



Universidad
Nacional
de Loja



Carrera de Ingeniería en
Sistemas / Computación

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

CARRERA DE INGENIERÍA EN SISTEMAS

Implementación de un sistema multiplataforma de voto electrónico basado en Blockchain

“TESIS PREVIA A LA
OBTENCIÓN DEL TÍTULO DE
INGENIERO EN SISTEMAS”

Autores:

- Jhon Alexander Carrión Piedra
- Luis Xavier Paredes Cuenca

Director:

- Ing. Cristian Ramiro Narváez Guillen, Mg.Sc.

LOJA – ECUADOR

2022

Certificado del director

Autoría

Nosotros, **Jhon Alexander Carrión Piedra** y **Luis Xavier Paredes Cuenca**, declaramos ser autores del presente Trabajo de Titulación y eximimos expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente aceptamos y autorizamos a la Universidad Nacional de Loja, la publicación de nuestro Trabajo de Titulación en el Repositorio Institucional, Biblioteca Virtual.

Loja, 10 de marzo de 2022

**CARTA DE AUTORIZACIÓN DE TESIS POR PARTE
DEL AUTOR, PARA LA CONSULTA,
REPRODUCCIÓN PARCIAL O TOTAL Y
PUBLICACIÓN ELECTRÓNICA DEL TEXTO
COMPLETO.**

Nosotros, **Jhon Alexander Carrión Piedra** y **Luis Xavier Paredes Cuenca**, declaramos ser los autores del Trabajo de Titulación que versa: **“IMPLEMENTACIÓN DE UN SISTEMA MULTIPLATAFORMA DE VOTO ELECTRÓNICO BASADO EN BLOCKCHAIN”**, como requisito para el grado de: **Ingeniero en Sistemas**; autorizamos al Sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el (RDI), en las redes de información del país y del exterior, con los cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los diez días del mes de marzo de dos mil veinte y dos

Dedicatoria

Agradecimiento

Índice de contenidos

Certificado del director.....	i
Autoría.....	ii
CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DEL AUTOR, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.	iii
Dedicatoria	iv
Agradecimiento	v
Índice de contenidos.....	vi
Índice de figuras	xi
Índice de tablas	xii
1 TITULO	13
2 RESUMEN	14
3 INTRODUCCIÓN.....	16
4 REVISIÓN DE LITERATURA	19
4.1 Voto.....	19
4.2 Voto electrónico o e-voting	19
4.2.1 Requisitos para el voto electrónico	19
4.2.2 Riesgos del voto electrónico.....	20
4.3 Blockchain	21
4.3.1 Características.....	24
4.3.2 Usos	25
4.3.3 Definiciones complementarias	25
4.3.4 Operación básica de Blockchain.....	27
4.3.5 Algoritmos de Consenso de Blockchain.....	28
4.3.6 Tipos de blockchain.....	29
4.4 Hyperledger.....	30

4.5	Hyperledger Fabric	32
4.5.1	Ventajas	33
4.5.2	Glosario de términos	33
4.5.2.1	Anchor Peer	33
4.5.2.2	Block	34
4.5.2.3	Chain.....	34
4.5.2.4	Chaincode o Smart contract	34
4.5.2.5	Channel.....	34
4.5.2.6	Commit.....	35
4.5.2.7	Concurrency Control Version Check.....	35
4.5.2.8	Configuration Block	35
4.5.2.9	Consensus	35
4.5.2.10	Consenter set.....	36
4.5.2.11	Consortium.....	36
4.5.2.12	Chaincode definition	36
4.5.2.13	Dynamic Membership.....	37
4.5.2.14	Endorsement.....	37
4.5.2.15	Endorsement policy	37
4.5.2.16	Follower.....	38
4.5.2.17	Genesis Block	38
4.5.2.18	Gossip Protocol	38
4.5.2.19	Hyperledger Fabric CA	38
4.5.2.20	Init	38
4.5.2.21	Install.....	39
4.5.2.22	Instantiate.....	39
4.5.2.23	Invoke.....	39
4.5.2.24	Leader	39

4.5.2.25	Leading Peer	40
4.5.2.26	Ledger	40
4.5.2.27	Log entry	40
4.5.2.28	Membership Service Provider	41
4.5.2.29	Membership Service	41
4.5.2.30	Ordering Service	41
4.5.2.31	Organization	42
4.5.2.32	Peer	42
4.5.2.33	Policy	42
4.5.2.34	Private Data	43
4.5.2.35	Private Data Collection	43
4.5.2.36	Proposal	43
4.5.2.37	Query	43
4.5.2.38	Quorum	44
4.5.2.39	Raft	44
4.5.2.40	Software Development Kit (SDK)	44
4.5.2.41	State Database	45
4.5.2.42	System Chain	45
4.5.2.43	Transaction	45
4.5.2.44	World State	46
4.6	Aplicación Descentralizada	46
4.7	Agile Blockchain DApp Engineering (ABCDE)	49
4.8	NodeJS	50
4.8.1	Express	50
4.9	Go	51
4.10	Flutter	52
4.11	Trabajos relacionados	52

5	MATERIALES Y MÉTODOS	57
5.1	Contexto	57
5.2	Procesos	58
5.3	Recursos	59
5.3.1	Científicos.....	60
5.3.2	Técnicos.....	60
5.4	Participantes.....	61
6	RESULTADOS	62
6.1	Objetivo 1: Definir la arquitectura del sistema de voto electrónico usando la ingeniería de requisitos.....	62
6.1.1	Fase 1: Objetivo del sistema.....	62
6.1.2	Fase 2: Identificar los actores	62
6.1.3	Fase 3: Historias de usuario	63
6.1.3.1	Requisitos funcionales.....	1
6.1.3.2	Requisitos no funcionales.....	2
6.1.3.3	Historias de usuario.....	3
6.1.3.4	Casos de uso	3
6.1.4	Fase 4: Dividir el sistema en dos subsistemas	3
6.1.4.1	Diagramas de arquitectura del sistema e-voting	4
6.2	Objetivo 2: Desarrollar el sistema de voto electrónico utilizando la arquitectura anteriormente definida.	6
6.2.1	Construir la red descentralizada de Blockchain.	6
6.2.1.1	Prerrequisitos	6
6.2.1.2	Generación del material criptográfico	7
6.2.1.3	Generación de las configuraciones para el bloque génesis	9
6.2.1.4	Generación de las configuraciones del canal.....	10
6.2.1.5	Generación de las configuraciones de los AnchorPeers.....	10

6.2.1.6	Creación del contenedor Portainer	11
6.2.1.7	Crear y ejecutar contenedores.....	11
6.2.1.8	Creación del canal.....	13
6.2.1.9	Añadir organizaciones al canal.....	13
6.2.1.10	Configuración de los AnchorPeers	16
6.2.2	Fase 5: Diseño del subsistema de contratos inteligentes.....	18
6.2.3	Fase 6: Codificación y prueba del subsistema de contratos inteligentes	18
6.2.3.1	Codificación de los contratos inteligentes	18
6.2.3.2	Pruebas de subsistema de contratos inteligentes	22
7	DISCUSIÓN	23
8	CONCLUSIONES.....	24
9	BIBLIOGRAFÍA	25
10	ANEXOS.....	28
A.	Anexo 1. Diagrama de procesos.....	29
B.	Anexo 2. Especificación de requisitos de software	30
C.	Anexo 3. Historias de Usuario	83
D.	Anexo 4. Arquitectura del sistema de e-voting	99
E.	Anexo 5. Plan de pruebas para subsistema de contratos inteligentes.	100

Índice de figuras

Figura 1. Esquema de la cadena de bloques	23
Figura 2. Estructura del invernadero Hyperledger	31
Figura 3. Proyectos del invernadero Hyperledger	32
Figura 4. Estructura de una DApp simplificada	47
Figura 5. Estructura de una DApp extendida	48
Figura 6. Diagrama de flujo de procesos	59
Figura 7. Diagrama de casos de uso	3
Figura 8. Arquitectura del sistema e-voting	4
Figura 9. Arquitectura de la red de Hyperledger Fabric.....	5
Figura 10. Componentes Peers de la red blockchain.....	6
Figura 11. Fragmento código fuente archivo crypto-config.yalm	8
Figura 12. Fragmento del código fuente del archivo configtx.yaml	9
Figura 13. Comandos para la creación archivos configuración de AnchorPeers de cada organización.	11
Figura 14. Declaración de variables necesarias para ejecutar los contenedores.	11
Figura 15. Comando para crear y ejecutar los contenedores.....	12
Figura 16. Resultado de la creación y ejecución de los contenedores Docker.	12
Figura 17. Vista de administración de contenedores del Portainer.....	13
Figura 18. Comandos para crear el canal	13
Figura 19. Comando para añadir el primer peer al canal.	14
Figura 20. Comandos para añadir las organizaciones restantes al canal.....	16
Figura 21. Comando para asignar la configuracion AnchorPeer a la organización 1... ..	16
Figura 22. Comandos para asignar las configuraciones AnchorPeers a las organizaciones restantes.	17
Figura 23. Diagrama de clases del sistema.	18
Figura 24. Código fuente del SC vote.go	21
Figura 25. Prueba del subsistema de contratos inteligentes desde la terminal	22
Figura 26. Validación en la base de datos.	22
Figura 27. Roles y actividades	6

Índice de tablas

TABLA I.....	1
TABLA II.....	2
TABLA III. PERSONAL INVOLUCRADO ESTUDIANTES DE LA CIS	4
TABLA IV. PERSONAL INVOLUCRADO DOCENTE DE LA CIS	4
TABLA V. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS	5
TABLA VI. REFERENCIAS.....	5
TABLA VII. CARACTERÍSTICAS USUARIO USUARIO.....	6
TABLA VIII. CARACTERÍSTICAS USUARIO VOTANTE	6
TABLA IX. CARACTERÍSTICAS USUARIO ADMINISTRADOR	6
TABLA X. CARACTERÍSTICAS USUARIO MEMBERSHIP SERVICE PROVIDERS....	7
TABLA XI. CARACTERÍSTICAS USUARIO SC-VOTES.....	7
TABLA XII. REQUISITO FUNCIONAL INICIO DE SESIÓN	8
TABLA XIII. REQUISITO FUNCIONAL GESTIÓN DE VOTANTES	9
TABLA XIV. REQUISITO FUNCIONAL GESTIÓN DE PARTIDOS	9
TABLA XV. REQUISITO FUNCIONAL GESTIÓN DE ELECCIONES.....	10
TABLA XVI. REQUISITO FUNCIONAL CONSULTA DE RESULTADOS.....	10
TABLA XVII. REQUISITO FUNCIONAL REALIZAR VOTO	11
TABLA XVIII. REQUISITO NO FUNCIONAL RENDIMIENTO.....	11
TABLA XIX. REQUISITO NO FUNCIONAL USABILIDAD	12
TABLA XX. REQUISITO NO FUNCIONAL FIABILIDAD	12
TABLA XXI. REQUISITO NO FUNCIONAL SEGURIDAD	12

1 TITULO

“Implementación de un sistema multiplataforma de voto electrónico basado en Blockchain”

2 RESUMEN

En la actualidad las votaciones dentro de la Universidad Nacional de Loja se han venido realizando de la forma convencional, centralizada. Es por esto que el presente trabajo tiene como objetivo implementar un sistema de voto electrónico, utilizando tecnología Blockchain, que permita registrar votos y realizar el escrutinio de los mismos en los procesos de votación de la Universidad.

Con esta meta el primer paso fue definir una metodología que se adapte a las necesidades del proyecto, con lo cual se determinó que la metodología “Agile Blockchain DApp Engineering” (ABCDE) es la más óptima para el desarrollo de aplicaciones descentralizadas (DApps); de igual forma, para la elicitación de requisitos se usó el estándar IEEE – 830 para generar el documento de especificación de requisitos de software.

El proyecto consta de tres fases bien definidas dentro de las cuales se aplican algunas de las fases de la metodología. En la primera fase se busca definir la arquitectura del sistema de voto electrónico usando la ingeniería de requisitos; resultando en la obtención de los objetivos del sistema, los actores y gracias al estándar IEE – 830 se extrajeron los requisitos en forma de historias de usuario, de acuerdo a la metodología se debe dividir el sistema en dos subsistemas, el primero es el Chaincode que se ejecuta dentro de la blockchain, que es quien comanda todas las transacciones entre esta y la aplicación, y el segundo es la aplicación que se comunica con el Chaincode para enviar solicitudes de transacción. Gracias a esto se obtuvo la arquitectura del sistema, y ya que se desarrolló una DApp haciendo uso de una blockchain privada Hyperledger, en dicha arquitectura se evidencia la topología de la red blockchain. Y de igual forma se obtuvieron los diagramas de casos de uso, y la arquitectura interna de la red de Hyperledger Fabric mostrando los permisos de cada uno de los componentes Peers de la red blockchain y el diagrama de la estructura interna de estos últimos.

En la segunda fase se pretende desarrollar el sistema de voto electrónico utilizando la arquitectura definida, y con esta se construyó la red blockchain configurando las características descritas, lograr que la red sea estable, funcional y segura es una de las etapas más importantes en el desarrollo de DApps con blockchain ya que ahí se resguardaran los datos generados y las transacciones se ejecutarán dentro de ella, para lograr esto último se escribió el código del Chaincode que comandara todas las operaciones relacionadas con la aplicación de voto electrónico; este fue escrito en el lenguaje de programación GO, lo que le da una estructura bastante robusta en la ejecución de las operaciones, antes de empezar con el desarrollo del Chaincode se diseñó el diagrama de clases que representa la estructura y funcionamiento del mismo. De forma paralela se desarrolló el sistema de aplicación, para el cual se decidió construirlo en Flutter, esto debido a que se puede generar aplicaciones para diferentes plataformas usando la misma base de código, lo que acorta en gran medida

el tiempo de desarrollo. Una vez culminados los dos subsistemas se procedió a la integración de estos, lo que conllevó al desarrollo de un REST – API intermedio (middleware), desarrollado en NodeJs, que permite la comunicación entre el sistema de aplicación y el Chaincode, usando el paquete de comunicación entre NodeJs y Hyperledger Fabric llamado fabric-sdk-node. La DApp es la integración de estas dos partes que funcionan como un solo sistema.

En la tercera fase y final se realizan las pruebas del sistema de voto electrónico realizando una votación en un ambiente de pruebas simulado.

Como resultado final se obtiene un sistema que permite realizar el proceso de votación de forma electrónica, cuyos datos están resguardados dentro de una blockchain privada, lo que les da mayor seguridad ya que al ser una red descentralizada los datos no se encuentran en un solo nodo de la red, esto también permite dar transparencia y trazabilidad al proceso de votación en la Universidad Nacional de Loja; esto también abre la puerta para que se desarrollen aplicaciones descentralizadas para distintas áreas y con distintos propósitos.

3 INTRODUCCIÓN

Durante los últimos años la palabra blockchain ha ido ganando notoriedad, esto debido a que han surgido varias iniciativas, impulsadas tanto por el sector público como por el privado, y ya que las aplicaciones, desarrolladas para funcionar con las redes blockchain, trabajan de una forma descentralizada; esto resulta ser muy atractivo para los proyectos que buscan brindar transparencia, trazabilidad y seguridad a las empresas y sus productos. Un ejemplo muy claro es la empresa “El Ordeño” que introdujo la tecnología blockchain, haciendo uso del proyecto de IBM “IBM Food Trust” [1], para dar trazabilidad a su producto TRU [2]; gracias a esto la empresa entrega a sus clientes una herramienta con la cual se puede conocer todo el proceso de producción y transporte del producto desde su origen. Otro sector que también ha apostado por esta tecnología es el camaronero cuyo propósito es dar a conocer desde la cosecha hasta el empaque del camarón y de esta forma dar trazabilidad a la producción del crustáceo [3].

La tecnología Blockchain, también conocida como “cadena de bloques”, se puede definir como una base de datos distribuida que se asegura usando métodos de criptografía. Estas son algunas de las razones por las que las empresas optan por implementarla ya que gracias a ella se abre un abanico de posibilidades de proyectos que, al ser desarrollados como aplicaciones distribuidas, adquieren un valor agregado muy elevado, esto además de los beneficios que otorga blockchain por si sola. Existen dos tipos de blockchain identificadas; se las conoce como privadas y públicas; las Blockchain públicas son las comerciales, un ejemplo de este tipo es la red Ethereum. Por otro lado, las redes privadas son redes que son gestionadas por una o varias instituciones y el acceso a estas está restringido a la autorización de quien la administra, es el caso de la red del proyecto “IBM Food Trust” que es una red privada que fue creada mediante proyecto Hyperledger. Este último es un proyecto que permite crear redes blockchain propias y administrarlas de acuerdo a las necesidades de la organización.

A pesar de que ya muchos sectores están apostando por implementar la tecnología blockchain aún hay áreas a las que su implementación no ha sido aprobada; una de estas áreas son las votaciones. El voto se puede definir como un medio para expresar algo, es la forma de participar en un proceso de toma de decisiones y/o de selección de representantes y gobernantes. En la mayoría de los casos cuando se realizan votaciones, se ejecutan de la forma tradicional, es decir haciendo uso de papel, esto es algo que la tecnología puede ayudar a reducir pues al brindar medios digitales para realizar esta acción se reduce el uso de papel y el proceso de escrutinio se puede realizar de forma inmediata.

La meta del presente trabajo es la de desarrollar una aplicación descentralizada que permita realizar la votación de forma electrónica y poder entregar los resultados de forma inmediata, valiéndose de una red blockchain para otorgar mayor seguridad y transparencia al proceso. La blockchain a implementar es una red privada, levantada dentro de la red institucional de la Universidad Nacional de Loja, esto permite tener mayor control sobre la misma.

