UNIVERSIDAD DE CARABOBO

FACULTAD DE CIENCIAS Y TECNOLOGÍA

LICENCIATURA EN COMPUTACIÓN

Diseño e implementación de un contrato inteligente sobre una plataforma de cadena de bloques con Ethereum

Elaborado por:
Cesar Nieves Aramendi

Tutor:
Jorge Ernesto Rodríguez

Naguanagua, 28 de mayo de 2020

Universidad de Carabobo
Facultad Experimental de Ciencias y
Tecnología

Departamento de Computación
Valencia-Venezuela

ACTA

Reunidos en Bárbula, en la sede del Departamento de Computación de la FACYT, los miembros del jurado designado por el Equipo Directivo del Departamento de Computación de la Facultad Experimental de Ciencia y Tecnología (FACYT), para evaluar el Trabajo Especial de Grado titulado: **"DISEÑO E IMPLEMENTACIÓN DE UN CONTRATO INTELIGENTE SOBRE UNA PLATAFORMA DE CADENA DE BLOQUES CON ETHEREUM"** el cual fue presentado por el bachiller **CÉSAR NIEVES ARAMENDI, C.I. 22.552.994**, a los fines de cumplir con los requerimientos legales para optar al título de Licenciado en Computación dejamos constancia de lo siguiente: Una vez leído el Trabajo por cada uno de los miembros del jurado, se fijó el **14 de AGOSTO de 2020**, para la presentación pública del trabajo, después de concluida ésta, el autor respondió satisfactoriamente a las preguntas que le fueron formuladas por el Jurado. Posteriormente el jurado deliberó y lo consideró: **APROBADO**.

En Valencia a los 14 días del mes de agosto de 2020.

Prof. Jorge Ernesto Rodríguez
Tutor
Dpto. de Computación – FACYT

Prof. Pedro Linares
Miembro
Dpto. de Computación - FACYT

Prof. Aldo Reyes
Miembro
Dpto. de Computación- FACYT

UNIVERSIDAD DE CARABOBO

FACULTAD DE CIENCIAS Y TECNOLOGÍA

LICENCIATURA EN COMPUTACIÓN

Diseño e implementación de un contrato inteligente sobre una plataforma de cadena de bloques con Ethereum

RESUMEN

Uno de los mayores problemas a la hora de establecer relaciones transaccionales entre pares a través de la red, es la dificultad de garantizar la seguridad de la información relativa a la transacción sin la intermediación de una entidad centralizadora que actúe como un tercero de confianza. Esta entidad se encarga de validar los intercambios entre los pares y garantiza la seguridad de la información y sus efectos consecuentes. Este es el caso, por ejemplo, de los intercambios de dinero o activos entre pares, los cuales deben estar mediados por la intermediación de una entidad financiera (un banco, Visa, Master, etc.). Esta entidad intermediadora de la transacción no es inherente a la relación entre los pares, sino que es una consecuencia de la fragilidad (inseguridad) del medio electrónico para intercambios de estas características, haciendo obligante la aparición de esta tercera parte en la transacción. La aparición de la tecnología de cadena de bloques (blockchain) plantea un esquema tecnológico en el que es posible realizar transacciones entre pares prescindiendo de esta entidad central. Aunque creada originalmente para permitir el intercambio de dinero (criptomonedas), su utilidad ha sido extendida hacia otro tipo de transacciones electrónicas entre pares, abriendo el camino a la implementación de aplicaciones varias, una de ellas es lo que se ha denominado contratos inteligentes. El objetivo de este estudio es explorar la potencialidad de las cadenas de bloques como plataforma confiable para la implementación de contratos inteligentes. En tal sentido, se plantea la implementación de una aplicación que soporte un contrato inteligente en forma experimental sobre una plataforma de cadena de bloques pública. La aplicación escogida responde a un sistema de votación electrónica para la elección de una persona entre un conjunto de candidatos específicos.

Palabras claves: Cadena de Bloques, Contratos Inteligentes, Aplicaciones Descentralizadas.

Estudiante: Cesar Nieves Aramendi **Tutor**: Jorge Ernesto Rodríguez

TABLA DE CONTENIDO

1. INTRODUCCIÓN

La cadena de bloques (*blockchain*) es la tecnología que está impactando el mundo de la computación actualmente como lo expresa Don Tapscott (ejecutivo empresarial) en su presentación "Cómo la cadena de bloques está cambiando los negocios" [1].

La cadena de bloques surge como alternativa al modo usual de intercambiar activos a través de la intermediación de un tercero de confianza, permitiendo la interacción directa entre las partes implicadas sin la intermediación de entidad alguna, encargándose de asegurar que las partes cumplan lo acordado y que el intercambio se realice según los términos establecidos. Ejemplos de intermediarios en operaciones de intercambios de activos bajo el esquema tradicional suelen ser los bancos, ciertas entidades gubernamentales, compañías de tarjetas de crédito, etc. los cuales ofrecen a personas jurídicas y naturales que desean o requieren hacer transacciones de compra/venta de bienes y/o servicios fungir de intermediarios en sus transacciones para asegurar el correcto pago y cobro entre las partes implicadas en determinada transacción. Para este propósito, manejan los datos de sus clientes de manera centralizada, garantizando que las transacciones entre las partes sean realizadas correctamente y asumiendo la responsabilidad y capacidad operativa asociados a este servicio a cambio, claro está de una lucrativa ganancia normalmente cobrada a través de comisiones a las transacciones realizadas. Sin embargo, este esquema, inevitable hasta la aparición del *blockchain*, conlleva un precio a pagar en términos de la burocracia añadida, tiempo y costos mayores, y las debilidades de seguridad asociadas a los sistemas centralizados.

En esta perspectiva, esta propuesta de investigación aspira indagar sobre los contratos inteligentes (*Smart Contracts*) que usan *blockchain* para el intercambio de activos en el marco de un conjunto de condiciones o cláusulas previamente pautadas entre las partes.

Para lograr este objetivo, nos hemos planteado la escogencia de un problema específico susceptible de ser modelado como un contrato entre un conjunto de partes, e implementarlo sin la intermediación de una entidad centralizadora de la información entre dichas partes, a través del uso de una plataforma pública e cadena de bloques.

El presente documento introduce la propuesta de investigación que se desea realizar durante el desarrollo del trabajo de grado. En tal sentido se ha estructurado en seis secciones, además de esta introducción:

Las secciones 2 y 3 comprenden el planteamiento del problema, donde se establece la problemática y los objetivos generales y específicos trazados, así como la justificación para desarrollar esta investigación. Vale resaltar que esta sección se corresponde con la fase de análisis del problema de investigación, es decir, el modelo conceptual de nuestra investigación en el marco de la metodología hipotético-deductiva escogida para el desarrollo de nuestra investigación.

La sección 4 comprende el Marco Teórico, donde se hace una revisión de algunos conceptos y desarrollos anteriores sobre los cuales se sustentará la investigación que

proponemos. Esta revisión bibliográfica del estado del arte en el área de cadenas de bloques y contratos inteligentes, viene a completar la información relativa al modelo conceptual de nuestra investigación.

En la sección 5, titulada Marco Metodológico, introducimos la metodología que será seguida para el desarrollo de la presente investigación, a saber, la metodología hipotético-deductiva. En tal sentido, por una parte, esta sección aborda los diferentes modelos relativos a la investigación desarrollada, a saber, el modelo conceptual, el modelo lógico, y el modelo físico que se corresponde al análisis del problema, diseño del experimento, y la fase de implementación del experimento, respectivamente. Por otra parte, en virtud de que nuestro experimento consiste en el desarrollo de una aplicación de software, encontraremos en la descripción del modelo físico, las etapas específicas de la metodología de desarrollo aplicada a la construcción de una aplicación de software, caracterizada por un análisis del problema específico a modelar, un diseño de la aplicación a ser implementada, y finalmente los detalles de la implementación de la misma.

La sección 6 recoge los resultados, discusión y conclusiones, donde presentamos los resultados obtenidos de la implementación de la aplicación, y exponemos nuestras conclusiones al respecto.

Finalmente, en la sección 7 encontrará el lector las referencias bibliográficas, que como su nombre lo indica, contiene las referencias a los documentos, artículos y libros citados en el texto del presente trabajo.

2. PLANTEAMIENTO DEL PROBLEMA

En 1995 Nick Szabo define el término “contrato inteligente” como: “Un conjunto de promesas, incluidos los protocolos dentro de los cuales las partes cumplen esas promesas. Los protocolos generalmente se implementan con programas en una red informática o en otras formas de electrónica digital, por lo que estos contratos son más inteligentes que sus antecesores en papel. El uso de inteligencia artificial no está implicado”. [2]

Ya que el modo usual de enviar información entre dos o más partes es crear una copia, y enviarla electrónicamente desde el emisor o remitente hacia el o los destinatarios, todo va bien mientras la información enviada no exija garantía de seguridad o confidencialidad. Si estamos hablando de intercambiar activos, por ejemplo, es fundamental garantizar la integridad del envío. Si alguien envía una cantidad de dinero a un conjunto de destinatarios, se debe garantizar que este dinero efectivamente llegue íntegro a sus destinatarios y que se descuente debidamente de las cuentas del emisor.

La solución que hemos dado, hasta el momento, al problema de garantizar la seguridad en este tipo de transacciones, es la intermediación de un tercero confiable, el cual centraliza el control de la transacción garantizando que las transacciones entre las partes sean realizadas correctamente y asumiendo la responsabilidad y capacidad operativa asociados a este servicio a cambio, claro está, de una lucrativa ganancia normalmente cobrada a través de comisiones a las transacciones realizadas.

Estos intermediarios pueden ser los bancos, el gobierno, grandes compañías de medios sociales, compañías de tarjetas de crédito, etc. los cuales manejan los datos de manera centralizada. Sin embargo, este esquema, inevitable hasta la aparición del *blockchain*, conlleva un precio a pagar en términos de la burocracia añadida, ralentización de la operación, y aumento en los costos asociados a la operación, amén de las debilidades de seguridad asociadas a los sistemas centralizados.

Los esquemas centralizados basados en un tercero de confianza suponen el mantenimiento por parte del ente centralizador de un libro contable con los registros de todas las transacciones de todos los clientes que éste posee en su haber. Ello implica una inversión importante en hardware, software y personal especializado para garantizar un funcionamiento eficaz, eficiente y seguro de la plataforma operativa que debe soportar este servicio cuando se hace masivo. Adicionalmente al equipamiento propiamente computacional, se requiere una infraestructura de climatización, energización y control de acceso físico. Obviamente, los costos asociados a la operatividad de esta plataforma son transferidos a los usuarios del servicio, aun cuando la naturaleza de una transacción entre pares no supondría teóricamente la intermediación de un tercero.

En particular, en cuanto a la seguridad, aun cuando un ataque informático a un banco parecería una idea descabellada a simple vista, no es algo imposible y de hecho sucede, y no nos referimos a un ataque a un blindado o una sucursal a mano armada como en una película, sino al robo y cambio de información. El mítico robo de 3,7$ millones a la

industria de servicios financieros Citibank en 1994, fue el primer robo internacional a un banco de forma completamente remota. Desde entonces el costo por robo informático ha aumentado junto a la inversión en seguridad y sigue en aumento (Ver gráfico1).

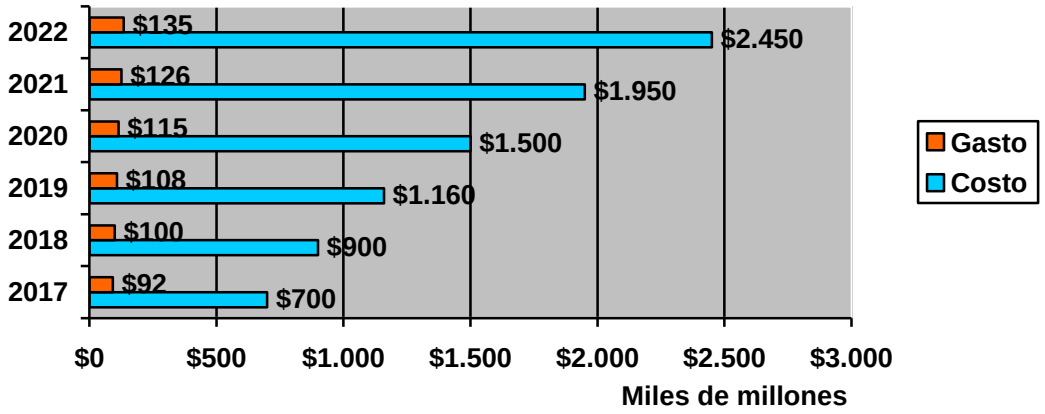


Gráfico 1 Incremento estimado en billones de dólares en los costos de la violación de datos y el gasto de ciberseguridad global del 2017 al 2022. Fuente "The Wall Street Journal".

Todas estas razones conllevan a plantearse la posibilidad de realizar intercambios entre pares sin la necesidad del intermediario o tercero de confianza, y la tecnología de la cadena de bloques (Blockchain) ha abierto esta posibilidad, y aún más, abre el espacio para ir más allá del intercambio de activos electrónicos, creando la posibilidad de crear aplicaciones varias que se puedan ejecutar en forma distribuida, manteniendo la integridad de los datos durante la ejecución y después de esta. Así las cosas, el planteamiento de Nick Szabo referido al inicio de esta sección encuentra apoyo tecnológico y pasa a ser una realidad posible. Así, la implementación de un contrato inteligente que permita concretar acuerdos y realizar transacciones acordes a éste entre pares de forma automatizada, y sin la necesidad de un intermediario, cuando ello no sea obligante por la naturaleza del contrato, empieza a ser posible, disminuyendo costos y tiempos de ejecución, a la par que garantiza la seguridad requerida.

En este contexto, se plantean las siguientes incógnitas: ¿Qué se necesita para crear un contrato inteligente?, ¿Cómo se desarrolla un contrato inteligente?, ¿Qué efecto tiene la aplicación de un contrato inteligente? Esta investigación se plantea el reto de responder estas interrogantes, y explorar este nuevo camino tecnológico que se abre ante nosotros y que podría aportar algunas soluciones donde antes no las había.

2.1. OBJETIVO GENERAL

Diseñar una aplicación que soporte un contrato inteligente en forma experimental sobre una plataforma de cadena de bloques pública.

2.2. OBJETIVOS ESPECÍFICOS

- Diagnosticar un problema susceptible de ser automatizado aprovechando las ventajas de los contratos inteligentes.
- Diseñar la aplicación escogida en forma de un contrato inteligente.
- Implementar el contrato inteligente acorde al diseño planteado que da respuesta al caso de estudio escogido.
- Evaluar el funcionamiento de la aplicación implementada.

3. JUSTIFICACIÓN

De acuerdo a la opinión del especialista en los negocios Don Tapscott: "*La tecnología que está impactando al mundo no son los medios sociales. Tampoco los grandes volúmenes de datos. No es la robótica, ni la inteligencia artificial, les sorprenderá saber que se trata de la tecnología subyacente a las divisas digitales como Bitcoin. Se llama cadena de bloques (Del inglés, blockchain). No es la palabra más clamorosa del mundo, pero creo que es la próxima generación de Internet, y que encierra una gran promesa para cada negocio, cada sociedad y cada uno de nosotros.*" [1].

Tal vez suene exagerado, pero considerando que se trata de dinero no es descabellado tener en cuenta tales afirmaciones. El blockchain o cadena de bloques es la tecnología que subyace en la creación y manejo de las criptomonedas, pero si extendemos el concepto funciona para los activos en general y son la base de los contratos inteligentes.

Como ya hemos indicado antes, las cadenas de bloques irrumpen en la escena de la informática como una alternativa para desarrollar aplicaciones que permitan el intercambio de activos electrónicos en forma segura sin la necesidad de un intermediario o tercero de confianza. Asimismo, abre las puertas a la implementación de acuerdos automatizados denominados contratos inteligentes. Corresponde a las instituciones académicas, específicamente en el ámbito computacional, explorar este tipo de tecnologías a fin de sistematizar el conocimiento relativo a ellas y poder incorporarlo al conocimiento formal tanto a nivel de investigación como de docencia.

En tal sentido, en el marco de la investigación académica, se justifica el desarrollo de un proyecto exploratorio de este tipo de tecnologías emergentes a través de los instrumentos propios de la investigación donde los trabajos especiales de grado juegan un papel de particular utilidad en este sentido.

Por todo lo anterior, la presente investigación se plantea diseñar e implementar un contrato inteligente basado en alguna plataforma de cadena de bloques que permita comprender el funcionamiento interno de este tipo de tecnologías emergentes.

4. MARCO TEÓRICO

4.1. BASES TEORICAS

Contrato inteligente

Nick Szabo, un científico informático y criptógrafo estadounidense, fue el primero en introducir el término contratos inteligentes en la última década del siglo pasado en su documento "*Smart Contracts: Building Blocks for Digital Markets*" [3]. De esta forma se convierte en el padre de los contratos inteligentes al desarrollar su investigación sobre contratos inteligentes planteando las siguientes interrogantes:

> ¿Cómo es posible que un software se conecte con activos reales? ¿Cómo puede asegurarme eso un software? Además, el manejo de dinero tiene fuertes regulaciones. ¿Estas no afectan de igual modo a los contratos inteligentes? ¿Y qué hay de la manipulación informática? ¿No puede alguna de las partes manipular a su beneficio la computadora donde el contrato fue escrito o inclusive el código de su software? [3]

Todas esas interrogantes sin una respuesta tecnológica conveniente para el momento, hacían que los contratos inteligentes no fuesen posibles en los 90s. Hubo que esperar 20 años para la aparición de la tecnología que permitiera dar respuesta adecuada a las interrogantes de Szabo, a través de las cadenas de bloques (Blockchain).

Aparición de la tecnología blockchain

La tecnología Blockchain o de cadena de bloques vio la luz en el 2008 con el artículo de Satoshi Nakamoto titulado "*Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer*" [4] y se basa en el aprovechamiento del esquema de interconexión *P2P* (peer-to-peer, persona a persona) para eliminar la figura del tercero de confianza (o tercero confiable) en el intercambio seguro de información, dando origen a la posibilidad de intercambiar dinero electrónico en forma directa y segura entre emisor y receptor sin la intermediación de terceros. La eliminación del tercero de confianza disminuye los costos y los tiempos adicionales que sobrecargan la operación cuando éste debe existir. Se crea así el *bitcoin*, la primera moneda intercambiable en forma íntegramente electrónica sin posibilidad de doble-gasto y en forma segura sin la necesidad de un tercero confiable.

Básicamente, el sistema consiste en mantener un registro de las transacciones en un libro contable (*ledger*) global, público y de difícil adulteración, el cual es construido en forma distribuida y progresiva por los nodos participantes de la red que soporta a la cadena de bloques (computadoras conectadas a la red blockchain). Para evitar la adulteración de este

libro contable público, cada transacción posee la firma electrónica del vendedor de la moneda (y dueño de la misma antes de la transacción) junto a un estampado de tiempo del momento en que es realizada la transacción, y son almacenadas por grupo en bloques que se encadenan correlativamente entre sí con un mecanismo que en términos de costo hace muy difícil o poco probable su adulteración. De esta manera, el libro contable público se traduce en la cadena de bloques que se construye en forma distribuida. (Ver figura 1).

