              si contrato.auditor = participante.clavePublica y no transaccion
                entonces
7                  regulaciones ← contrato.regulaciones;
8                  contratoTerminado ← Verdadero;
9                  acciones ← contrato.acciones;
10                 para regulacion en regulaciones hacer
11                     si no verificarRegulacion(regulacion) entonces
12                         contratoTerminado ← Falso;
13                 si contratoTerminado entonces
14                     blockdag.crearTransaccion(contrato.hash);
15                     para accione en acciones hacer
16                         aplicarAccion(accion);
17 fin def
```


De forma similar, se presentan los Algoritmos 3.2 y 3.3, donde la verificación de regulaciones y ejecución de acciones es llevada a cabo, en función del *script* recibiendo como parámetros el contrato y la red, con esto se espera que en ninguno de los dos casos este abierto a la posibilidad de ejecutar eventos que no sean otros que los de la

misma red, los cuales ya están regulados según en los roles de la misma.

Algoritmo 3.2: Algoritmo de verificación de regulaciones.

```
1 def revisarRegulacion(regulacion, contrato, blockdag):
2     retornar blockdag.ejecutar(regulacion.instrucciones, { contrato, blockdag })
        == regulacion.valor;
3 fin def
```

Algoritmo 3.3: Algoritmo de verificación de acciones.

```
1 def aplicarAccion(accion, contrato, blockdag):
2     blockdag.ejecutar(accion.accion, { contrato, blockdag });
3 fin def
```

En términos generales, en un contrato inteligente una vez que dos o más partes aceptan todos los términos del contrato, firman criptográficamente el contrato inteligente y lo transmiten a la red entre pares (P2P) para su verificación. Cuando se activan las condiciones predefinidas, los contratos inteligentes ejecutarán automáticamente las estipulaciones del acuerdo y desencadenarán la acción de respuesta correspondiente sin ninguna intervención de terceros.

3.3. Protocolo de comunicación

Entre los procesos que más importancia tiene en la red se encuentra, la forma en que los participantes de la misma se comunican, debido a que los puntos críticos del sistema se basan en decisiones que estos toman. Es por esto que el desarrollo se basa en protocolos que la mayoría de las *blockchains* modernas usan, en tal sentido, se optó por el protocolo de sockets usado por redes como Foundation (2018) para la transmisión de mensajes en la red.

Este protocolo establece una comunicación persistente con cada uno de los participantes, de forma que cuando suceden eventos dentro de la red (como el proceso de entrada de un miembro), éste se encuentra listo para recibir los datos del evento. En

el proceso de registro en la red, el participante crea una conexión activa con cada uno de los miembros de la misma, gracias a que durante el consenso de participantes, cada uno obtiene los datos de conexión del resto, haciendo que el proceso sea computacionalmente simple. Durante la permanencia del participante en la red, estas conexiones se pueden renovar, similar a lo que sucede en la salida o entrada de un miembro de la misma.

En el proceso de creación de una transacción, el participante debe conocer la dirección de destino al que quiere dirigirla, ahora bien para que estos datos sean seguros se hace uso del protocolo de clave pública y clave privada como se observa en la Fig. 3.5. La red facilita la aplicación del protocolo debido a que entre los datos almacenados de los participantes se encuentra su clave pública, de forma que solo con conocer la dirección es posible enviar de forma rápida una transacción.

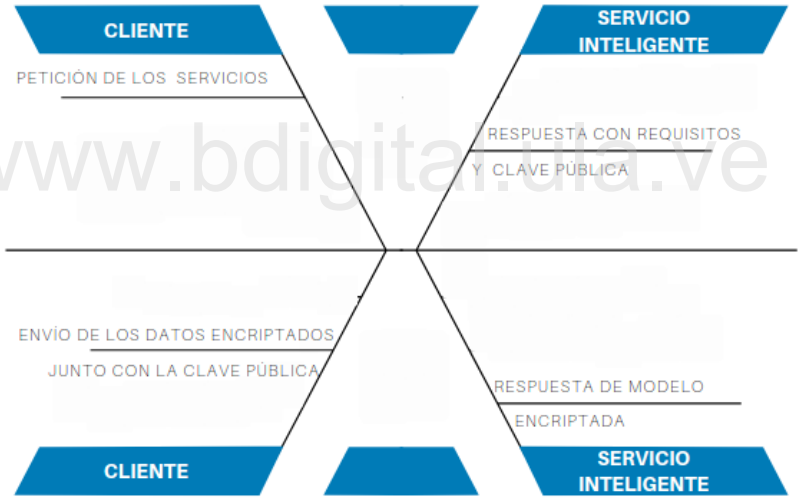

Figura 3.5: Protocolos de comunicación de la red.

Sin embargo, surge un problema ¿qué sucede si se quiere realizar un proceso de audición de los datos y estos solo son conocidos por el receptor?, en primer lugar se puede solicitar a éste que los envié también usando la red, pero ¿y si ya no se encuentra conectado? es por esto, que se hace uso dentro de la red de un participante especial. El cual como se puede observar en la Fig. 3.6, cambia la forma en que el protocolo acontece, recibiendo una copia del todo el proceso, permitiendole así, además de corroborar la

veracidad de los contratos visto en el la SECCIÓN 3.2, recibir datos de otros eventos para en caso de ser necesario realizar una audición con la información que este posee.

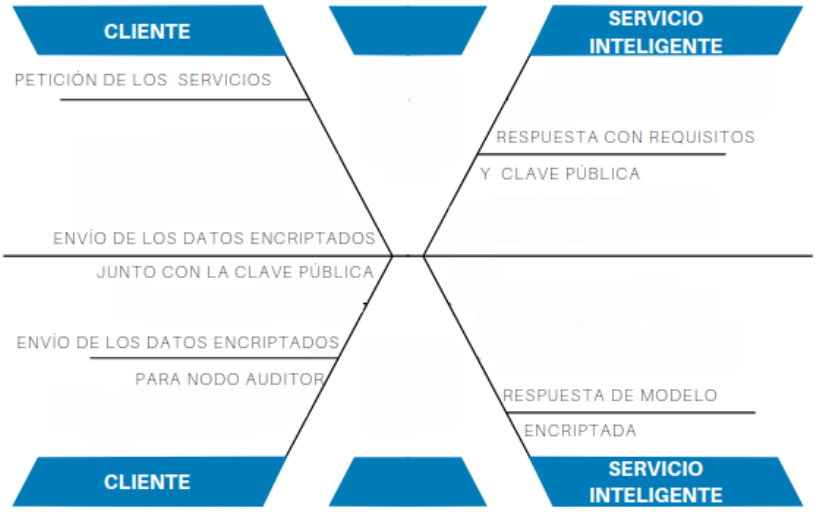

Figura 3.6: Protocolos de comunicación de la red con la presencia de auditor.

Este protocolo en la red esta implementado en el Algoritmo 3.4 donde se puede observar, en la primera parte de este, cómo el participante origen accede de forma sencilla a la clave pública de destino, esto se debe a que dentro de la información que comparte cada participante de la red sobre sus compañeros se encuentra la clave pública, facilitando completar la primera fase del protocolo. Una vez que la transacción esta realizada y confirmada, el destino tiene la posibilidad de responder la misma con

otra transacción.

Algoritmo 3.4: Algoritmo de comunicación entre participantes tomado de Guerrero (2020).

```
1  def crearTransaccion(autor, contenido, dirección):
2      origen ← fecha ← objetivo ← vacio;
3      para participante in participantes hacer
4          si autor == participante.direccion entonces
5              origen ← participante.clavePublica;
6          si direccion == participante.direccion entonces
7              contenido ← encriptar(contenido, participante.clavePublica);
8              fecha fechaActual();
9              objetivo ← participante.clavePublica;
10     blockchain.agregarTransaccion(origen, objetivo, contenido, fecha);
11 fin def
```


Ahora bien, la inclusión del participante auditor, trae consigo un conjunto de cambios, siendo el principal la creación de una transacción también a nombre del

participante auditor, estos cambios se pueden observar en el Algoritmo 3.5.

Algoritmo 3.5: Adaptación del algoritmo de comunicación entre participantes.

```
1  def crearTransaccion(verificarAutor, VerificarObjectivo, autor, contenido,
     dirección, blockdag):
2      origen ← fecha ← objetivo ← vacio;
3      auditor ← blockdag.getAuditorDisponible();
4      para participante in participantes hacer
5          si verificarAutor(autor, participante) entonces
6              origen ← participante.clavePublica;
7          si verificarObjetivo(participante) entonces
8              contenido ← encriptar(contenido, participante.clavePublica);
9              fecha fechaActual();
10             objetivo ← participante.clavePublica;
11     blockchain.agregarTransaccion(origen, objetivo, contenido, fecha);
12     blockchain.agregarTransaccion(origen, auditor, contenido, fecha);
13 fin def
```


Esta aproximación da cabida a la inundación de transacciones, debido a que con la dirección es posible generar transacciones no deseadas a un participante, sin embargo es algo que se puede solucionar agregando un costo por transacción, que es uno de los objetivos que se plantean en trabajos futuros.