Al tener implementada una blockchain institucional, la Universidad, tiene un medio fiable para desarrollar diferentes tipos de proyectos descentralizados, basados en blockchain, que sean de utilidad para la misma institución y para la sociedad.

Es por ello que se planteó como objetivo principal “Implementar un sistema de voto electrónico utilizando tecnología Blockchain que permita registrar votos y realizar el escrutinio de los mismos en el proceso de votación de la Universidad Nacional de Loja.” y con el fin de darle cumplimiento se han definido tres objetivos específicos: el primero reza “Definir la arquitectura del sistema de voto electrónico usando la ingeniería de requisitos”, el segundo proclama “Desarrollar el sistema de voto electrónico utilizando la arquitectura anteriormente definida”, y como ultimo objetivo específico se plantea “Probar el sistema de voto electrónico realizando una votación en un ambiente de pruebas simulado”

Para poder lograr los objetivos planteados y con el fin de obtener una aplicación descentralizada que sea estable, robusta y fiable se utilizó la metodología “Agile Blockchain DApp Engineering”, la cual establece que las DApps deben ser desarrolladas dividiendo en dos subsistemas, dentro de los cuales las evaluaciones de seguridad son bastante estrictas para que, finalizado el desarrollo de las dos partes, se haga una integración de las mismas y se realicen pruebas de funcionamiento. Además, se hizo uso del estándar IEEE – 830 para realizar la especificación de requisitos.

En las diferentes secciones del documento se encuentra todo lo necesario para entender cómo se desarrolló el proyecto. A continuación, se describen de forma general, las secciones del mismo: En la sección Revisión de Literatura, se presenta toda la base teórica necesaria para entender cómo funciona una blockchain, que es el voto electrónico, las herramientas necesarias para el desarrollo del proyecto y los trabajos relacionados al mismo. La sección Materiales y Métodos describe el contexto y los procesos a seguir para el desarrollo del proyecto, así como los recursos, tanto científicos como técnicos, y los participantes que intervinieron en el desarrollo del mismo. La sección de resultados está dividida en tres secciones, una por cada objetivo específico, dentro de las cuales se presentan las evidencias de los resultados obtenidos tras culminar cada una de las fases necesarias para lograr cada uno de ellos. En la sección de Discusión se analizan los resultados, basado en las evidencias obtenidas, con lo que se describe el cumplimiento de cada uno de los objetivos. Finalmente, en la sección de Conclusiones se describe los sucesos más relevantes que se obtuvieron tras el desarrollo del proyecto y los trabajos futuros.

4 REVISIÓN DE LITERATURA

En este apartado se realiza la recolección de información bibliográfica importante que sustentan el desarrollo del TT, así como también, los conceptos que permiten una mayor comprensión del tema.

Se inicia con una breve introducción a las definiciones generales acerca del voto electrónico.

4.1 Voto

De acuerdo a [4] el voto es, para todo efecto práctico, el resultado de la acción de participar en un proceso de toma de decisiones y/o de selección de representantes y gobernantes. Es sinónimo, entre otros, del término sufragio.

4.2 Voto electrónico o e-voting

El termino e-voting, según [5], hace referencia a la “votación electrónica” y hace alusión a la opción de utilizar medios electrónicos para votar en los referendos y las elecciones.

Hay sistemas como máquinas de registro electrónico directo de la votación, en que la información no es transmitida a través de Internet u otra red. La información es entonces registrada y almacenada en la máquina. También está la votación a través de Internet, que se utiliza una computadora una computadora personal con una conexión remota. Las agendas electrónicas y lo teléfonos fijos o móviles también pueden ser utilizados para emitir un voto electrónicamente.

4.2.1 Requisitos para el voto electrónico

Los sistemas de votación tradicionales se han desarrollado para garantizar que los principios necesarios para la celebración de elecciones y referendos democráticos se cumplan, como la garantía de la libertad de voto, la secrecía del voto, la no modificación de la intención

expresada en el voto y la no intimidación durante la realización de la votación. Es esencial que estos principios no resulten perjudicados por la introducción de nuevos métodos de votación y, en consecuencia, que los sistemas de votación electrónica estén diseñados y operados de manera que se garantice la fiabilidad y la seguridad del proceso de votación. Los sistemas de votación tradicionales se han desarrollado para garantizar que los principios necesarios para la celebración de elecciones y referendos democráticos se cumplan, como la garantía de la libertad de voto, la secrecía del voto, la no modificación de la intención expresada en el voto y la no intimidación durante la realización de la votación. Es esencial que estos principios no resulten perjudicados por la introducción de nuevos métodos de votación y, en consecuencia, que los sistemas de votación electrónica estén diseñados y operados de manera que se garantice la fiabilidad y la seguridad del proceso de votación [5].

Un sistema de voto electrónico, por lo tanto, debe considerar los siguientes requisitos mínimos, según [5]:

- Asegurar que sólo las personas con derecho a voto están en condiciones de votar.
- Garantizar que cada voto sea contado y que sea contado sólo una vez.
- Mantener el derecho del elector a formar y expresar su opinión de una manera libre, sin ningún tipo de coacción o influencia indebida.
- Proteger la secrecía del voto en todas las fases del proceso de votación.
- Garantizar la accesibilidad al mayor número posible de votantes, especialmente a las personas con discapacidad.
- Aumentar la confianza de los electores al maximizar la transparencia de la información sobre el funcionamiento de cada sistema.

4.2.2 Riesgos del voto electrónico

Según [5], se debe prestar especial atención a los siguientes riesgos que presenta el voto electrónico:

- La intervención no autorizada de terceros en el proceso de votación. En la etapa actual de la tecnología de la información, no existe garantía de que un programa no puede ser manipulado para permitir el almacenamiento e impresión de un documento diferente del que aparece en la pantalla.
- En comparación con los procedimientos convencionales, es más difícil detectar e identificar el origen de los errores y de las fallas técnicas.
- Existe la posibilidad de que un sistema completamente digitalizado no reporte los resultados, y al carecer de un respaldo físico sea sumamente difícil, si no imposible, hacer un recuento público.

En el contexto del voto electrónico, se debe prestar especial atención al proceso que garantice el voto libre y secreto. Sólo los electores debidamente acreditados deben ser capaces de ejercer el voto, para lo que su identidad debe ser corroborada, al tiempo que se verifica que tenga derecho al voto. Para evitar que se vote en repetidas ocasiones, o cualquier otro tipo de fraude, se debe llevar un registro y revisado para establecer si una persona ya ha votado. En un sistema de voto electrónico debe existir una distinción electrónica entre el voto y la identificación del votante.

4.3 Blockchain

De acuerdo a [6], la cadena de bloques, más conocida por el término en inglés blockchain, es una tecnología que permite la transferencia de datos digitales con una codificación muy sofisticada y de una manera completamente segura, es decir, se habla de un texto de acontecimientos digitales, esta transferencia o procedimiento no requiere de un intermediario centralizado (como los bancos en el caso del dinero y notarias para el caso de los contratos o acuerdos) que identifique y certifique la información allí contenida, sino que esta información está distribuida en múltiples nodos independientes entre sí, que la registran y la validan. Una vez la información este en la

cadena de bloques, "la información no puede ser borrada, solo se podrán añadir nuevos registros, y no será legitimada a menos que la mayoría de ellos se pongan de acuerdo para hacerlo".

También se conoce a esta tecnología como una base red de procesamiento distribuida donde los datos y las tareas de procesamiento de las transacciones son repartidos a los participantes la red, sin embargo esta tecnología no consiste en una única técnica, sino que se vale de la criptografía, las matemáticas, distintos modelos económicos, combinados con redes "peer-to-peer" (punto a punto) y algoritmos de consenso distribuido para resolver el problema tradicional de sincronización de bases de datos distribuidas. Es una construcción de infraestructura que integra muchos campos [7].

En blockchain cada contrato, proceso, tarea y pago podría tener un registro digital y una firma que podría ser identificable, validada, almacenada y compartida. Cualquier aplicación que presente altos costos de intermediación, problemas de agilidad y/o vulnerabilidad por la necesidad de involucrar un intermediario que genere confianza entre proveedores y clientes se puede beneficiar del uso de la blockchain [8].

En cada bloque se almacena: [9]

- Una cantidad de registros o transacciones válidas.
- Información referente a ese bloque.
- Su vinculación con el bloque anterior y el bloque siguiente a través del hash de cada bloque un código único que sería como la huella digital del bloque.

Por lo tanto, cada bloque tiene un lugar específico e inamovible dentro de la cadena, ya que cada bloque contiene información del hash del bloque anterior. La cadena completa se guarda en cada nodo de la red que conforma la blockchain, por lo que se almacena una copia exacta de la cadena en todos los participantes de la red, así como se observa en la Figura 1 tomada de [9].

A medida que se crean nuevos registros, estos son primeramente verificados y validados por los nodos de la red y luego añadidos a un nuevo bloque que se enlaza a la cadena.

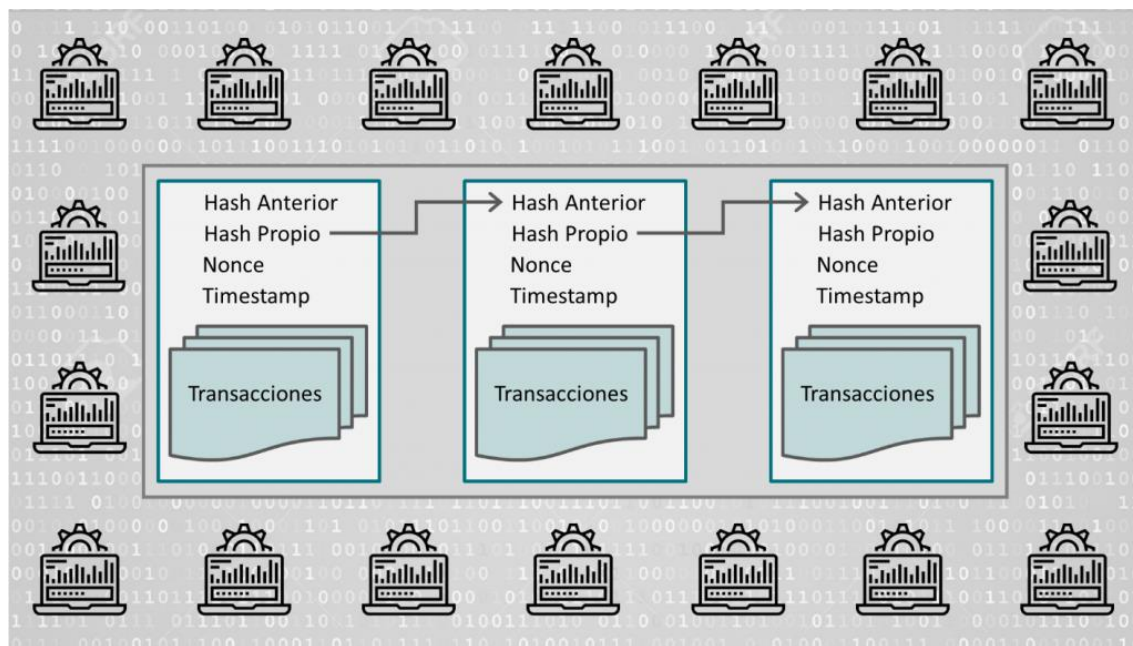


Figura 1. Esquema de la cadena de bloques

4.3.1 Características

Las principales características de la tecnología Blockchain que tenemos según [7]:

- **Descentralización:** aquí no se confía en un único nodo centralizado (sistema centralizado). Por el contrario, cada transacción es controlada, compartida y autorizada por todos los nodos que participan de la red blockchain (sistema descentralizado).
- **Transparencia:** los datos almacenados por los sistemas que usan la tecnología Blockchain son transparentes para cada nodo. En una analogía con los libros contables, en la cual, cada transacción queda registrada y cualquier nodo puede consultarla.
- **Código abierto:** por lo general los sistemas Blockchain son abiertos para cualquier individuo que desee participar en la red. Esto significa que cualquiera puede acceder a la información de la red. Además, los registros pueden ser verificados públicamente y usar esta tecnología para crear cualquier aplicación que se requiera.
- **Autonomía:** debido a la base del consenso, cada nodo del sistema de Blockchain, tienen la capacidad de transferir o actualizar datos de manera segura y confiable. El objetivo es poder confiar tanto en una sola persona, como en todo el sistema.
- **Inmutabilidad:** en este caso cualquier registro se preservará para siempre, debido a que las operaciones que se realizan no se pueden alterar y son únicas. Las transacciones efectuadas son realizadas con base a un sistema criptográfico, lo que resulta que las operaciones sean casi imposibles de hackear.
- **Anonimato:** la tecnología Blockchain solucionó el problema de confianza entre nodos, es decir, la transferencia de datos. En donde, cada transacción puede ser anónima, solo es necesario conocer la dirección de Blockchain de la otra persona.
- **Trazabilidad:** blockchain garantiza la capacidad de registrar la traza de los productos a lo largo de la cadena de suministro interna o externa, empaquetarlos en un formato legible y prepararlos para poder ser gestionados por el propio software o como respuesta a una solicitud de servicio.

4.3.2 Usos

En esta tecnología se puede usar básicamente cualquier tipo de información que requiera ser preservada de forma intacta y que deba permanecer disponible puede ser almacenada en blockchain de manera segura, descentralizada y más económica que a través de intermediarios. A continuación, se describe algunos usos según [9]:

- **Usos en la salud:** Por ejemplo, en los registros de salud podrían ser unificados y almacenados en blockchain. De tal manera, la historia médica de cada paciente estaría segura y a la vez disponible para cada médico autorizado, independientemente del centro de salud donde se haya atendido el paciente. Además, en la industria farmacéutica se puede utilizar para verificar medicamentos y evitar falsificaciones.
- **En documentos:** en este caso, resultaría muy útil para la gestión de bienes y documentos digitales. Blockchain permite registrar compras, escrituras, documentos o cualquier tipo de bien digital y que no pueda ser falsificado.
- **Otros:** también se puede revolucionar el mercado de Internet de las Cosas (IoT), donde existe millones de dispositivos conectados a Internet que deben ser gestionados por las empresas proveedoras. En un futuro próximo, el modelo centralizado no va a soportar tantos dispositivos, sin contar que muchos de ellos no son lo suficientemente seguros. Con la tecnología blockchain los dispositivos pueden comunicarse a través de la red de manera directa, segura y confiable, sin intermediarios.

4.3.3 Definiciones complementarias

Según [8], para entender la forma en la que opera blockchain es necesario conocer algunas definiciones complementarias:

- **Hoja única contable abierta y distribuida (Open Ledger):** la tecnología Blockchain está constituida por una base de datos distribuida, que consta de un registro compartido que contiene la información de los dueños de los activos y el histórico del intercambio de la propiedad de los mismos (transacciones). Esto quiere decir que se puede establecer la cadena de propiedad del activo desde su origen o emisión.
- **Nodos:** cada participante de la red Blockchain constituye un nodo. Cada vez que se generan transacciones estas se transmiten a los nodos, un bloque se confirma y se añade a la cadena, esta operación actualiza la hoja contable que cada participante almacena.
- **Mecanismos o algoritmos de consenso:** La propiedad de los activos y la transferencia de los mismos son validadas por los participantes de la red. La mayoría de nodos debe de estar de acuerdo sobre el estado resultante y el orden de cada transacción, esto se logra verificando en la hoja contable distribuida i) si el activo existe (origen y emisión), ii) si este pertenece a quien transfiere (cadena de propiedad), y iii) si previamente no ha sido transferido a otro participante (evitar el fraude por doble desembolso). El mecanismo de consenso también se encarga de verificar que los nodos sean “honestos”, es decir, sean reales y no varios participantes falsos dominados por un atacante para ganar más participación en la decisión. Una vez verificada, una transacción no podrá ser eliminada o modificada.
- **Bloque:** para brindar mayor seguridad las transacciones se almacenan en bloques y este se encadena a bloques anteriores. Cada bloque contiene una estampilla de tiempo que indica el momento en el cual se hicieron las operaciones contenidas en él y un número cifrado (hash) que se crea a partir de la combinación de las transacciones contenidas y la referencia al bloque anterior. Las transacciones quedan en firme cuando su bloque contenedor se enlaza a la cadena de bloques. Cualquier modificación en las transacciones modifica el hash del bloque, esto rompe la cadena y las transacciones se anulan.

- **Mineros:** son los nodos que se encargan de las operaciones de validación de transacciones y de la creación y enlace de nuevos bloques dentro de la cadena. Los mineros prestan sus recursos para mantener la seguridad y confianza de las transacciones. Los mineros que validan bloques son recompensados por la blockchain con tokens, estos llegan a tener valor de mercado.

4.3.4 Operación básica de Blockchain

A continuación, se describe la operación básica de la Blockchain según [8]:

- Acuerdo entre participantes antes de la transacción: Se determinan los componentes para el acuerdo, como las variables de transacción, el tipo de activo, tamaño o cantidad, dirección del cliente, entre otras.
- Publicación de la operación: La transacción se transmite a los nodos de la blockchain.
- Creación del bloque: Los nodos combinan la operación con otras transacciones en un bloque, el bloque genera un número Hash, el cual se crea a partir de:
 - Un número hash resultante de la combinación de todas las transacciones, para ello usa una estrategia conocida como árbol de Merkle. Un cambio en una de las transacciones produce un Hash totalmente diferente.
 - La estampilla de tiempo de la creación del bloque, esto permite validar que el bloque nuevo es creado posteriormente al bloque anterior.
 - La combinación con el Hash del bloque anterior, esto permite determinar si el nuevo bloque hace referencia a un bloque anterior existente y válido. Si el bloque anterior no existe se anulan las operaciones.
 - Un número aleatorio conocido como Nounce.