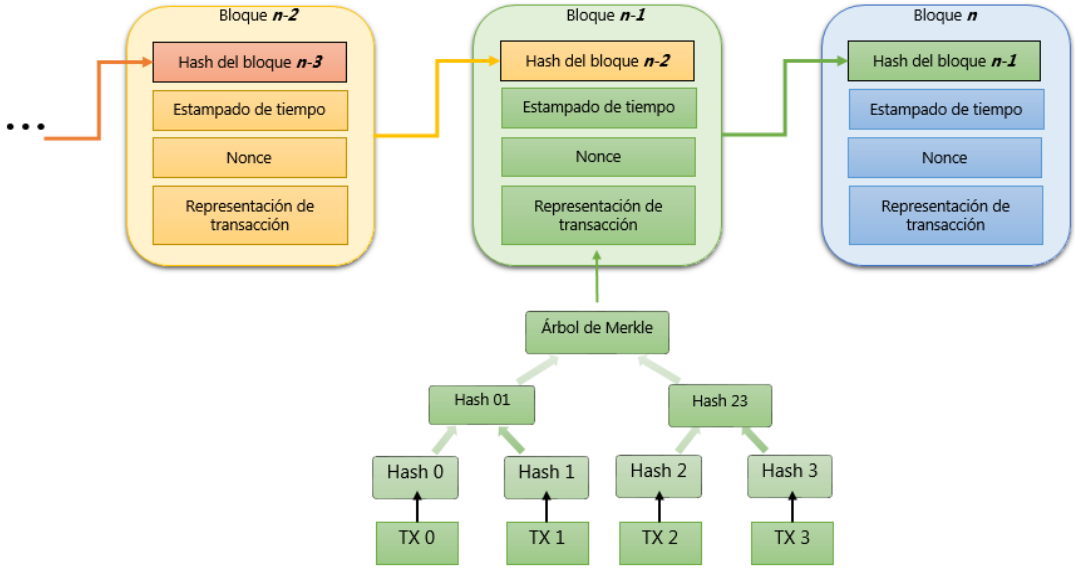


Figura 1 estructura de cadena de bloques.

El proceso de construcción distribuida del libro contable público (cadena de bloques), bajo el enfoque de Satoshi, se ejecuta de la siguiente forma (Ver figura 2):

1. Una vez que una transacción nueva ha sido realizada, con su debida firma y estampado de tiempo, ésta es emitida a todos los nodos de la red, pero ésta no se completa hasta tanto no sea validada y confirmada (consenso).

2. Cada nodo recolecta un grupo de nuevas transacciones para formar un nuevo bloque de la cadena. Crear bloques con grupos de transacciones es más económico en requerimientos de espacio, y más veloz en tiempo de ejecución, a generar un bloque por cada transacción.

3. Para generar un nuevo bloque es necesario resolver una "*prueba de trabajo*" [5], la cual consiste en generar el hash identificador del bloque de forma tal que satisfaga una cierta condición de dificultad, que a su vez está determinada por una cierta cantidad de ceros al inicio de dicho hash. Cuando un nodo resuelve una prueba de trabajo, construye el bloque y lo emite.

4. Los nodos validan el bloque verificando la integridad del hash que identifica el bloque, lo que supone verificar que la prueba de trabajo fue resuelta correctamente y que se cumple la relación de correlatividad con el último nodo de la cadena. Esta relación de correlatividad entre los bloques obliga a que para alterar el bloque n, el atacante está obligado a alterar todos los bloques posteriores al adulterado, lo que a su vez supondría

resolver la prueba de trabajo para dichos bloques, y todo ello antes de que un nuevo bloque sea añadido a la cadena (Ver figura 3). Vale destacar que, si bien realizar la prueba de trabajo suele exigir gran poder de cómputo, su validación es una tarea sencilla.

5. Si el 51% de los nodos de la red expresan su aceptación del nuevo bloque (consenso), éste es incorporado a la cadena como último bloque de la misma, y todas las transacciones contenidas en él se confirman como válidas y se ejecutan. En caso contrario, el bloque se rechaza y pasa a la piscina (pool) de bloques huérfanos (ver figura 4). De esta forma, la cadena de bloques garantiza que no haya doble-pago de monedas.

La figura 2 ilustra el procedimiento descrito arriba para realizar una transacción bajo el esquema de cadena de bloques.

El nivel de dificultad de la prueba de trabajo suele ajustarse de acuerdo al poder de cálculo presente en la red en un momento determinado, para garantizar un tiempo promedio fijo para la creación de los bloques. Es decir, a mayor poder de cómputo, mayor nivel de dificultad y viceversa.

El nodo que crea un bloque con éxito es recompensado con dinero electrónico recién creado (de allí el término de minería para esta actividad) cuya cantidad es establecida en las condiciones iniciales de lanzamiento de la moneda, en el llamado libro blanco. Esto incentiva a la participación ya que a mayor participación mayor seguridad en las transacciones.

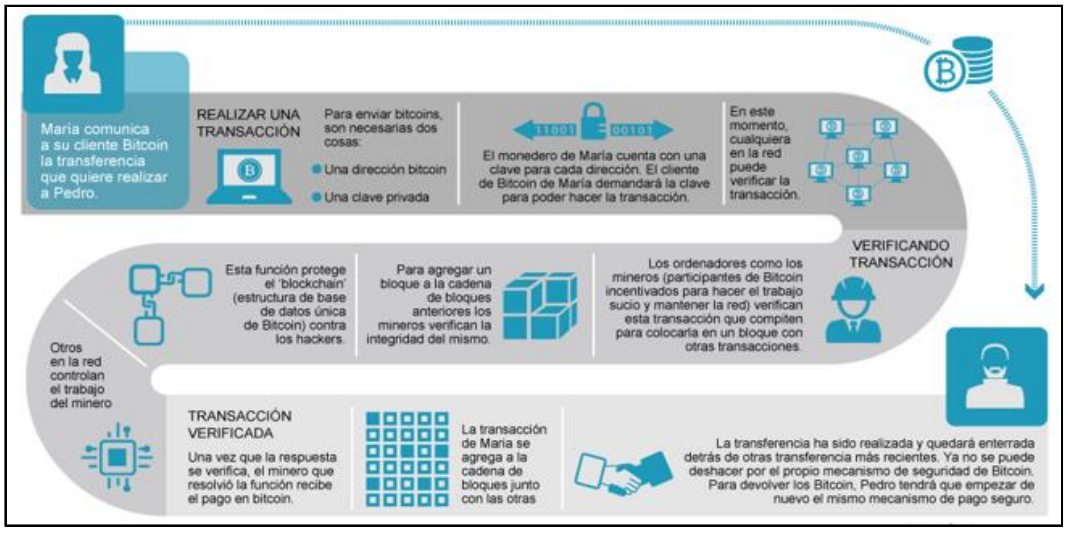


Figura 2 Procedimiento para realizar una transacción en Blockchain. Fuente: Unimet

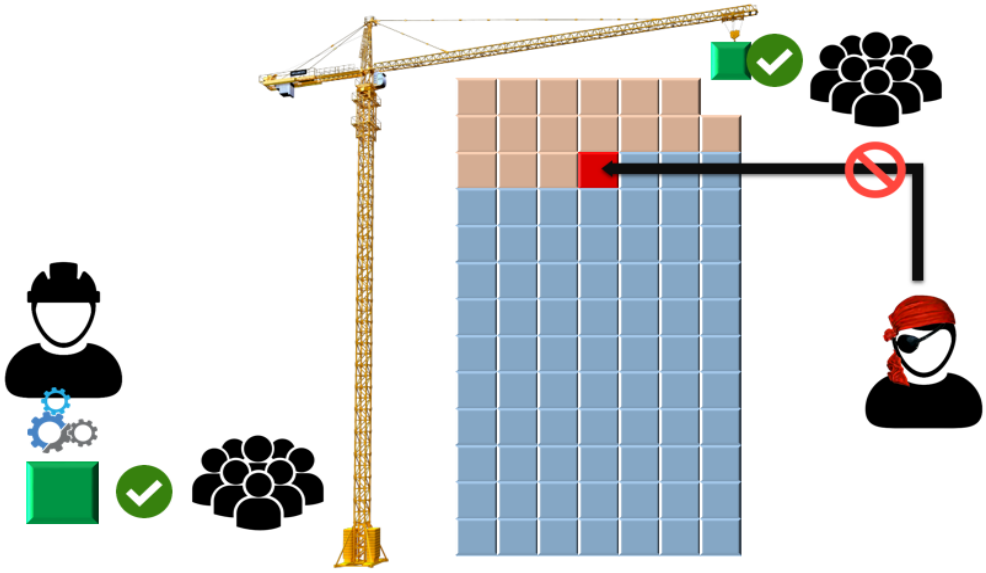


Figura 3 Simbolización gráfica de la dificultad para adulterar un bloque en la cadena: 1) Todos los nodos están trabajando en bloque 91 (bloque verde). 2) Un minero malicioso quiere alterar una transacción en el bloque 74 (bloque rojo). 3. Para lograr su objetivo tendría que adulterar el bloque rojo y recalcular el hash para todos los bloques desde el 74 hasta el 90 antes que sea incorporado a la cadena el bloque 91.

La confiabilidad del sistema está determinada por: a) la relación de correlatividad entre los bloques, b) la dificultad inherente a falsificar esta correlatividad, c) la cantidad de bloques de la cadena en un momento dado, y d) la cantidad de nodos validadores de la red. Mientras más bloques previos y más nodos, mayor el poder de cómputo necesario para falsificar un bloque.

Figura 4 Gentile (2017). Representación de mayoría de nodos de red Bitcoin rechazando un bloque inválido. Tomado de https://www.youtube.com/watch?v=YBNr69vrscw&t=751s

Esto trae consigo una notable mejora en seguridad pues para que alguien altere un bloque sin que la red de nodos validadores lo detecte, tendría que acceder a todos los bloques posteriores al falsificado, y generarlos nuevamente para preservar la correlatividad entre

ellos y, simultáneamente, lidiar con el tiempo que tardan los nodos honestos en validar un nuevo bloque real e incorporarlo a la cadena, porque de lo contrario, el atacante tendría que controlar también el 51% de los nodos validadores. Una labor de conectividad y procesamiento que se torna virtualmente imposible, haciendo que sea más factible crear bloques nuevos reales y ser recompensado por ello que crear un bloque falso.

¿Cuándo usar la tecnología basada en cadena de bloques?

Su filosofía basada en registros públicos y auditables, la virtual imposibilidad de su adulteración, y la verificación descentralizada hecha por los propios pares sin necesidad de un tercero confiable, ha hecho que la tecnología blockchain esté ganando un lugar privilegiado en el mundo de ciertas aplicaciones computacionales. Así, un problema es susceptible de ser automatizado usando la tecnología blockchain cuando satisface las siguientes características:

1. Cuando es necesario que varias entidades distintas escriban y consulten datos en forma compartida.
2. Cuando no se pueden asumir relaciones de confianza entre los entes participantes del sistema y de sus resultados.
3. Cuando no sea posible o sea preferible prescindir de esta entidad centralizadora por razones de eficiencia, costos o dificultad excesiva de implementación.
4. Cuando sea requerido que los registros sean inmutables, es decir, que no puedan ser modificados ni eliminados, una vez sean creados, y estén prestos para ser auditados.

Una forma alternativa para decidir la conveniencia del uso de la cadena de bloques está expresada en el siguiente diagrama de flujo:

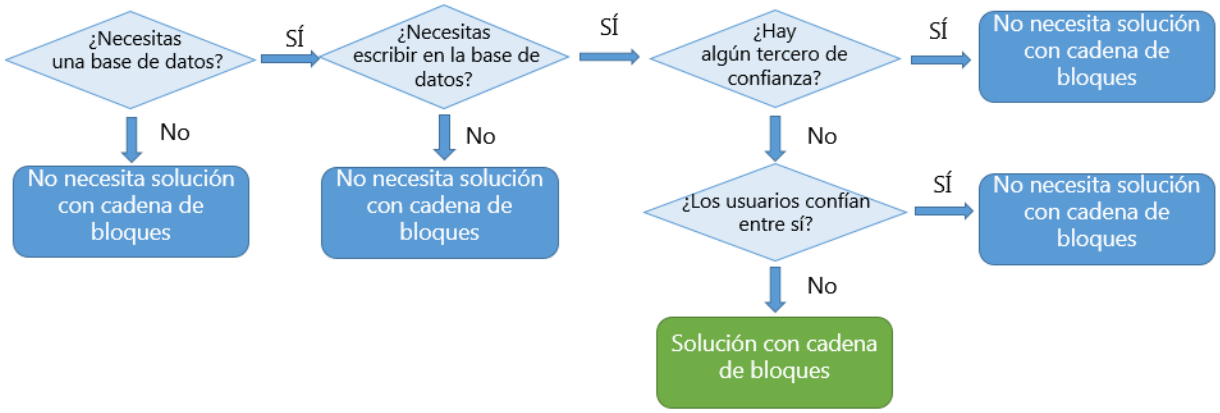


Figura 5 Diagrama de flujo para decidir cuándo utilizar cadena de bloques

Ethereum y los contratos inteligentes sobre la blockchain

Como ya hemos indicado anteriormente, la tecnología Blockchain da pie a la implementación de los contratos inteligentes (s*mart contracts*), que requieren el intercambio de activos como parte de los acuerdos pactados, ya que pueden usar la plataforma de criptoactivos para dicho intercambio al cumplirse las condiciones pactadas, siempre que las mismas puedan validarse a través de una o varias condiciones lógicas de forma automática. Para ilustrar el funcionamiento de un contrato electrónico hipotético, veamos el siguiente ejemplo: supongamos que tenemos dos usuarios que quieren concretar la compra/venta de una casa, el usuario 1 obtendrá la casa (registro automático del título de propiedad a su nombre) una vez que se compruebe el pago del importe acordado (verificación automática de la transferencia al usuario 2 por el valor acordado). Como se puede ver, siempre que se pueda asegurar la confianza en las operaciones electrónicas directas entre los pares, podrá automatizarse el contrato sin necesidad de un intermediario o tercero confiable.

El 30 de julio de 2015, Vitalik Buterin crea la plataforma de cadena de bloques llamada Ethereum, cuya criptomoneda de intercambio es el Ether [6]. Vitalik modificó la estructura de los bloques para poder almacenar en ellos una información adicional a las transacciones de cambio de mano de la criptomonedas. Ethereum ofrece un lenguaje de programación Turing completo, lo que significa que teóricamente se pueden crear programas que modelen cualquiera de los problemas que en la actualidad se pueden modelar con los lenguajes conocidos. En este sentido, es posible programar sobre la blockchain contratos arbitrarios para cualquier tipo de transacción o aplicación directamente entre las partes implicadas. Es por ello que Ethereum se ha perfilado como la plataforma de cadena de bloques pública pionera para el desarrollo de las aplicaciones distribuidas conocidas como *Dapps* de las que hablaremos más adelante.

Las criptomonedas como el Ether son unidades de valor o métodos de pago (en sustitución de dinero tipo FIAT, tales como dólares o euros), mientras que los tokens son fichas de intercambio que pueden representar cualquier activo, pero solo tienen valor en un contexto específico o para un fin determinado. Un token de Ethereum es el ERC-20, del inglés *Ethereum Request for Comment* (20 es el número asignado a esa solicitud), y es un estándar de Ethereum para la creación de otros tokens sobre su plataforma.

Las aplicaciones de contratos inteligentes basadas en la cadena de bloques utilizan tecnologías provistas por la red blockchain sobre la cual será desarrollada. En el caso de Ethereum se provee el lenguaje de alto nivel Solidity. Su sintaxis es similar a la de JavaScript y está enfocado específicamente a un entorno de ejecución en la cadena de bloques llamado Máquina Virtual de Ethereum (EVM). Solidity es orientado a objetos (OO) como C++, por lo que puede resultar familiar si se conoce JavaScript, pero siendo este último para el uso web en HTML.

Solidity cumple con las características de un lenguaje de Turing completo, es decir, posee las estructuras de control necesarias para codificar cualquier aplicación; por ello permite el

uso de un recurso muy valioso que son los ciclos, haciendo que la plataforma Ethereum se considere una gran computadora descentralizada, donde los códigos se ejecutan en los nodos de la red, y no en una entidad centralizada, y los datos relativos a los resultados de la aplicación igualmente quedan almacenados en la cadena de bloques sin necesidad de una base de datos que centralice dichos datos.

Aplicaciones distribuidas o Đapps

Las aplicaciones cuyo funcionamiento interno se soportan sobre una blockchain se denominan aplicaciones descentralizadas, del inglés *decentralized application* (Đapp), por el hecho de que su ejecución y registro de operaciones se distribuye entre la red de nodos del blockchain. Asimismo, los datos resultantes de la aplicación se almacenan en la cadena de bloques haciendo innecesaria la existencia de una base de datos centralizada para este propósito. Por supuesto, podemos inferir limitaciones en el tamaño o cantidad de datos a ser generados a partir de una misma operación ya que ello podría incidir en el tamaño de los bloques haciendo improcedente su propagación rápida entre los nodos de la red. He allí una de las limitaciones de este esquema distribuido.

La literatura identifica una jerarquía entre los tipos de Dapps dependiendo de su forma de interacción con la cadena de bloques. Las Đapp que operan directamente sobre su propia cadena de bloques son de Tipo 1. Ethereum, por ejemplo, es una ĐApp de tipo 1. Las ĐApps que operan directamente sobre cadenas de bloques, pero que dicha cadena no es propia, y por ende requiere hacer uso de las API para su interacción con ella son de Tipo 2. Un ejemplo es la aplicación *Raiden Network* que es “una solución de escalamiento fuera de la cadena, que permite pagos casi instantáneos, de bajo costo y escalables” [7]. Finalmente, las ĐApp que usan ĐApps de Tipo 2 para establecer su contacto con la cadena de bloques son de Tipo 3. Un ejemplo de una ĐApp Tipo 3 sería una tienda de contenido *online*, que use una ĐApp de Tipo 2 para agilizar el intercambio de activos.

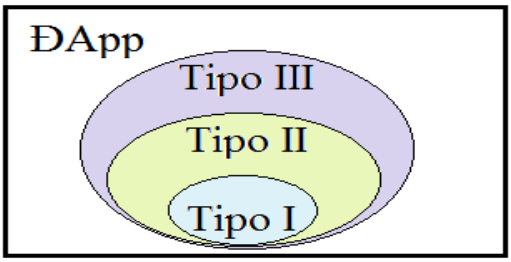


Figura 6 Representación de las ĐApp como conjuntos.

Ethereum resalta utilizar la redundancia para crear seguridad en la cadena de bloques. Además, agrega porque los principales objetivos de su uso son de lectura y escritura en la blockchain en lugar de la realización de cálculos:

“Cada nodo que participa en la red ejecuta el EVM (*Ethereum Virtual machine*) como parte del protocolo de verificación de bloque. Revisan las transacciones enumeradas en el bloque que están verificando y ejecutan el código según lo desencadenado por la transacción dentro del EVM. Todos y cada uno de los nodos completos de la red realizan los mismos cálculos y almacenan los mismos valores. Su procesamiento paralelo es redundantemente paralelo. Esto es para ofrecer una manera eficiente de llegar a un consenso sobre el estado del sistema sin necesidad de terceros confiables, oráculos o monopolios.

Pero lo más importante es que no están allí para un cálculo óptimo. El hecho de que las ejecuciones de contratos se repliquen de forma redundante a través de los nodos, naturalmente los hace costosos, lo que generalmente crea un incentivo para no usar blockchain para los cálculos que se puedan hacer fuera de la cadena” [8].

Para evitar que los usuarios en sus programas consuman recursos excesivos de los nodos, por avaricia o por error (ciclos infinitos, ineficiencia en el código, etc.), se impone un costo por ejecución a través de una unidad de pago denominada “Gas”. El Gas es una tarifa que representa “el coste que tiene el realizar una operación o un conjunto de operaciones en la red Ethereum… se refiere al “**GASto**” de tipo computacional dentro de Ethereum” [9]. El precio del gas se fija al momento de hacer una operación de escritura en la cadena de bloques, por lo que el costo total será la cantidad de Gas que requiere la operación multiplicada por el precio ofertado a los mineros. De allí que la eficiencia de ejecución sea un aspecto muy importante a tomar en cuenta a la hora de crear un contrato inteligente en Ethereum, sino las partes involucradas perderán dinero al derrochar Gas. Las consultas a la cadena de bloques no cuestan gas. En la tabla 1 podemos ver el costo en gas para un conjunto de instrucciones propias del lenguaje Solidity.