3.4. Arquitectura de capas

La arquitectura de capas implementada en el desarrollo permite procesar y garantizar transacciones confiables que son intercambiadas dentro de la red. La estructura de la red tiene siete capas: *middleware*, datos, contrato, consenso, red, incentivo y aplicación.

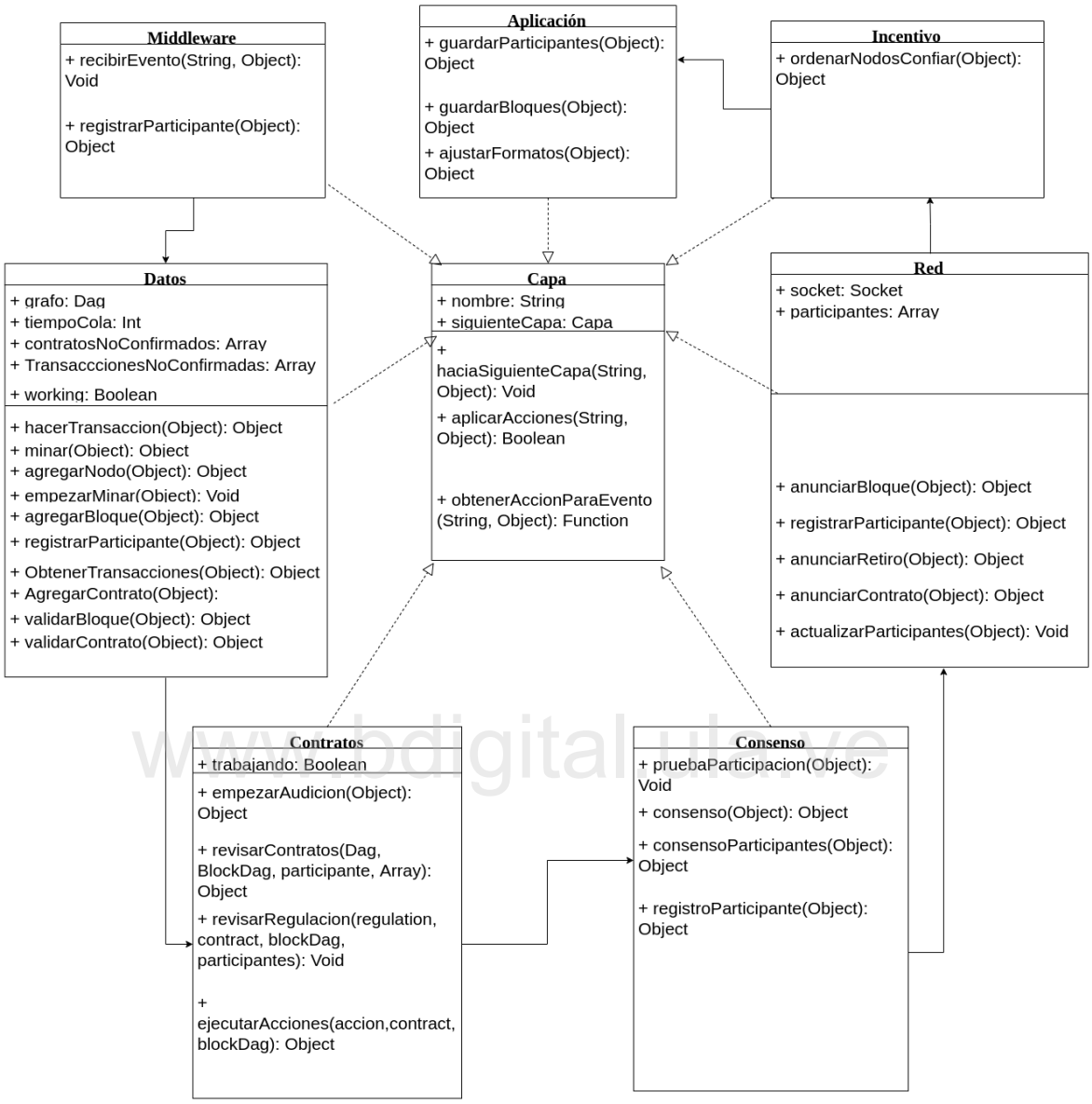

Figura 3.7: Diagrama de clases de la capas de la red.

En la Fig. 3.7 se puede observar que cada una de las capas presenta el parámetro **siguienteCapa** que sirve para llevar la dependencia entre capas, de igual forma el campo **nombre** que viene a servir de identificador en la red. El centro del flujo es llevado por la clase capas, siendo las demás especializaciones de esta; aquí se puede observar que donde más se centra la carga computacional de la red es en las capas de datos, contrato, consenso y red, porque aunque las de aplicación y *middleware* tienen

acción en casi todos los eventos, su trabajo es más genérico y difiere muy poco de evento a evento.

La forma en que se lleva a cabo el manejo de las capas puede verse representado en el Algoritmo 3.6 donde se puede observar que cada evento se ve ejemplificado en forma de una función en cada capa. En caso de que la capa que se está evaluando no presente una acción en el evento en ejecución, simplemente pasa el testigo a la siguiente capa, esto representa una implementación sencilla que permite la encapsulación de la lógica de cada capa y la fácil depuración de errores.

Algoritmo 3.6: Algoritmo de control de capas de la red.

```
1  def haciaSiguienteCapa(evento, parametros):
2      si capa.siguienteCapa entonces
3          capa.siguienteCapa.aplicarAcciones(evento, parametros)
4  fin def
5  def aplicarAcciones(evento, parametros):
6      accion ← obtenerAccionParaEvento(evento);
7      si accion entonces
8          resultado ← accion(parametros);
9          si resultado.estatus entonces
10             haciaSiguienteCapa(evento, parametro);
11         retornar resultado;
12     en otro caso
13         haciaSiguienteCapa(evento, parametro);
14 fin def
15 def obtenerAccionParaEvento(evento):
16     retornar obtenerAtributo(evento);
17 fin def
```

3.4.1. Capa middleware

La capa *middleware* debe garantizar una recopilación de datos fluida y confiable entre las entidades de la red. Proporciona los medios funcionales para que los datos recopilados sean transmitidos a la siguiente capa. Para verificar que los datos no contienen errores proporciona una función para verificar y validar datos, así como una función para corroborar que los datos presenten un formato específico, de tal forma que sea legible por todas las capas de la red.

Gracias al uso del protocolo TCP dentro de esta capa se comprueba y modifica los errores al agrupar y transferir datos de bloque a paquetes, y luego volver a calcular y comparar la suma de comprobación de cada paquete de datos. Además, realiza una tarea de control de flujo para evitar la pérdida de tramas verificando y regulando la velocidad de transferencia de datos, así como una tarea de control de errores que procesa estos para verificar si todas las tramas se han entregado a su destino en el orden correcto. De igual forma, gestiona posibles colisiones que pueden ocurrir.

De esta manera, se garantiza que los datos que son recibidos por esta capa lleguen de manera segura, ahora bien, este protocolo es más lento que el UDP así que se esta sacrificando velocidad por seguridad en el envío de información.

Si se observa en la Fig. 3.8 se puede percatar que esta capa tiene un funcionamiento genérico, donde en todos los eventos cumple la misma función, recibir datos, y una función que comprueba el formato de los mismos, en caso de que esta no confirme la veracidad, rechaza el evento entrante. Aunque esta capa no presente gran complejidad, es de las más importante porque tiene que comprobar la correctitud en formato de los datos que están ingresando a la red.

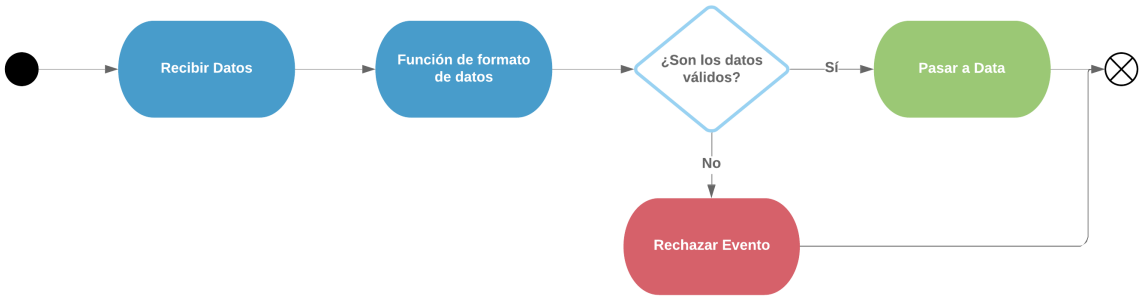

Figura 3.8: Diagrama de actividades de la capa *middleware*.

3.4.2. Capa de datos

La capa de datos proporciona los bloques de datos encadenados, incluyendo el cifrado asimétrico de los datos, el sellado del tiempo, los Algoritmos *hash* y es el lugar propicio para incluir la verificación con árboles de Merkle.