- Validar el bloque: Esta es la operación de minería, para ello se usan diferentes aproximaciones como la prueba de trabajo (Proof of work) o la prueba de participación (Proof of Stake). El resultado de esto es un número que combinado con el Hash del nuevo bloque permite enlazarlo a la Blockchain.
- Distribución de la solución y verificación de transacciones: La solución se transmite a la red, los nodos verifican cada transacción del bloque verificando la existencia, pertenencia y orden de las transacciones.
- Encadenamiento del bloque: Una vez las transacciones del bloque son verificadas por la mayoría de nodos, el bloque nuevo se combina con los bloques anteriores, creando una cadena de bloques.
- Confirmación de la transacción: De acuerdo al tipo de Blockchain para confirmar una transacción se debe esperar que se enlacen en promedio otros 5 bloques.

4.3.5 Algoritmos de Consenso de Blockchain

El algoritmo de consenso es fundamental en la blockchain y se encarga del encadenamiento de los bloques. El objetivo que tiene es proveer de un conjunto de normas bien definidas que deben cumplir todos los usuarios para mantener la consistencia de la cadena. Este debe actuar frente a las bifurcaciones de la cadena (forks), los participantes deben ponerse de acuerdo sobre que cadena es la válida. Además, debe decidir que participante puede o no añadir un nuevo bloque a la cadena y se encarga del encadenamiento de los bloques.

Los algoritmos de consenso más populares son Proof of Work (PoW) y Proof of Stake (PoS):

Proof of Work (PoW)

En PoW (prueba de trabajo) los nodos tienen que resolver un problema matemático para poder añadir un bloque a la cadena. La cadena valida es aquella que más bloques contenga. Si alguien intentase modificarla necesitaría un poder computacional más elevado que el resto de nodos de la red juntos, algo poco probable y requeriría de mucha energía eléctrica [10].

El principal inconveniente es su enorme gasto de recursos. Además, requiere que se actualice el hardware frecuentemente [10]

Proof of Stake (PoS)

En PoS (prueba de participación), los participantes con más participación en la red, es decir, aquellos que más tokens poseen, tienen más probabilidad de añadir un bloque a la cadena. Se asume que un usuario con mucha participación velará por la seguridad de la cadena (no se perjudicaría a el mismo) de este modo los bloques que genere no serían maliciosos [10].

4.3.6 Tipos de blockchain

La tecnología de Blockchain puede ser dividida en dos tipos:

Blockchain pública

La Blockchain pública es aquella que no tiene restricciones en cuanto a la lectura de sus datos y a la visualización de las transacciones para su inclusión en la cadena de bloques, todos los participantes tienen derecho de enviar transacciones y en caso que sean válidas, estas serán incluidas en la Blockchain. Para el proceso de consenso todos los nodos tienen la posibilidad de participar [8].

La blockchain pública es considerada una arquitectura “totalmente descentralizada”. Por ejemplo, Bitcoin y Ethereum son Blockchain públicas [7].

Blockchain privada

La Blockchain privada es aquella que tiene acceso directo a los datos de la cadena de bloque, pero la vista de las transacciones es limitada a una lista predefinida de entidades, el permiso de escritura es mantenido por solo una organización y los permisos de lectura pueden ser públicos o en cierta manera restringidos arbitrariamente {Plazaes2019}.

La blockchain privada, por su naturaleza, en la mayoría de los casos no brinda ventajas significativas en comparación con el modelo tradicional centralizado de confianza. [8]

4.4 Hyperledger

"Hyperledger es una comunidad de código abierto centrada en desarrollar un conjunto de marcos, herramientas y bibliotecas estables para implementaciones de blockchain de nivel empresarial." [11]

Hyperledger es un proyecto colaborativo organizado por The Linux Foundation, que integra a expertos en diferentes áreas como: finanzas, banca, IoT, cadena de suministro, fabricación y tecnología.

El Hyperledger greenhouse o invernadero Hyperledger, alberga el desarrollo de diferentes proyectos empresariales enfocados en blockchain, que nacen desde el laboratorio de semillas Hyperledger Labs, hasta llegar al código funcional que será puesto en producción. La estructura del invernadero Hyperledger se muestra en las Figura 2 y Figura 3 extraídas de [11].

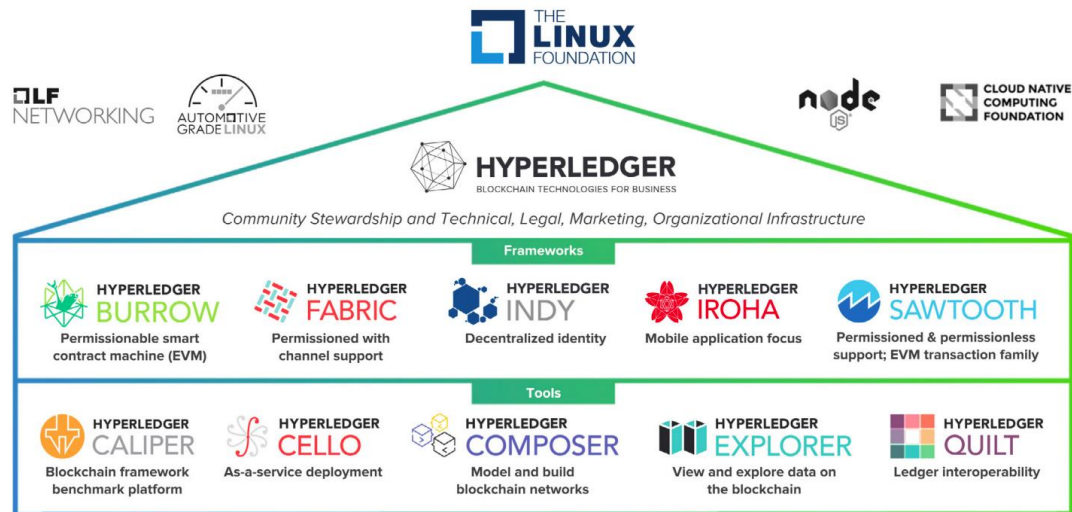


Figura 2. Estructura del invernadero Hyperledger

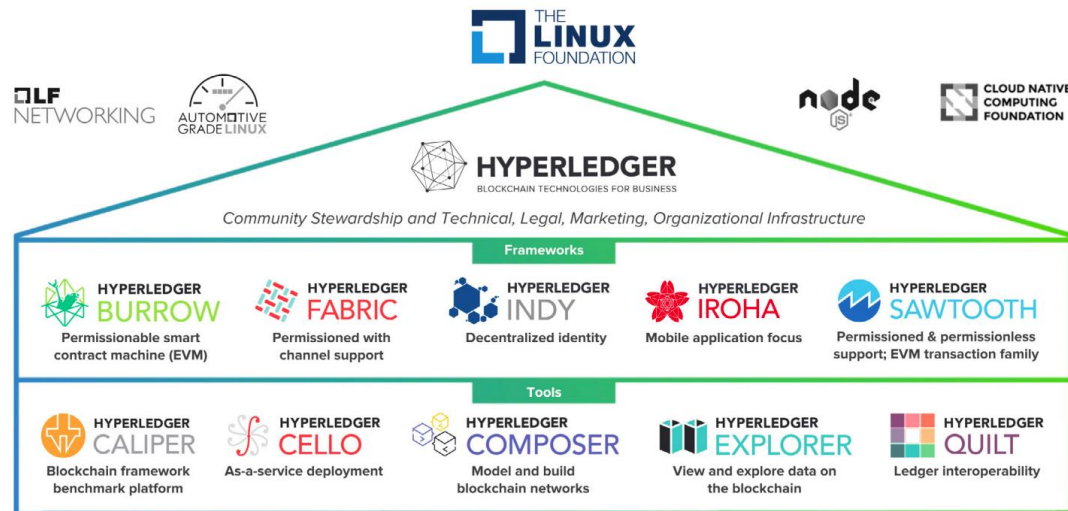


Figura 3. Proyectos del invernadero Hyperledger

4.5 Hyperledger Fabric

Es uno de los proyectos más conocidos de Hyperledger que es impulsado por grandes empresas como IBM. No es una Blockchain publica que esté disponible para todo el mundo, como Bitcoin o Ethereum. Hyperledger Fabric es una infraestructura Blockchain flexible; en otras palabras, es una Blockchain privada orientada al uso empresarial, está diseñada para ser la base del desarrollo de aplicaciones empresariales, permite la creación de contratos inteligentes, llamados chaincode, que pueden ser escritos en cualquier lenguaje, pues en recientes versiones se ha integrado el soporte para NodeJS, Java, JavaScript, etc. [11]–[13]

4.5.1 Ventajas

- **Red con permisos:** permite establecer confianza en una red descentralizada de participantes conocidos a diferencia de las redes abiertas en la que existen participantes anónimos.
- **Transacciones confidenciales:** se puede exponer solo los datos que se desea distribuir con los participantes con los que se apetece compartir.
- **Arquitectura conectable:** la blockchain es flexible, por lo que se puede adaptar a las necesidades del proyecto en el que se necesite usar.
- **Inicio sencillo:** al tener soporte para varios lenguajes de programación no se necesita aprender arquitecturas y lenguajes personalizados, si no que se puede usar la tecnología con la que el equipo este trabajando actualmente.

4.5.2 Glosario de términos

A continuación, se define los términos que se utiliza en Hyperledger Fabric, para facilitar al lector la comprensión en la sección de resultados :

4.5.2.1 Anchor Peer

Utilizado por los cotillas para que los compañeros de diferentes organizaciones se conozcan entre sí.

Cuando se confirma un bloque de configuración que contiene una actualización para los pares ancla, los pares se ponen en contacto con los pares ancla y aprenden de ellos sobre todos los pares conocidos por el/los pares ancla. Una vez que al menos un peer de cada organización ha contactado con un peer ancla, el peer ancla aprende sobre todos los peers del canal. Dado que la comunicación de cotilleo es constante, y que los pares siempre piden que se les informe de la existencia de cualquier par que no conozcan, se puede establecer una visión común de los miembros de un canal.

Como la comunicación entre organizaciones depende de los chismes para funcionar, debe haber al menos un peer ancla definido en la configuración del canal. Se recomienda encarecidamente que cada organización proporcione su propio conjunto de pares de anclaje para una alta disponibilidad y redundancia.

4.5.2.2 Block

Un bloque contiene un conjunto ordenado de transacciones. Está vinculado criptográficamente al bloque anterior y, a su vez, está vinculado a los bloques siguientes. El primer bloque de esta cadena de bloques se denomina bloque génesis. Los bloques son creados por el servicio de ordenación, y luego validados y comprometidos por los pares.

4.5.2.3 Chain

La cadena del libro mayor es un registro de transacciones estructurado como bloques de transacciones vinculadas por hash. Los pares reciben bloques de transacciones del servicio de pedidos, marcan las transacciones del bloque como válidas o inválidas en función de las políticas de aprobación y las violaciones de la concurrencia, y añaden el bloque a la cadena hash en el sistema de archivos del par.

4.5.2.4 Chaincode o Smart contract

Un contrato inteligente es un código -invocado por una aplicación cliente externa a la red de blockchain- que gestiona el acceso y las modificaciones a un conjunto de pares clave-valor en el Estado Mundial a través de la Transacción. En Hyperledger Fabric, los contratos inteligentes se empaquetan como chaincode. El chaincode se instala en los pares y luego se define y utiliza en uno o más canales.

4.5.2.5 Channel

Un canal es una cadena de bloques privada que permite el aislamiento y la confidencialidad de los datos. Un libro de contabilidad específico del canal se comparte entre los pares del canal, y las partes que realizan transacciones deben estar autenticadas en un canal para poder interactuar con él. Los canales se definen mediante un bloque de configuración.

4.5.2.6 Commit

Cada Peer en un canal valida los bloques ordenados de transacciones y luego compromete (escribe/aplica) los bloques a su réplica del Ledger del canal. Los pares también marcan cada transacción en cada bloque como válida o inválida.

4.5.2.7 Concurrency Control Version Check

La comprobación de versiones de control de concurrencia es un método para mantener sincronizado el estado del libro mayor entre los pares de un canal. Los pares ejecutan las transacciones en paralelo y, antes de consignarlas en el libro mayor, los pares comprueban si el estado leído en el momento en que se ejecutó la transacción ha sido modificado en un nuevo bloque que estaba en marcha en el momento de la ejecución o en una transacción anterior en el mismo bloque. Si los datos leídos para la transacción han cambiado entre el tiempo de ejecución y el tiempo de consignación, entonces se ha producido una violación de la Comprobación de la Versión de Control de Concurrencia, y la transacción se marca como no válida en el libro mayor y los valores no se actualizan en la base de datos de estado.

4.5.2.8 Configuration Block

Contiene los datos de configuración que definen los miembros y las políticas de una cadena del sistema (servicio de pedidos) o de un canal. Cualquier modificación de la configuración de un canal o de la red en general (por ejemplo, la salida o entrada de un miembro) dará lugar a un nuevo bloque de configuración que se añadirá a la cadena correspondiente. Este bloque contendrá el contenido del bloque génesis, más el delta.

4.5.2.9 Consensus

Un término más amplio que abarca todo el flujo transaccional, que sirve para generar un acuerdo sobre el orden y para confirmar la corrección del conjunto de transacciones que constituyen un bloque.

4.5.2.10 *Consenter set*

En un servicio de pedidos de Raft, estos son los nodos ordenadores que participan activamente en el mecanismo de consenso en un canal. Si existen otros nodos ordenadores en el canal del sistema, pero no forman parte de un canal, no forman parte del conjunto de consentidores de ese canal.

4.5.2.11 *Consortium*

Un consorcio es un conjunto de organizaciones no ordenantes en la red blockchain. Son las organizaciones que forman y se unen a los canales y que poseen pares. Aunque una red blockchain puede tener múltiples consorcios, la mayoría de las redes blockchain tienen un único consorcio. En el momento de la creación del canal, todas las organizaciones añadidas al mismo deben formar parte de un consorcio. Sin embargo, una organización que no esté definida en un consorcio puede añadirse a un canal existente.

4.5.2.12 *Chaincode definition*

Las organizaciones utilizan una definición de código de cadena para acordar los parámetros de un código de cadena antes de que se pueda utilizar en un canal. Cada miembro del canal que quiera utilizar el chaincode para avalar transacciones o consultar el libro mayor debe aprobar una definición de chaincode para su organización. Una vez que un número suficiente de miembros del canal ha aprobado una definición de código de cadena para cumplir con la política de aprobación del ciclo de vida (que se establece en una mayoría de organizaciones en el canal por defecto), la definición de código de cadena puede ser confirmada en el canal. Una vez confirmada la definición, la primera invocación del chaincode (o, si se solicita, la ejecución de la función Init) iniciará el chaincode en el canal.

4.5.2.13 *Dynamic Membership*

Hyperledger Fabric admite la adición/eliminación de miembros, pares y nodos de servicio de pedidos, sin comprometer la operatividad de la red en general. La membresía dinámica es fundamental cuando las relaciones comerciales se ajustan y las entidades necesitan ser añadidas/eliminadas por diversas razones.

4.5.2.14 *Endorsement*

Se refiere al proceso en el que nodos pares específicos ejecutan una transacción chaincode y devuelven una respuesta de propuesta a la aplicación cliente. La respuesta a la propuesta incluye el mensaje de respuesta a la ejecución del chaincode, los resultados (conjunto de lectura y conjunto de escritura) y los eventos, así como una firma que sirve como prueba de la ejecución del chaincode por parte del par. Las aplicaciones chaincode tienen sus correspondientes políticas de endoso, en las que se especifican los pares que las endosan.

4.5.2.15 *Endorsement policy*

Define los nodos pares de un canal que deben ejecutar transacciones asignadas a una aplicación de código de cadena específica, y la combinación requerida de respuestas (endosos). Una política puede requerir que una transacción sea endosada por un número mínimo de pares endosantes, un porcentaje mínimo de pares endosantes, o por todos los pares endosantes que estén asignados a una aplicación chaincode específica. Las políticas pueden ser curadas en base a la aplicación y al nivel de resistencia deseado contra el mal comportamiento (deliberado o no) de los pares endosantes. Una transacción enviada debe satisfacer la política de endoso antes de ser marcada como válida por los pares que se comprometen.

4.5.2.16 *Follower*

En un protocolo de consenso basado en el líder, como Raft, estos son los nodos que replican las entradas de registro producidas por el líder. En Raft, los seguidores también reciben mensajes "heartbeat" del líder. En caso de que el líder deje de enviar esos mensajes durante un tiempo configurable, los seguidores iniciarán una elección de líder y uno de ellos será elegido líder.

4.5.2.17 *Genesis Block*

El bloque de configuración que inicializa el servicio de pedidos, o sirve como primer bloque de una cadena.

4.5.2.18 *Gossip Protocol*

El protocolo de difusión de datos de cotilleo realiza tres funciones: 1) gestiona el descubrimiento de pares y la pertenencia al canal; 2) difunde los datos del libro mayor entre todos los pares del canal; 3) sincroniza el estado del libro mayor entre todos los pares del canal.

4.5.2.19 *Hyperledger Fabric CA*

La CA de Hyperledger Fabric es el componente por defecto de la Autoridad de Certificación, que emite certificados basados en la PKI a las organizaciones miembros de la red y a sus usuarios. La CA emite un certificado raíz (rootCert) a cada miembro y un certificado de inscripción (ECert) a cada usuario autorizado.