Tabla 1 Costo del GAS (Resumida). Fuente: Ethereum

Parámetro	**Costo Gas**	Parámetro	**Costo Gas**	Parámetro	**Costo Gas**
DUP	**3**	SHA3BASE	**30**	POP	**2**
SWAP	**3**	SHA3WORD	**6**	MLOAD	**3**
PUSH	**3**	ECRECOVER	**3000**	MSTORE	**3**
ADD	**3**	SHA256BASE	**60**	MSTORE8	**3**
MUL	**5**	SHA256WORD	**12**	SLOAD	**50**
SUB	**3**	RIPEMD160BASE	**600**	STORAGEADD	**20000**
DIV	**5**	RIPEMD160WORD	**120**	STORAGEMOD	**5000**
SDIV	**5**	IDENTITYBASE	**15**	STORAGEKILL	**5000, plus 15000 refund**
MOD	**5**	IDENTITYWORD	**3**	JUMP	**8**
SMOD	**5**	ADDRESS	**2**	JUMPI	**10**
ADDMOD	**8**	BALANCE	**20**	PC	**2**
MULMOD	**8**	ORIGIN	**2**	MSIZE	**2**
EXPBASE	**10**	CALLER	**2**	GAS	**2**
EXPBYTE	**10**	CALLVALUE	**2**	JUMPDEST	**1**

SIGNEXTEND	**5**	CALLDATALOAD	**3**	GLOG	**375**
LT	**3**	CALLDATASIZE	**2**	GLOGTOPIC	**375**
GT	**3**	CALLDATACOPYBASE	**3**	GLOGDATA	**8**
SLT	**3**	CODESIZE	**2**	CREATE	**32000**
SGT	**3**	CODECOPYBASE	**3**	CREATEDATA	**200**
EQ	**3**	GASPRICE	**2**	GCALL	**40**
ISZERO	**3**	EXTCODESIZE	**20**	GCALLVALUETRANSFER	**9000**
AND	**3**	EXTCODECOPYBASE	**20**	GCALLSTIPEND	**2300**
OR	**3**	GCOPYWORD	**3**	GCALLNEWACCOUNT	**25000**
XOR	**3**	BLOCKHASH	**20**		
NOT	**3**	COINBASE	**2**	RETURN	**0**
BYTE	**3**	TIMESTAMP	**2**	STOP	**0**
		NUMBER	**2**	SUICIDE	**0**
		DIFFICULTY	**2**	GSUICIDEREFUND	**24000 refund**
		GASLIMIT	**2**	MEMWORD	**3**
				QUADCOEFFDIV	**512 (divisor)**
				GTX	**21000**
				GTXDATANONZERO	**68**
				GTXDATAZERO	**4**

5.

5. MARCO METODOLÓGICO

Al respecto de la metodología para una investigación de carácter científico cabe citar a Popper [10] cuando señala:

> "El éxito de la ciencia no se basa en reglas de inducción, sino que depende de la suerte, el ingenio y las reglas puramente deductivas de argumentación crítica.
>
> Puedo resumir algunas de mis conclusiones de la manera siguiente:
>
> (1) La inducción, es decir, la inferencia basada en muchas' observaciones, es un mito. No es un hecho psicológico, ni un hecho de la vida cotidiana, ni un procedimiento científico.
>
> (2) El procedimiento real de la ciencia consiste en trabajar con conjeturas: en saltar a conclusiones, a menudo después de una sola observación (como lo destacan, por ejemplo. Hume y Born).
>
> (3) Las observaciones y los experimentos repetidos funcionan en la ciencia como test de nuestras conjeturas o hipótesis, es decir, como intentos de refutación.
>
> (4) La errónea creencia en la inducción se fortifica por la necesidad de un criterio de demarcación que, según se cree tradicional pero erróneamente, sólo lo puede suministrar el método inductivo."

En correspondencia con la cita anterior, en el presente capítulo se describen los procedimientos y pasos que se llevaron a cabo para el logro de los objetivos propuestos, siempre enmarcados en la metodología de investigación hipotético-deductiva. Por ello analizamos el problema y formulamos una hipótesis que próximamente será develada (modelo conceptual), se diseñó un experimento que permitiera comprobar o refutar la hipótesis planteada (modelo lógico) para, finalmente, implementar dicho experimento (modelo físico) siguiendo el flujo representado en la figura 7. Cabe destacar que en esta investigación la etapa de implementación (modelo físico o experimento) se corresponde con el desarrollo de una aplicación de software: el desarrollo de un contrato inteligente sobre una plataforma de cadena de bloques. De allí que esta etapa comprenda en sí misma, las fases propias de desarrollo de software, a saber: análisis, diseño e implementación, referidas en este caso a la construcción de la aplicación de interés que servirá, en el marco de la investigación propuesta, como el experimento que permitirá comprobar o no la hipótesis planteada.

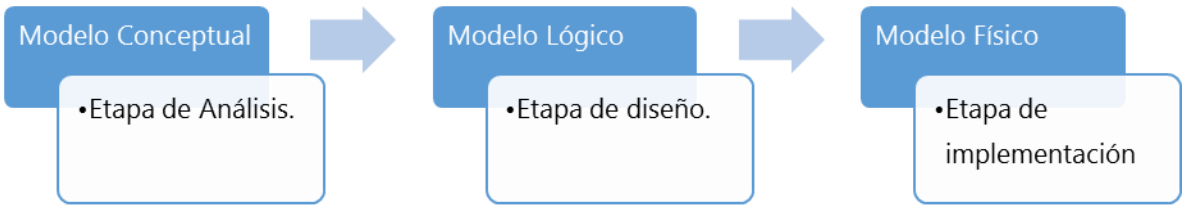


Figura 7 Interpretación de la metodología hipotética-deductiva.

5.1. ETAPA DE ANÁLISIS DE LA INVESTIGACIÓN.

Observación del fenómeno

Las transacciones entre pares, en su naturaleza, no contemplan la existencia de un tercero de confianza. La compra de un bien, por ejemplo, es una transacción entre el vendedor y el comprador del bien, sin que medie en la operación ningún tercero. La extensión de esta transacción al espacio electrónico, no tendría por qué modificar la naturaleza de la operación, a menos que limitaciones tecnológicas así lo obliguen. La aparición del tercero de confianza en las transacciones electrónicas surge de la imposibilidad de garantizar la transparencia y honestidad de la transacción entre pares con los recursos tecnológicos existentes. Sin embargo, la inclusión del tercero de confianza supone las siguientes desventajas:

- Se centraliza en el tercero de confianza la validación de la operación.
- Se agregan a la operación los costos derivados del proceso centralizado de validación incluyendo la ganancia esperada por prestar dicho servicio.
- La centralización en un tercero introduce retardos propios de la burocracia requerida para efectuar la validación que no tendrían por qué estar presentes en la transacción directa entre pares.
- La centralización en un tercero de múltiples transacciones hace más atractivo a potenciales criminales el arremeter contra el tercero, obligando a éste a realizar importantes inversiones en la seguridad de su infraestructura, aumentando aún más los costos de la transacción.

La etapa de análisis en esta investigación consistió fundamentalmente en el estudio documental relacionado con las tecnologías Blockchain y su funcionamiento en la implementación de contratos inteligentes para realizar operaciones entre pares sin la intermediación de un tercero de confianza. Gran parte de este trabajo documental ha sido recogido en la sección de Marco Teórico del presente trabajo, donde se identifica y justifica el problema a ser atendido (observación), los objetivos trazados en la investigación, y los

antecedentes de esta investigación. Todo ese proceso de análisis derivó en la formulación de la siguiente hipótesis que sintetiza nuestro modelo conceptual.

Hipótesis

El uso de la tecnología basada en cadena de bloques (Blockchain) permite la implementación de transacciones entre pares de forma transparente y segura, sin la intermediación de un tercero de confianza, disminuyendo en consecuencia costos y retrasos derivados de las operaciones de validación centralizadas en dicho tercero.

De la cual se derivaron, a su vez, los objetivos planteados en la investigación descritos en la sección 2, cuyo desarrollo abordaremos en las siguientes secciones.

5.2. ETAPA DE DISEÑO DE LA INVESTIGACIÓN

5.2.1. Definición del experimento

Para realizar la comprobación de nuestra hipótesis hemos decidido implementar una aplicación (app) de contrato inteligente basada en cadena de bloques. La escogencia del problema concreto a ser modelado a través de un contrato inteligente respondió, por una parte, al cumplimiento de los criterios arriba esbozados para ser susceptible de ser abordado con tecnología basada en cadena de bloques, y por otra parte, a un interés particular relacionado con otras investigaciones en curso dentro del grupo de desarrollo en el que se inscribe este trabajo.

En tal sentido, el problema escogido para ser modelado y desarrollado consistió en un sistema de votación electrónico para dar soporte a un proceso electoral, el cual sería modelado, a los efectos de esta investigación, a partir de una aplicación basada en tecnología de cadenas de bloques. En esencia, se quiere desarrollar una aplicación de software que permita representar el contrato implícito que existe entre electores y candidatos al momento de organizar y ejecutar un evento electoral.

5.2.2. ¿Es un problema susceptible de ser modelado con cadena de bloques?

En el marco del problema escogido para ser modelado, el uso de la tecnología de cadena de bloques (o Blockchain) se justifica debido a que el problema del sistema electoral cumple con las características arriba mencionadas. Analicemos dichas características en el marco del problema:

1. ***Múltiples partes interesadas y datos compartidos:*** Para un sistema electoral son indispensables los electores y los candidatos. El ente rector entra en juego para ofrecer el sistema, como cuarta parte interesada estarán quienes espectarán el proceso electoral y la totalización de los votos. Los datos utilizados de los electores serán los necesarios para autenticarle, mientras que los datos utilizados de los candidatos serán los necesarios por las personas al realizar el acto de vota para identificarle de entre otros candidatos.
2. ***Bajo nivel de confianza entre las partes:*** La cadena de bloques puede permitir un sistema de voto en el que las identidades de los votantes sean infalsificables, estuviesen protegidas al no asociar a las direcciones utilizadas en la cadena de bloques, todo a un coste prácticamente nulo y de acceso público a los registros.
3. ***Ausencia de un tercero de confianza:*** A pesar de que el ente rector seria quien pode a disposición del sistema electoral y se encarga de su funcionamiento, la transferencia de los votos de votantes a candidatos estará en un registro descentralizado al ser una cadena de bloques pública.
4. ***Exigencia de resultados confiables y 100% auditables:*** Convirtiendo los votos en transacciones podemos crear una cadena que lleve cuentas de los votos, de esta manera todos podemos estar de acuerdo en el conteo final porque podemos auditar los registros del sistema electoral en la cadena de bloques de disposición pública.

De todo lo anterior, se puede concluir que el uso de la tecnología de cadena de bloques, entendida como un registro descentralizado de intercambios de activos se ajusta al problema del sistema electoral, ya que el acto de votar se puede homologar al acto de transferir un activo (voto) entre las partes, es decir de un emisor (elector) a un receptor (candidato), en el marco de las condiciones y características que deberán regir el proceso.

5.2.3. Características del proceso electoral que modelaremos como un contrato inteligente

Como ya hemos referido, el problema escogido para ser modelado a través de un contrato inteligente, será el correspondiente a un proceso electoral. En un proceso electoral interactúan esencialmente dos partes: los electores y los candidatos. Por supuesto, existe también un ente rector del proceso, cuyo rol se concentra en la organización logística del evento electoral, mas no tiene porqué erigirse en un intermediario del acto de emisión del voto.

Así las cosas, en el contexto de nuestro contrato inteligente, los votos serán interpretados como transferencias de tokens desde las billeteras de los electores, hacia las billeteras de los candidatos previamente inicializadas en cero, siendo el resultado de la elección los saldos contenidos en las billeteras de los candidatos una vez finalizada. Las premisas establecidas para este sistema de votación son las siguientes:

1. **Votación universal y uninominal**: El sistema de votación a ser modelado será un sistema universal y uninominal donde todos los electores tienen idéntico derecho a voto, y pueden hacerlo por un único candidato.
2. **Elección del ganador por mayoría simple**: El candidato ganador será aquel que al final del evento electoral acumule mayor cantidad de votos a su favor.
3. **"Un elector – un voto":** Se debe garantizar el respeto a la relación "un elector-un voto", es decir, cada elector podrá emitir un único voto por el candidato de su preferencia.
4. **Inviolabilidad del secreto del voto:** Se debe garantizar el anonimato del votante, es decir que la identidad del votante no quede vinculada al voto emitido, ni almacenada en la cadena de bloques, solo el voto queda registrado en forma inmutable. Sin embargo, sería deseable que el votante, y solo el votante, pudiera auditar su voto durante o después del proceso.
5. **Blindaje de la transparencia de los resultados**: La aplicación debe garantizar la transparencia del proceso, siendo los resultados expresión fiel de la voluntad de los electores participantes. Para ello los resultados deben quedar almacenados en un registro inmutable e inalterable antes, durante o después del evento electoral, pudiendo ser auditado en cualquier momento.

En resumen, el diseño de nuestro experimento para comprobar la factibilidad y conveniencia del uso de contratos inteligentes basado en cadena de bloques como estrategia para la implementación de transacciones entre pares de forma transparente y segura, sin la intermediación de un tercero de confianza, disminuyendo en consecuencia costos y retrasos derivados de las operaciones de validación centralizadas en dicho tercero, es decir, para comprobar nuestra hipótesis, consiste en el desarrollo de un contrato inteligente que modele un proceso electoral bajo las premisas arriba descritas. En adelante, abordaremos el proceso de implementación de este experimento, que como ya hemos indicado, consiste en el desarrollo (análisis, diseño e implementación) de una aplicación de software.

5.3. ETAPA DE IMPLEMENTACIÓN

En esta etapa de la investigación abordaremos el proceso de desarrollo del contrato inteligente arriba descrito. En tal sentido, en virtud de que nuestro experimento se corresponde con el desarrollo de una aplicación de software, nos corresponde cumplir las fases características del ciclo de desarrollo de software, a saber, análisis, diseño e implementación, con las iteraciones que corresponda realizar en el proceso de ajuste de la aplicación a los objetivos trazados.

5.3.1. Análisis de la aplicación escogida

En esta fase nos corresponde identificar los actores que participan del sistema a ser modelado e identificar los casos de uso que serán transformados en funcionalidades de la aplicación.

5.3.1.1. Identificación de los actores (participantes) y tareas del sistema

Los actores se subdividen en dos grupos, como primer grupo tenemos a los usuarios naturales del sistema: *Los administradores, los candidatos, los electores, y los observadores*. Como segundo grupo, tenemos a las entidades, no usuarias, del propio sistema: *La base de datos y la cadena de bloques*.

Vale destacar que un *observador* es todo usuario que sin estar registrado como votante puede consultar los resultados parciales o totales del proceso electoral. En tal sentido, un *elector* tiene la posibilidad de consultar la totalización, pero además puede ingresar al sistema al registrarse para así votar en caso de que unas elecciones estén en curso. A su vez, un *candidato* es al mismo tiempo un *elector*. Las *Bases de Datos* y la *Cadena de bloques* realizarán tareas de suministrar y registrar la información actuando en conjunto para el funcionamiento del sistema siendo especializaciones del registro electoral

- El ente rector.
- Los candidatos.
- Los electores.
- Observadores
- La Base de datos.
- La Cadena de Bloques.

5.3.1.2. Identificación de los casos de uso del sistema.

Estos actores operarán el sistema y darán origen a transacciones sobre el mismo, entre las cuales identificamos las siguientes:

- Registrar candidato: Un administrador registrara los datos de los candidatos a estar disponibles para ser seleccionados en las elecciones.
- Registrar elector: Una persona registrara sus datos para ingresar al sistema y poder votar en elecciones que estén abiertas.
- Iniciar evento electoral: Un administrador cargara los datos para iniciar el evento electoral.
- Asignación de voto a elector: El sistema asignará a un elector su voto.
- Realizar el acto de votar: Los electores autenticados seleccionaran de entre una lista de candidatos al candidato de su preferencia y votan por él.

- Cerrar evento electoral: Un usuario administrador podrá finalizar el evento electoral.
- Consultar lista de candidatos: Los administradores autenticados en el sistema podrán ver la lista de los candidatos registrados en sistema.
- Consultar lista de personas: Los administradores autenticados en el sistema podrán ver la lista de las personas registradas disponibles participar como electores.
- Consultar lista de administradores: Los administradores autenticados en el sistema podrán ver la lista de los administradores registrados en sistema.

5.3.1.3. Modelado de los casos de uso

Cada una de estas operaciones da origen a los casos de uso del sistema, los cuales son diagramados en las figuras 8, 9 y 10.

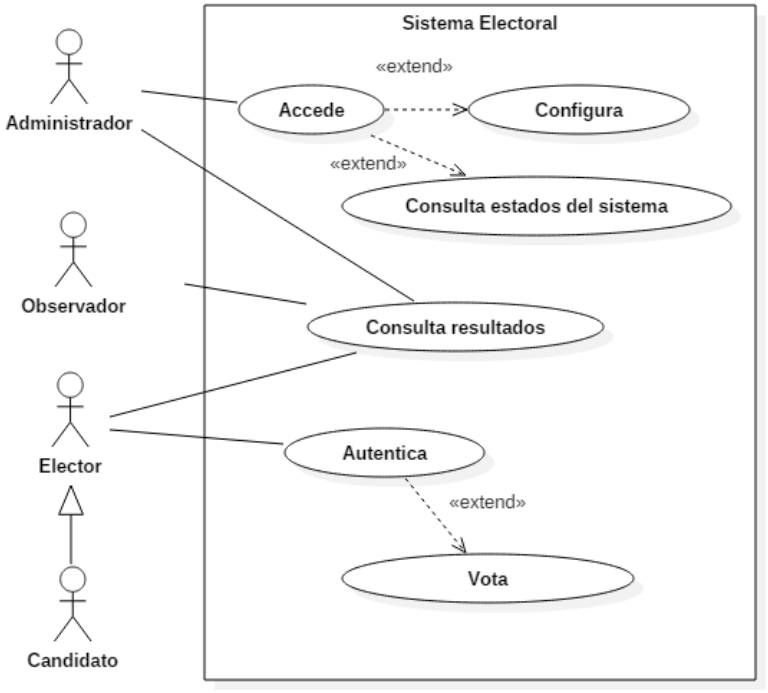


Figura 8 Diagrama de casos de usos de un sistema de elecciones

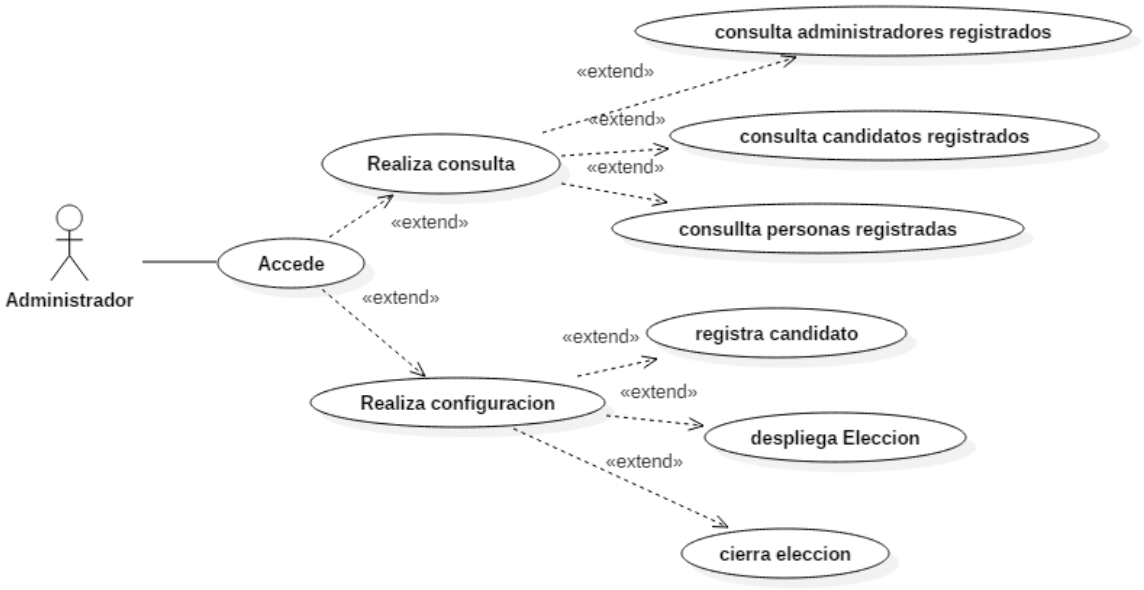


Figura 9 Vista detallada del actor "Administrador" del diagrama de casos de uso.