En la red *blockdag* cuando a un participante le corresponde crear un nuevo bloque, es esta capa donde se realiza la supervisión de los participantes que contribuyen en la red, verifica datos como la existencia de orígenes y destinos de las transacciones. La capa de datos se comunica con sus capas contiguas a través del paso de eventos, recibe los datos de la capa *middleware* y después de verificarlos, los envía a la capa contrato para realizar las siguientes operaciones; analiza el tráfico recibido de la capa *middleware* y clasifica los tipos de eventos. La capa de datos también comprueba los errores para garantizar la precisión en el envío de datos y organiza el orden de los datos de eventos. De igual forma, en esta capa actúa el participante validador de transacciones donde, gracias al árbol de Merkle puede hacer este proceso de forma eficiente.

La estructura de datos de la capa de datos tiene el siguiente arquetipo:

- Grafo acíclico dirigido (DAG)
- Tiempo de cola
- Contratos no confirmados
- Número de transacciones
- Árbol de Merkle

El primer parámetro es el **grafo acíclico dirigido** (DAG) que es la estructura de datos que maneja los bloques de la red. Por otro lado, el **tiempo de cola** es un parámetro para manejar el tiempo de espera promedio que están teniendo las transacciones en la red para ser verificadas, el **número de transacciones** indica la cantidad de transacciones que a manejado el participante desde que comenzó a trabajar, por último, el **árbol de Merkle** se usa con propósitos de validación.

3.4.3. Capa contrato

La capa contrato empaqueta los algoritmos y contratos inteligentes de la red, los cuales sirven como activadores importantes de los datos pertenecientes a contratos inteligentes de la misma. Como se comentó en el SECCIÓN 3.2, los contratos inteligentes son un grupo de reglas de respuesta autoverificables y autoejecutables que se almacenan dentro de la red. En esta capa se verifica que los datos que provienen de la capa de datos pertenezcan a un contrato; tiene las funciones de crear un contrato, firmar el contrato por ambas partes, registrar todos los detalles que se requieren para el contrato y ejecutar el contrato.

3.4.4. Capa de consenso

La capa consenso es la que se encarga de validar cada proceso de la red, en ella se realiza el proceso de votación para la elección de los participantes testigos, así como también, de la aceptación o el rechazo de los bloques. De igual forma, se encarga del consenso de participantes para la aceptación o rechazo de un nuevo participante conectado a la red. La capa consenso, es una de las capa más importantes de la red, debido a que tiene la responsabilidad de mantener la red segura y confiable.

En el caso de este desarrollo, la red ya contaba con un protocolo de transacciones como prueba de participación, por lo que el consenso pasa por la elección de los participantes que mayor número de transacciones presentan dentro de la red, basándose en el hecho de que al tener un gran uso de la red, a estos les interesa mantener la seguridad de la misma.

Como se puede observar en la Fig. 3.9, esta capa tiene como entrada un

evento solicitando la verificación de información, la cual puede ser un participante, transacciones o bloques. Para esto la red consultará a los participantes que el protocolo de participación determine como válidos y empezará una votación, donde en caso de tener una mayoría, se aceptará como válida la entidad escrutada y en caso contrario se rechazará.

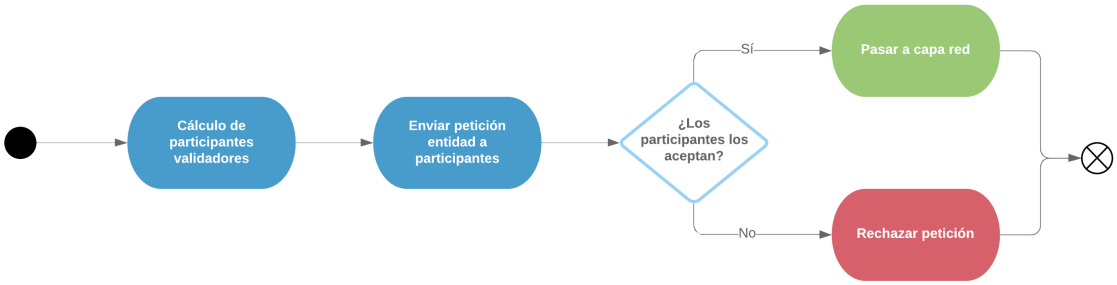

Figura 3.9: Diagrama de actividades de la capa *middleware*.

Este tipo de solución tiene la ventaja de que al hacer uso de la mayoría para la toma de decisiones, es más complicado tomar control de la red. Sin embargo hace que el proceso tenga un tiempo de validación que depende de cada participante, entre más participantes sean los que contribuyan, más tiempo va a llevar el proceso.

3.4.5. Capa de red

La capa de red debe proveer los mecanismos de red distribuida, reenvío de datos y verificación de los mismos bajo el modelado de una red entre pares (P2P).

Otras funciones de la capa de red incluyen asignar y mediar la ruta para la transferencia de datos entre sistemas de red, controlar el flujo y controlar los errores. Proporciona una función de control de congestión, que surge tras la transferencia de datos, así como la función para establecer, mantener y terminar las conexiones de red en las líneas de comunicación de datos entre sistemas y sistemas superiores.

Además, los datos del bloque se dividen en paquetes y se reúnen después de ser transferidos. Para determinar la ruta óptima para la comunicación de datos, se utiliza un algoritmo de enrutamiento para asignar la dirección lógica, y los paquetes se envían desde el remitente al receptor.

Esta capa está basada en sockets, es decir, que se busca tener una conexión lo más estable posible. Debido a que tampoco es viable que cada participante esté conectado el 100 % del tiempo, se tiene en cuenta las posibles desconexiones de los participantes, para esto, con el objetivo de mantener seguridad, el participante tiene que realizar una reincorporación completa cada vez que se desconecta, de forma que se pueda validar identidad y su cadena de bloques.

3.4.6. Capa de incentivo

La capa incentivo genera una recompensa o penalización a los participantes testigos de acuerdo a su actuación dentro de la red. Si el participante testigo crea un bloque exitosamente recibe una transacción de recompensa, en cambio si se detecta un comportamiento deshonesto el participante pierde su estatus de testigo, así como la posibilidad de volver a optar por el cargo.

En este proyecto no se hizo énfasis en esta capa, debido a que el desarrollo no tenía pensado la inclusión de un *token* para representar el valor, de ahí que se usara un protocolo de transacciones como prueba de participación, sin embargo eliminar esta capa del protocolo no era una opción óptima debido a que el desarrollo tiene miras a incluir un sistema de *token* en el futuro.

3.4.7. Capa de aplicación

En ésta última capa, se encuentran los posibles escenarios de aplicación y casos de uso, en ella es donde se mostrará la información proporcionada y gestionada por la red. La capa de aplicación proporciona una API de la red para ser adaptada a cualquier sistema compatible. Es de vital importancia que en esta capa final donde se entregan los datos, no se comprometa la seguridad de los mismos, para esto el API esta desarrollada bajo el protocolo HTTP para satisfacer una arquitectura cliente-Servidor consiguiendo así mantener la seguridad de los datos.

3.5. Eventos

La forma en que cada proceso en la red se adaptó para trabajar en esta arquitectura, es basada a la presentada en Foundation (2018) donde los procesos son denominados eventos, y cada uno es recibido y atacado por un conjunto de capas en caso de ser necesario. Pero esto no es una medida arbitraria, un evento que sea introducido en la red y no este referenciado a ninguna capa, pasara por la misma sin aplicarle ninguna acción sobre la red; una modificación en algún protocolo de las capas para alterar la red, no podrá ser confirmado por los demás miembros, aislando el problema al participante fraudulento.

3.5.1. Eventos de transacciones

En la Tabla 3.1 se puede observar la lista de eventos relacionados a la creación de transacciones, esto es atacado por el participante de la red capaz de crearlas, y una vez validadas, la capa de red propaga la información con el evento *announceNewTransaction*. Existen procesos para obtener información como son *getTransaction*, que permiten acceder a las transacciones de la red, este proceso no pierde seguridad debido a que todos estas transacciones están cifradas con la clave pública de sus creadores.

Tabla 3.1: Eventos de la red: transacciones.

Código del evento	**Nombre del evento**	**Capas que trabajan sobre el evento**
makeTransaction	Crear transacciones.	Datos, *Middleware*, Red
getTransaction	Obtener transacciones.	Datos, Aplicación
announceNewTransaction	Anunciar a la red una nueva transacción por verificar.	Datos, Red

3.5.2. Eventos de información

La red mantiene también cierta información para medir el rendimiento, para esto, se crearon los eventos presentes en la Tabla 3.2 donde se accede a información de diversas capas, de forma que se tenga un conocimiento general del estado de la red.

Tabla 3.2: Eventos de la red: Información.