4.5.2.20 *Init*

Un método para inicializar una aplicación chaincode. Todos los chaincodes necesitan tener una función Init. Por defecto, esta función nunca se ejecuta. Sin embargo, se puede utilizar la definición de chaincode para solicitar la ejecución de la función Init con el fin de inicializar el chaincode.

4.5.2.21 *Install*

El proceso de colocar un código de cadena en el sistema de archivos de un compañero.

4.5.2.22 *Instantiate*

El proceso de iniciar e inicializar una aplicación chaincode en un canal específico. Después de la instanciación, los pares que tienen el chaincode instalado pueden aceptar invocaciones de chaincode.

4.5.2.23 *Invoke*

Se utiliza para llamar a las funciones de chaincode. Una aplicación cliente invoca chaincode enviando una propuesta de transacción a un par. El par ejecutará el chaincode y devolverá una respuesta de propuesta endosada a la aplicación cliente. La aplicación cliente reunirá suficientes respuestas de propuesta para satisfacer una política de endoso, y luego enviará los resultados de la transacción para su ordenamiento, validación y confirmación. La aplicación cliente puede optar por no enviar los resultados de la transacción. Por ejemplo, si la invocación sólo consulta el libro mayor, la aplicación cliente normalmente no enviará la transacción de sólo lectura, a menos que se desee registrar la lectura en el libro mayor con fines de auditoría. La invocación incluye un identificador de canal, la función chaincode a invocar y una matriz de argumentos.

4.5.2.24 *Leader*

En un protocolo de consenso basado en el líder, como Raft, el líder es responsable de la ingesta de nuevas entradas de registro, replicándolas a los nodos de ordenación seguidores, y gestionando cuando una entrada se considera comprometida. No se trata de un tipo especial de ordenador. Es sólo un papel que un ordenador puede tener en ciertos momentos, y no en otros, según determinen las circunstancias.

4.5.2.25 *Leading Peer*

Cada organización puede poseer múltiples peers en cada canal al que se suscriba. Uno o más de estos peers deben servir como peer líder del canal, para comunicarse con el servicio de pedidos de la red en nombre de la organización. El servicio de pedidos entrega los bloques a los pares líderes de un canal, que luego los distribuyen a otros pares dentro de la misma organización.

4.5.2.26 *Ledger*

Un libro de contabilidad consta de dos partes distintas, aunque relacionadas: una "cadena de bloques" y la "base de datos de estado", también conocida como "estado mundial". A diferencia de otros libros de contabilidad, las cadenas de bloques son inmutables, es decir, una vez que un bloque se ha añadido a la cadena, no puede cambiarse. En cambio, el "estado mundial" es una base de datos que contiene el valor actual del conjunto de pares clave-valor que han sido añadidos, modificados o eliminados por el conjunto de transacciones validadas y comprometidas en la cadena de bloques.

Es útil pensar que hay un libro de contabilidad lógico para cada canal de la red. En realidad, cada usuario de un canal mantiene su propia copia del libro mayor, que se mantiene coherente con la copia de todos los demás usuarios mediante un proceso llamado consenso. El término Tecnología de Libro Mayor Distribuido (DLT) se asocia a menudo con este tipo de libro mayor: uno que es lógicamente singular, pero que tiene muchas copias idénticas distribuidas a través de un conjunto de nodos de la red (pares y el servicio de ordenación).

4.5.2.27 *Log entry*

La unidad primaria de trabajo en un servicio de ordenación de Balsa, las entradas de registro se distribuyen desde el líder ordenador a los seguidores. La secuencia completa de estas entradas se conoce como "registro". Se considera que el registro es coherente si todos los miembros están de acuerdo con las entradas y su orden.

4.5.2.28 *Membership Service Provider*

El Proveedor de Servicios de Membresía (MSP) se refiere a un componente abstracto del sistema que proporciona credenciales a los clientes y a los pares para que participen en una red Hyperledger Fabric. Los clientes utilizan estas credenciales para autenticar sus transacciones, y los pares utilizan estas credenciales para autenticar los resultados del procesamiento de transacciones (endosos). Aunque está fuertemente conectada a los componentes de procesamiento de transacciones de los sistemas, esta interfaz pretende tener definidos los componentes de los servicios de afiliación, de tal manera que las implementaciones alternativas de esto puedan ser conectadas sin modificar el núcleo de los componentes de procesamiento de transacciones del sistema.

4.5.2.29 *Membership Service*

Los servicios de afiliación autentican, autorizan y gestionan las identidades en una red de blockchain con permisos. El código de los servicios de afiliación que se ejecuta en los pares y en los ordenantes autentifica y autoriza las operaciones de la cadena de bloques. Es una implementación basada en PKI de la abstracción del Proveedor de Servicios de Membresía (MSP).

4.5.2.30 *Ordering Service*

También conocido como ordenador. Un colectivo definido de nodos que ordena las transacciones en un bloque y luego distribuye los bloques a los pares conectados para su validación y confirmación. El servicio de ordenación existe independientemente de los procesos de los pares y ordena las transacciones por orden de llegada para todos los canales de la red. Está diseñado para admitir implementaciones enchufables más allá de las variedades de Kafka y Raft listas para usar. Es un enlace común para toda la red; contiene el material de identidad criptográfica vinculado a cada miembro.

4.5.2.31 *Organization*

También conocidos como "miembros", las organizaciones son invitadas a unirse a la red blockchain por un proveedor de la red blockchain. Una organización se une a una red añadiendo su Proveedor de Servicios de Membresía (MSP) a la red. El MSP define cómo otros miembros de la red pueden verificar que las firmas (como las de las transacciones) fueron generadas por una identidad válida, emitida por esa organización. Los derechos de acceso particulares de las identidades dentro de un MSP se rigen por políticas que también se acuerdan cuando la organización se une a la red. Una organización puede ser tan grande como una corporación multinacional o tan pequeña como un individuo. El punto final de la transacción de una organización es un Peer. Un conjunto de organizaciones forma un Consorcio. Aunque todas las organizaciones de una red son miembros, no todas las organizaciones formarán parte de un consorcio.

4.5.2.32 *Peer*

Una entidad de la red que mantiene un libro mayor y ejecuta contenedores de cadena para realizar operaciones de lectura/escritura en el libro mayor. Los pares son propiedad de los miembros y son mantenidos por ellos.

4.5.2.33 *Policy*

Las políticas son expresiones compuestas por propiedades de las identidades digitales, por ejemplo, `OR('Org1.peer', 'Org2.peer')`. Se utilizan para restringir el acceso a los recursos de una red blockchain. Por ejemplo, dictan quién puede leer o escribir en un canal, o quién puede utilizar una API de cadena específica a través de una ACL. Las políticas pueden definirse en `configtx.yaml` antes de arrancar un servicio de pedidos o crear un canal, o pueden especificarse al instanciar `chaincode` en un canal. En el `configtx.yaml` de ejemplo se incluye un conjunto de políticas por defecto que será apropiado para la mayoría de las redes.

4.5.2.34 *Private Data*

Datos confidenciales que se almacenan en una base de datos privada en cada par autorizado, lógicamente separada de los datos del libro mayor del canal. El acceso a estos datos está restringido a una o más organizaciones en un canal a través de una definición de recopilación de datos privados. Las organizaciones no autorizadas tendrán un hash de los datos privados en el libro mayor del canal como prueba de los datos de la transacción. Además, para mayor privacidad, los hashes de los datos privados pasan por el Servicio de Pedidos, no por los datos privados en sí, por lo que esto mantiene la confidencialidad de los datos privados frente a los Ordenadores.

4.5.2.35 *Private Data Collection*

Se utiliza para gestionar los datos confidenciales que dos o más organizaciones de un canal quieren mantener en privado frente a otras organizaciones de ese canal. La definición de colección describe un subconjunto de organizaciones en un canal con derecho a almacenar un conjunto de datos privados, lo que por extensión implica que sólo estas organizaciones pueden realizar transacciones con los datos privados.

4.5.2.36 *Proposal*

Una solicitud de aprobación que se dirige a compañeros específicos en un canal. Cada propuesta es una solicitud Init o Invoke (lectura/escritura).

4.5.2.37 *Query*

Una consulta es una invocación de chaincode que lee el estado actual del libro mayor pero no escribe en él. La función chaincode puede consultar ciertas claves en el libro mayor, o puede consultar un conjunto de claves en el libro mayor. Dado que las consultas no cambian el estado del libro mayor, la aplicación cliente no suele enviar estas transacciones de sólo lectura para su ordenación, validación y confirmación. Aunque no es típico, la aplicación cliente puede optar por enviar la transacción de sólo lectura para ordenar, validar y

confirmar, por ejemplo, si el cliente quiere una prueba auditable en la cadena del libro mayor de que tenía conocimiento del estado específico del libro mayor en un momento determinado.

4.5.2.38 *Quorum*

Describe el número mínimo de miembros del clúster que deben afirmar una propuesta para que se puedan ordenar las transacciones. Para cada conjunto de consentidores, se trata de una mayoría de nodos. En un clúster con cinco nodos, tres deben estar disponibles para que haya quórum. Si un quórum de nodos no está disponible por cualquier motivo, el clúster deja de estar disponible para operaciones de lectura y escritura y no se pueden consignar nuevos registros.

4.5.2.39 *Raft*

Nuevo para la v1.4.1, Raft es una implementación del servicio de ordenación tolerante a fallos (CFT) basada en la biblioteca etcd del protocolo Raft. Raft sigue un modelo de "líder y seguidor", donde se elige un nodo líder (por canal) y sus decisiones son replicadas por los seguidores. Los servicios de pedidos de Raft deberían ser más fáciles de configurar y gestionar que los servicios de pedidos basados en Kafka, y su diseño permite a las organizaciones contribuir con nodos a un servicio de pedidos distribuido.

4.5.2.40 *Software Development Kit (SDK)*

El SDK de cliente de Hyperledger Fabric proporciona un entorno estructurado de bibliotecas para que los desarrolladores escriban y prueben aplicaciones de código en cadena. El SDK es totalmente configurable y ampliable a través de una interfaz estándar. Los componentes, incluidos los algoritmos criptográficos para las firmas, los marcos de registro y los almacenes de estado, se intercambian fácilmente dentro y fuera del SDK. El SDK proporciona APIs para el procesamiento de transacciones, servicios de membresía, cruce de nodos y manejo de eventos.

4.5.2.41 *State Database*

Los datos del estado del mundo se almacenan en una base de datos de estado para realizar lecturas y consultas eficientes desde chaincode.

4.5.2.42 *System Chain*

Contiene un bloque de configuración que define la red a nivel de sistema. La cadena del sistema vive dentro del servicio de pedidos, y de forma similar a un canal, tiene una configuración inicial que contiene información como: Información del MSP, políticas y detalles de configuración. Cualquier cambio en la red global (por ejemplo, la incorporación de una nueva organización o la adición de un nuevo nodo de pedido) dará lugar a la adición de un nuevo bloque de configuración a la cadena del sistema.

La cadena del sistema puede considerarse como el enlace común de un canal o grupo de canales. Por ejemplo, un conjunto de instituciones financieras puede formar un consorcio (representado a través de la cadena del sistema), y luego proceder a crear canales relativos a sus agendas de negocio alineadas y variadas.

4.5.2.43 *Transaction*

Las transacciones se crean cuando se invoca un código de cadena desde una aplicación cliente para leer o escribir datos del libro mayor. Los clientes de la aplicación Fabric envían propuestas de transacciones a los pares que las respaldan para su ejecución y aprobación, reúnen las respuestas firmadas (aprobadas) de esos pares que las respaldan, y luego empaquetan los resultados y las aprobaciones en una transacción que se envía al servicio de pedidos. El servicio de pedidos ordena y coloca las transacciones en un bloque que se transmite a los pares que validan y consignan las transacciones en el libro mayor y actualizan el estado mundial.

4.5.2.44 *World State*

También conocido como "estado actual", el estado mundial es un componente del HyperLedger Fabric Ledger. El estado mundial representa los últimos valores de todas las claves incluidas en el registro de transacciones de la cadena. Chaincode ejecuta las propuestas de transacción contra los datos del estado mundial porque el estado mundial proporciona acceso directo al último valor de estas claves en lugar de tener que calcularlo recorriendo todo el registro de transacciones. El estado mundial cambiará cada vez que el valor de una clave cambie (por ejemplo, cuando la propiedad de un coche -la "clave"- se transfiera de un propietario a otro -el "valor"-) o cuando se añada una nueva clave (se cree un coche). En consecuencia, el estado del mundo es fundamental para el flujo de una transacción, ya que debe conocerse el estado actual de un par clave-valor antes de poder modificarlo. Los pares consignan los últimos valores en el estado del mundo del libro mayor para cada transacción válida incluida en un bloque procesado.

4.6 Aplicación Descentralizada

Decentralized Application (DApp) es una aplicación que en su mayoría o completamente esta descentralizada. La estructura de una DApp se muestra en Figura 4 y Figura 5, tomadas de [14]. Considerando los siguientes aspectos:

- Backend
- Frontend
- Almacenamiento
- Comunicación
- Resolución de nombre

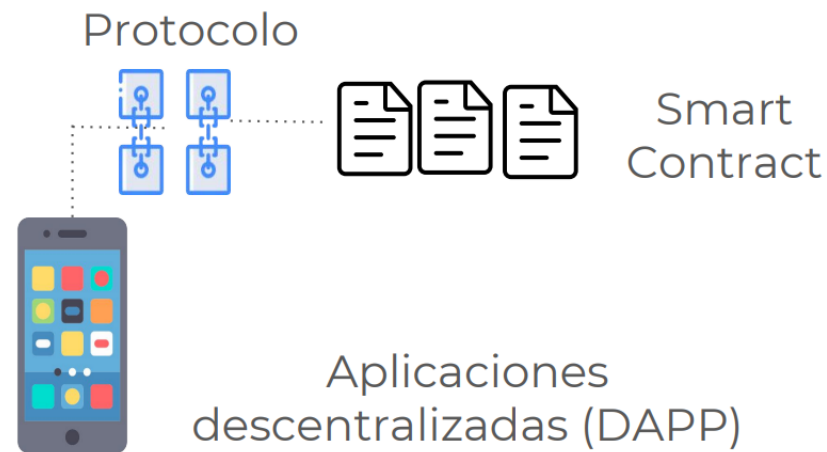


Figura 4. Estructura de una DApp simplificada

Las DApps según [14], deben cumplir con las siguientes tres características:

- **Código abierto**
 - Debe funcionar como una aplicación independiente.
 - Ninguna organización puede reclamar la posesión de la mayor parte de sus tokens.
 - Un Dapp puede adaptar su protocolo en respuesta a las mejoras sugeridas y los comentarios del mercado, pero todos los cambios deben ser adoptados por consenso de todos sus usuarios.
- **Encriptación:** los datos de Dapp y los informes operativos deben encriptarse y almacenarse en un dominio público, el llamado blockchain descentralizado, para evitar cualquier posible interrupción de la red.

- **Token:** una Dapp debe requerir un token criptográfico (bitcoin o token de la aplicación original) para acceder a él. Cada aportación aportada por los mineros debe ser recompensada en los tokens de Dapp.