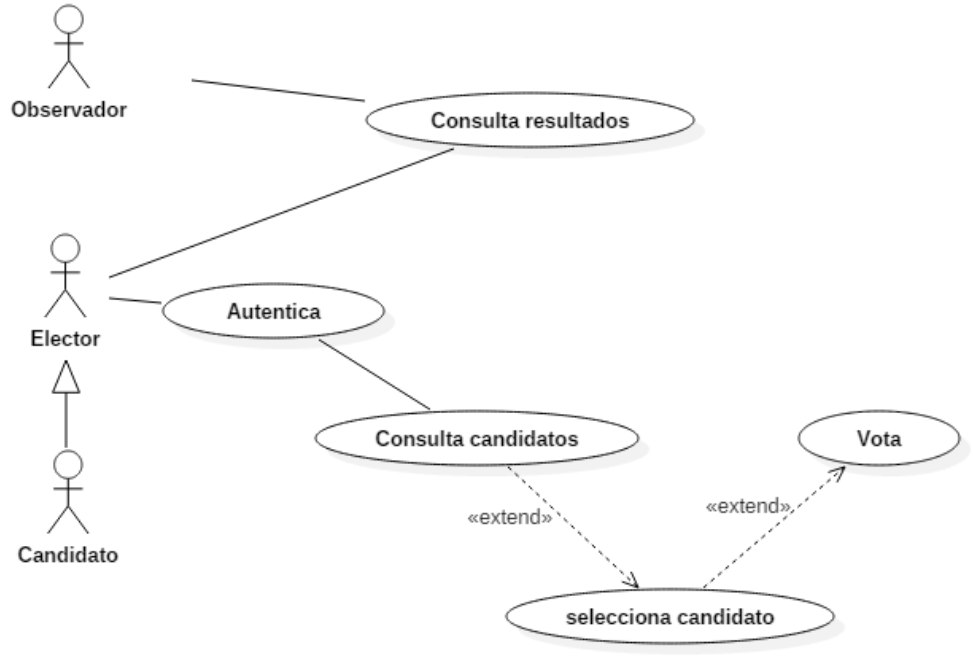


Figura 10 Visualización detallada de actores "Observador", "Elector" y "Candidato" del diagrama de casos de uso.

5.3.2. Diseño de la aplicación escogida

Selección de la tecnología

La plataforma de cadena bloques (Blockchain) escogida para montar sobre ella la aplicación de votación fue Ethereum, por ser una plataforma pública ampliamente reconocida como segura y confiable, diseñada para crear sobre ella contratos inteligentes, con APIs disponibles, además de poseer una *tesnet* (red de prueba) para poder realizar pruebas. La escogencia de Ethereum supone el uso del lenguaje *Solidity* provisto por Ethereum como lenguaje de programación del funcionamiento interno del contrato inteligente que, como hemos dicho anteriormente, es un lenguaje completo de Turing, permitiendo una implementación amplia de cláusulas si así se requiriera.

Para almacenar información pertinente de actores, así como eventos del funcionamiento y resultados de la aplicación que necesiten ser imperecederos utilizaremos bases de datos relacionales. Asimismo, las transacciones y los resultados intermedios y finales de las elecciones quedarán registrados de manera inmutable en la cadena de bloques.

Como *framework web* para el desarrollo de la interfaz de usuario de la aplicación se utiliza *Node.js* que funciona como una capa de software sobre el sistema operativo que permite integrar, a través de programas escritos en *ECMAscript*, el uso de módulos tales como *express*, *web3.js*, y *mysql*, entre otros.

Como ya indicáramos, la plataforma de cadenas de bloques escogida es *Ethereum* la cual proporciona varias herramientas para el uso de su cadena de bloques, entre ellas *Remix* que es su entorno de desarrollo integrado o *IDE* (*Integrated Development Environment*) con el cual poder visualizar, editar, compilar, desplegar y ejecutar los contratos inteligentes escritos en *Solidity*, de forma rápida y sencilla haciendo uso de un navegador web. Para la comprensión de *Solidity* es imperativa la lectura de su documentación. La versión de *Solidity* escogida es la **0.5.3** debido a su amplia documentación disponible para la fecha y compatibilidad con superiores versiones hasta anteriores a la 0.7.0 (para la fecha de abril de 2020 la versión actual es 0.6.4)

Ethereum pone a disposición la *API* (*Application Programming Interface*) *web3.js* cuyos métodos son invocados desde un programa escrito en el lenguaje de programación *ECMAscript*, el cual se ejecuta sobre el entorno multiplataforma *Node.js*.

Como manejador de base de datos se ha escogido *MySQL* por lo que para acceder a los datos se debe incluir el módulo *mysql* provisto por *Node.js*, como interfaz entre el contrato inteligente y las funcionalidades del manejador de base de datos.

Arquitectura de la ĐApp

Todo este conjunto de herramientas permitirá el desarrollo de dos aplicaciones descentralizadas (ĐApp) con vistas y controladores distintos que utilizan el mismo tipo de

tecnologías (*frameworks, APIs*, etc) y comparten el uso de la cadena de bloque y la base de datos. Ambas aplicaciones quedarán integradas en una arquitectura como la que se aprecia en la figura 11.

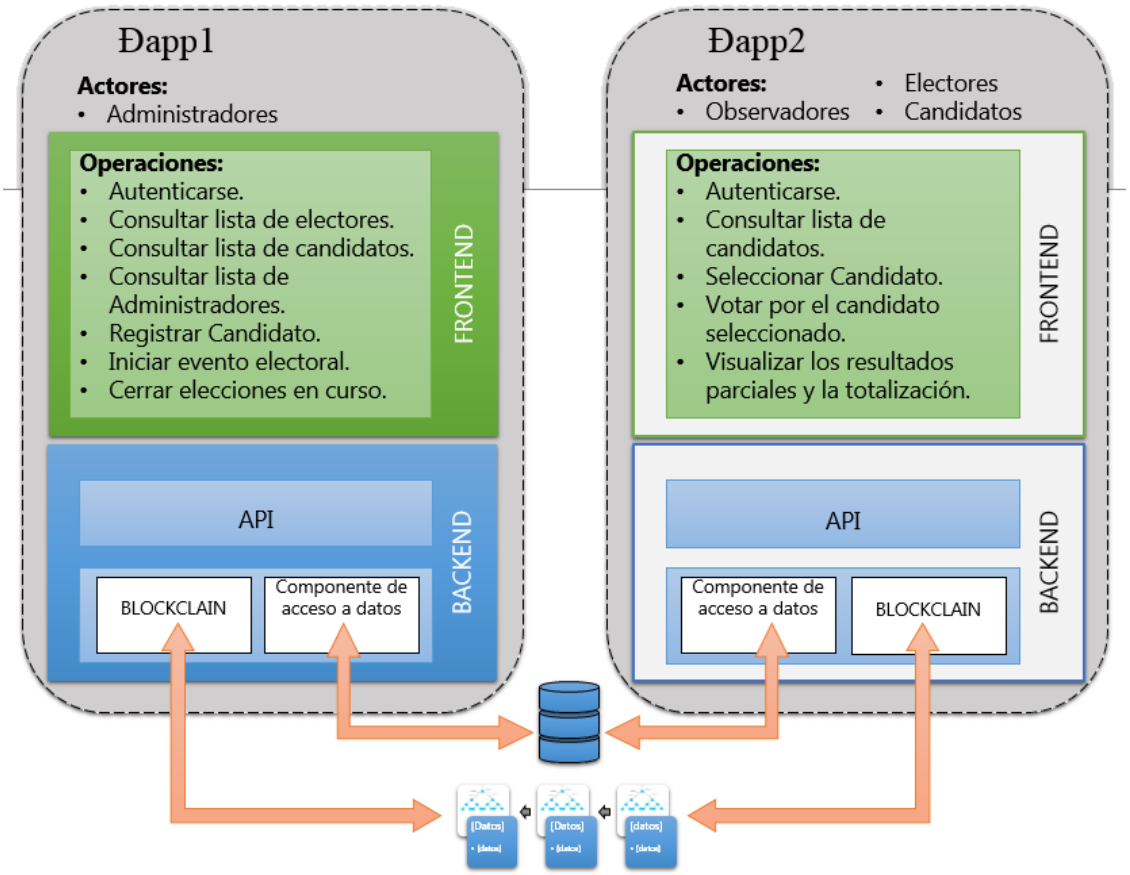


Figura 11 Representación de las aplicaciones descentralizadas para gestión de elecciones y ejercicio del voto.

La Đapp 1 la hemos denominado ¨Transfer E-Government¨ y será la encargada de la carga de datos de inicialización y configuración del evento electoral, así como de la administración de la aplicación.

La Đapp 2 denominada "Decisión Block" modela las funcionalidades relativas al acto de votar propiamente dicho, permitiendo a los usuarios electores que realicen el proceso de autenticación, y efectúen el acto de votar por un candidato, además de permitir visualizar los resultados parciales y la totalización, en el marco de las políticas definidas para tal acción por la autoridad rectora del evento electoral.

En el contrato inteligente se almacenarán el nombre del token que representará al voto, las siglas que lo identificarán, la cantidad total de tokens a crear, los decimales a utilizar (en nuestro caso será cero debido a que los votos son valores enteros), y la dirección de la

billetera del propietario del contrato inteligente que corresponde a la entidad rectora del evento electoral. Además, se almacenarán las direcciones de las billeteras receptoras de los votos (billeteras de los candidatos), las direcciones de las billeteras desde donde se emitirán los votos (billeteras de los electores) y los votos (tokens) asignados a cada candidato.

Se crearán métodos en el contrato con las cuales almacenar las direcciones de las billeteras de los candidatos, asignar un voto (token) a la billetera de un elector y transferir el voto de la billetera de un elector a la de un candidato al votar.

Entidades de la Base de Datos

En el proceso de modelado del sistema electoral identificamos las entidades siguientes:

Entidad Persona: Esta entidad permite almacenar toda la información de los potenciales votantes del sistema y se corresponde con lo que sería el Registro Electoral correspondiente al proceso electoral que se va a realizar. Los atributos de esta entidad son los siguientes:

- Nacionalidad (nacionalidad del votante).
- Cedula (cédula del potencial votante).
- Apellido (apellido del potencial votante).
- Nombre (nombre del potencial votante).
- Password (contraseña de acceso del potencial votante al sistema).
- Fecha_de_nacimiento (año, mes y día de nacimiento del potencial votante).
- Direccion_de_habitacion (Dirección en la que reside el potencial votante).

Vale destacar que el proceso de autenticación originalmente fue pensado para ser realizado a través de algún rasgo biométrico del votante (huella dactilar, iris o rostro), por lo que dicho atributo debe estar en la base de datos. En este caso, por las limitaciones de este prototipo experimental, dicho atributo ha sido sustituido por una clave de acceso (password) razón por la cual aparece dicho campo como un atributo de la entidad.

Entidad administrador: Esta entidad permite almacenar la información de los usuarios que fungirán como administradores de la aplicación (en sus distintos roles y distintas perisologías) y representa al actor “órgano rector”. Sobre los actores representados por esta entidad recaen funciones tales como registrar a un nuevo candidato, actualizar el registro electoral, inicializar las variables de un evento electoral, y dar inicio a un evento electoral. Los atributos de esta entidad son los siguientes:

- Email (email de acceso del usuario administrador al sistema).
- Pass (contraseña de acceso del usuario administrador al sistema).

- Rol (rol y permisos de administración asignados al usuario).
- Facultad (comentario corto del usuario administrador).

Entidad candidato: Esta entidad permite almacenar toda la información de los candidatos para un evento electoral específico. Los atributos de esta entidad son los siguientes:

- Nacionalidad_candidato (nacionalidad del candidato).
- Cedula_candidato (cédula del candidato).
- Apellido_candidato (apellido del candidato).
- Nombre_candidato (nombre del candidato).
- Billetera_candidato (cuenta que almacena los votos del candidato en la blockchain).
- Votos_candidato (cantidad de votos del candidato en el sistema).
- Contrato_candidato (Hash de transacción del contrato inteligente de las elecciones en curso).

Entidad elección: Esta entidad permite almacenar la información del evento electoral específico que se está realizando en un momento determinado. Podemos decir que esta entidad representa los datos de identificación del contrato inteligente asociados al evento electoral que se va a realizar. Los atributos de esta entidad son los siguientes:

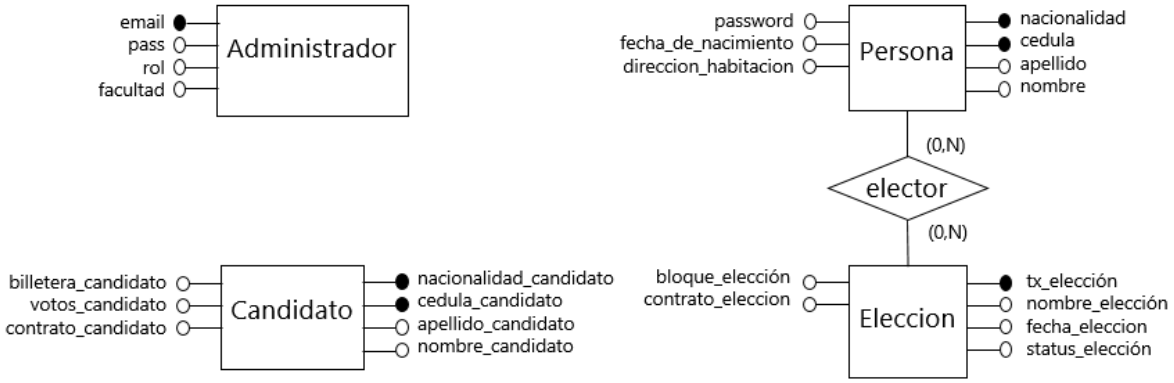
- Tx_eleccion (Hash de transacción del contrato inteligente de la elección).
- Nombre_eleccion (Nombre de la elección).
- Fecha_eleccion (Fecha de creación de la elección).
- Status_eleccion (Estado en él se encuentra la elección)
    1. Elección registrada.
    2. Elección en curso.
    3. Elección cerrada.
- Bloque_eleccion (El bloque de la cadena en la que se encuentra la elección).
- Contrato_eleccion (Contrato inteligente de la elección).

Relaciones de la Base de Datos

Relación Elector (Entidades: Persona, Elección) Esta relación es la responsable de registrar a las personas que ya votaron en las elecciones.

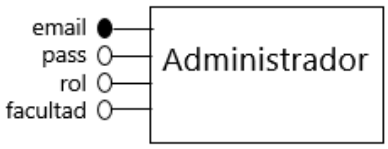
Modelo Entidad-Relación

A continuación, se procederá a presentar el modelado conceptual de la base de datos.



Transformación de las entidades

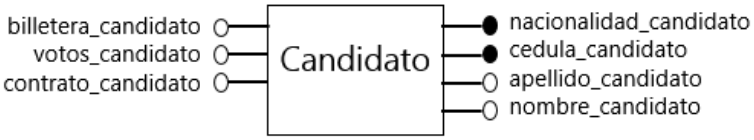
Entidad Administrador: La entidad administradora en el modelo entidad - relación es la mostrada a continuación:



Al realizar la transformación para obtener el modelo lógico de esta entidad se genera la siguiente tabla:

Administrador (email, pass, rol, facultad)

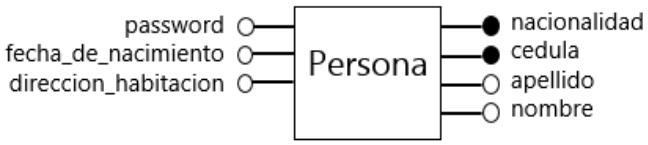
Entidad Candidato: La entidad candidato en el modelo entidad - relación es la mostrada a continuación:



Al realizar la transformación para obtener el modelo lógico de esta entidad se genera la siguiente tabla:

Candidato (nacionalidad_candidato, cedula_candidato, apellido_candidato,
nombre_candidato, billetera_candidato, votos_candidato, contrato_candidato)

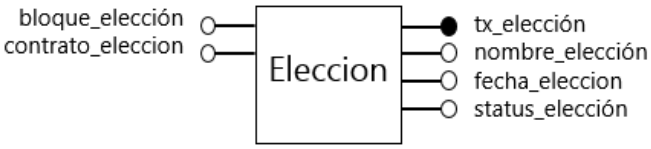
Entidad Persona: La entidad persona en el modelo entidad - relación es la mostrada a continuación:



Al realizar la transformación para obtener el modelo lógico de esta entidad se genera la siguiente tabla:

Persona (nacionalidad, cedula, apellido, nombre, password, fecha_nacimiento, dirección_habitacion)

Entidad Eleccion: La entidad eleccion en el modelo entidad - relación es la mostrada a continuación:

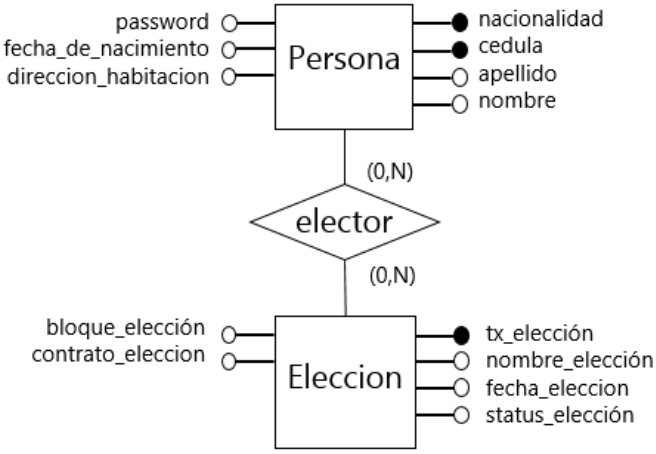


Al realizar la transformación para obtener el modelo lógico de esta entidad se genera la siguiente tabla:

Eleccion (tx_eleccion, nombre_eleccion, fecha_eleccion, status_eleccion, bloque_eleccion, contrato_eleccion)

Transformación de las relaciones

Relación Elector (Entidades: PERSONA, ELECCION): Esta relación perteneciente al modelo entidad – relación es mostrada a continuación:



Al realizar la transformación para la obtención del modelo lógico, debido a que la relación recibe una relación de muchos a muchos, es necesario unir a esta tabla las claves primarias de las entidades involucradas en esta relación. Como resultado se obtiene la siguiente tabla:

Elector (persona_nacionalidad, persona_cedula, elección_tx_eleccion)

Proceso de normalización

El proceso de normalización permite transformar un conjunto de relaciones, a una forma más sencilla. Para empezar a estudiar las tres primeras formas normales, además de la de Boyce-Codd, se debe introducir el concepto de dependencia funcional.

"Dada una relación R, el atributo Y de R depende funcionalmente del atributo X de R si siempre que dos tuplas de R concuerden en su valor de X, deben por fuerza, concordar en su valor de Y".

Para normalizar las tablas obtenidas en la fase anterior es necesario aplicar a estas el proceso de normalización que incluye las 3 primeras formas normales y la de Boyce-Codd.

Este proceso de normalización es presentado a continuación.

Primera Forma Normal (1FN)

Para que una relación se encuentre en 1era forma normal es necesario que todos los dominios simples y subyacentes solo contengan valores atómicos, no se permiten atributos compuestos ni polivalentes.

La 1FN es cumplida por todas las tablas obtenidas anteriormente, ya que todas ellas solo están compuestas por atributos que contienen valores atómicos, por lo tanto, estas tablas se encuentran en 1era forma normal.

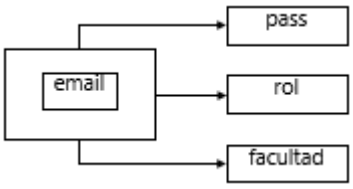
Segunda Forma Normal (2FN)

Para que una relación se encuentre en 2FN debe estar en 1FN y adicionalmente todos los atributos no clave deben depender por completo de la clave primaria o de las claves candidatas. A continuación, se realizará un análisis de todas las tablas obtenidas en la fase anterior para verificar que estas cumplan con la 2FN candidatas.