Codigo del evento	**Nombre del evento**	**Capas que trabajan sobre el evento**
getNumTrasactions	Obtener el número de transacciones totales de la red.	Datos, Red
getTimeStack	Obtener el tiempo promedio de la red en validar un transaccion.	Datos, Red
getTimeStack	Obtener el tamaño de la cola de transacciones esperando por validar en la red.	Datos, Red

3.5.3. Eventos de comunicación

De igual forma existen eventos relacionados con los procesos de comunicación de los participantes de la red, los cuales se pueden apreciar en la Tabla 3.3, donde se encuentran procesos como los de conexión y desconexión de la red, así también como los consensos.

Tabla 3.3: Eventos de la red: Comunicación.

Código del evento	**Nombre del evento**	**Capas que trabajan sobre el evento**
registerNode	Registrar participante en la red.	Datos, *middleware*, Red
removeNode	Desconectar participante en la red.	Datos, *middleware*, Red
consensoNodes	Consenso de participantes.	Consenso, Red
addNode	Agrega un participante a la red.	Datos, Red
anounceRetire	Anunciar a la red un del retiro de un participante.	Datos, Red

3.5.4. Eventos de verificacion de bloques

Otro proceso vital en el protocolo de la cadena de bloques es el de la creación y validación de bloques, donde su verificación puede ser encontrada en la Tabla 3.4.

Tabla 3.4: Eventos de la red: Validación de bloques.

Codigo del evento	**Nombre del evento**	**Capas que trabajan sobre el evento**
verifyBlock	Verificación de un bloque.	Datos, Red
addBlock	Agregar un bloque a la red.	Datos, Red
announceNewBlock	Anunciar a la red un nuevo bloque válido.	Datos, Red

3.5.5. Eventos de contratos inteligentes

Por último se presentan los eventos relacionados a los contratos inteligentes encontrados en la Tabla 3.5 donde se puede encontrar la creación de contratos y su

verificación así como también la ejecución de sus acciones.

Tabla 3.5: Eventos de la red: Contratos inteligentes.

Codigo del evento	**Nombre del evento**	**Capas que trabajan sobre el evento**
addSmartContract	Crear contrato inteligente no verificado	Datos, *middleware*, Contratos, Red
makeActionsContract	Aplicar acciones del contrato inteligente.	Datos, *middleware*, Contratos, Red
announceNewContract	Anunciar a la red un nuevo contrato por verificar.	Datos, Contrato, Red

3.6. Diseño de los eventos

Como se trató en la SECCIÓN 3.5, la red usa evento para la ejecución de acciones dentro de la misma, en esta sección se presenta la forma en que los eventos principales son tratados por cada capa y cómo se trata de mantener encapsulado cada proceso en pro de obtener un flujo seguro y lo más eficiente posible.

3.6.1. Evento crear transacciones

Basándose en la Fig. 3.10, se puede observar que la capa de *middleware* tiene la función de validar que los datos introducidos tenga el formato correcto, además validar que no hay ningún dato extra que pueda ser utilizado en el futuro de forma no segura.

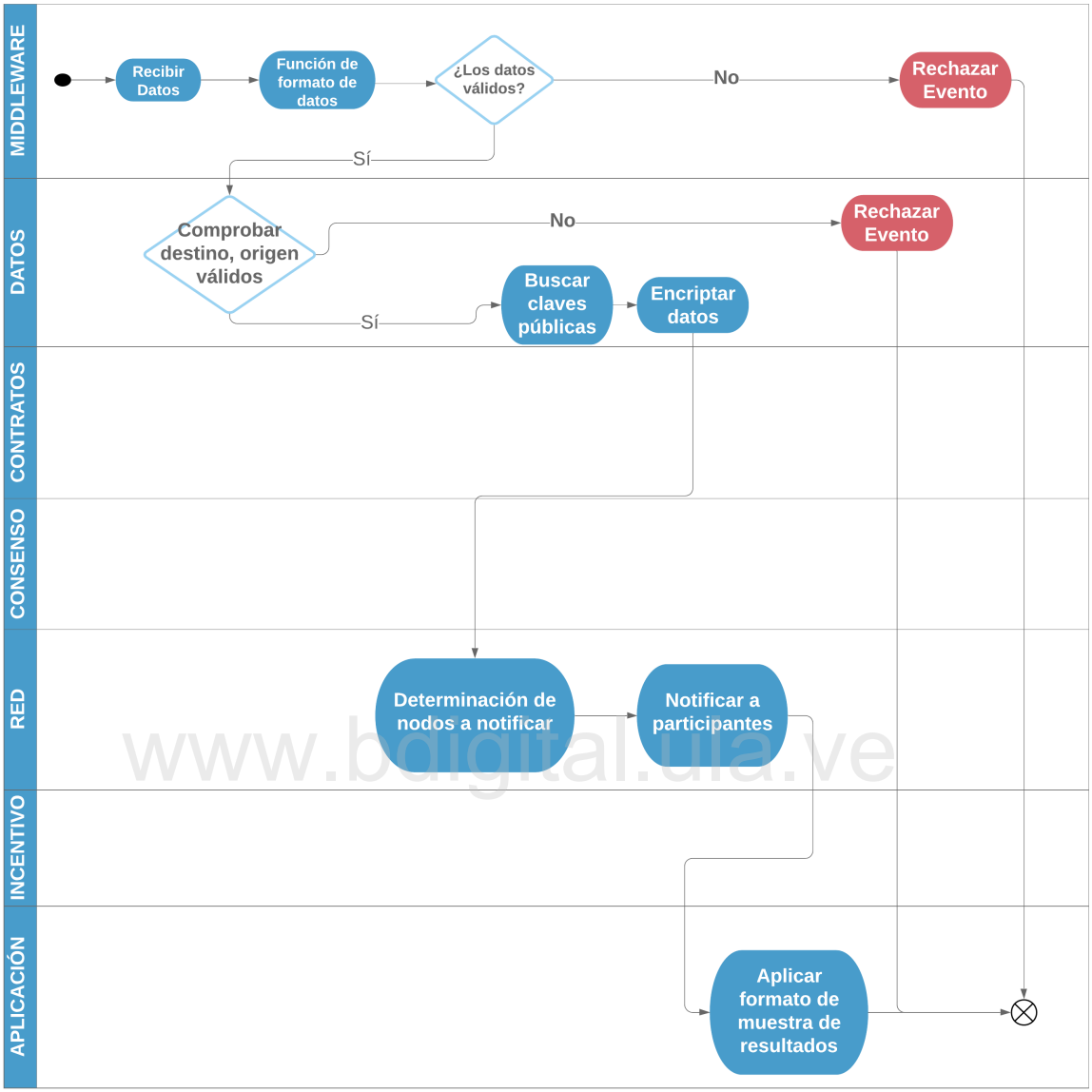

Figura 3.10: Diagrama de actividades del evento crear transacción.

En la capa de datos se observa la lógica aplicada en el Algoritmo 3.5, donde se buscan dentro de la red los parámetros faltantes. Al proseguir el flujo, se accede a la capa de red, por lo tanto, el mensaje se repartirá al conjunto de participantes que estén conectados, por último, en la capa de aplicaciones se realiza una verificación de los datos de salida de forma que no se escape ningún dato de interés en la red.

3.6.2. Evento crear contrato

Este evento tiene cierta similitud al de crear transacciones tratado con anterioridad, sin embargo, se está aplicado una lógica extra en la capa de contrato, si se observa la Fig. 3.11, el flujo sigue las reglas que se trataron en el CAPITULO 3.2, donde se definió que el contrato tenía un participante auditor a su cargo y que el mismo trabajaría en segundo plano, una vez la capa de contratos termina su labor, el sistema es transmitido.