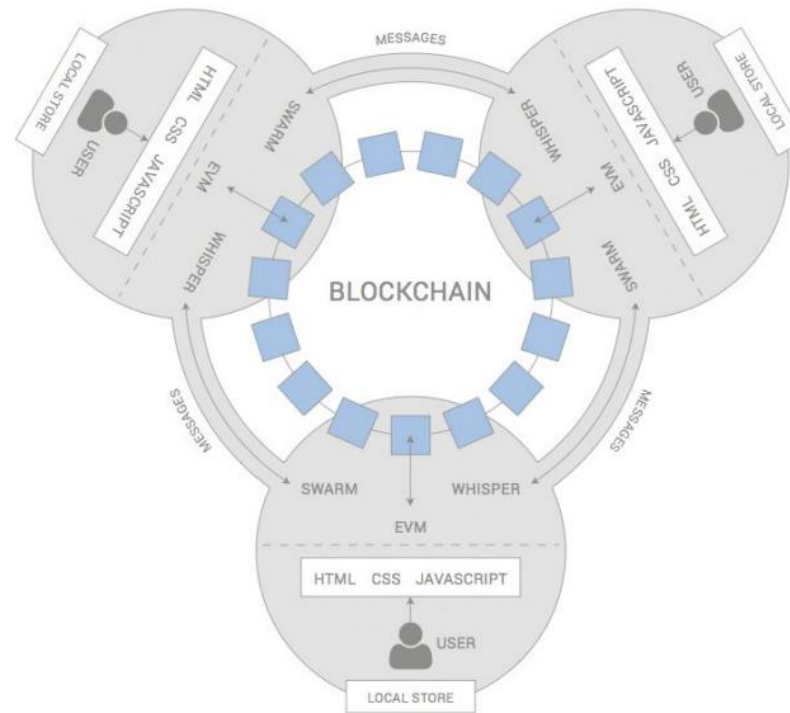


Figura 5. Estructura de una DApp extendida

4.7 Agile Blockchain DApp Engineering (ABCDE)

ABCDE es una metodología de desarrollo de software ágil, lo que significa que sigue los principios del Manifiesto Agile. Sin embargo, se complementa el proceso ágil con un enfoque más formal, utilizando diagramas UML con una notación específica para contratos inteligentes y una lista de verificación específica para la evaluación de seguridad. [15]

Este es un método completo que aborda el desarrollo de software blockchain. El método considera la integración de software entre los componentes de la cadena de bloques (contratos inteligentes, bibliotecas, estructuras de datos) y los componentes fuera de la cadena, como aplicaciones web o móviles, que en conjunto constituyen un sistema DApp completo. En esta metodología se hace hincapié en el uso de las prácticas ágiles, porque estas son adecuadas para desarrollar sistemas cuyos requisitos no se comprenden completamente desde el principio, o tienden a cambiar durante el desarrollo, como es el caso de la mayoría de las aplicaciones basadas en blockchain. [15]

ABCDE está basado en Scrum y, por lo tanto, es iterativo e incremental. Desde Scrum, algunas de las cosas rescatadas de Scrum son: la recopilación de requisitos con historias de usuarios, el enfoque iterativo-incremental, los roles clave y las reuniones. La principal diferencia con Scrum es la separación de las actividades de desarrollo en dos flujos, uno para contratos inteligentes y el otro para software de aplicación que interactúa con la cadena de bloques, cada uno realizado de forma iterativa, con actividades de integración cada 2-3 iteraciones. [15]

ABCDE hace explícitas las actividades que deben realizarse para diseñar, desarrollar, probar e integrar contratos inteligentes y software de aplicación, y documenta los contratos inteligentes mediante diagramas formales para ayudar al desarrollo, la evaluación de la seguridad y el mantenimiento. Un diagrama derivado del diagrama de clases UML ayuda a modelar eficazmente la estructura de datos de los contratos inteligentes, mientras que el intercambio de mensajes entre las entidades del sistema se modela utilizando un diagrama de secuencia UML.

modificado. El método propuesto también tiene actividades específicas para la evaluación de la seguridad y la optimización de los costos de las transacciones, mediante el uso sistemático de patrones y listas de verificación. ABCDE se centra en la cadena de bloques Ethereum y su lenguaje de programación Solidity, pero conserva la generalidad y, con las modificaciones adecuadas, se puede aplicar a cualquier proyecto de software de Blockchain. [15]

4.8 NodeJS

De acuerdo a [16], Node.js es un entorno de ejecución multiplataforma de código abierto que se utiliza para el desarrollo de aplicaciones web del lado del servidor. Las aplicaciones Node.js están escritas en JavaScript y pueden ejecutarse en una amplia variedad de sistemas operativos. Node.js utiliza un modelo de entradas y salidas no bloqueante y dirigido por eventos que hace que las cosas sean rápidas y eficientes, en otras palabras, nos permite ejecutar código en JavaScript en los servidores, y utiliza modelos impulsados por eventos, así mismo, ejecuta las operaciones de entrada y salida en paralelo y no espera o bloquea otras operaciones mientras realiza operaciones. [17]

4.8.1 Express

Según menciona [18], Express es un framework construido sobre NodeJS que proporciona una API simplificada para algunas de las funcionalidades principales de NodeJS.

Puede describirse como una capa de abstracción sobre el módulo HTTP de la API principal de NodeJS que busca simplificar sus APIs y añadir nuevas y útiles características.

Facilita la organización de la funcionalidad con middleware (en lugar de una función monolítica de gestión de solicitudes, se llama a varias funciones de gestión de solicitudes que se encargan de una pequeña parte del trabajo) y enrutamiento; añade convenientes utilidades a

los objetos HTTP de Node.js; facilita la representación de vistas HTML dinámicas y define un estándar de extensibilidad fácil de implementar.

¿Qué añade Express a NodeJS?

A grandes rasgos, Express añade dos grandes características al servidor HTTP de Node.js [19]:

- Añade una serie de comodidades útiles al servidor HTTP de Node.js, abstrayendo una gran parte de su complejidad. Por ejemplo, el envío de un solo archivo JPEG es bastante complejo en Node.js en bruto (tomando en cuenta si se tiene en mente el rendimiento); en cambio Express lo reduce a una sola línea.
- Te permite refactorizar una función monolítica que gestiona las solicitudes en muchas funciones más pequeñas que gestionarán las solicitudes y que manejan sólo partes específicas. Esto es lo vuelve mantenible y más modular.

4.9 Go

En [20] menciona que, el lenguaje de programación Go es un proyecto de código abierto para que los programadores sean más productivos.

Go es expresivo, conciso, limpio y eficiente. Sus mecanismos de concurrencia facilitan la escritura de programas que sacan el máximo partido a las máquinas multinúcleo y en red, mientras que su novedoso sistema de tipos permite la construcción de programas flexibles y modulares. Go compila rápidamente a código máquina, pero tiene la comodidad de la recolección de basura y la potencia de la reflexión en tiempo de ejecución. Es un lenguaje compilado, rápido y de tipado estático que se siente como un lenguaje interpretado de tipado dinámico.

4.10 Flutter

En [21], menciona que Flutter es un kit de desarrollo de software de interfaz gráfica open source creado por Google. Fue mencionado por primera vez en la cumbre de desarrollo de Dart de 2015 con el nombre en clave “Sky” y su primer lanzamiento antes de la versión estable sucedió en los Días de los Desarrolladores de Google de 2017 en Shanghái. El lanzamiento de la versión estable fue el 4 de diciembre del 2018 en el Flutter Live Event.

Es utilizado actualmente en el desarrollo de aplicaciones para Android, iOS, Windows, Mac, Linux, Google Fuchsia y web a partir de un único código base. El proceso utilizado es la compilación del código en el lenguaje Dart a código nativo de cada plataforma.

Este framework escrito en C, C++ y Dart se encuentra bajo la licencia New BSD. Este hecho hace que se apoye mucho en una comunidad cada vez más numerosa y que se está convirtiendo en la habilidad con más expansión entre ingenieros de software en la red profesional LinkedIn.

Los paquetes oficiales y todos aquellos desarrollados por la comunidad, tanto para Flutter como para Dart están disponibles en <https://pub.dev/>. Mientras que la página oficial con toda la documentación es <https://flutter.dev/>.

4.11 Trabajos relacionados

Titulo	Resumen	Ref.
Implementación de un sistema de votación electrónica basado en la tecnología	La votación electrónica ha buscado mejorar los procesos de elección popular hechas a papel, tratando puntos claves como aumentando la eficiencia y reduciendo los errores que se puedan	[22]

Blockchain para las elecciones estudiantiles en la Universidad de Córdoba.	presentar en este proceso complejo por la cantidad de personas que participan en estos eventos, claves para el desarrollo de una sociedad. Blockchain es una tecnología en auge, posee características que podrían aportar al desarrollo de sistemas capaces de dar confianza a las personas que intervienen en una elección de un cargo popular. Con este trabajo, se hace un esfuerzo para implementar la tecnología blockchain en un sistema de votación electrónica, que esté orientado a la transparencia. Se desarrolló una interfaz web para la interacción del administrador de la elección (para registrar los candidatos y votantes) y los votantes (que efectuarán el voto) con el sistema, implementando Hyperledger Fabric, que provee la red blockchain y una base de datos de estado de la red (CouchDB).	
Modelo y sistema de votación electrónica aplicando la tecnología de cadena de bloques.	Durante los últimos años se han implementado diferentes mecanismos para asegurar los requerimientos necesarios de un proceso	[23]

	<p>electoral: libertad, equidad, franqueza, secreto y democracia. Existen procesos electorales tradicionales de votación física y procesos de votación electrónica que utilizan herramientas tecnológicas. Lamentablemente, los procedimientos aplicados no aseguran el cumplimiento de estos requerimientos en su totalidad, por lo cual la integridad de la información o la lucha contra el fraude se podría ver afectada. Este artículo presenta un modelo de votación electrónica que integra aspectos del modelo tradicional, la tecnología Blockchain y la infraestructura transaccional de la moneda criptográfica Bitcoin, para implementar una votación descentralizada y anónima, asegurando la integridad de los datos ante cualquier posible dificultad que pueda surgir. Así mismo, este artículo presenta una implementación del modelo aplicado a los distintos procesos electorales que Bolivia tiene y un caso de estudio para la evaluación de la implementación del modelo.</p>	
--	---	--

<p>Desarrollo de un sistema de votación electrónica utilizando una tecnología de contabilidad distribuida para el almacenamiento seguro de la información.</p>	<p>Las votaciones son importantes para la democracia de cualquier país, empresa o institución educativa. En la Universidad Técnica de Machala, la Escuela de Informática, conformada por las carreras de Ingeniería de Sistemas y Tecnologías de la Información, anualmente realiza la elección de sus representantes estudiantiles de forma presencial mediante votación en papel, pero actualmente debido al inconveniente que generan las aglomeraciones de personas, requiere de una alternativa segura y confiable de votación electrónica que permita ejercer el derecho al voto de manera online. El principal problema con los sistemas de votación electrónica que guardan la información en bases de datos centralizadas, es que son mucho más vulnerables al fraude que la votación en papel, puesto que, si el ente administrador o un atacante informático manipulara indebidamente los datos, difícilmente sería detectado. Por tal motivo, el objetivo de la presente propuesta es desarrollar un sistema de votación electrónica utilizando una tecnología de contabilidad distribuida (DLT) para el almacenamiento seguro de la información. La ventaja que ofrece este tipo de tecnología es la descentralización, ya que una DLT consta de una</p>	<p>[24]</p>
--	---	-------------

	red de nodos distribuidos que según su configuración de acceso puede ser pública, privada o federada. Los puntos de la red verifican mediante mecanismos de consenso cada transacción realizada y almacenan la información estructurando los datos en una cadena de bloques o un grafo acíclico dirigido (DAG), haciendo uso de métodos criptográficos que garantizan su seguridad e inmutabilidad.	
Implementación de un prototipo de una red descentralizada Blockchain para el voto electrónico en la universidad de Guayaquil.	En la actualidad la seguridad de la información en las redes de datos es sumamente importante al momento de proteger la integridad de los mismos, la diversidad de propuestas para resguardar nuestra información ha logrado significativos avances tecnológicos en cuanto a mantener los datos protegidos y fuera del alcance de terceros. Estamos tan habituados a usar las redes centralizadas que para nosotros es inconcebibles intercambiar información en internet sin que estos datos pasen por los sistemas centrales de las grandes empresas que en teoría podrían vender, borrar e incluso modificar la información a su antojo. Aparece una tecnología llamada Blockchain que quiere decir cadena de bloques que hace que el sistema informático sea seguro. Esta tecnología permite realizar las transacciones sin intermediarios, es	[25]

	<p>decir, de una manera descentralizada y es precisamente esto lo que le da seguridad. Blockchain permite un gran número de posibilidades para su implementación y una de estas es el voto electrónico. Actualmente en algunos países se ha aplicado el voto electrónico sin embargo como sabemos la gran debilidad de un sistema informático es que es hackeable, la tecnología Blockchain es capaz de solucionar estos problemas haciendo que las identidades de los votantes estén protegidas y los votos no puedan ser manipulados.</p>	
--	---	--

5 MATERIALES Y MÉTODOS

De acuerdo con el Reglamento de Régimen Académico que rige a las Instituciones de Educación Superior de Ecuador, en el artículo 21, numeral 3, se estipula que un Trabajo de Titulación (TT) se basará en procesos de investigación e intervención. Por otro lado, todo TT deberá consistir en una propuesta innovadora que contenga como mínimo una investigación exploratoria y diagnóstica, además, de acuerdo con el artículo 72 del mismo reglamento, la investigación a nivel de grado es de carácter exploratorio y descriptivo.

5.1 Contexto

El proyecto de Titulación se desarrolló en la Universidad Nacional de Loja, en la Facultad de Energía, Carrera de Ingeniería en Sistemas; la experimentación se llevó a cabo en la Dirección de Tecnología de la Información de la Dirección de Tecnologías de la Información.

5.2 Procesos

Para alcanzar el objetivo general del presente proyecto de investigación se usó el siguiente proceso para cada uno de los objetivos específicos, tal como se muestra en la Figura 6, para una mejor apreciación puede encontrar el diagrama en el **Anexo 1**. Diagrama de procesos.

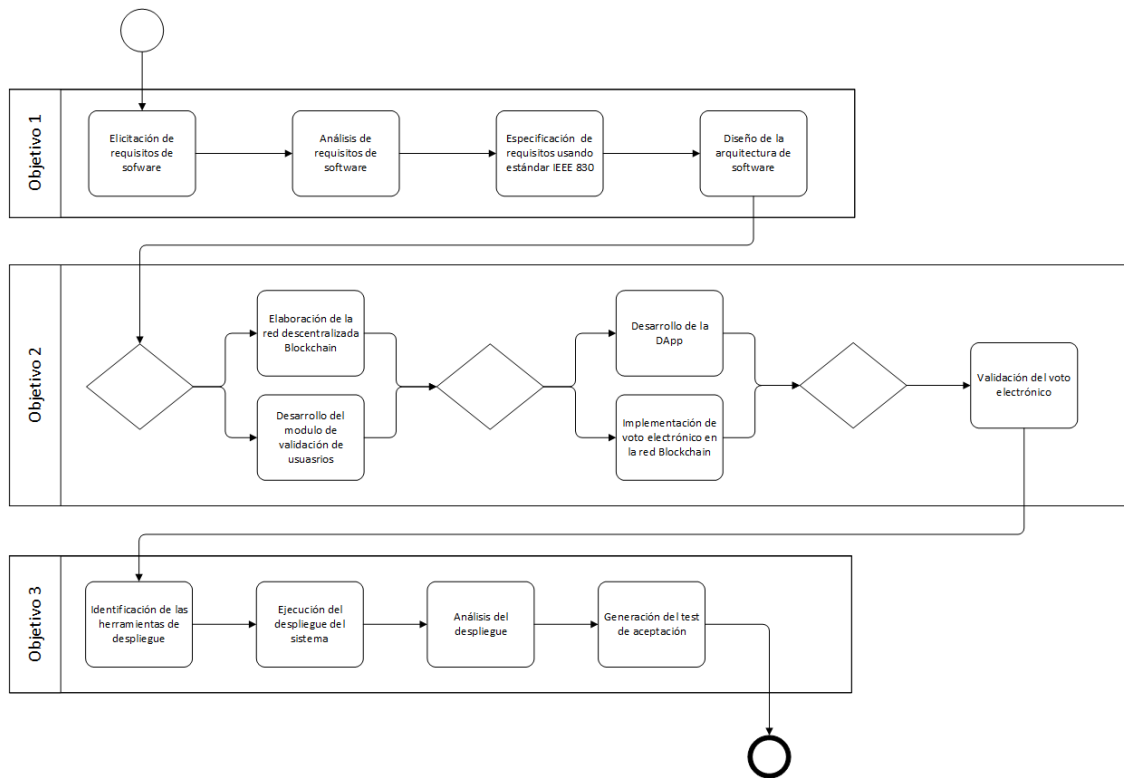


Figura 6. Diagrama de flujo de procesos

5.3 Recursos

Para dar respuesta a la pregunta de investigación y cumplir los objetivos planteados se usarán los siguientes recursos:

5.3.1 Científicos

- **Estudios de caso:** esta técnica permitió identificar casos donde se ha implementado la tecnología blockchain en los sistemas de voto electrónico en el Ecuador y Latinoamérica, véase sección **Trabajos relacionados**.
- **Método científico:** este método es la base para el desarrollo del presente TT, desde la presentación de la propuesta de TT, hasta la culminación del mismo.
- **Análisis estático:** este método permite realizar exámenes minuciosos a la estructura del producto con la finalidad de descubrir errores.

5.3.2 Técnicos

- **Herramientas colaborativas:** se utilizaron herramientas colaborativas para la comunicación y trabajo en equipo tales como: Draw.io para la elaboración de diagramas, Zoom, para las reuniones entre los participantes del presente TT y Drive como servicio de almacenamiento y herramienta colaborativa para documentos.
- **Metodología ABCDE:** este método considera la integración de software entre los componentes de la cadena de bloques (contratos inteligentes, bibliotecas, estructura de datos) y los componentes fuera de la cadena, como aplicaciones web o móviles, que en conjunto constituyen un sistema Dapp completo. [26]
- **Encuestas:** este método se lo utilizó con la finalidad de obtener conocimiento referente a la aceptación del presente TT.
- **Reuniones:** Esta técnica permite, a través de las personas vinculadas con el TT, revisar los diferentes avances en cada una de las fases, siempre generando una retroalimentación.

5.4 Participantes

El proyecto de titulación fue ejecutado por Jhon Alexander Carrión Piedra y Luis Xavier Paredes Cuenca, estudiantes de la Carrera de Ingeniería en Sistemas, con la dirección técnica del Ingeniero Cristian Ramiro Narváez Guillen, docente de la Universidad Nacional de Loja.

6 RESULTADOS

En esta sección se detallan los resultados obtenidos del desarrollo del TT, para lo cual se planteó tres objetivos, con sus respectivas actividades y tareas, mismos que se llevaron a cabo siguiendo la metodología de desarrollo ABCDE.

6.1 Objetivo 1: Definir la arquitectura del sistema de voto electrónico usando la ingeniería de requisitos.

Para el desarrollo de este objetivo, intervinieron las fases de la 1 a la 4 de la metodología de software ABCDE para el TT, las cuales se detallan a continuación.

6.1.1 Fase 1: Objetivo del sistema

Este se relaciona con el objetivo general del TT que es “Implementar un sistema de voto electrónico utilizando tecnología Blockchain que permita registrar votos y realizar el escrutinio de los mismos en el proceso de votación de la UNL”.