Tabla administrador: La tabla administrador es la siguiente:

Administrador (email, pass, rol, facultad)

Para indicar las dependencias funcionales utilizaremos el siguiente gráfico:

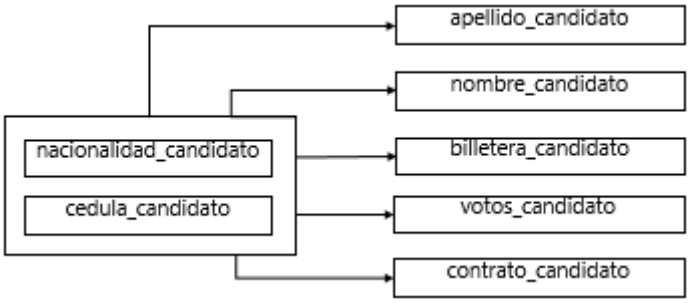


El atributo email es la clave primaria, y esta determina el resto de los atributos no-claves pertenecientes a la relación, por lo tanto, esta relación se encuentra en 2FN.

Tabla Candidato: La tabla candidatos es la siguiente:

Candidato (nacionalidad_candidato, cedula_candidato, apellido_candidato, nombre_candidato, billetera_candidato, votos_candidato, contrato_candidato)

Para indicar las dependencias funcionales utilizaremos el siguiente gráfico:

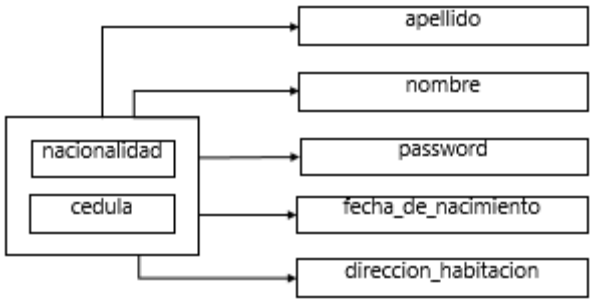


Los atributos nacionalidad_candidato y cedula_candidato son las claves primarias, y estas determinan el resto de los atributos no-claves pertenecientes a la relación, por lo tanto, esta relación se encuentra en 2FN.

Tabla Persona: La tabla persona es la siguiente:

Persona (nacionalidad, cedula, apellido, nombre, password, fecha_nacimiento, direccion_habitacion)

Para indicar las dependencias funcionales utilizaremos el siguiente gráfico:

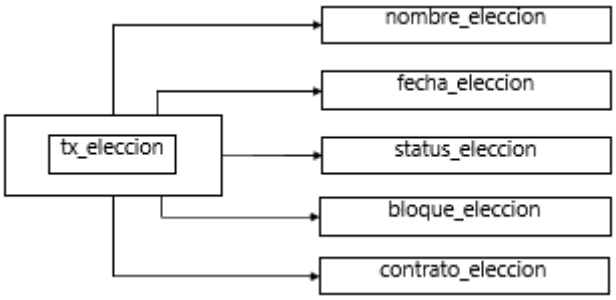


Los atributos nacionalidad y cedula son las claves primarias, y estas determinan el resto de los atributos no-claves pertenecientes a la relación, por lo cual esta relación se encuentra en 2FN.

Tabla Eleccion: La tabla eleccion es la siguiente:

Eleccion (tx_eleccion, nombre_eleccion, fecha_eleccion, status_eleccion, bloque_eleccion, contrato_eleccion)

Para indicar las dependencias funcionales utilizaremos el siguiente gráfico:

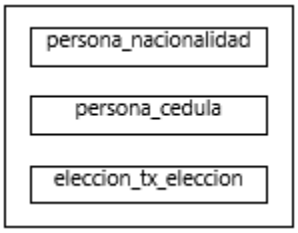


El atributo tx_eleccion es la clave primaria, y esta determina el resto de los atributos no-claves pertenecientes a la relación, por lo que esta relación se encuentra en 2FN.

Tabla Elector: La tabla elector es la siguiente:

Elector (persona_nacionalidad, persona_cedula, elección_tx_eleccion)

Para indicar las dependencias funcionales utilizaremos el siguiente gráfico:



Debido a que todos los atributos pertenecientes a esta relación son claves primarias, no existen dependencias funcionales entre ellos, entonces esta relación se encuentra en 2FN.

Debido a que todas las relaciones se encuentran en segunda forma normal, el modelo se encuentra en segunda forma normal. Es importante hacer notar que el modelo no sufrió ningún tipo de transformación en esta fase debido a la normalización.

Tercera Forma Normal (3FN)

Para que una relación se encuentre en 3FN debe estar en 2FN y adicionalmente todos los atributos no clave deben depender no transitivamente de la clave primaria.

Como se observó en los diagramas mostrados anteriormente, todos los atributos no clave dependen de forma no transitiva de la clave primaria en cada una de las relaciones. Ningún atributo no clave determina otro atributo no clave en ninguna de las relaciones, por lo cual podemos aseverar que el modelo se encuentra en tercera forma normal. En esta fase el modelo no sufre ninguna transformación por normalización.

Forma Normal De Boyce-Codd (BCNF)

Para que un modelo se encuentre en BCNF debe estar en 3FN y además se requiere que toda clave candidata sea un determinante. Un atributo es un determinante si este determina por completo a algún otro atributo. Debido a que todas las claves primarias y claves candidatas son las únicas que determinan a los atributos pertenecientes a todas las relaciones de nuestro modelo, este se encuentra en BCNF, por lo cual el mismo no sufre ningún tipo de transformación.

Finalizado el proceso de normalización del modelo, no se requirió modificar el esquema lógico, así el esquema lógico es el siguiente:

Administrador (email, pass, rol, facultad)

Candidato (nacionalidad_candidato, cedula_candidato, apellido_candidato,
nombre_candidato, billetera_candidato, votos_candidato, contrato_candidato)

Persona (nacionalidad, cedula, apellido, nombre, password, fecha_nacimiento, direccion_habitacion)

Eleccion (tx_eleccion, nombre_eleccion, fecha_eleccion, status_eleccion, bloque_eleccion, contrato_eleccion)

Elector (persona_nacionalidad, persona_cedula, elección_tx_eleccion)

El hecho de que el modelo no haya sufrido transformaciones de ningún tipo en el proceso de normalización, indica que el modelo diseñado originalmente es consistente y seguro.

5.3.3. Implementación de la aplicación

Tablas de la base de datos

Ahora, se realizará un despliegue detallado de los tipos y tamaño de datos asignados a cada uno de los campos de las tablas mencionadas recientemente:

Tabla persona

Campo	Tipo	Bytes
Nacionalidad	VARCHAR(1)	1 byte
Cedula	INT(12)	12 bytes
Apellido	VARCHAR(40)	40 bytes
Nombre	VARCHAR(40)	40 bytes
Password	VARCHAR(40)	40 bytes
fecha_de_nacimiento	DATE	7 byte
dirección_habitacion		180 bytes
Total		**320 bytes**

Tabla administrador

Campo	Tipo	Bytes
Email	VARCHAR(35)	35 bytes
Pass	VARCHAR(40)	40 bytes
Rol	TINYINT(1) UNSIGNED	1 byte

Facultad	VARCHAR(100)	100 bytes
Total		**176 bytes**

Tabla candidato

Campo	Tipo	Bytes
nacionalidad_candidato	VARCHAR(1)	1 byte
cedula_candidato	INT(12) UNSIGNED	12 bytes
apellido_candidato	VARCHAR(40)	40 bytes
nombre_candidato	VARCHAR(40)	40 bytes
billetera_candidato	VARCHAR(42)	42 bytes
votos_candidato	INT(12) UNSIGNED	12 bytes
contrato_candidato	VARCHAR(42)	42 bytes
Total		**189 bytes**

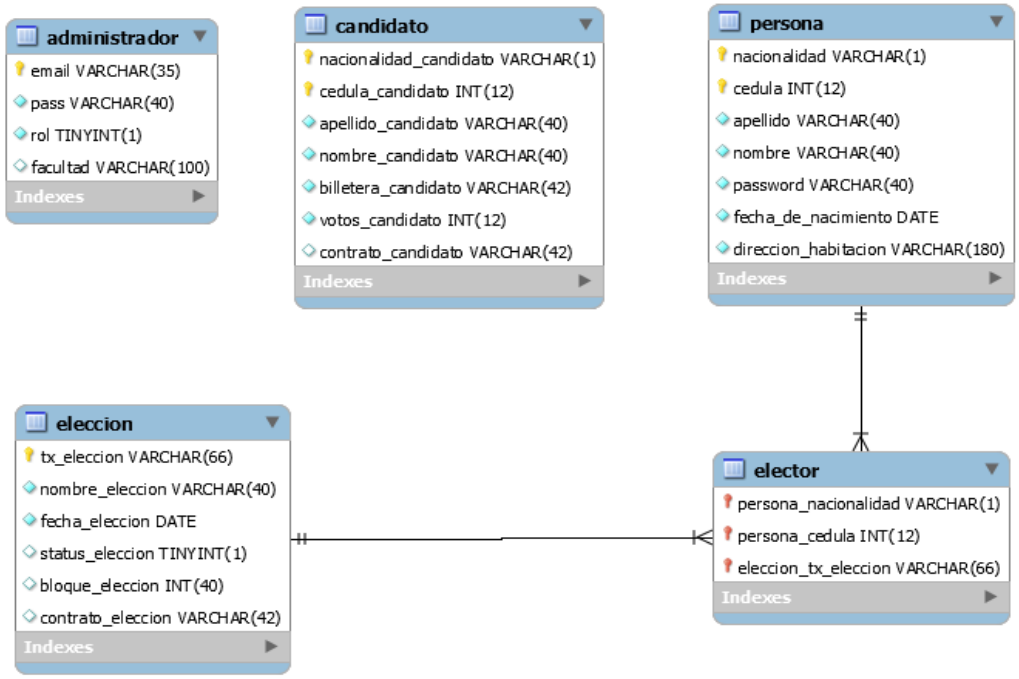
Tabla elección

Campo	Tipo	Bytes
tx_eleccion	VARCHAR(66)	66 bytes
nombre_eleccion	VARCHAR(40)	40 bytes
fecha_eleccion	DATE	7 bytes
status_eleccion	TINYINT(1)	1 byte
bloque_eleccion	INT(40)	40 bytes
contrato_eleccion	VARCHAR(42)	42 bytes
Total		**196 bytes**

Tabla elector

Campo	Tipo	Bytes
persona_nacionalidad	VARCHAR(1)	1 byte
persona_cedula	INT(12)	12 bytes
eleccion_tx_eleccion	VARCHAR(66)	66 bytes
Total		**79 bytes**

Modelos Entidad-Relación con MySQL Workbench



Código de la base de datos

```
-- MySQL Script generated by MySQL Workbench
-- Sun Mar  1 23:01:07 2020
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema interoperables
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `interoperables` ;
USE `interoperables` ;

-- -----------------------------------------------------
-- Table `interoperables`.`persona`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `interoperables`.`persona` (
  `nacionalidad` VARCHAR(1) NOT NULL,
  `cedula` INT(12) UNSIGNED NOT NULL,
  `apellido` VARCHAR(40) NOT NULL,
  `nombre` VARCHAR(40) NOT NULL,
  `password` VARCHAR(40) NOT NULL,
  `fecha_de_nacimiento` DATE NOT NULL,
  `direccion_habitacion` VARCHAR(180) NOT NULL,


  PRIMARY KEY (`nacionalidad`, `cedula`));
```

```
-- -----------------------------------------------------
-- Table `interoperables`.`usuarios`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `interoperables`.`administrador` (
  `email` VARCHAR(35) NOT NULL,
  `pass` VARCHAR(40) NOT NULL,
  `rol` TINYINT(1) UNSIGNED NOT NULL,
  `facultad` VARCHAR(100) NULL DEFAULT NULL,
  PRIMARY KEY (`email`));


-- -----------------------------------------------------
-- Table `interoperables`.`candidato`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `interoperables`.`candidato` (
  `nacionalidad_candidato` VARCHAR(1) NOT NULL,
  `cedula_candidato` INT(12) UNSIGNED NOT NULL,
  `apellido_candidato` VARCHAR(40) NOT NULL,
  `nombre_candidato` VARCHAR(40) NOT NULL,
  `billetera_candidato` VARCHAR(42) NOT NULL,
  `votos_candidato` INT(12) UNSIGNED NOT NULL,
  `contrato_candidato` VARCHAR(42) NULL DEFAULT NULL,
  PRIMARY KEY (`nacionalidad_candidato`, `cedula_candidato`));


-- -----------------------------------------------------
-- Table `interoperables`.`eleccion`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `interoperables`.`eleccion` (
  `tx_eleccion` VARCHAR(66) NOT NULL,
  `nombre_eleccion` VARCHAR(40) NOT NULL,
  `fecha_eleccion` DATE NOT NULL,
  `status_eleccion` TINYINT(1) NULL DEFAULT NULL,
  `bloque_eleccion` INT(40) UNSIGNED NULL DEFAULT NULL,
  `contrato_eleccion` VARCHAR(42) NULL DEFAULT NULL,
  PRIMARY KEY (`tx_eleccion`));


-- -----------------------------------------------------
-- Table `interoperables`.`elector`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `interoperables`.`elector` (
  `persona_nacionalidad` VARCHAR(1) NOT NULL,
  `persona_cedula` INT(12) UNSIGNED NOT NULL,
  `eleccion_tx_eleccion` VARCHAR(66) NOT NULL,
  PRIMARY KEY (`persona_nacionalidad`, `persona_cedula`, `eleccion_tx_eleccion`),
  INDEX `fk_persona_has_eleccion_eleccion1_idx` (`eleccion_tx_eleccion` ASC),
  INDEX `fk_persona_has_eleccion_persona_idx` (`persona_nacionalidad` ASC, `persona_cedula`
ASC),
  CONSTRAINT `fk_persona_has_eleccion_persona`
    FOREIGN KEY (`persona_nacionalidad` , `persona_cedula`)
    REFERENCES `interoperables`.`persona` (`nacionalidad` , `cedula`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_persona_has_eleccion_eleccion1`
    FOREIGN KEY (`eleccion_tx_eleccion`)
    REFERENCES `interoperables`.`eleccion` (`tx_eleccion`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Elementos del contrato inteligente

El contrato está compuesto de los siguientes elementos:

Variables globales:

- name (nombre del token)
- symbol (siglas del token)
- totalSupply (cantidad de tokens a crear)
- decimals (cantidad de decimales que usan los tokens)
- owner (dirección del propietario para contrato inteligente)
- i (índice para ciclos)

Estructuras

- candidatesArray (vector con la dirección de los candidatos)
- mapping(address => uint256) public balanceOf ()
- mapping(address => bool) public voted ()
- mapping(address => bool) public candidatesMapping ()

Funciones

- Constructor() public {asignara los valores al crear el contrato inteligente}
- function addCandidates(address[] memory _candidates) {agrega candidatos al contrato inteligente}
- function getCandidates() view public returns(address[] memory){devuelve el vector con los candidatos almacenados}
- function tranfer(address _to) {transfiere un token de la cuenta propietario al votante}
- function vote(address _from, address _candidate) public returns (bool success){ transfiere un token de una cuenta votante a un candidato}

Pseudocódigo del contrato inteligente

```
Clase Elección

Cadena nombreToken
Cadena siglasToken
Entero numeroToken
Dirección de wallet propietario

Vector balanceOf de direcciones
Mapeo balanceOf direcciones de wallet => entero
Mapeo voto direcciones de wallet => lógico
Mapeo candidatos direcciones de wallet =>lógico

Función constructora ()
Inicio
   Inicializar las variables globales del contrato
Fin función

Función agregarCandidatos (Vector candidatosAux de direcciones de wallet)
Inicio
   Si quien ejecuta la función es el propietario del contrato entonces
      Para i hasta cantidad de candidatos en 1
         Si el candidato no ha sido agregado entonces
            Agrega el candidato al vector Candidatos
         Fin si
      Fin para
   Fin Si
Fin Función

Función getCandidatos ()
Inicio
   Retornar cantidad de candidatos registrados en el contrato
Fin Función

Función transfer (dirección de wallet elector)
Inicio
   Si el propietario tiene tokens entonces
      Transfiere un token de la cuenta del propietario a la cuenta del elector
   Fin si
Fin Función

Función transfer (dirección de wallet elector)
Inicio
   Si el elector tiene token, no es propietario y no ha votado aun entonces
      Asigna el token del elector al candidato
   Fin si
Fin Función
```

Código del contrato Inteligente

```
1  ////////////////////////////////////////////////////////////////////////////
2  // VERSION DE solidity
3  ////////////////////////////////////////////////////////////////////////////
4  pragma solidity >=0.5.3 <0.7.0;
5
6  ////////////////////////////////////////////////////////////////////////////
7  // NOMBRE DEL CONTRATO
8  ////////////////////////////////////////////////////////////////////////////
9
10 contract Eleccion{
11
12 ////////////////////////////////////////////////////////////////////////////
13 // VARIABLES
14 ////////////////////////////////////////////////////////////////////////////
15
16     string public name;
17     string public symbol;
18     uint256 public totalSupply;
19     uint8 decimals;
20     address owner;
21     uint8 i;
22
23 ////////////////////////////////////////////////////////////////////////////
24 // ESTRUCTURAS
25 ////////////////////////////////////////////////////////////////////////////
26
27     address[] candidatesArray;
28     mapping(address => uint256) public balanceOf;
29     mapping(address => bool) public voted;
30     mapping(address => bool) public candidatesMapping;
31
32
33 ////////////////////////////////////////////////////////////////////////////
34 // METODOS
35 ////////////////////////////////////////////////////////////////////////////
36     constructor() public {
37         name = "Transfer E-Government";
38         symbol = "TEG-Vote";
39         decimals = 0;
40         totalSupply = 11000000; //cantidad de tokens a crear
41         balanceOf[msg.sender] = totalSupply;
42         owner = msg.sender;
43     }
44
45
46     event Transfer(address indexed _from, address indexed _to, uint256 _value);
47
48     function lengthCandidatesArray()public view returns(uint256){
49         return candidatesArray.length;
50     }
51
52
53     function addCandidates(address[] memory _candidates) public{
54         require(msg.sender == owner);
55         for (i = 0; i < _candidates.length; i++){
56             if(candidatesMapping[_candidates[i]] != true){
57                 candidatesArray.push( _candidates[i]);
58                 candidatesMapping[_candidates[i]] = true;
59             }
60         }
61     }
62
63     function getCandidates() view public returns(address[] memory){
64         return candidatesArray;
65     }
66
67     function transfer(address  to) private returns (bool success){
```

```
68          require(balanceOf[msg.sender]> 0);
69
70
71          balanceOf[msg.sender] -= 1;
72          balanceOf[_to] += 1;
73          //llama al evento
74          emit Transfer(msg.sender, _to,  1);
75          return true;
76      }
77
78      function vote(address _from, address _candidate) public returns (bool success){
79          require(msg.sender == owner);            //Si el propietario lo ejecuta.
80          require(msg.sender != _from );           //El propietario NO es quien vota.
81          require(msg.sender != _candidate);       //El propietario NO es el candidato.
82          require(voted[_from] != true);              //El votante NO voto anteriormente
83          require(candidatesMapping[_candidate] == true);    //El candidato existe
84          require(transfer(_from));
85
86          balanceOf[_from] -= 1;
87          balanceOf[_candidate] += 1;
88          emit Transfer(_from, _candidate,  1);
89          voted[_from] = true;
90
91          return true;
92      }

    }
```

Web3.js para conectar frontend, backend y blockchain

Debemos asegurarnos que las versiones tanto de node.js, npm y solidity sean las necesarias para utilizar web3.

```
pragma solidity ^0.5.3;
```

Nos indica que estamos trabajando con el lenguaje de programación solidity.

```
pragma solidity ^0.5.3;
```

La versión a utilizada funciona con 0.5.3 o superior.

Luego instanciamos el objeto con:

```
var Web3 = require('web3')
```

Además de web3 la documentación indica como necesario el uso del módulo ethereumjs-tx:

```
const Tx = require('ethereumjs-tx').Transaction
```

Ethereumjs-tx es un módulo simple para crear, manipular y firmar transacciones en Ethereum.

Cómo conectar con un nodo

```
var nodoUrl = 'https://dirección web nodo'
```

Es importante además de la dirección especificar el protocolo utilizado, ya sea "http", "https" u otro.

La url de conexión del nodo será una dirección web o una IP (Internet Protocol). Suele ser una dirección web cuando el nodo es proporcionado por una empresa como infura, pero una IP cuando es una blockchain local como ganache. En la elaboración de este proyecto la blockchain local se usa como entorno de prueba por proporcionar un mayor control sobre la misma, como numero de cuentas, direcciones, precio del gas, etc. mientras que en la Dapp ya finalizada se utiliza la blockchain de Ethereum.