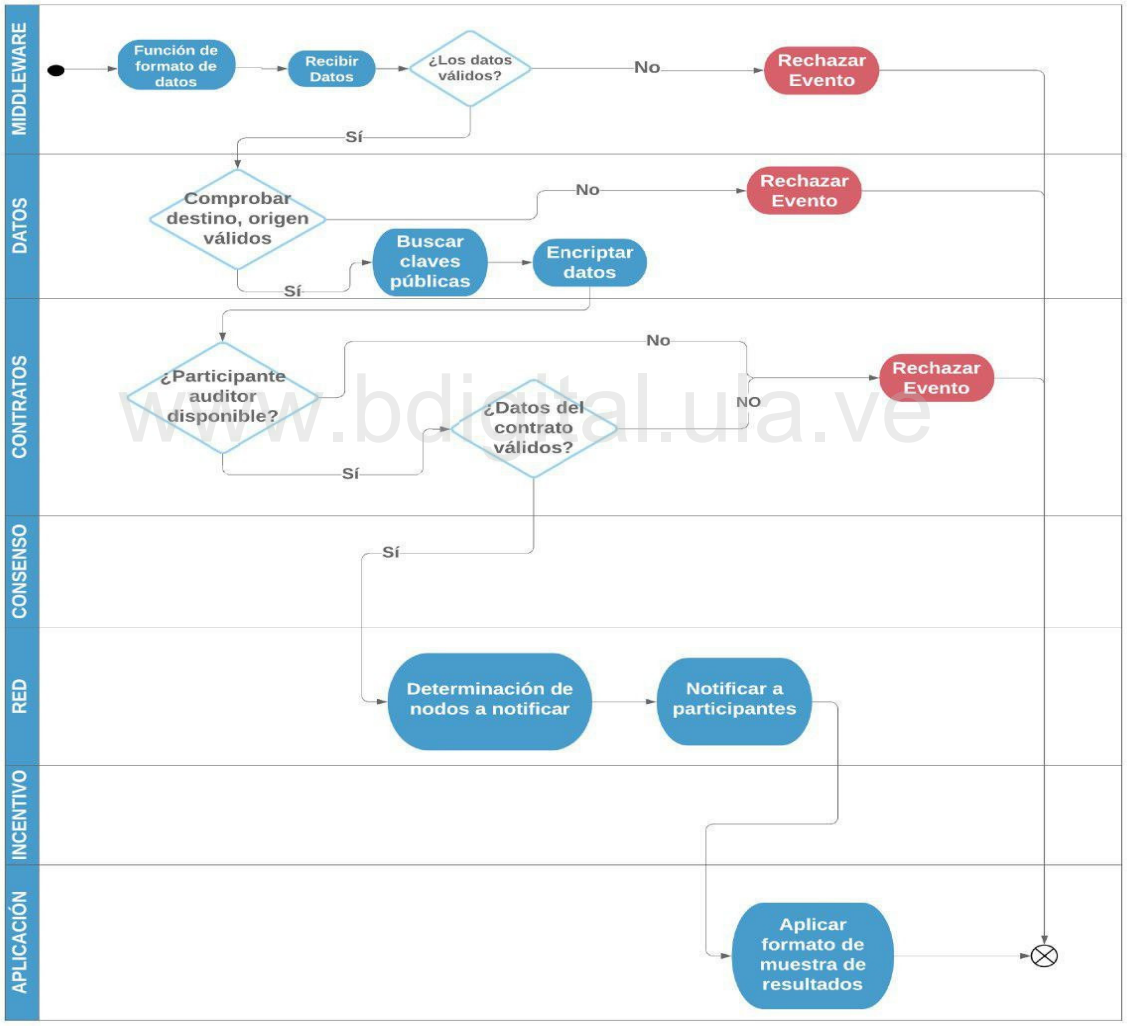

Figura 3.11: Diagrama de actividades del evento crear contrato.

3.6.3. Evento registrar participante

En este caso, se puede observar en la Fig. 3.12 que el evento *registerNode* presenta más énfasis en la capa de red y esto tiene sentido porque éste es el único encargado del manejo de los participantes de la red. Sin embargo, se observa que aunque la red en la capa de datos solo maneja la lógica referente a la cadena basada en DAG, si aplica una verificación dentro de la cadena para confirmar si este participante ha tenido una participación previa y de esta forma notificar a la capa de consenso que el participante cuenta con una valoración a la hora de participar en el protocolo de participación basado en transacciones.

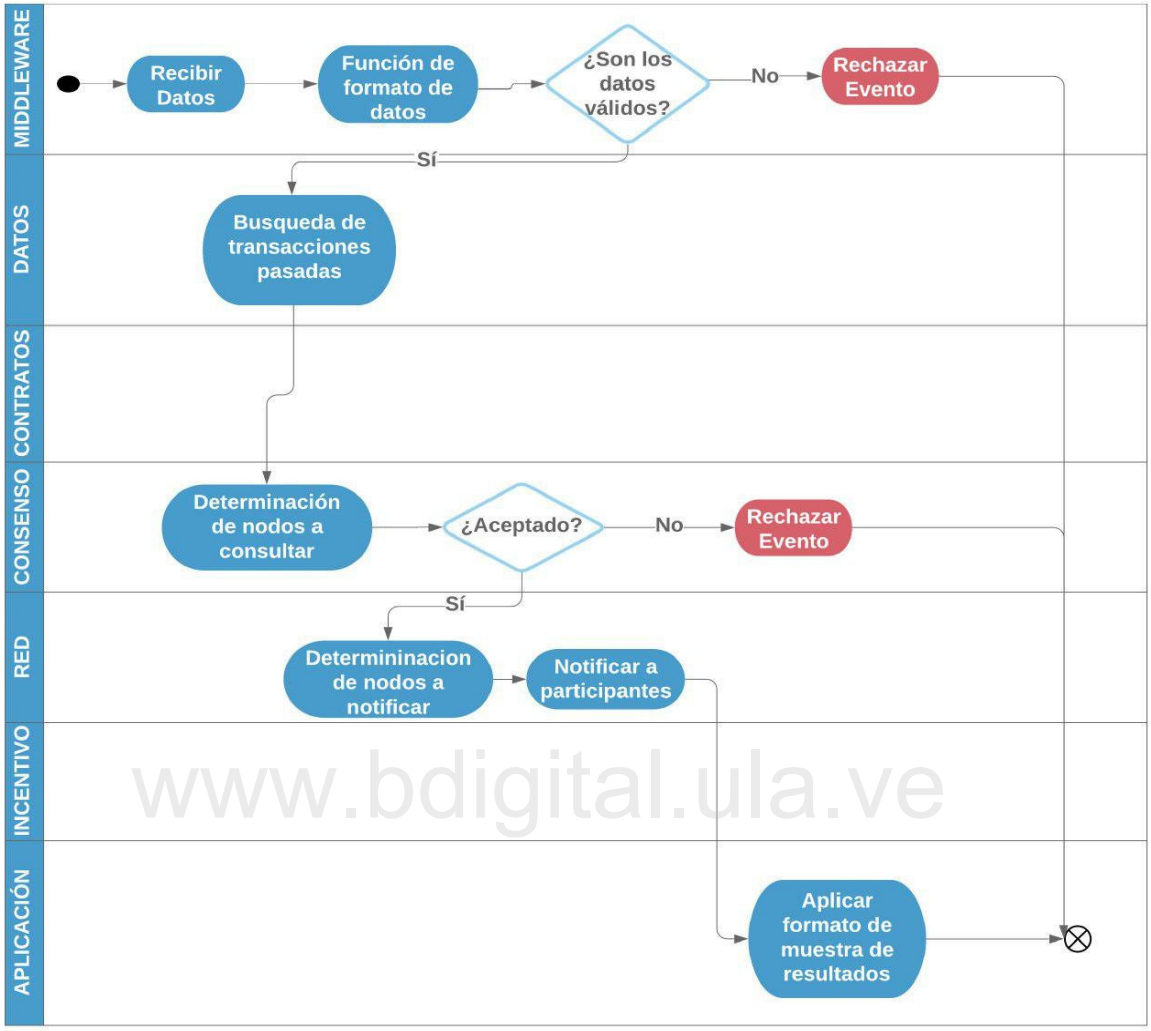

Figura 3.12: Diagrama de actividades del registro de un participante.

3.6.4. Evento de verificar bloque

Para este evento se puede observar en la Fig. 3.13, al igual que los demás eventos, la capa *middleware* realiza una revisión en los datos, una vez que el bloque parece contener datos válidos, la capa de datos realiza un revisión exhaustiva sobre los datos que contiene, entre estas verificaciones se encuentra si las transacciones son válidas, de igual forma, se verifica cada uno de los cambios del bloque como el *hash*. Una vez que se encuentre validado, la capa de red pasa la información a cada participante anunciando que existe un nuevo bloque en la red.