6.1.2 Fase 2: Identificar los actores

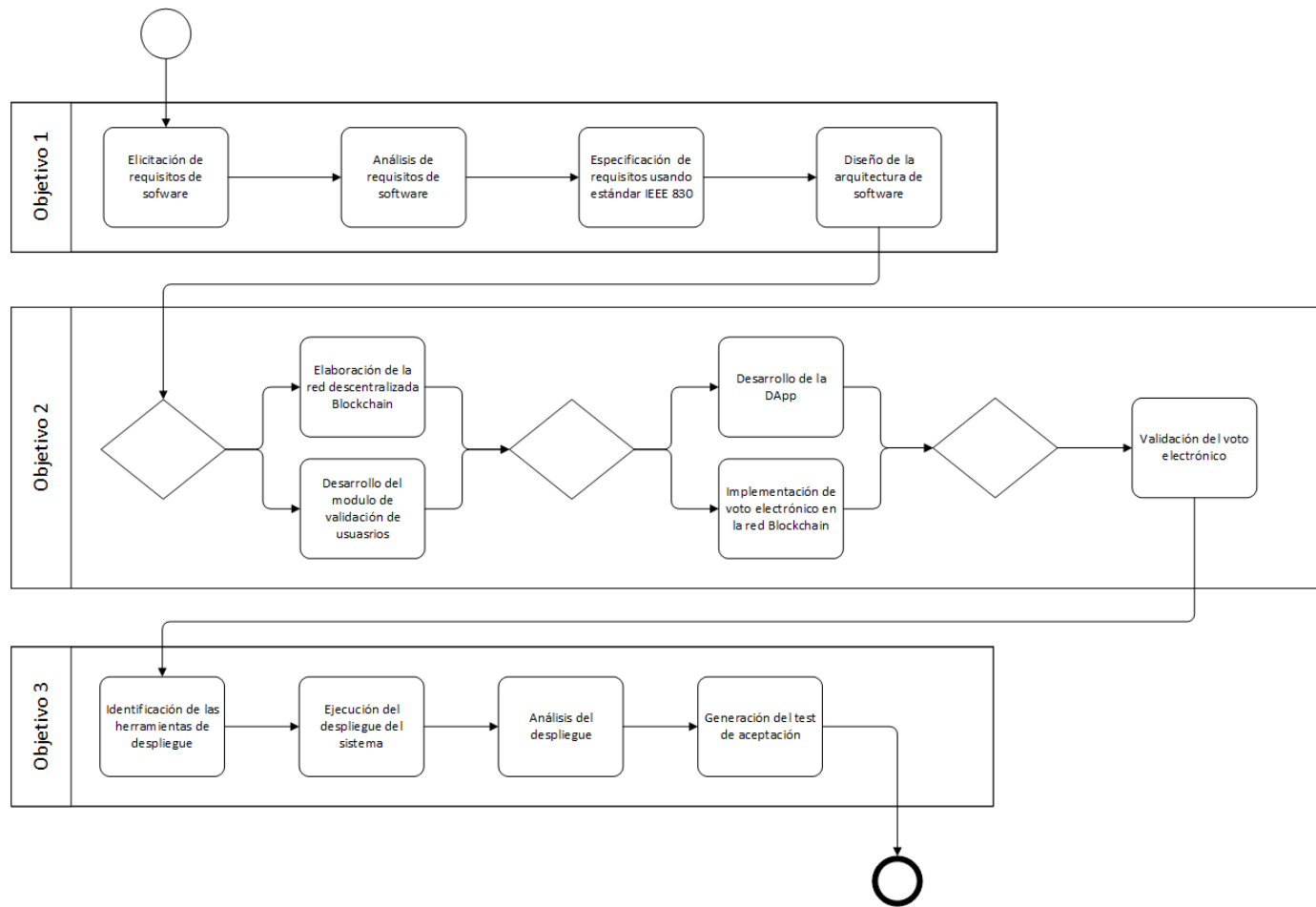
Para el sistema a desarrollar, se identificaron los siguientes actores:

- **Administrador:** El usuario encargado de administrar el sistema e-voting.
- **Usuario:** El usuario promedio que puede visualizar los resultados de una votación en proceso o alguna votación antigua en el sistema e-voting.
- **Votante:** El usuario que puede participar en alguna votación que se esté llevando a cabo.

- **SC-Users:** Contrato(s) inteligente(s) en la blockchain de Hyperledger Fabric que tiene la funcionalidad de registrar y validar usuarios.
- **SC-Votes:** Contrato(s) inteligente(s) en la blockchain de Hyperledger Fabric que tiene la funcionalidad de registrar y validar votos.

6.1.3 Fase 3: Historias de usuario

En esta fase se detallan los requisitos del sistema, tanto los funcionales como los no funcionales (extraídos del **Anexo 1**. Diagrama de procesos



Anexo 2. Especificación de requisitos de software), siguiendo el estándar IEEE 830, las historias de usuario y el diagrama de casos de uso.

6.1.3.1 Requisitos funcionales

En la TABLA I se presenta los requisitos funcionales del sistema e-voting basado en blockchain.

TABLA I
REQUISITOS FUNCIONALES

Código	Requisito	Descripción	Prioridad
RF01	Iniciar Sesión	Para poder hacer uso del sistema, el administrador y el votante deben iniciar sesión con usuario y contraseña, además se debe validar si el usuario que está tratando de ingresar es un usuario que está registrado en la Blockchain.	ALTA
RF02	Gestión de Votantes	El administrador podrá crear votantes, actualizar sus datos y visualizar su información.	ALTA
RF03	Gestión de Partidos	El administrador puede crear partidos, actualizar sus datos y ver el listado. Además, puede agregar candidatos a estos partidos; los candidatos pueden ser cualquier votante registrado. Los partidos pertenecen a un periodo de elecciones específico.	ALTA
RF04	Gestión de Elecciones	El administrador puede crear elecciones, editar sus datos y ver el listado de elecciones registradas y añadir el listado de votantes habilitados para esta elección.	ALTA
RF05	Consulta de Resultados	Cualquier usuario con o sin cuenta dentro del sistema puede visualizar los resultados de las elecciones, en curso o las pasadas.	ALTA
RF06	Realizar voto	El votante debe tener su sesión abierta para poder visualizar las elecciones a las que tiene permitido sufragar, dentro de cada elección podrá visualizar los partidos y sus candidatos más la opción de otorgar su voto a uno de ellos. al registrar el voto se debe confirmar la decisión, con esto el sistema registra la elección	ALTA

		del votante al partido y envía este dato a registrar en la Blockchain.	
--	--	--	--

6.1.3.2 Requisitos no funcionales

En la TABLA II se presentan los requisitos no funcionales del sistema e-voting basado en blockchain.

TABLA II
REQUISITOS NO FUNCIONALES

Código	Requisito	Descripción	Prioridad
RNF01	Rendimiento	El sistema debe proporcionar un tiempo de respuesta aceptable aproximadamente entre 2 a 7 segundos. La transacción tarda de 2 a 5 segundos en un ambiente simulado de red blockchain y de 15 a 5 minutos en un ambiente real.	ALTA
RNF02	Usabilidad	El sistema de software debe proporcionar una interfaz amigable e intuitiva, haciendo que el proceso sea comprensible y fácil de llevar a cabo. Además, debe permitir ser utilizado en cualquier navegador web.	ALTA
RNF03	Fiabilidad	El sistema de software debe permitir la disponibilidad las 24 horas del día y los 7 días de la semana, y en caso de que el módulo de software presente algún error, se debe recuperar en el menor tiempo posible. El sistema de software debe permitir recuperar los datos que se vean afectados en el caso de alguna falla en el módulo de software respecto al tiempo y esfuerzo que este genere.	ALTA
RNF04	Seguridad	El sistema de software debe garantizar disminuir las vulnerabilidades de ataques de fuerza bruta.	ALTA

		Garantizar la seguridad del módulo de software con respecto a la información y datos que se manejan tales sean documentos o archivos.	
--	--	---	--

6.1.3.3 Historias de usuario

En la tabla se muestra el resumen de las historias de usuario del sistema de e-voting basado en blockchain. El desarrollo completo de las historias de usuario se encuentra en **Anexo 3. Historias de Usuario**.

6.1.3.4 Casos de uso

En la Figura 7 se muestra los diferentes actores con su respectivo caso de uso.

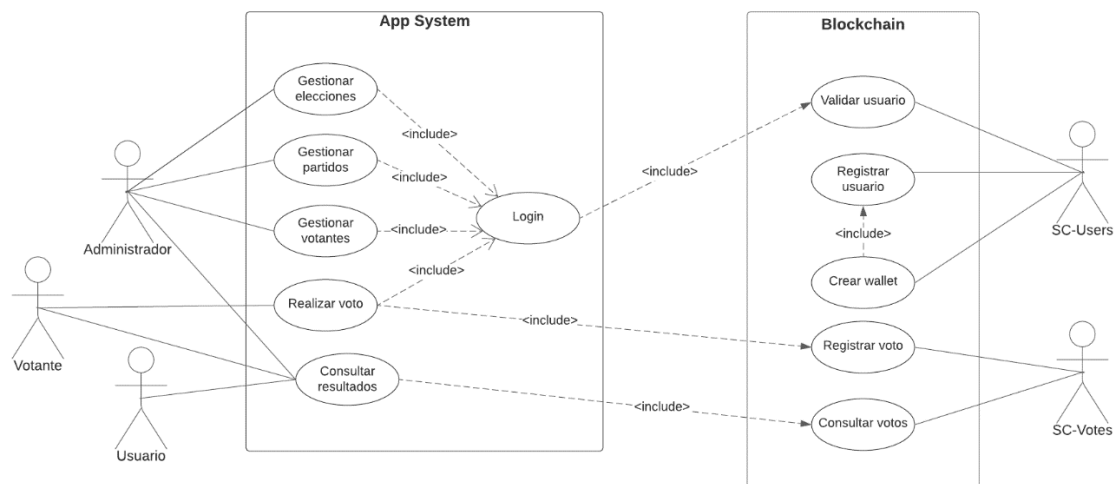


Figura 7. Diagrama de casos de uso

6.1.4 Fase 4: Dividir el sistema en dos subsistemas

El sistema de e-voting se procede a dividir de la siguiente manera:

- Subsistema de los contratos inteligentes que se ejecutan en la blockchain (aquí intervienen: Fase 5: Diseño del subsistema de contratos inteligentes y Fase 6: Codificación y prueba del subsistema de contratos inteligentes de la metodología ABCDE).
- Subsistema de aplicaciones, que es el sistema externo que interactúa con la blockchain (aquí intervienen: **¡Error! No se encuentra el origen de la referencia.** y **¡Error! No se encuentra el origen de la referencia.** de la metodología ABCDE).

En base a estos dos subsistemas, se procede a determinar los diagramas de arquitectura respectivos que se detallan a continuación.

6.1.4.1 Diagramas de arquitectura del sistema e-voting

En esta sección se plantea el diagrama de arquitectura del sistema e-voting; en la Figura 8 se muestra la arquitectura general del sistema, donde se puede observar los subsistemas de aplicaciones y blockchain, el subsistema de aplicaciones, a su vez se subdivide en la parte del cliente y la parte del servidor REST-API, para una mejor apreciación de la arquitectura propuesta puede revisar el **Anexo 4**. Arquitectura del sistema de e-voting.

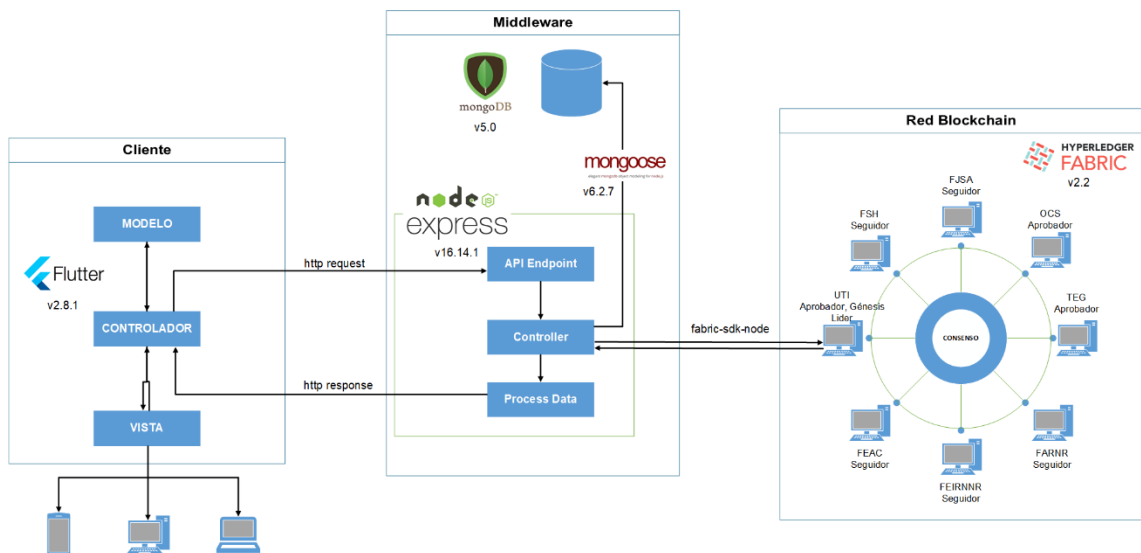


Figura 8. Arquitectura del sistema e-voting

En la arquitectura propuesta, por medio del medio del Middleware, desarrollado con NodeJS, la aplicación cliente (Flutter) se comunica con la red blockchain, esto dado que a la fecha del desarrollo del presente TT, no existe una forma de comunicación directa entre Flutter y Hyperledger Fabric. Es por ello que por medio del SDK fabric-sdk-node se establece la comunicación entre la red descentralizada y el middleware, y este a su vez, se comunica con la aplicación cliente por haciendo uso de la librería http-response.

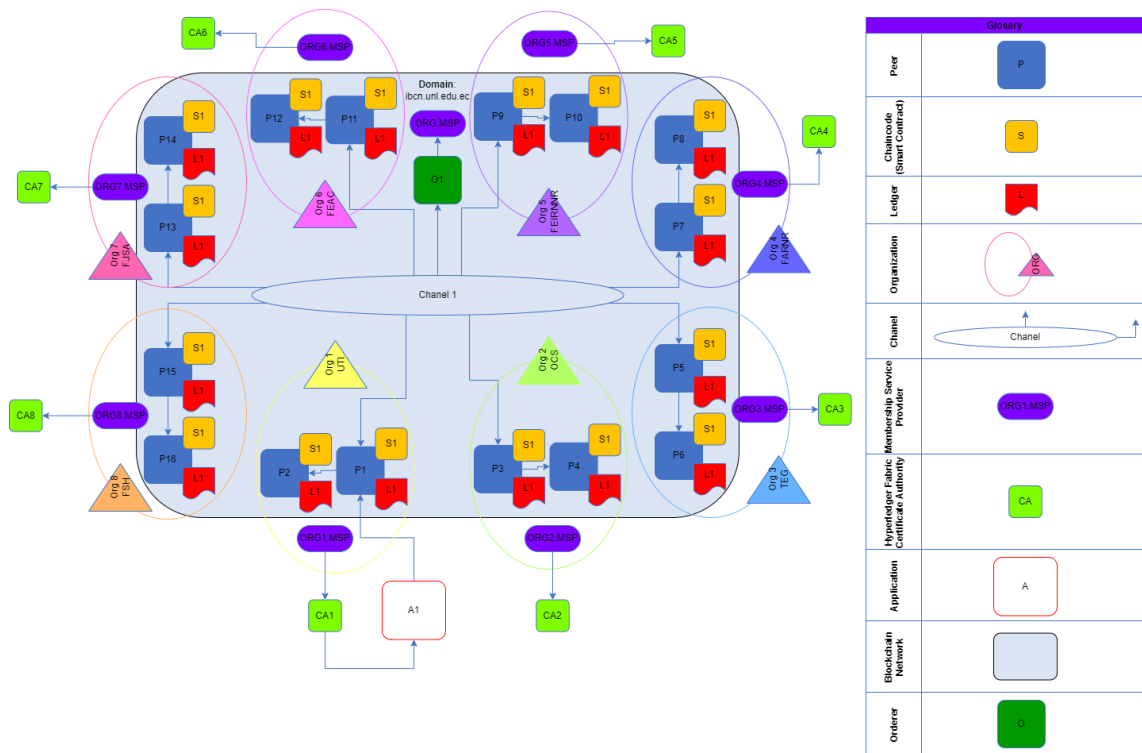


Figura 9. Arquitectura de la red de Hyperledger Fabric

En la Figura 10 se muestran los componentes peers de la red blockchain de una forma más organizada y fácil de comprender. Este diagrama se obtiene a partir de la arquitectura de la red de Hyperledger Fabric mostrado en la Figura 9, mismo que se utilizará para realizar el despliegue de la red.



Figura 10. Componentes Peers de la red blockchain

6.2 Objetivo 2: Desarrollar el sistema de voto electrónico utilizando la arquitectura anteriormente definida.

En los resultados de este objetivo hay que considerar que en la Fase 4: Dividir el sistema en dos subsistemas de la metodología ABCDE, dentro del objetivo 1, se dividió el módulo de software de la siguiente manera:

- Subsistema de los contratos inteligentes que se ejecutan en la blockchain (aquí intervienen las fases 5 y 6 de la metodología ABCDE).
- Subsistema de aplicaciones, que es el sistema externo que interactúa con la blockchain (aquí intervienen las fases 7 y 8 de la metodología ABCDE).