```
var web3 = new Web3(nodoUrl)
```

De esa manera instanciamos el objeto "web3" que se conectara con el nodo especificado.

Como firmar transacciones

A la hora de interactuar con la blockchain nos podemos encontrar con dos escenarios:

- **Consultas a la blockchain**: que por no cambiar el estado de la misma no es necesaria la firma de transacciones.
- **Escritura en la blockchain**: que al cambiar el estado de la cadena de bloques y los estados de los contratos es necesario la firma de estas transacciones.

Para el desarrollo y prueba de ĐApps en la blockchain se utiliza una cadena de bloques de prueba (*testnet*), la *tesnet* de Ethereum tiene el nombre de *ropsten*. Las *testnet* existen debido a que las transacciones de la cadena de bloques principal (*mainnet*) tienen un costo real agregado.

Además, está la alternativa de utilizar una blockchain de pruebas local como *ganache*.

Para realizar pruebas en la *ropsten* en cierta forma es necesario disponer de *Ether*; pero no es necesario pagar para ello, basta con usar un Grifo de Ether en la ropsten (*Ropsten Ethereum Faucet*), con los cuales suministramos Ether de prueba dada una dirección de un wallet en la *ropsten*.

Ya al disponer de una *wallet* con Ether procederemos a guardarla para usarla en la Đapp. Podríamos utilizar una función de web3 y cifrarlo, por razones de practicidad guardaremos

la dirección del propietario del contrato inteligente en una constante *ownerAddress* y la clave en otra constante de nombre *ownerAddressKey*.

```
const ownerAddress = '0x3492406B0B18c4B9B579aA0ED046A9c34D5B0241'

  const ownerAddressKey = new
Buffer.from('f37cdfd90707671b4acd12643c6e591dfdbf1a4070856b6c51c0cfae7b9c7605','hex')
```

El envío de la transacción firmada la haremos con web3 mientras que la firma es realizada con Ethereum-tx.

Como crear cuentas en la Blockchain

```
var cuenta = web3.eth.accounts.create ([entropía])
```

web3.eth.accounts.create () Genera un objeto de cuenta con clave privada y clave pública.

Parámetros

Entropía - Cadena (opcional): una cadena aleatoria para aumentar la entropía. Si se proporciona, debe tener al menos 32 caracteres. Si no se proporciona ninguno, se generará una cadena aleatoria usando *randomhex*.

Devoluciones

Objeto: el objeto de cuenta con la siguiente estructura:

- *address* - *string*: la dirección de la cuenta.
- *privateKey* - *string*: la clave privada de las cuentas. Esto nunca debe compartirse o almacenarse sin cifrar en el almacenamiento local. También asegúrese de anular la memoria después del uso.
- *signTransaction* (tx [, callback]) - Función: La función para firmar transacciones.
- signo (datos) - Función: La función para firmar transacciones.

Como desplegar contratos en la cadena de bloques de Ethereum

Cuando instanciamos un contrato inteligente en web3 generamos un objeto en Javascript que tiene una referencia a todas las funciones del contrato inteligente en un documento JSON de nombre *Application Binary Interface* (*ABI*).

Para poder compilarlo y usarlo se puede hacer de dos formas:

- Compilarlo con IDE de Ethereum *Remix* y copiar el *ABI* a un documento, guardar el contrato en un archivo de tipo *solidity* y el *ABI* a un archivo de tipo JSON. Este método es más sencillo e intuitivo, ideal para familiarizarse con los conceptos, pero es un proceso manual.
- Instalar el módulo de node "*solc*" cuya versión debe corresponder con la del contrato inteligente, en caso de este proyecto se utilizó *solc* 0.5.3, de tal forma es necesario guardar el contrato en un archivo de tipo *Solidity* para, a partir del mismo, generar el *bytecode* y el *ABI* de forma automatizada.

El uso de *solc* consta de los siguientes pasos:

6. Cargar el modulo

```
const solc = require('solc')
```

7. Verificar que la versión del compilador corresponde a la utilizada en el contrato

```
solc.version()
```

8. Cargamos el contenido del contrato

```
const content = fs.readFileSync('Documentos/myToken.sol').toString()
```

'Documentos' corresponde al subdirectorio, 'myToken.sol' corresponde al archivo con el contrato inteligente a cargar

9. Creamos un objeto

```
const objectSolc = {
  language: 'Solidity',
  //entradas
  sources: {
    'myToken':{
      content: content
    }
  },
  //salidas
  settings: {
    outputSelection:{
      '*':{
        '*':['*']
      }
```

```
    }
  }
}
```

10. Compilamos y transformamos a JSON

```
const output = JSON.parse(solc.compile(JSON.stringify(objectSolc)))
```

A partir de ello podemos obtener el bytecode y el ABI de la forma siguiente:

Bytecode:

```
const bytecodeContract = output.contracts.myToken.Eleccion.evm.bytecode.object
```

ABI:

```
const abiArray = output.contracts.myToken.Eleccion.abi
```

Para el despliegue del contrato usaremos dos métodos de web3:

1. getTransactionCount: Se obtendrá el número de transacciones enviadas desde esta dirección, con ello se evita el doble gasto. La dirección usada será la del propietario del contrato.

```
web3.eth.getTransactionCount (dirección [, defaultBlock] [, devolución de llamada])
```

Parámetros

Cadena: la dirección de la cual obtener el número de transacciones.

Número | Cadena: (opcional) Si pasa este parámetro, no utilizará el bloque predeterminado establecido con web3.eth.defaultBlock.

Función: (opcional) Devolución de llamada opcional, devuelve un objeto de error como primer parámetro y el resultado como segundo.

Devoluciones

Promesa que devuelve un número: el número de transacciones enviadas desde la dirección indicada.

2. sendSignedTransaction: Envía una transacción ya firmada.

```
web3.eth.sendSignedTransaction (ignedTransactionData [, devolución de llamada])
```

Parámetros

Cadena: datos de transacción firmados en formato HEX.

Función: (opcional) devuelve un objeto de error como primer parámetro y el resultado como segundo.

Devoluciones

PromiEvent: Una promesa emisor combinada con eventos.. Se resolverá cuando el recibo de la transacción esté disponible.

Las partes necesarias para la construcción de la transacción son:

5. Empaquetar los datos en un objeto con los datos de la transacción.
    5.1. nonce: el número de la transaccion.
    5.2. gasPrice: El precio de cada unidad expresado en wei.
    5.3. gasLimit: La cantidad máxima de gas disponible a utilizar en la transaccion, debe ser mayor o igual a la capacidad computacional que requiera el contrato.
    5.4. to: La dirección de destino, que será nula al ser el despliegue de un contrato.
    5.5. data: El bytecode del contrato, que previamente compilamos.
6. Instanciar el objeto para la firma pasando como parámetro el objeto con los datos de la transacción.
7. Firmar con la clave de la billetera emisora.
    7.1. El parámetro de entrada es el objeto para la firma
    7.2. Especificamos la cadena utilizada (*main*, *ropsten*, entre otras).
8. Serializar, es decir, transformamos el objeto JavaScript a una cadena hexadecimal.
9. Difusión de la transacción con el método sendSignedTransaction.

```
 await web3.eth.getTransactionCount(ownerAddress, (err, txCount) => {
      //1 EMPAQUETADO
      const txOject = {
        nonce: web3.utils.toHex(txCount),
        gasPrice: web3.utils.toHex(web3.utils.toWei('2','gwei')),
        gasLimit: web3.utils.toHex(1500000),
        to: null,
        data: '0x'+ bytecodeContract
      }
      //3.INSTANCIACION
      const tx = new Tx(txOject,{'chain':'ropsten'})

      //3.FIRMA
      tx.sign(ownerAddressKey)
```

```
        //4.SERIALIZACION
        const serializedTx = tx.serialize().toString('hex')