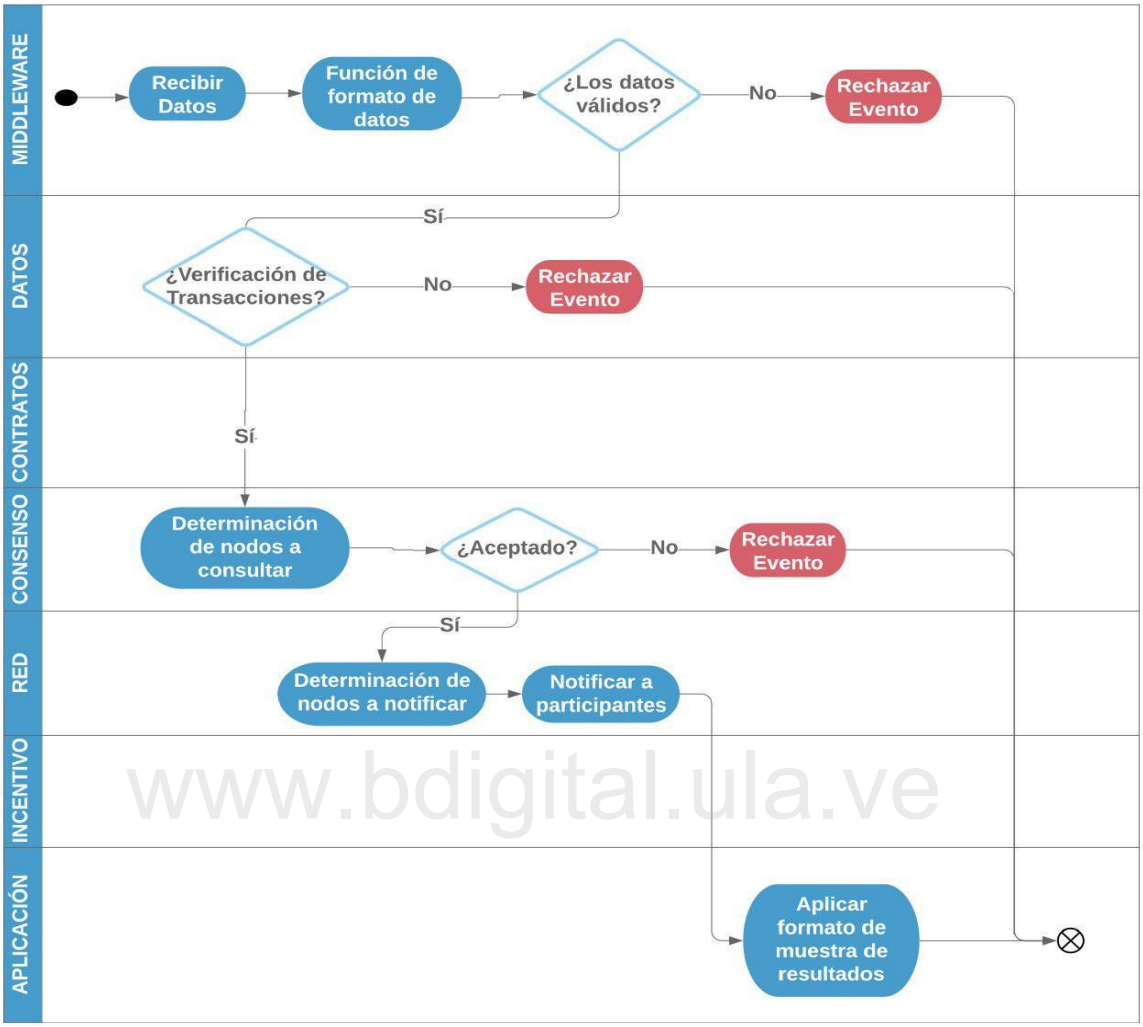

Figura 3.13: Diagrama de actividades del evento de verificar bloque.

Capítulo 4

Pruebas de funcionamiento general de la red

En este capítulo se demostrará empíricamente cada uno de los procesos de la red, que fueron diseñados en los capítulos anteriores. De igual forma, comprobar con pruebas de rendimiento que la red presenta un desempeño similar al que tenía antes de aplicar los cambios.

4.1. Demostración de funcionalidad

Para tratar el primer punto basado en la demostración de la capacidades de la red, se hizo uso principalmente de la capa de aplicación, explicada en la SECCIÓN 3.4.7. A través del uso de una API conectada directamente con la red, se logra comunicar un sistema externo a la misma. Este agregado podía otorgar ciertas vulnerabilidades dentro de la red, de modo que se aplicó una conexión directa proveniente de un participante válido dentro de la red; de esta forma se asegura que la misma mantenga cada unos de los protocolos de seguridad antes expuestos.

La demostración corre bajo una arquitectura cliente-servidor, siendo el servidor un participante de la red. Si bien la red permite conectarse a otro ente que también cumpla el rol del servidor, este desarrollo no presentaba dichos fines, por lo que acceder directamente al servidor de la red cumple el propósito de demostrar sus protocolos.

Como se puede observar en Fig. 4.1 donde se representa la interfaz gráfica de la demostración, en primer lugar cuenta con la selección de origen del contrato, en éste se listan cada una de las direcciones capaces de recibir uno. Los siguientes parámetros son el título y texto. Para finalizar, se prosiguió con las regulaciones y acciones.

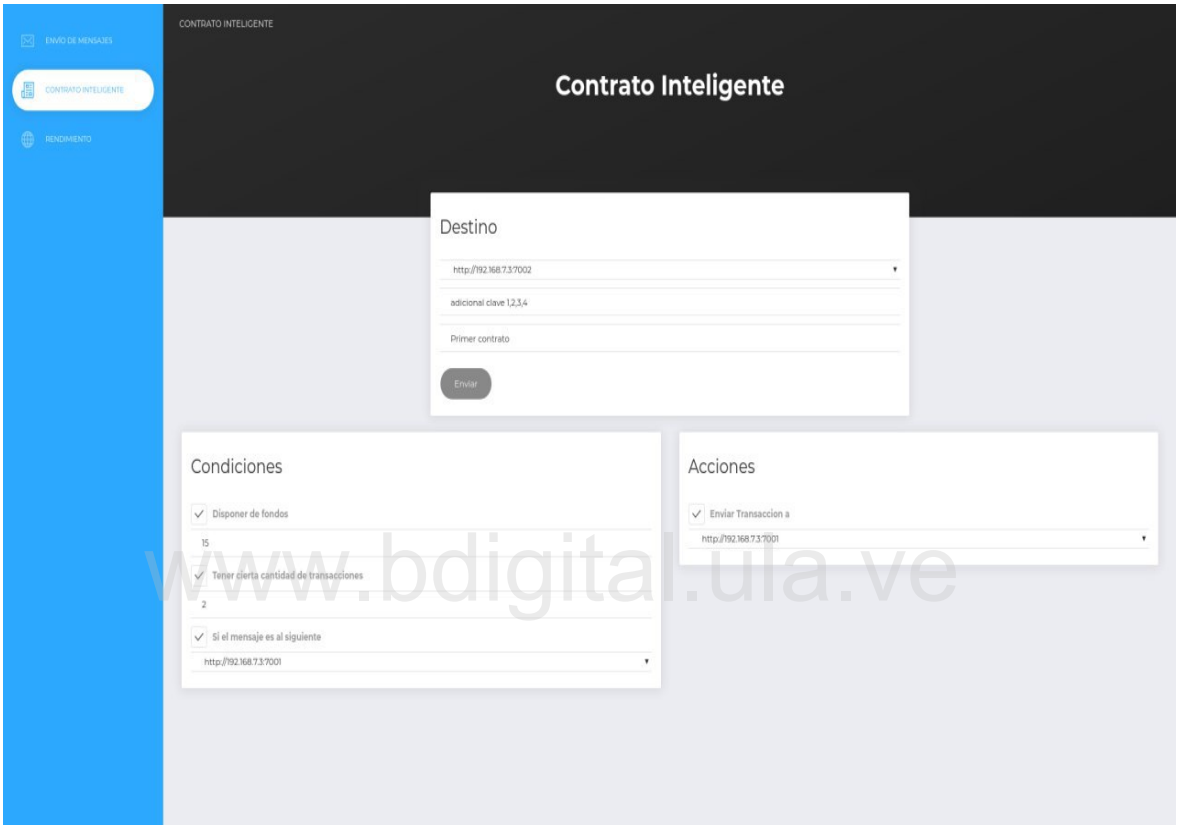

Figura 4.1: Interfaz gráfica de la demostración.

La demostración trata de simular un ambiente donde se pretende contabilizar la cantidad de fondos que tenga un cliente para luego emitir una transacción, en esta demostración las transacciones están basadas solo en texto plano, pero existe la posibilidad de brindar soporte a diversos formatos.

4.1.1. Regulaciones

En el caso de las regulaciones se contó con las siguientes:

- Disponer de fondos

- Tener cierta cantidad de transacciones
- Si tiene un mensaje a un contacto

4.1.2. Acciones

En el caso de las acciones se contó con las siguientes:

- Enviar transacción a un contacto

De estos parámetros son obligatorios, el **nombre**, la **dirección** y el **texto**, aunque las regulaciones y acciones parecieran ser obligatorias no tienen esta regla, un contrato sin estos, simplemente es visto como una transacción más en el sistema; el participante auditor corroborara las regulaciones, al no existir dará como completado el contrato, creando así, la transacción para confirmarlo, procediendo a aplicar acciones que en esto caso no existen.

Una vez un contrato entra en el sistema, la demostración notifica el registro exitoso siempre que el evento *addSmartContract* (ver Fig. 3.11) sea terminado con un resultado positivo, una vez el contrato es introducido a la red, el participante que envió el contrato no tiene más acción sobre el mismo.

Para el caso específico de contabilizar los fondos, se hizo uso del campo **datos** del participante, que permite almacenar la información extra del mismo. La contabilización de las transacciones se pueden ejecutar gracias a los eventos de información; al igual que la última regulación que tiene como objetivo, verificar si el participante destino tiene al menos una transacción enviada a otro destino especifico.

4.2. Prueba de rendimiento

El objetivo de esta sección en corroborar que el rendimiento de la red no se haya visto afectado debido al cambio en el funcionamiento de los protocolos. La arquitectura de desarrollo en capas tiene entre sus desventajas la posibilidad de generar retraso en los protocolos; para comprobar que esto no haya sucedido, se hará uso de tres estadísticos que permiten medir que tan eficiente se comporta una red *blockchain* (Guerrero, 2020).

El primero son las **transacciones por segundo** (Tps), que permiten determinar que tan rapido es una transacción verificada por el sistema. El segundo parámetro, es el **tamaño de cola de espera** (TS), que sirve para determinar sobre el tiempo, como es el comportamiento de la cola de espera que hacen las transacciones para ser verificadas. Por último, el **tiempo de cola promedio**, que al igual que el anterior da una perspectiva de el tiempo de espera que cada una de las transacciones demora por ser validadas.