6.2.1 Construir la red descentralizada de Blockchain.

Previo a continuar con la siguiente fase de la metodología ABCDE, es necesario el desarrollo de la red descentralizada de Hyperledger Fabric. Para ello se realizó lo siguiente:

6.2.1.1 Prerrequisitos

Para poder realizar las configuraciones necesarias para que la red Blockchain, de Hyperledger Fabric, esté lista para poder crear canales o ejecutar chaincodes, es necesario tener previamente instalado y configurado lo siguiente:

- Git
- Docker
- Docker Compose
- Golang
- Hyperledger Fabric.

6.2.1.2 Generación del material criptográfico

En este paso se crea el material criptográfico, todos los certificados, componentes, llaves privadas. Para esto se necesita la librería cryptogen, de Hyperledger Fabric, la que va a necesitar como parámetro de entrada el archivo `crypto-config.yaml`. En archivo se definen los siguientes componentes de la red:

- Orderer
- Dominio de la red
- Organizaciones
- Peers
- Usuarios

Un fragmento del código fuente del archivo se visualiza en la Figura 11 .


```

! crypto-config.yaml
1  # Define el orderer para la red
2  OrdererOrgs:
3      - Name: Orderer # Nombre que se le asigna al orderer
4        Domain: ibcn.unl.edu.ec # Dominio de la red
5        EnableNodeOUs: true
6        Specs:
7            - Hostname: orderer
8              SANS:
9                - localhost
10 # Define las organizaciones que tendrá la red
11 PeerOrgs:
12     # Se define la organización DTI
13     - Name: DTI # Nombre de la organización
14       Domain: org1.ibcn.unl.edu.ec # Dominio de la organización
15       EnableNodeOUs: true
16       Template:
17           Count: 2 # Cantidad de nodos que se van a crear
18           SANS:
19               - localhost
20       Users:
21           Count: 1 # Cantidad de usuarios que se van a crear
22     # Se define la organización OCS
23     - Name: OCS
24       Domain: org2.ibcn.unl.edu.ec
25       EnableNodeOUs: true
26       Template:
27           Count: 2
28           SANS:
29               - localhost
30       Users:
31           Count: 1

```

Figura 11. Fragmento código fuente archivo crypto-config.yaml

Al ejecutar el comando `cryptogen generate --config=./crypto-config.yaml`, se genera el directorio, en el cual se encuentran dos carpetas, una con el material criptográfico del orderer de la red y la segunda con el material criptográfico de cada organización.

Dentro del material criptográfico se encuentra los certificados y llaves para CA, MSP, Orderers, TLS y Users, estos certificados (x509) se utilizan para identificar la identidad de quienes participan en la red. Si se desea validar estos certificados generados por cryptogen, se lo puede hacer en el siguiente sitio <https://www.dondominio.com/products/ssl/tools/ssl-checker/>, donde se especifica toda

la información referente a cada certificado. Los certificados contienen la información de las llaves públicas, mientras que las llaves privadas se crean de forma autónoma.

6.2.1.3 Generación de las configuraciones para el bloque génesis

Luego de haber creado el material criptográfico, se debe crear el bloque Genesis, las transacciones de configuraciones, que son archivos de información de cómo queda el canal, información de los AnchorPeers. Todo esto se lo especifica en el archivo configtx.yaml, en la Figura 12 se muestra un fracción del código fuente, dado que el archivo es muy extenso.

```
! configtx.yaml
1 Organizations:
2   - &OrdererOrg
3     Name: OrdererOrg
4     ID: OrdererMSP
5     MSPDir: crypto-config/ordererOrganizations/ibcn.unl.edu.ec/msp
6     # Políticas por defecto
7     Policies:
8       Readers:
9         Type: Signature
10        Rule: "OR('OrdererMSP.member')"
11      Writers:
12        Type: Signature
13        Rule: "OR('OrdererMSP.member')"
14      Admins:
15        Type: Signature
16        Rule: "OR('OrdererMSP.admin')"
17      OrdererEndpoints:
18        - orderer.ibcn.unl.edu.ec:7050
19   - &DTI
20     Name: DTIMSP
21     ID: Org1MSP
22     MSPDir: crypto-config/peerOrganizations/org1.ibcn.unl.edu.ec/msp
23     Policies:
24       Readers:
25         Type: Signature
26         Rule: "OR('Org1MSP.admin', 'Org1MSP.peer', 'Org1MSP.client')"
27       Writers:
28         Type: Signature
29         Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"
30       Admins:
31         Type: Signature
32         Rule: "OR('Org1MSP.admin')"
33       Endorsement:
34         Type: Signature
35         Rule: "OR('Org1MSP.peer')"
36     # Se define el AnchorPeer para la organización
37     AnchorPeers:
38       - Host: peer0.org1.ibcn.unl.edu.ec
39       Port: 7051
```

Figura 12. Fragmento del código fuente del archivo configtx.yaml

En la Figura 12 se puede observar que se establece atributos que tendrán de las organizaciones que conforman el canal. Además, se define las capacidades del canal, las políticas de la aplicación que se ejecutará en el canal, las configuraciones del servicio de ordenamiento, los perfiles y políticas del canal.

Configurado el archivo, se utiliza el comando `configtxgen`, se crea el bloque Genesis, el canal y los AnchorPeers de las organizaciones. Para almacenar las configuraciones se crea una carpeta denominada 'channel-artifacts'. El comando completo es el siguiente:

```
mkdir channel-artifacts
```

```
configtxgen -profile ThreeOrgsOrdererGenesis -channelID system-channel  
-outputBlock ./channel-artifacts/genesis.block
```

6.2.1.4 Generación de las configuraciones del canal

Luego de crear las configuraciones de gel bloque génesis, para crear la configuración del canal, se utiliza el mismo archivo, se modifica el comando anterior y queda como se muestra a continuación:

```
configtxgen -profile ThreeOrgsChannel -outputCreateChannelTx  
./channel-artifacts/channel.tx -channelID evoting
```

6.2.1.5 Generación de las configuraciones de los AnchorPeers

Para crear los archivos de configuración de los AnchorPeers, se lo hace de la misma forma, los comandos para cada organización quedarían como se muestra a continuación:

```

# AnchorPeer Org1
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -
channelID evoting -asOrg DTIMSP
# AnchorPeer Org2
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -
channelID evoting -asOrg OCSMSP
# AnchorPeer Org3
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org3MSPanchors.tx -
channelID evoting -asOrg TEGMSP
# AnchorPeer Org4
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org4MSPanchors.tx -
channelID evoting -asOrg FARNRMSP
# AnchorPeer Org5
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org5MSPanchors.tx -
channelID evoting -asOrg FEIRNRMSP
# AnchorPeer Org6
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org6MSPanchors.tx -
channelID evoting -asOrg FEACMSP
# AnchorPeer Org7
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org7MSPanchors.tx -
channelID evoting -asOrg FJSAMSP
# AnchorPeer Org8
configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org8MSPanchors.tx -
channelID evoting -asOrg FSHMSP

```

Figura 13. Comandos para la creación archivos configuración de AnchorPeers de cada organización.

6.2.1.6 Creación del contenedor Portainer

Para el despliegue, se utiliza la herramienta denominada Portainer para administrar las imágenes Docker, para ello se crea un volumen para la herramienta y se crea y ejecuta el contenedor para la herramienta, esto se lo hace con los siguientes comandos:

```
docker volume create portainer_data
```

```

docker      run      -d      -p      8000:8000      -p      9000:9000      -v
/var/run/docker.sock:/var/run/docker.sock      -v      portainer_data:/data
portainer/portainer

```

Luego se debe configurar el usuario administrador para ello se ingresa a la dirección <http://localhost:9000/>.

6.2.1.7 Crear y ejecutar contenedores

Primero se crean variables las cuales facilitaran el despliegue de la red blockchain. Las variables se muestran en la Figura 14.

```

export CHANNEL_NAME=evoting #Define el nombre del canal
export VERBOSE=false #Define el estado del VERBOSE
export FABRIC_CFG_PATH=$PWD #Define el directorio de las configuraciones de Fabric

```

Figura 14. Declaración de variables necesarias para ejecutar los contenedores.

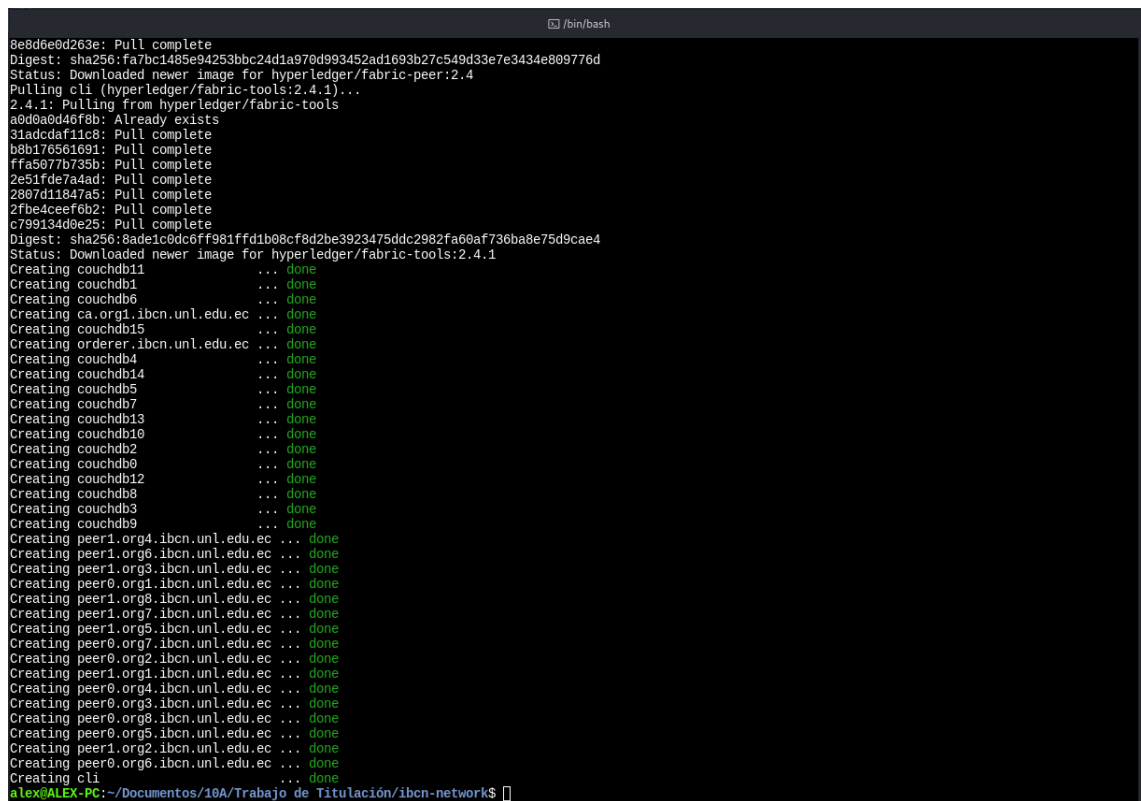
Luego se crea y ejecuta los contenedores de para Cli, CA, Orderer, Peers y CouchDB, para ello se utiliza el siguiente comando:

```
CHANNEL_NAME=$CHANNEL_NAME docker-compose -f docker-compose-cli-couchdb.yaml up -d
```

Figura 15. Comando para crear y ejecutar los contenedores.

Como se puede observar en la Figura 15, se utiliza el archivo `docker-compose-cli-couchdb.yaml`, donde se les asignan las configuraciones creadas en los pasos anteriores a cada peer, según como correspondan.

La salida al ejecutar este comando debe ser la que se muestra en la Figura 16, esto nos indica que todos los contenedores especificados en el archivo con extensión `.yaml` se han creado y ejecutado correctamente.



```
/bin/bash
8e8d6e0d263e: Pull complete
Digest: sha256:fa7bc1485e94253bbc24d1a970d993452ad1693b27c549d33e7e3434e809776d
Status: Downloaded newer image for hyperledger/fabric-peer:2.4
Pulling cli (hyperledger/fabric-tools:2.4.1)...
2.4.1: Pulling from hyperledger/fabric-tools
a0d0a0d46f8b: Already exists
31adcdaf11c8: Pull complete
b8b176561691: Pull complete
ffa5077b735b: Pull complete
2e51fde7a4ad: Pull complete
2807d11847a5: Pull complete
2fbc4ceef6b2: Pull complete
c799134d0e25: Pull complete
Digest: sha256:8ade108dc6ff981ffdb08cf8d2be3923475ddc2982fa60af736ba8e75d9cae4
Status: Downloaded newer image for hyperledger/fabric-tools:2.4.1
Creating couchdb11 ... done
Creating couchdb1 ... done
Creating couchdb6 ... done
Creating ca.org1.ibcn.unl.edu.ec ... done
Creating couchdb15 ... done
Creating orderer.ibcn.unl.edu.ec ... done
Creating couchdb4 ... done
Creating couchdb14 ... done
Creating couchdb5 ... done
Creating couchdb7 ... done
Creating couchdb13 ... done
Creating couchdb10 ... done
Creating couchdb2 ... done
Creating couchdb0 ... done
Creating couchdb12 ... done
Creating couchdb8 ... done
Creating couchdb3 ... done
Creating couchdb9 ... done
Creating peer1.org4.ibcn.unl.edu.ec ... done
Creating peer1.org6.ibcn.unl.edu.ec ... done
Creating peer1.org3.ibcn.unl.edu.ec ... done
Creating peer0.org1.ibcn.unl.edu.ec ... done
Creating peer1.org8.ibcn.unl.edu.ec ... done
Creating peer1.org7.ibcn.unl.edu.ec ... done
Creating peer1.org5.ibcn.unl.edu.ec ... done
Creating peer0.org7.ibcn.unl.edu.ec ... done
Creating peer0.org2.ibcn.unl.edu.ec ... done
Creating peer1.org1.ibcn.unl.edu.ec ... done
Creating peer0.org4.ibcn.unl.edu.ec ... done
Creating peer0.org3.ibcn.unl.edu.ec ... done
Creating peer0.org8.ibcn.unl.edu.ec ... done
Creating peer0.org5.ibcn.unl.edu.ec ... done
Creating peer1.org2.ibcn.unl.edu.ec ... done
Creating peer0.org6.ibcn.unl.edu.ec ... done
Creating cli ... done
alex@ALEX-PC: ~/Documentos/10A/Trabajo de Titulación/ibcn-network$
```

Figura 16. Resultado de la creación y ejecución de los contenedores Docker.

Si se desea verificar, en entorno gráfico, se lo puede hacer ingresando al contenedor que ejecuta Portainer, ahí debe dirigirse al submenú de contenedores del host, donde se podrá observar todos los contenedores con respectiva información actual, tal y como se puede observar en la Figura 17






















































































<input type="checkbox"/> cli	running	    	ibcn-network	hyperledger/fabric-tools:2.4.1	2022-01-26 16:07:45	-	
<input type="checkbox"/> peer1.org6.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7554:7051 7554:7051 7556:7053 7556:7053	
<input type="checkbox"/> peer1.org8.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7754:7051 7754:7051 7756:7053 7756:7053	
<input type="checkbox"/> peer0.org7.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7653:7053 7653:7053 7651:7051 7651:7051	
<input type="checkbox"/> peer0.org3.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7251:7051 7251:7051 7253:7053 7253:7053	
<input type="checkbox"/> peer1.org7.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7654:7051 7654:7051 7656:7053 7656:7053	
<input type="checkbox"/> peer0.org5.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7451:7051 7451:7051 7453:7053 7453:7053	
<input type="checkbox"/> peer0.org1.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7051:7051 7051:7051 7053:7053 7053:7053	
<input type="checkbox"/> peer0.org6.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7551:7051 7551:7051 7553:7053 7553:7053	
<input type="checkbox"/> peer1.org2.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7154:7051 7154:7051 7156:7053 7156:7053	
<input type="checkbox"/> peer0.org8.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7751:7051 7751:7051 7753:7053 7753:7053	
<input type="checkbox"/> peer0.org4.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7351:7051 7351:7051 7353:7053 7353:7053	
<input type="checkbox"/> peer1.org1.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7056:7053 7056:7053 7054:7051 7054:7051	
<input type="checkbox"/> peer0.org2.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7151:7051 7151:7051 7153:7053 7153:7053	
<input type="checkbox"/> peer1.org5.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7454:7051 7454:7051 7456:7053 7456:7053	
<input type="checkbox"/> peer1.org4.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7354:7051 7354:7051 7356:7053 7356:7053	
<input type="checkbox"/> peer1.org3.ibcn.unl.edu.ec	running	    	ibcn-network	hyperledger/fabric-peer:2.4	2022-01-26 16:06:49	7254:7051 7254:7051 7256:7053 7256:7053	

Figura 17. Vista de administración de contenedores del Portainer

6.2.1.8 Creación del canal

Para crear el canal, se debe ingresar a la consola del contenedor *Cli*, que es donde se culminara con las configuraciones de la red, esto se lo puede hacer utilizando la herramienta Portainer.

Una vez dentro de la consola, se utiliza el archivo `channel.tx`, primeramente se asigna el nombre del canal a una variable y se crea el bloque del canal, esto se lo realiza con los siguientes comandos:

```
export CHANNEL_NAME=evoting
peer channel create -o orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/channel.tx --
tls true --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/msp/tlscac
erts/tlsca.ibcn.unl.edu.ec-cert.pem
```

Figura 18. Comandos para crear el canal

Este comando crea un archivo llamando `evoting.block`, el cual tendrá las configuraciones del canal de la red.

6.2.1.9 Añadir organizaciones al canal

Luego de haber ejecutado el comando de la Figura 18, se tiene el canal creado, pero este no tiene asociada ninguna organización, para ello se debe añadir el `peer0` de la organización 1, esto se lo realiza con el comando que se muestra a continuación:

```
peer channel join -b evoting.block
```

Figura 19. Comando para añadir el primer peer al canal.