        //5.DIFUSION

        web3.eth.sendSignedTransaction('0x'+serializedTx, (err, txHash) => {
          log('Usuario '+req.session.user)
          if (err) {
            console.error(err);
            forbid(res);
          }else{
            console.log('txHash contrato:'+ txHash);
            txHashGenesis = txHash
            res.json({data: txHashGenesis });
            //Ahora verifica la transaccion en etherscan
          }
        })
      })
```

Al utilizar el contrato inteligente bastara con la dirección del contrato y su *ABI*, de tal forma sabremos el contrato utilizado y las propiedades de sus funciones.

6. RESULTADOS

Como ya indicáramos en la fase de Diseño el desarrollo del sistema de elecciones consta de dos aplicaciones web descentralizadas (Dapp):

- ĐApp 1: denominada “Transfer E-Government” con las siglas T.E.G. es la aplicación que permite al órgano rector o autoridad rectora responsable de organizar y ejecutar el proceso electoral realizar la administración del sistema. Entre las funcionalidades disponibles en el desarrollo actual realizado para esta investigación tenemos:

    1. Consultar registro electoral.
    2. Consultar candidatos registrados.
    3. Consultar usuarios permisados.
    4. Registrar candidato.
    5. Iniciar evento electoral.
    6. Cierre de evento Electoral.

La figura 12 muestra la pantalla principal de esta aplicación, donde se observa el menú de opciones del módulo de administración.

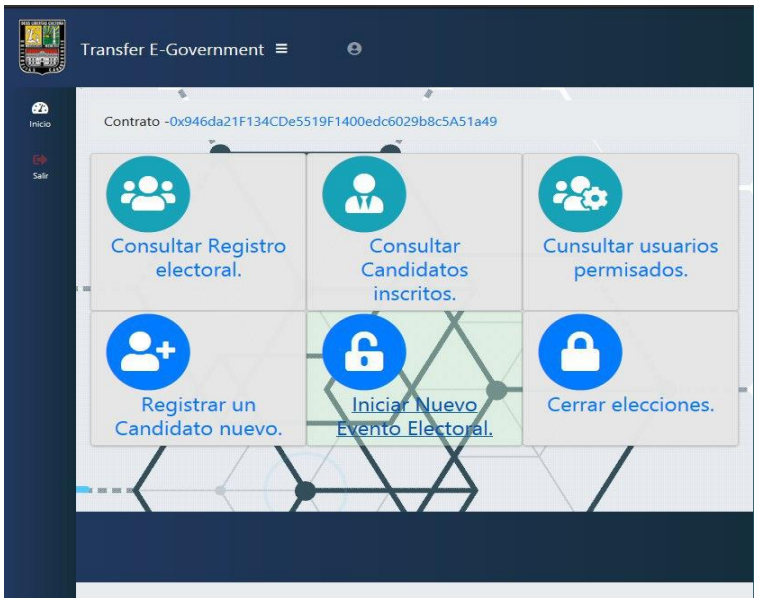


Figura 12 Vista de opciones para usuario administrativo con T.E.G

- ĐApp 2: denominada "Transfer E-Government **Decisión Block**" cuyas siglas son T.E.G.D.B. es la aplicación diseñada para los electores, y será la aplicación que podrá instalarse en las máquinas de votación o incluso en los teléfonos inteligentes de los votantes para hacer ejercicio de su derecho a votar:

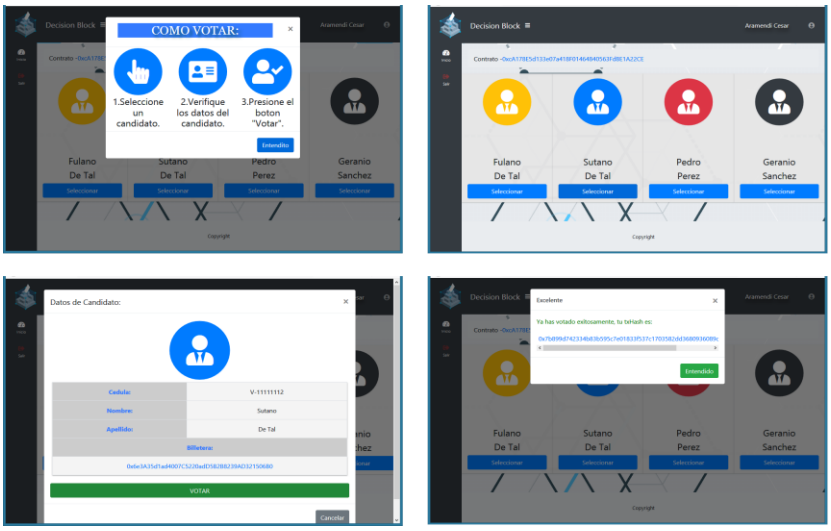


Figura 13 Vista de la realización del acto de votar con T.E.G.D.B

La figura 13 muestra captura de pantalla de la ĐApp 2 (***Decision Block***) donde el elector podrá ver los candidatos (el boletín electoral), podrá escoger el candidato de su preferencia, y podrá emitir su voto.

Por su parte, las figura 14 y 15 muestran captura de pantalla de la visualización de resultados con la ĐApp Decisión Block, una vez ha concluido el evento electoral. En este modelo experimental de la aplicación los resultados se presentan en forma de un gráfico de tipo torta (*pie*), pero el módulo de visualización puede y debe enriquecerse con muchas modalidades diferentes y desagregadas de visualización de los resultados.

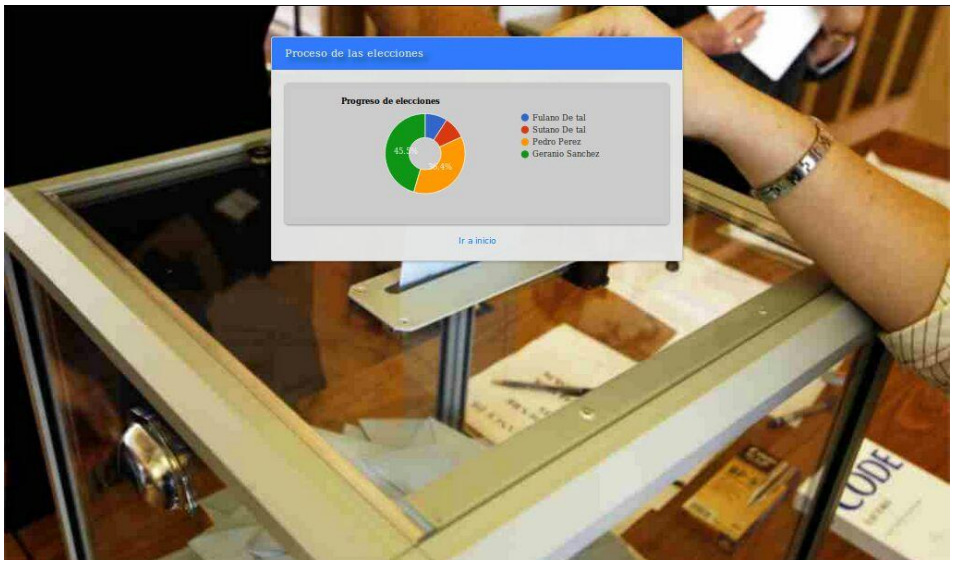


Figura 14 Vista –A de la totalización de las elecciones

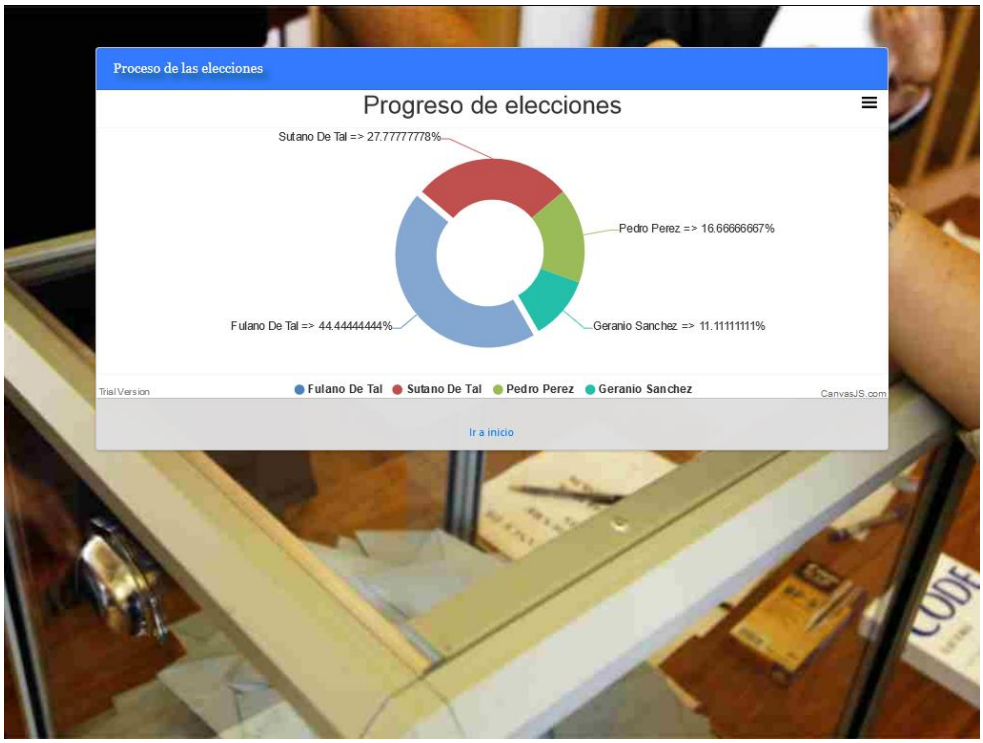


Figura 15 Vista –B de la totalización de las elecciones

En la Figuras 16 y 17 se visualizan demostraciones de que las transacciones y resultados han quedado almacenados en la cadena de bloques de Ethereum.

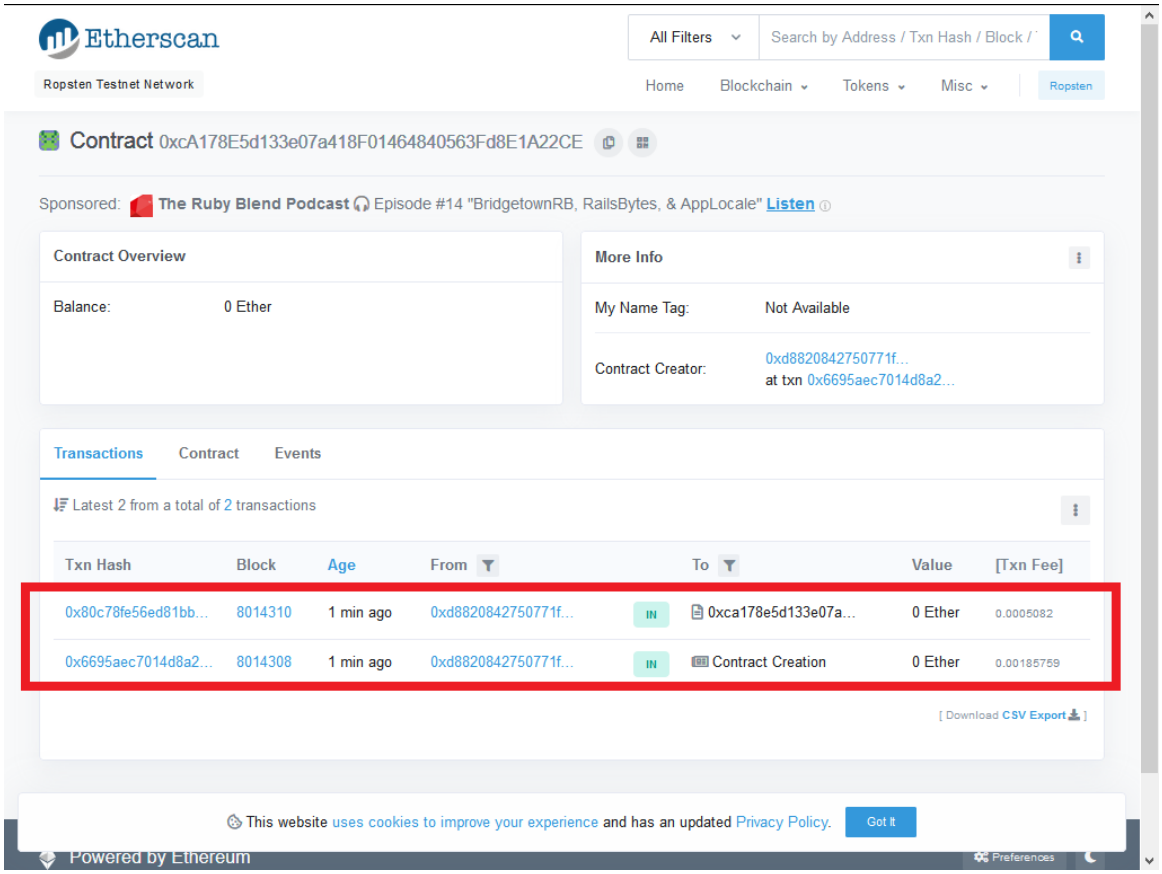


Figura 16 Etherscan (2020). Transacciones del despliegue del contrato y carga de candidatos

La figura 16 muestra el estado del contrato inteligente con el explorador de la cadena de bloques *Etherscan*, se pueden visualizar dos transacciones que corresponden a la creación del contrato de la elección y a la carga de los candidatos disponibles en dicha elección en el boletín electoral.

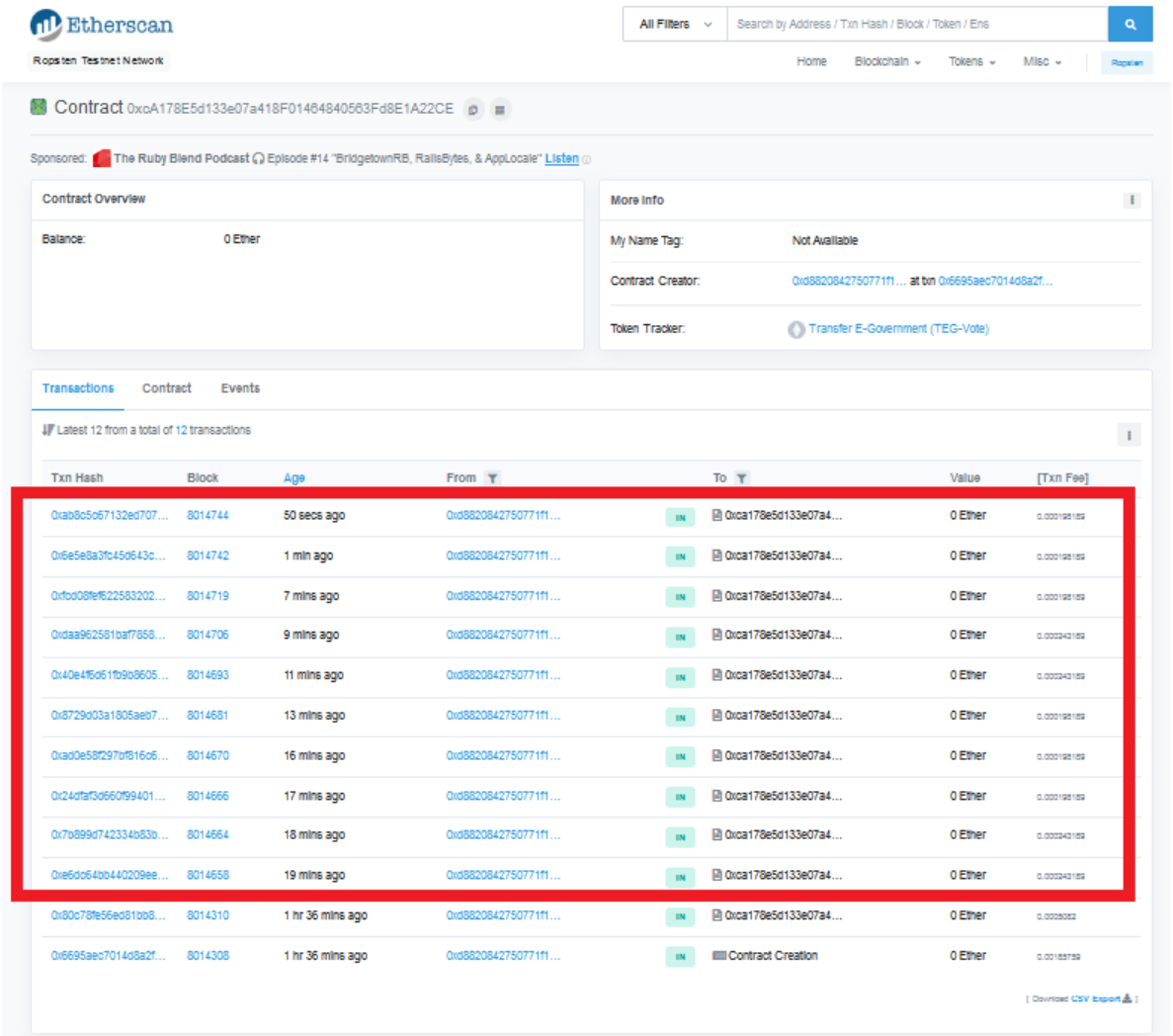


Figura 17 Etherscan (2020). Transacciones durante el desarrollo de elecciones

La figura 17 muestra diversas transacciones en el contrato inteligente luego del inicio de las elecciones, las transacciones corresponden a los votos realizados por los electores que son visualizados con Etherscan.

Figura 18 Visualización de candidatos al inicio de las elecciones con ĐApp Transfer E-Government

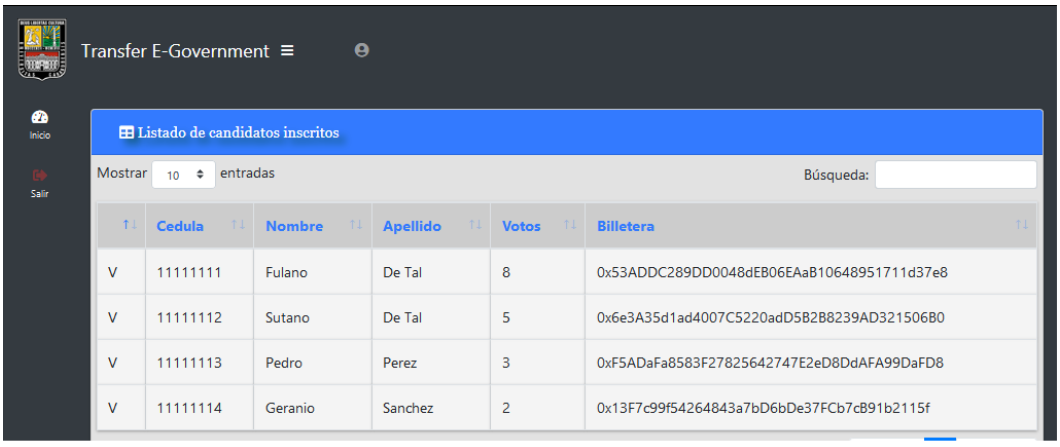


Figura 19 Visualización de candidatos tras finalizadas las elecciones con ĐApp Transfer E-Government

En las figuras 18 y 19 usando la ĐApp Transfer E-Government se observan la lista con los datos personales de los candidatos, la cantidad de votos a cada candidato y dirección de billeteras que almacenan en Ethereum los tokens que equivalen a dichos votos a los candidatos. En la figura 18 la cantidad de votos es cero debido a que la captura de pantalla fue tomada al inicio del evento electoral, mientras que en la figura 19 esto cambia ya que es imagen del resultado de las elecciones.

En las figuras 20, 21, 22 y 23 se muestran imágenes de Etherscan donde se ven los estatus iniciales de las billeteras de los candidatos en las elecciones, no se observan transacciones en las billeteras de los candidatos ya que son imágenes del inicio de las elecciones, por lo que las billeteras están vacías, esto se indica con el mensaje *There are no matching entries* (No hay entradas coincidentes).

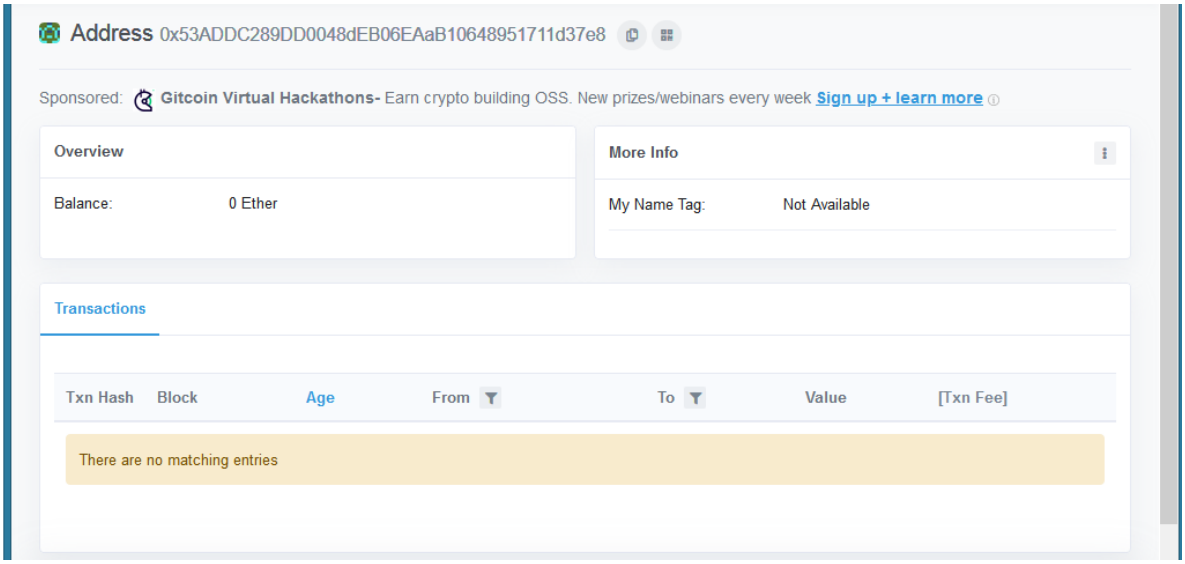


Figura 20 Etherscan (2020). Billetera del primer candidato al iniciar las elecciones

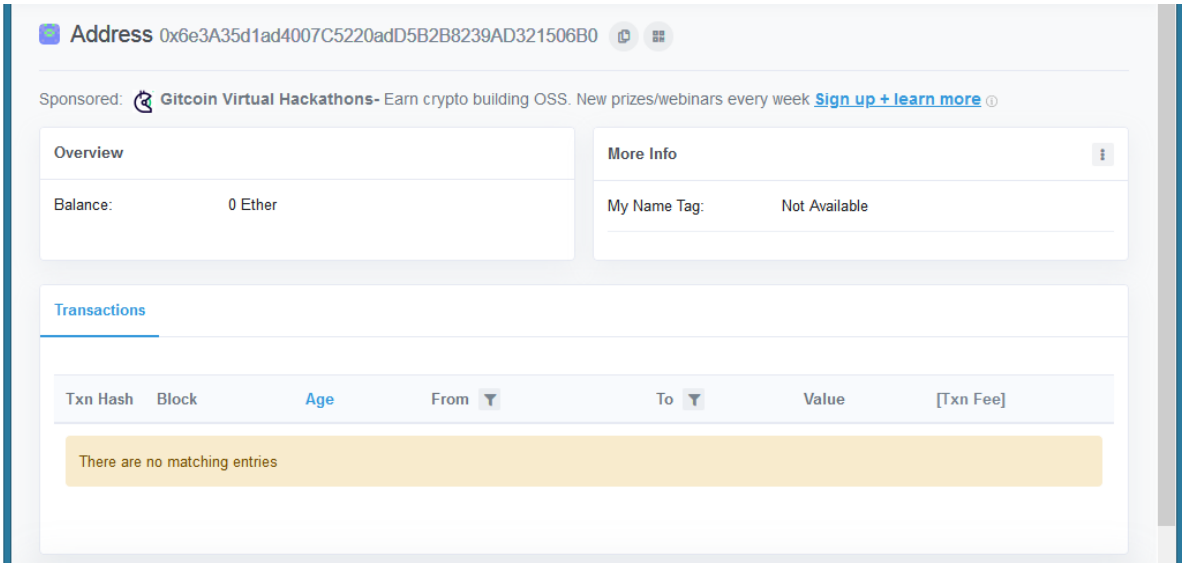


Figura 21 Etherscan (2020). Billetera del segundo candidato al iniciar las elecciones

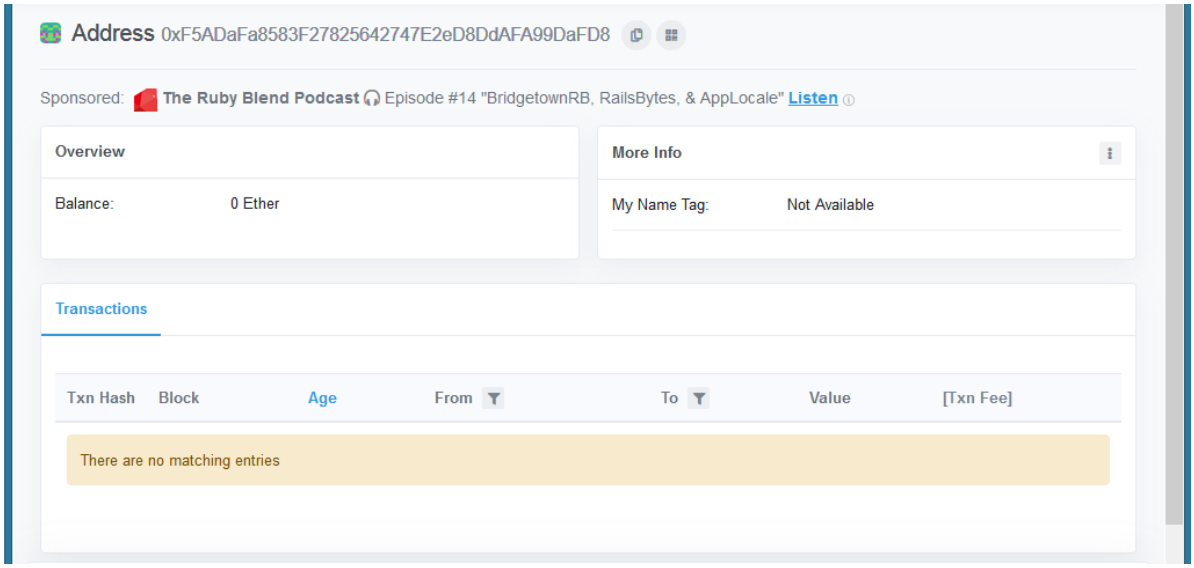


Figura 22 Etherscan (2020). Billetera del tercer candidato al iniciar las elecciones

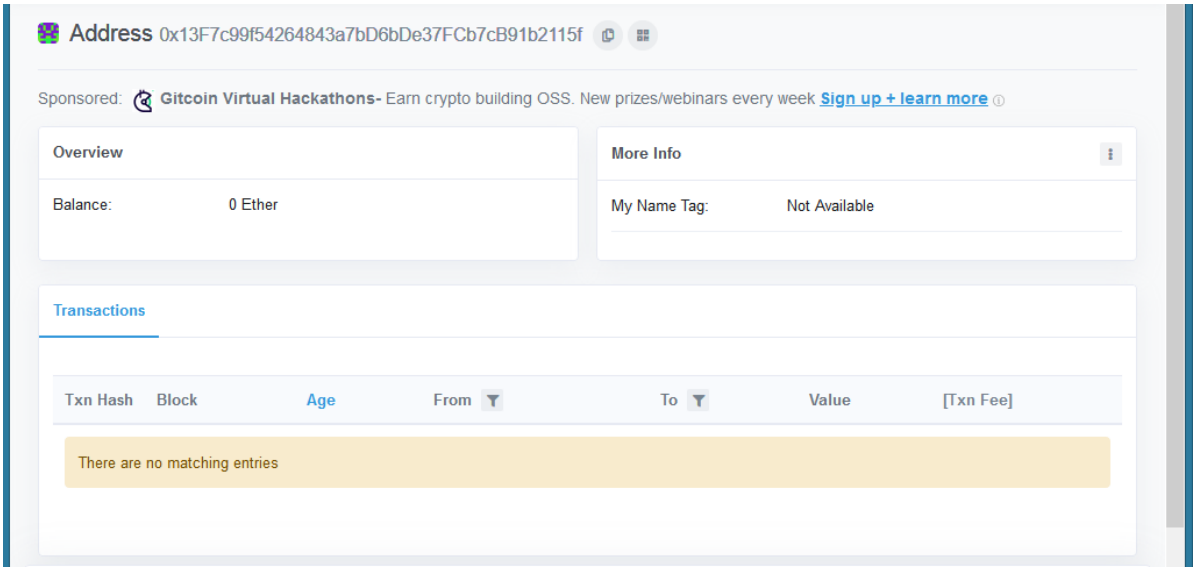


Figura 23 Etherscan (2020). Billetera del cuarto candidato al iniciar las elecciones

En las figuras 24, 25, 26 y 27 se muestran imágenes de Etherscan en las cuales se observan los estatus finales de las billeteras de los candidatos en las elecciones. Desde el transcurso de las elecciones hasta su finalización, las billeteras de los candidatos registran las transacciones equivalentes a los votos de los electores.

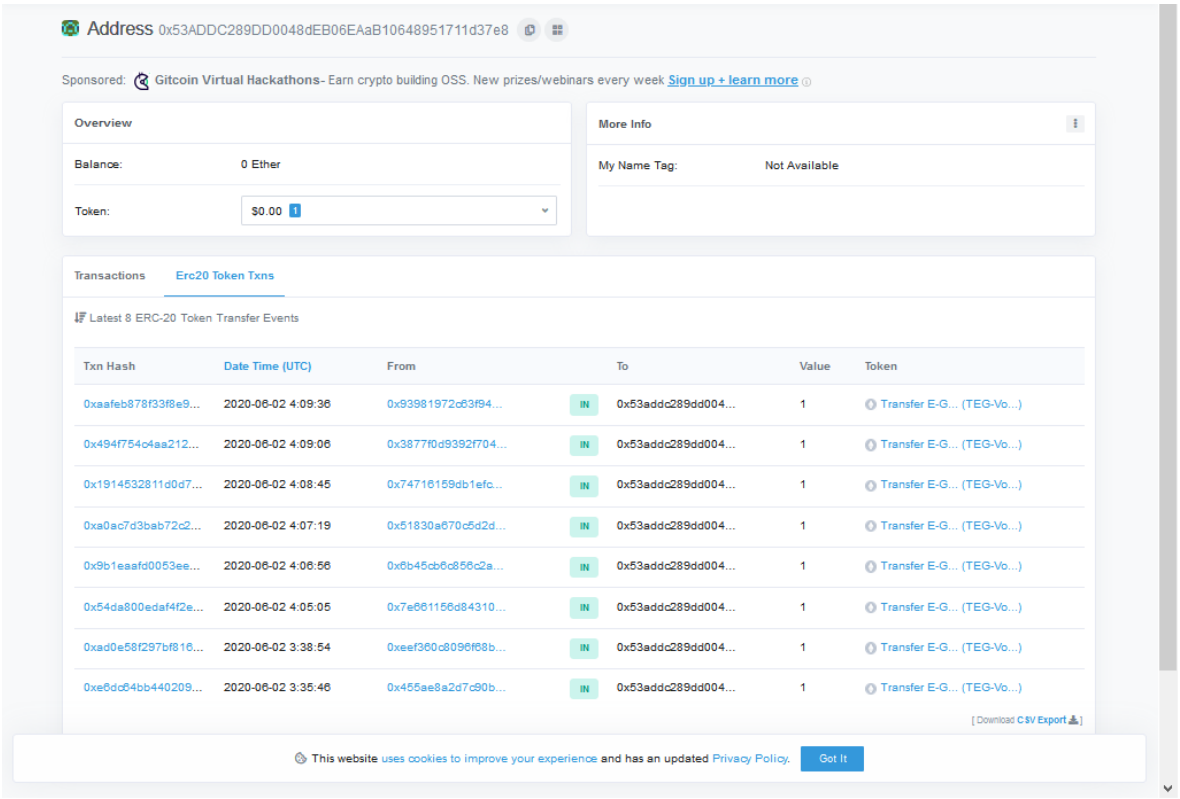


Figura 24 Etherscan (2020). Billetera de primer candidato al finalizar las elecciones

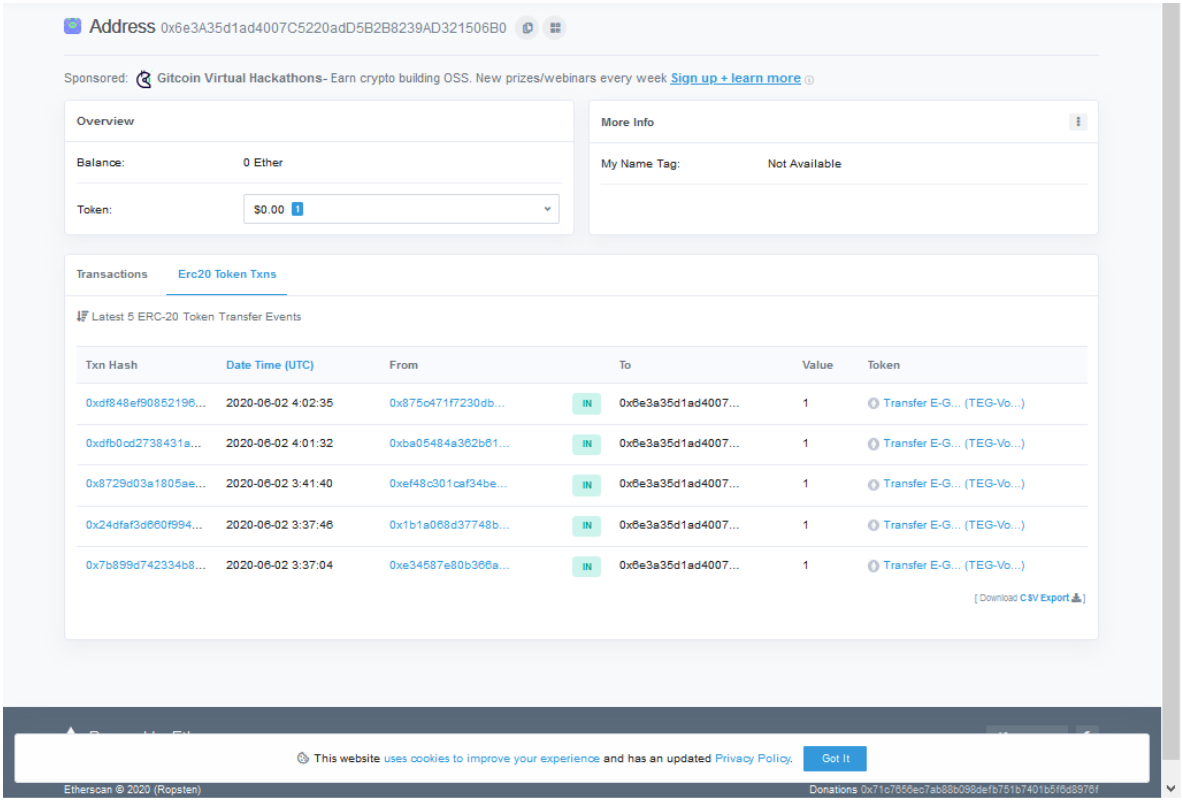


Figura 25 Etherscan (2020). Billetera del segundo candidato al finalizar las elecciones

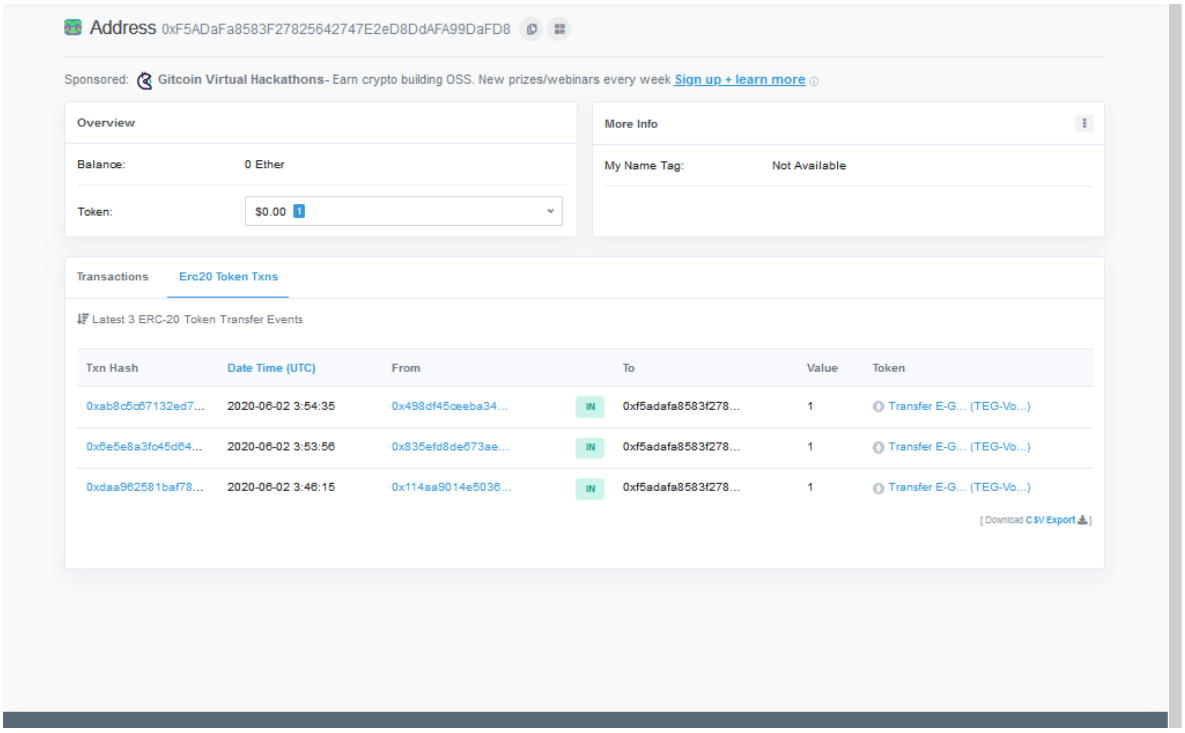


Figura 26 Etherscan (2020). Billetera del tercer candidato al finalizar las elecciones

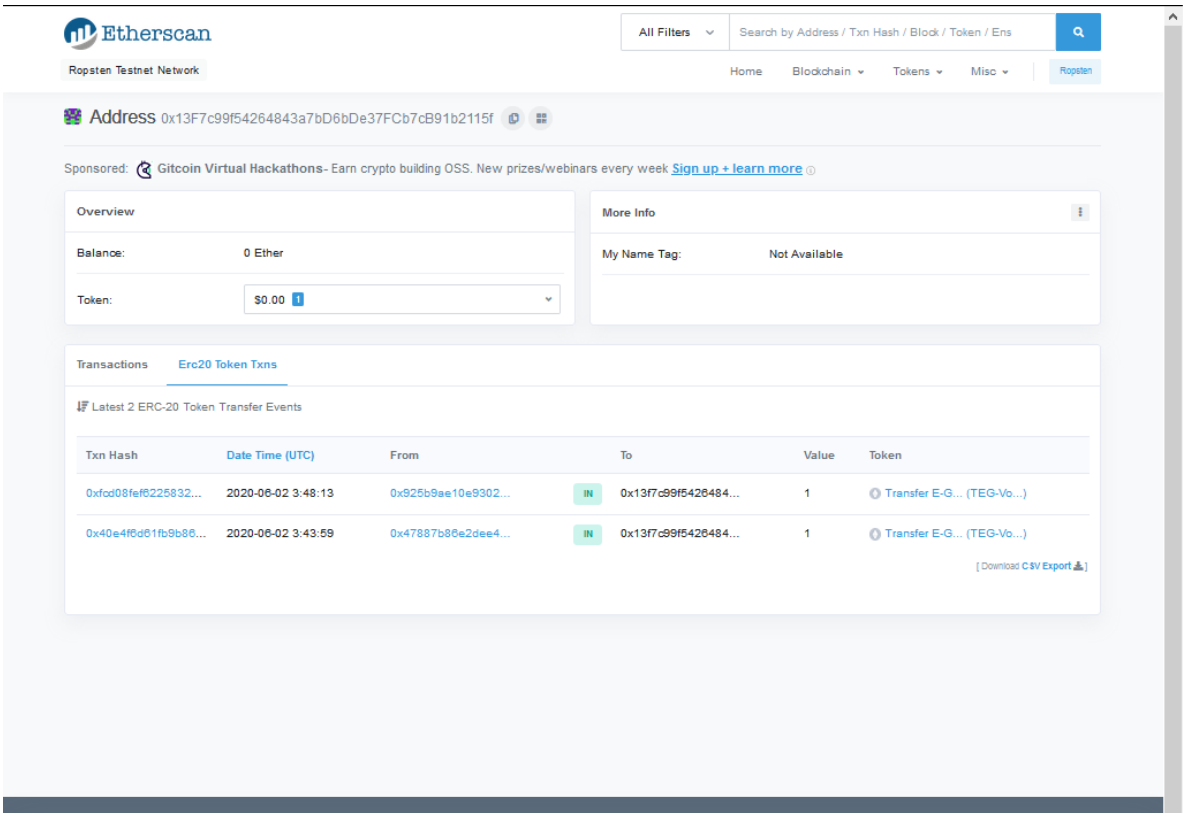


Figura 27 Etherscan (2020). Billetera del cuarto candidato al finalizar las elecciones

En la figura 28 se evidencia la relación de la cantidad de votos de los candidatos en la Dapp "T.E.G." respecto a la cantidad de transacciones del token "TEG-vote" en las billeteras de los candidatos mostradas con Etherscan. Por ejemplo, para el primer candidato se resaltan en naranja 8 votos en "T.E.G." y 8 transacciones del token "TEG-vote" listadas en Etherscan. Para los tres candidatos restantes se resaltan 5 votos para el segundo candidato, 3 el tercer candidato y 2 el cuarto candidato en la imagen.

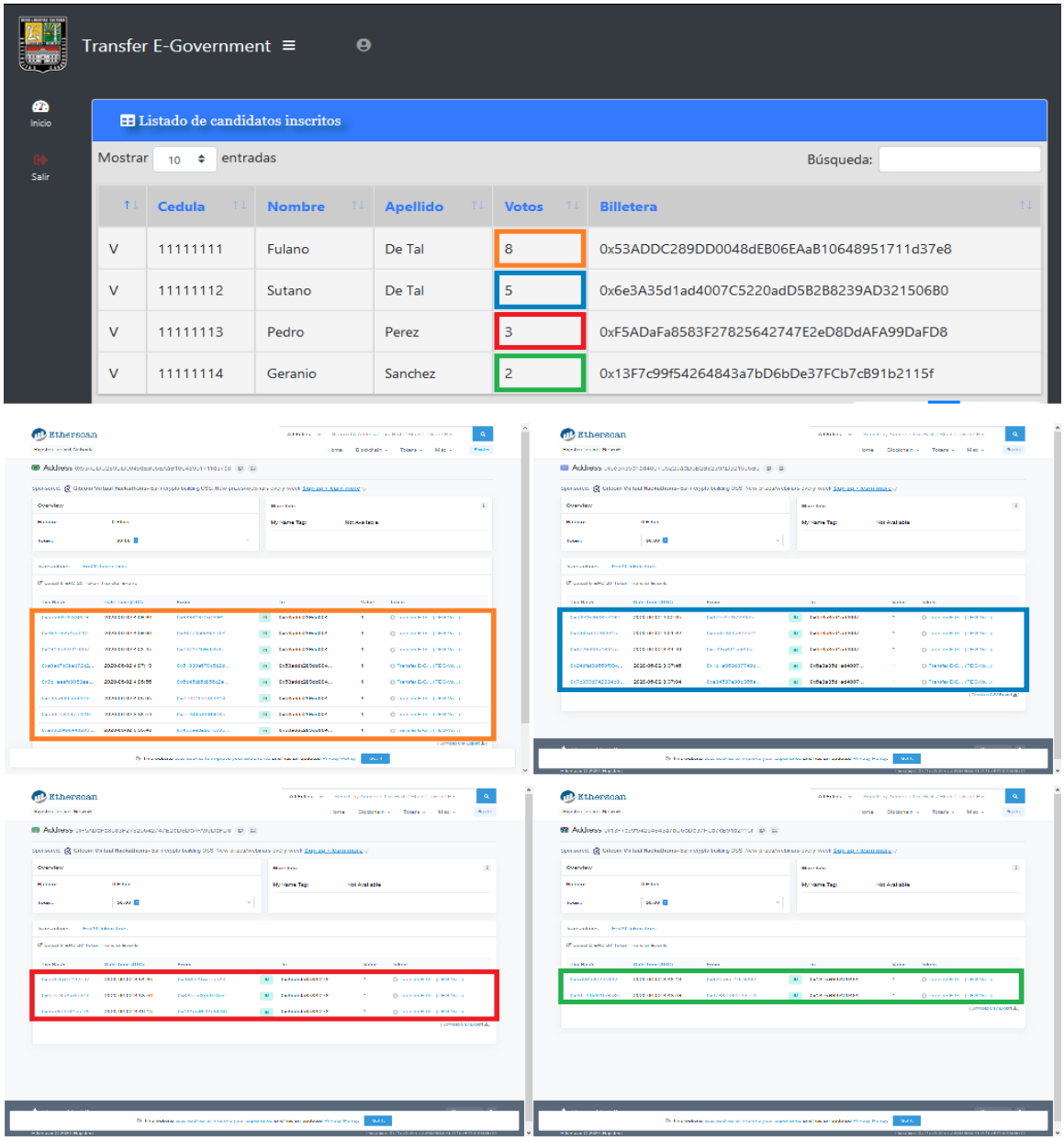


Figura 28 Comparativa de los candidatos al finalizar las elecciones. La Dapp1 "Transfer E-Government" y el visualizador de cadena de bloques "Etherscan"