Para comparar, se hizo uso de los resultados expuestos por Guerrero (2020), los cuales se pueden observar en la Tabla 4.1, donde para el caso de la prueba uno, donde la cantidad de participantes y transacciones introducidas son menores, se consiguió un total de 23,22 Tps. En el caso de la prueba dos, se consiguen resultados más altos, debido a que tanto como las transacciones, como el número de participantes son mayores.

Tabla 4.1: Resultados de Guerrero (2020)

Prueba	Número de participantes	Transacciones introducidas	Tps	Tamaño de la cola	Tiempo de cola promedio
1	2	50	23,22	29	0,332
2	12	200	57,40	403,63	2,73

Cuando se ejecuto las mismas pruebas, con el cambio en la arquitectura propuestos en este proyecto, los resultados fueron los presentados en la Tabla 4.2, y en la Figuras 4.2, 4.3, 4.4, 4.5, 4.6 y 4.7. Donde se puede observar que aunque hay un aumento en el Tps de las dos pruebas, también se tiene un aumento en el tiempo de cola promedio y en el tamaño de cola; lo que quiere decir que el sistema valida más transacciones por bloques, en la verificación hay un mayor número de transacciones a validar, entendiendo así que los participantes validadores de transacciones tardan más en empezar a realizar su labor.

Tabla 4.2: Resultados de pruebas de rendimiento.

Prueba	Número de participantes	Transacciones introducidas	Tps	Tamaño de la cola	Tiempo de cola promedio
1	2	50	26,36	26,23	0.35
2	12	200	63,40	726,86	6,05

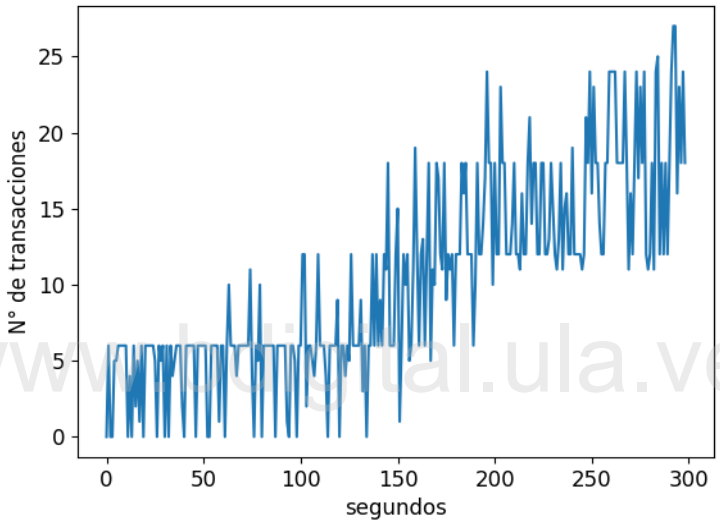

Figura 4.2: Tamaño de cola con dos participantes trabajando.

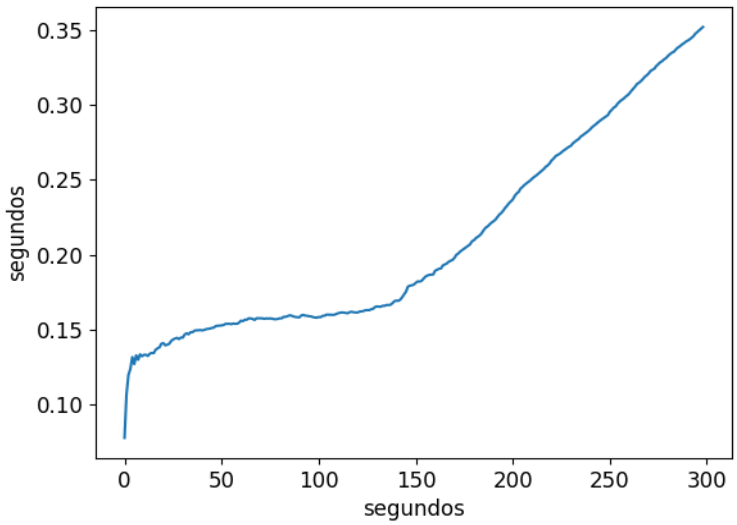

Figura 4.3: Tiempo de cola promedio con dos participantes trabajando.

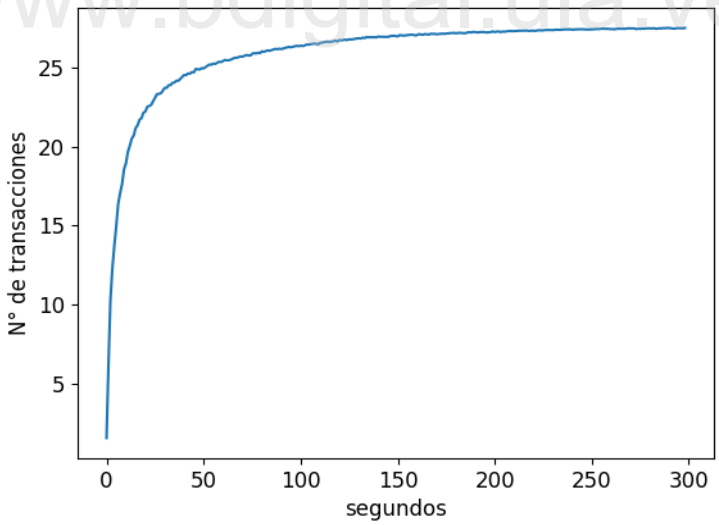

Figura 4.4: Transacciones por segundo con dos participantes trabajando.

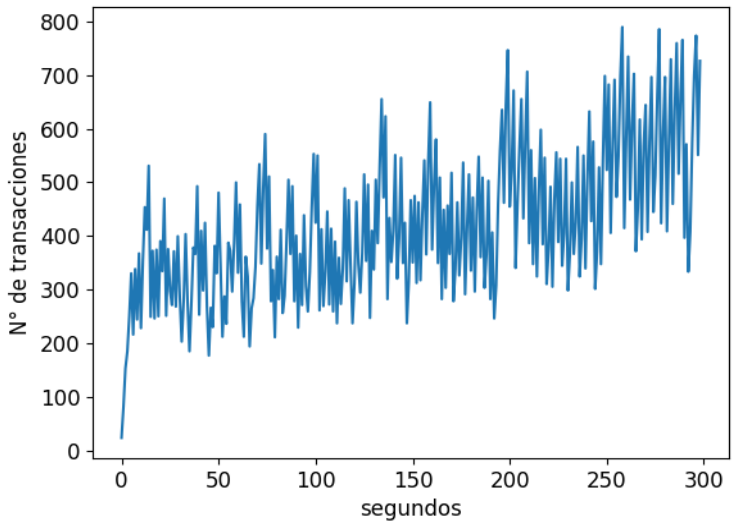

Figura 4.5: Tamaño de cola con doce participantes trabajando.

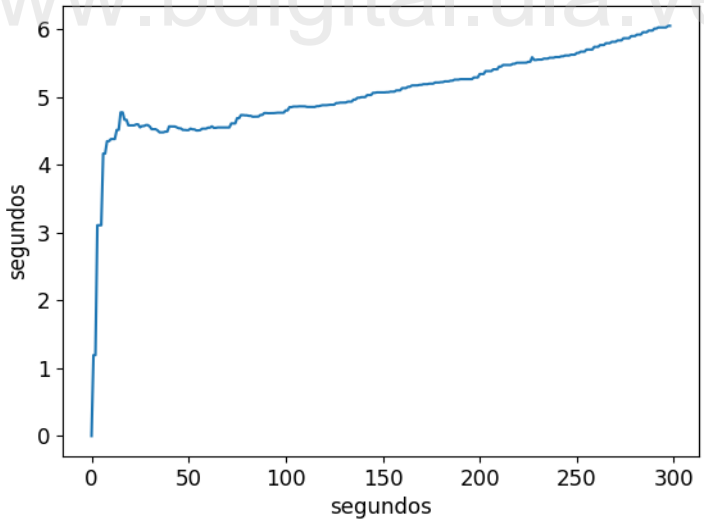

Figura 4.6: Tiempo de cola promedio con doce participantes trabajando.