CONCLUSIONES

A partir de la construcción de nuestro experimento, consistente en el desarrollo de un software que implementa el contrato inteligente que está por detrás de un proceso de votación, hemos podido demostrar la factibilidad de implementar dicho contrato sin la intermediación de un tercero confiable. El uso de la tecnología basada en cadena de bloques (Blockchain) permite la implementación de transacciones entre pares de forma transparente, segura, y sin la intermediación de entidades centralizadoras de la transacción.

Vale hacer notar que el acto de votar queda registrado directamente en la cadena de bloques sin necesidad de pasar por un servidor centralizador y validador de la transacción. La autoridad rectora del proceso, para ejercer su rol de emitir públicamente los resultados, los tomará de la propia cadena de bloques, como podría hacerlo cualquier ciudadano u otra entidad con acceso a la misma. Dependerá de las políticas acordadas para decidir los momentos y modalidades de acceso a dicha cadena de bloques, pero tecnológicamente el registro íntegro e inalterable de las transacciones realizadas durante el evento quedó allí sin pasar nunca, previamente, por una entidad centralizadora. En tal sentido consideramos que nuestra hipótesis fue validada por el experimento desarrollado.

Cada elector se autentico en el sistema, para escoger un candidato de la lista dada y votar por única vez por él candidato de su preferencia, siendo así registrado por el sistema como votante, dicho proceso se muestra en la figura 13. Dado que la billetera del elector se crea al momento de votar y no se almacena una relación entre los electores y los candidatos el voto se mantiene como secreto. Las diferencias entre el número de votos de los candidatos se expresan en las gráficas de totalización en las figuras 14 y 15, el candidato ganador es aquel que al final del evento electoral acumule mayor cantidad de votos a su favor. los resultados quedan almacenados en un registro inmutable e inalterable antes, durante o después del evento electoral, pudiendo ser auditado en cualquier momento, como muestran desde las figuras16 hasta 22. Con lo expuesto se satisfacen las cinco condiciones establecidas para el modelo de votación escogido descritas en la sección 5.2.3.

Por otra parte, los tiempos de consumación de la transacción, en un ambiente de interconexión convencional, como el de nuestra internet pública, han demostrado ser suficientemente eficaces para lograr el objetivo de la aplicación. Es decir, no se introduce una ralentización especialmente negativa del proceso. Vale agregar que, dependiendo del nivel de criticidad de la aplicación en cuestión, se podrán tomar las medidas convenientes para mejorar la velocidad de interconexión de la red sobre la cual se ejecutarán las transacciones.

El funcionamiento de las tecnologías implicadas en este experimento son prueba suficiente del nivel de transparencia proporcionado por la cadena de bloques. Su funcionamiento con un modelo automatizado en un entorno distribuido garantiza la eliminación de la centralización de las transacciones que pudiera ser objeto de manipulación por elementos maliciosos afectando la integridad de la información, y garantiza también el acceso a la

información por parte de los propios pares (electores en este caso), siendo ese acceso un aspecto clave para garantizar dicha transparencia.

Y es que la transparencia y la seguridad van de la mano, a diferencia de la obsoleta idea de que el método de funcionamiento secreto era más seguro. En el caso de la cadena de bloques, la transparencia (el cómo se hace) es la mayor garantía de seguridad en su funcionamiento, no el secreto.

Las partes involucradas en el intercambio pueden auditar por si mismas la información debido al funcionamiento transparente e independiente de una cadena de bloques de carácter público, haciendo innecesaria la intermediación de un tercero de confianza, y disminuyendo en consecuencia costos y retrasos derivados de las operaciones de validación centralizadas en dicho tercero.

Por lo que la teoría e implementación proporcionadas en este documento crea oportunidades para profundizar y aportar conocimiento de herramientas con medios distribuidos.

Entre las limitaciones identificadas, está el hecho de requerir conexión permanente y universal para los participantes del contrato inteligente. Esto supone que no sea posible establecer centros de votación bajo el esquema Blockchain en lugares que no posean conectividad. Asimismo, los nodos de la red deben ser distribuidos con relativa uniformidad numérica y de poder de cómputo entre todas las partes involucradas en el proceso de validación y construcción de bloques de las transacciones asociadas al contrato inteligente que se desea ejecutar.

Otras limitaciones propias de nuestro desarrollo tienen que ver con no haber implementado el módulo de autenticación sobre la cadena de bloques. Para ello debíamos contar con un sistema de autenticación biométrico, y distribuir previamente la base de datos de autenticación en forma cifrada entre los nodos de la cadena de bloque. Este desarrollo queda propuesto como una ampliación futura de esta aplicación a fin de garantizar que el proceso de autenticación se realice igualmente entre pares y no de forma centralizada.

7. REFERENCIAS BIBLIOGRAFICAS

[1] D. Tapscott, "Como la cadena de bloques está cambiando los negocios", 2017. [Online]. Disponible: https://www.ted.com/talks/don_tapscott_how_the_blockchain_is_changing_money_and_business?language=es

[2] N. Szabo, “Smart Contracts Glossary”, 1995. [Online]. Disponible: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_glossary.html

[3] N. Szabo, "Smart Contracts: Building Blocks for Digital Markets," 1996. [Online]. Disponible: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

[4] S. Nakamoto, "Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer", [Online]. Disponible: https://bitcoin.org/files/bitcoin-paper/bitcoin_es.pdf

[5] A. Black, “Hashcash – a denial of service counter-measure”, 2002. [Online]. Disponible: http://www.hashcash.org/papers/hashcash.pdf

[6] V. Buterin, "A Next Generation Smart Contract & Decentralized Application Platform," [Online]. Disponible: http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[7] Raiden Network, "Raiden Network - Basic Mechanics," [Online]. Disponible: https://raiden.network/.

[8] Ethereum, " Account Types, Gas, and Transactions," 2017. [Online]. Disponible: https://github.com/ethereum/homestead-guide/blob/master/source/contracts-and-transactions/account-types-gas-and-transactions.rst#example-transaction-cost

[9] Miethereum, "El ‘Gas’ en Ethereum," 2018. [Online]. Disponible: https://miethereum.com/ether/gas/

[10] K. R. Popper, “Conjeturas y refutaciones, El desarrollo del conocimiento científico”, 1ra. ed. Barcelona: Ediciones Paidos Oberica, S. A, 1983.