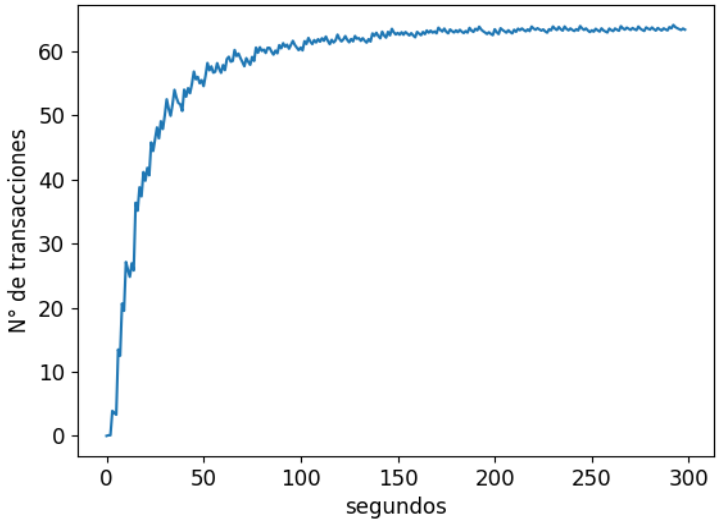

Figura 4.7: Transacciones por segundo con doce participantes trabajando.

En cada una de las Figuras 4.2, 4.3, 4.4, 4.5, 4.6 y 4.7 se puede observar que tienen un comportamiento creciente, en particular el cálculo del número de transacciones por segundo, en los dos casos se puede observar que tienden a sobrecargarse llegando un valor de equilibrio, que es un valor que la red alcanza una vez todos los participantes han terminado de conectarse a la misma y la comunicación es manejada por la capa de red de manera constante.

Para finalizar, la red muestra la capacidad de adaptarse a un caso de estudio sin tener que comprometer sus protocolos, dándole un dinamismo gracias a la inclusión de contratos inteligentes, todo esto sin perder rendimiento. Por el contrario, obteniendo en algunas métricas resultados mejores que los del desarrollo base, que es todo un logro teniendo en cuenta el antecedente de que arquitectura basada en capas tiende a generar lentitud en los procesos.

Capítulo 5

Conclusiones

Durante todo el desarrollo se propuso la idea de la implementación de un sistema de información basado en la red *blockchain*, traía un sin fin de ventajas. Pero más que esto, la forma en que la arquitectura *blockchain* se puede modelar, con el fin de obtener diferente resultados es fascinante. Es por razones como estas que una vez terminado la investigación se llegaron a los siguientes resultados:

- La implementación de una arquitectura en capas permite tener un desarrollo más escalable, la capacidad de adaptar o eliminar lógica en futuro ha aumentado gracias a la inclusión de esta arquitectura.

- La red *blockdag*, basada en capas permite tener un control más exhaustivo de la información, cada capa tiene una labor y es mantener la seguridad de los datos que a ella le concierne, ademas de encapsular la lógica computacional, basado en el objetivo de la capa.

- Los cambios en el protocolo de comunicación, con la inclusión del participante auditor, dota a la red de un sistema extra de seguridad en el mantenimiento de la información; otorgándole a una red, como la presentada en este documento, la posibilidad de corroborar las transacciones y contrato generados.

- La inclusión de contratos inteligentes, le brindo a la red *blockdag*, de una agilidad única en la manipulación de los datos, permitiendo que diversas aplicaciones puedan gestionar la forma en que reglas se cumplen. La red *blockdag* dota todo

un sistema autoadmistrado, con la labor de soporte del participante auditor para la validación de los contratos.

- El sistema de eventos, junto con el sistema de capas, permite aislar a la red de flujos ajenos a la misma, los evento que no sean reconocidos por el resto de la red, sera aislados y el participante, reprendido de toda posibilidad de toma de decisiones.

- La red no pierde rendimiento, a pesar de que el sistema de capas genera más regulaciones en los procesos de la misma, es capaz de procesar más transacciones por segundo, que es una de las métricas más utilizadas para valorar una red *blockchain*.

- La red demuestra ser capaz de recibir las peticiones de un sistema externos sin perder seguridad, dotándolo de la posibilidad de convertirse en una plataforma, que permita tener múltiples sistemas con lógicas diferentes en la mismo entorno.

5.1. Recomendaciones

Con el objetivo de obtener un desarrollo mas robusto y de realizar pruebas mas exigentes al desarrollo propuesto, se presentan las siguientes recomendaciones:

- La no inclusión de un *token* o moneda digital, nativo en la red, genero que ciertos protocolos fueran complejos de adaptar, es por esto que se recomienda en un futuro la inclusión del mismo.

- Si bien el árbol de merkle es usado en la red para validar transacciones, la utilización del mismo como mecanismo para la validación rápida de datos relacionados con una cartera digital es necesario.

- La inclusión de una cartera digital única para clientes, es uno de los puntos que es necesario desarrollar en el futuro.

- Si bien las pruebas realizadas permitieron medir el rendimiento de la red, es necesario generar prueba en entornos reales, como los vistos en un ambiente en

producción, donde temas como la latencia en la comunicación o la desconexión de participantes sea una posibilidad real.

Bibliografía

Amazon.com (2019). AWS Partner Story: GuildOne and R3. Seattle, Washington, Estados Unidos. Recuperado de https://aws.amazon.com/es/partners/success/guildone/. Recuperado el 01-09-2019.

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., McCallum, P., y Peacock, A. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 100:143–174.

Benchoufi, M. y Ravaud, P. (2017). Blockchain technology for improving clinical research quality. *Trials*, 18(1):335.

Cámara Albuixech, R. (2018). Estudio de tecnologías Bitcoin y Blockchain.

Czepluch, J. S., Lollike, N. Z., y Malone, S. O. (2015). The use of block chain technology in different application domains. *The IT University of Copenhagen, Copenhagen*.

David Schatsky, Amanpreet Arora, A. D. (2018). Blockchain and the five vectors of progress. Recuperado de https://www2.deloitte.com/us/en/insights/focus/signals-for-strategists/value-of-blockchain-applications-interoperability.html. Recuperado el 01-09-2019.

Engelhardt, M. A. (2017). Hitching healthcare to the chain: An introduction to blockchain technology in the healthcare sector. *Technology Innovation Management Review*, 7(10).

Foundation, F. (2018). Fantom: Whitepaper. Reporte técnico, FANTOM. Recuperado de: https://fantom.foundation. Consultado el 01-04-2019.

Galvin, D. (2017). Ibm and walmart: Blockchain for food safety. *PowerPoint presentation*.

Guerrero, A. (2020). Integración de grafos acíclicos dirigidos para mejorar la escalabilidad en la cadena de bloques. *Tesis de pregrado. Universidad de los Andes, Venezuela*.

Kosba, A., Miller, A., Shi, E., Wen, Z., y Papamanthou, C. (2016). Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. En *2016 IEEE symposium on security and privacy (SP)*, p. 839–858. IEEE.

Llumiugsi, T., Anabel, M., Felicita, M., y Eduardo, R. (2018). Sistema para mejorar la seguridad de la información en identidades digitales aplicando tecnología Blockchain. B.S. thesis, Universidad de las Fuerzas Armadas ESPE. Carrera de Ingeniería en Sistemas.

Lopez, J., Rios, R., Bao, F., y Wang, G. (2017). Evolving privacy: From sensors to the Internet of Things. *Future Generation Computer Systems*, 75:46–57.

money transfer, T. (2019). Tempo money transfer. Paris, Francia. Recuperado de https://tempo.eu.com/en. Recuperado el 01-09-2019.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Reporte técnico, www.bitcoin.org. Recuperado de: https://bitcoin.org/bitcoin.pdf. Consultado el: 01-03-2019.

Nugent, T., Upton, D., y Cimpoesu, M. (2016). Improving data transparency in clinical trials using blockchain smart contracts. *F1000Research*, 5.

Outchakoucht, A., Hamza, E., y Leroy, J. P. (2017). Dynamic access control policy based on blockchain and machine learning for the internet of things. *Int. J. Adv. Comput. Sci. Appl*, 8(7):417–424.

Pop, C., Cioara, T., Antal, M., Anghel, I., Salomie, I., y Bertoncini, M. (2018). Blockchain based decentralized management of demand response programs in smart energy grids. *Sensors*, 18(1):162.

Preukschat, A. (2017). *Blockchain: la revolución industrial de internet*. Gestión 2000, 08034 Barcelona, España.

Retamal, C. D., Roig, J. B., y Tapia, J. L. M. (2017). La blockchain: fundamentos, aplicaciones y relación con otras tecnologías disruptivas. *Economía industrial*, 405:33–40.

Swan, M. (2015). *Blockchain: Blueprint for a new economy*. .ºReilly Media, Inc.".

Swan, M. (2018). Blockchain for business: Next-generation enterprise artificial intelligence systems. En *Advances in Computers*, volumen 111, p. 121–162. Elsevier.

Tapscott, D. y Tapscott, A. (2017). La revolución blockchain. *Descubre cómo esta nueva tecnología transformará la economía global. ediciones deusco. séptima edición. recuperado en webdelprofesor. ula. ve/economia/oscared/materias/E_E_Mundial/Economia_ Internacional_Krugman_Obstfeld. pdf*.

Vileriño, S. (2017). *Estudio de los límites de generación de bloques en blockchain*. Tesis de Doctorado, Universidad de Buenos Aires.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32.

Yuan, Y. y Wang, F.-Y. (2016). Towards blockchain-based intelligent transportation systems. En *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, p. 2663–2668. IEEE.