Para añadir el peer1 de la Org1 y los peers de las organizaciones restantes, se debe añadir la identidad digital de cada peer, como se muestra en el siguiente comando:

[illegible]

Figura 20. Comandos para añadir las organizaciones restantes al canal

6.2.1.10 Configuración de los AnchorPeers

Para esto se utiliza los archivos de configuración de los AnchorPeers de las organizaciones para configurar los en el canal, esto se lo hace solo con un peer de cada organización. Para configurar el AnchorPeer de la primera organización se haciendo uso del siguiente comando:

```
peer channel update -o orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-  
artifacts/Org1MSPanchors.tx --tls --cafile  
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers  
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem
```

Figura 21. Comando para asignar la configuracion AnchorPeer a la organización 1.

Como en el paso anterior, el comando para el peer0 de la Org2 en adelante, se debe añadir la identidad digital, como se muestra a continuación:

```

# Org2
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org
2.ibcn.unl.edu.ec/users/Admin@org2.ibcn.unl.edu.ec/msp
CORE_PEER_ADDRESS=peer0.org2.ibcn.unl.edu.ec:7051 CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.ibcn.unl.edu.ec/peers/peer0.org2.ibcn.unl.edu.ec/tls/ca.crt peer channel update -o
orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org2MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem
# Org3
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org
3.ibcn.unl.edu.ec/users/Admin@org3.ibcn.unl.edu.ec/msp
CORE_PEER_ADDRESS=peer0.org3.ibcn.unl.edu.ec:7051 CORE_PEER_LOCALMSPID="Org3MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org3.ibcn.unl.edu.ec/peers/peer0.org3.ibcn.unl.edu.ec/tls/ca.crt peer channel update -o
orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org3MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem
# Org4
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org
4.ibcn.unl.edu.ec/users/Admin@org4.ibcn.unl.edu.ec/msp
CORE_PEER_ADDRESS=peer0.org4.ibcn.unl.edu.ec:7051 CORE_PEER_LOCALMSPID="Org4MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org4.ibcn.unl.edu.ec/peers/peer0.org4.ibcn.unl.edu.ec/tls/ca.crt peer channel update -o
orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org4MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem
# Org5
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org
5.ibcn.unl.edu.ec/users/Admin@org5.ibcn.unl.edu.ec/msp
CORE_PEER_ADDRESS=peer0.org5.ibcn.unl.edu.ec:7051 CORE_PEER_LOCALMSPID="Org5MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org5.ibcn.unl.edu.ec/peers/peer0.org5.ibcn.unl.edu.ec/tls/ca.crt peer channel update -o
orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org5MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem
# Org6
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org
6.ibcn.unl.edu.ec/users/Admin@org6.ibcn.unl.edu.ec/msp
CORE_PEER_ADDRESS=peer0.org6.ibcn.unl.edu.ec:7051 CORE_PEER_LOCALMSPID="Org6MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org6.ibcn.unl.edu.ec/peers/peer0.org6.ibcn.unl.edu.ec/tls/ca.crt peer channel update -o
orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org6MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem
# Org7
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org
7.ibcn.unl.edu.ec/users/Admin@org7.ibcn.unl.edu.ec/msp
CORE_PEER_ADDRESS=peer0.org7.ibcn.unl.edu.ec:7051 CORE_PEER_LOCALMSPID="Org7MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org7.ibcn.unl.edu.ec/peers/peer0.org7.ibcn.unl.edu.ec/tls/ca.crt peer channel update -o
orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org7MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem
# Org8
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org
8.ibcn.unl.edu.ec/users/Admin@org8.ibcn.unl.edu.ec/msp
CORE_PEER_ADDRESS=peer0.org8.ibcn.unl.edu.ec:7051 CORE_PEER_LOCALMSPID="Org8MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org8.ibcn.unl.edu.ec/peers/peer0.org8.ibcn.unl.edu.ec/tls/ca.crt peer channel update -o
orderer.ibcn.unl.edu.ec:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org8MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/ibcn.unl.edu.ec/orderers
/orderer.ibcn.unl.edu.ec/msp/tlscacerts/tlsca.ibcn.unl.edu.ec-cert.pem

```

Figura 22. Comandos para asignar las configuraciones AnchorPeers a las organizaciones restantes.

Y con esto concluye la configuración de la red Blockchain, es decir, la red se encuentra lista para realizar transacciones y ejecutar contratos inteligentes.

6.2.2 Fase 5: Diseño del subsistema de contratos inteligentes

En esta fase se encuentra el diagrama de clases que tiene la finalidad de diseñar la estructura del módulo de software con sus respectivos atributos, operaciones y las relaciones. Todo esto se observa en la Figura 23. Diagrama de clases del sistema..

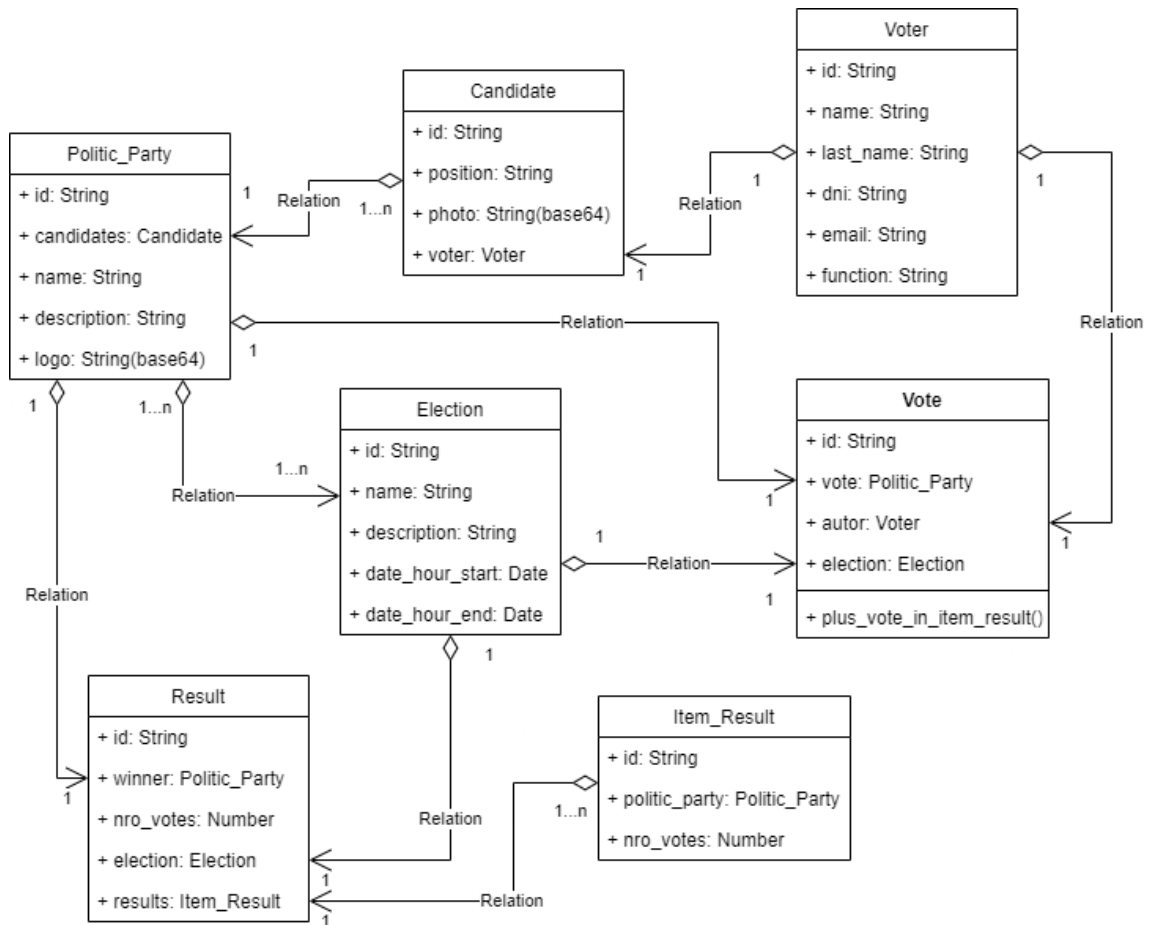


Figura 23. Diagrama de clases del sistema.

6.2.3 Fase 6: Codificación y prueba del subsistema de contratos inteligentes

6.2.3.1 Codificación de los contratos inteligentes

Los chaincodes o contratos inteligentes codificados son **vote.go** y **wallet.go**, el primer chaincode corresponde al Figura 23. Diagrama de clases del sistema., cuyo código fuente se observa en la Figura 24, en el cual se tiene lo siguiente:

- El modelo de dominio de cada estructura(clase), conforme a diagrama de clases descrito en la **¡Error! No se encuentra el origen de la referencia..**
- Las funciones de control de cada clase, las cuales son:
 - **create()**: la cual permite instanciar un objeto de la estructura en mención.
 - **getById()**: esta función permite obtener el objeto de la base de datos utilizando el identificador único de dicho objeto.
 - **getAll()**: esta función nos devuelve una lista de todos los objetos pertenecientes a la estructura correspondiente.
 - **exists()**: con esta función se puede validar mediante el identificador si el objeto se encuentra registrado en la base de datos.
 - **history()**: esta función no devuelve el historial de modificaciones en la base de datos obtenido a través del identificador del objeto en cuestión.

La función principal denominada **main()** la cual se encarga de crear e iniciar el chaincode(contrato inteligente) dentro del sistema.

```

1 package main
2
3 import (
4     "encoding/json"
5     "fmt"
6     "time"
7
8     "evoting/voter"
9     "evoting/political_party"
10    "evoting/election"
11
12    "github.com/hyperledger/fabric-contract-api-go/tree/main/contractapi"
13    "github.com/asaskevich/govalidator"
14 )
15
16 // SmartContract provides functions for the control of votes
17 type SmartContract struct {
18     contractapi.Contract
19 }
20
21 // Vote describes the basic data of a vote
22 type Vote struct {
23     Author Voter `json:"author" valid:"required"`
24     Vote PoliticalParty `json:"vote" valid:"required"`
25     Election Election `json:"election" valid:"required"`
26     CreatedAt time.Time `json:"created_at" valid:"required"`
27 }
28
29 func (s *SmartContract) CreateVote(ctx contractapi.TransactionContextInterface, voteId string, autor Voter, vote PoliticalParty, election Election) error {
30     // Validaciones de negocio
31     exists, err := s.VoteExists(ctx, voteId)
32     if err != nil {
33         return err
34     }
35     if exists {
36         return fmt.Errorf("El voto %s ya existe", voteId)
37     }
38     // Crear instancia de Vote
39     vote := Vote{
40         ID: voteId,
41         Author: autor,
42         Vote: vote,
43         Election: election,
44         CreatedAt: time.Now()
45     }
46     // Validaciones de sintaxis
47     valid, err := govalidator.ValidateStruct(vote)
48     if err != nil {
49         fmt.Printf("Errores de validacion de campos: %s", err.Error())
50         return err
51     }
52     // convertir Vote en arreglo de bytes para enviar al ledger
53     voteAsBytes, err := json.Marshal(vote)
54     if err != nil {
55         fmt.Printf("Marshall error: %s", err.Error())
56         return err
57     }
58     // guardar nuevo votante
59     return ctx.GetStub().PutState(voteId, voteAsBytes)
60 }
61
62 func (s *SmartContract) GetById(ctx contractapi.TransactionContextInterface, voteId string) (*Vote, error) {
63     voteAsBytes, err := ctx.GetStub().GetState(voteId)
64     if err != nil {
65         return nil, fmt.Errorf("Failed to read from world state: %s", err.Error())
66     }
67     if voteAsBytes != nil {
68         return nil, fmt.Errorf("%s does no exist", voteId)
69     }
70
71     vote := new(Vote)
72     err = json.Unmarshal(voteAsBytes, vote)
73     if err != nil {
74         return nil, fmt.Errorf("Unmarshal error: %s", err.Error())
75     }

```

```

    return vote, nil
}

func (s *SmartContract) GetAllVotes(ctx contractapi.TransactionContextInterface) ([]*Vote, error) {
    // La consulta de rango con una cadena vacía para startKey y endKey realiza una consulta abierta de todos los activos en el espacio de nombres del código de cadena.
    resultsIterator, err := ctx.GetStub().GetStateByRange("", "")
    if err != nil {
        return nil, err
    }
    // cerrar comunicación
    defer resultsIterator.Close()
    // crear variable array para contener los registros
    var votes []*Vote
    // iterar mientras haya registros que leer HasNext()
    for resultsIterator.HasNext() {
        // obtener el siguiente registro
        queryResponse, err := resultsIterator.Next()
        if err != nil {
            return nil, err
        }
        // crear variable para instanciar Vote
        vote := new(Vote)
        // convertir Bytes array en JSON y asignar a la variable vote
        err = json.Unmarshal(queryResponse.Value, vote)
        if err != nil {
            return nil, fmt.Errorf("Unmarshal error: %s", err.Error())
        }
        votes = append(votes, vote)
    }
    return votes, nil
}

func (s *SmartContract) VoteHistory(ctx contractapi.TransactionContextInterface, voteId string) ([]*Vote, error) {
    votesIterator, err := ctx.GetStub().GetHistoryForKey(voteId)
    if err != nil {
        return nil, fmt.Errorf("Failed to get from history: %s", err.Error())
    }
    if votesIterator != nil {
        return nil, fmt.Errorf("%s does not exist", voteId)
    }
    defer votesIterator.Close()
    var votes []*Vote
    for votesIterator.HasNext() {
        queryResponse, err := votesIterator.Next()
        if err != nil {
            fmt.Println(err.Error())
            return nil, err
        }
        vote := new(Vote)
        err = json.Unmarshal(queryResponse.Value, vote)
        if err != nil {
            fmt.Println(err.Error())
            return nil, err
        }
        votes = append(votes, vote)
    }
    return votes, nil
}

```

Figura 24. Código fuente del SC vote.go

El chaincode **wallet.go** el código fuente se observa en la f, en el que se tiene las siguientes funciones:

- Las funciones mencionadas en el chaincode **vote.go** que son **create()**, **getWalletBy()**, **getTokenBy()**, **exists()**, **history()**.
- Adicional, la función de registro de voto desde un votante hacia un partido denominada **defray()**, que está relacionada con el **RF06**, la cual recibe como parámetro el id del votante, el id del partido político y el id de la elección. Esta función no retorna un true si se realizó el voto con éxito y false si hubo algún error al momento de registrar el voto.

6.2.3.2 Pruebas de subsistema de contratos inteligentes

Las pruebas unitarias realizadas en este punto permitieron comprobar el correcto funcionamiento del subsistema de contratos inteligentes por unidad de código, asegurando que cada unidad funcione correcta y eficientemente por separado.

[illegible]

Figura 25. Prueba del subsistema de contratos inteligentes desde la terminal

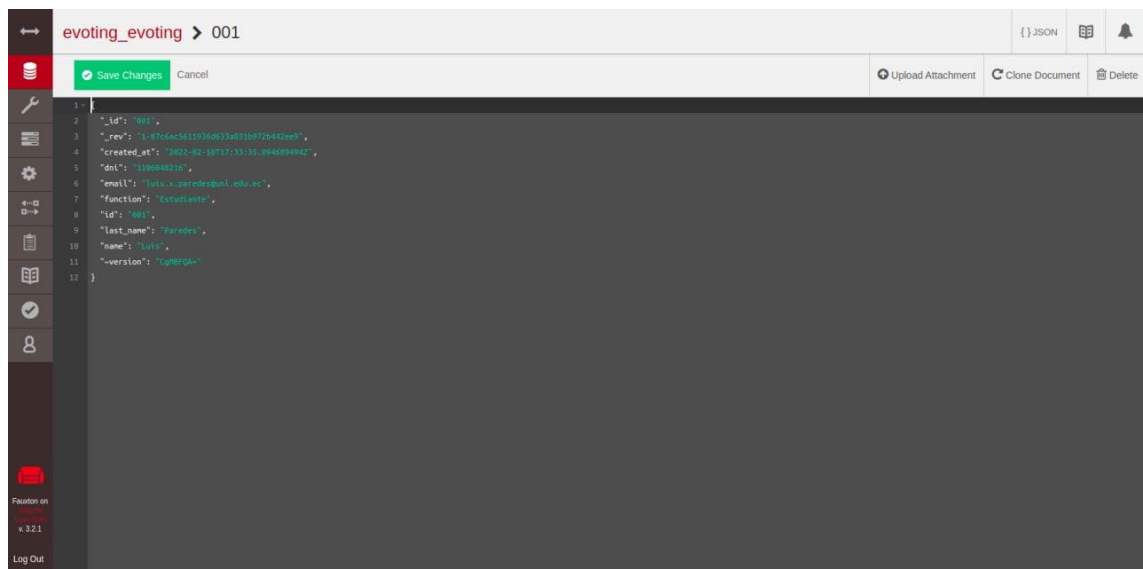


Figura 26. Validación en la base de datos.

Para mayor detalle, observe el

7 DISCUSIÓN

Aquí discusiones

8 CONCLUSIONES

Seguir la metodología de desarrollo de aplicaciones descentralizadas ABCDE permite obtener de forma sencilla la arquitectura del sistema, ya que obliga a separarlo en dos subsistemas y desarrollarlos de forma separada de esta forma cada subsistema tiene su propio análisis, y gracias a esto se puede evidenciar claramente la forma en la que levantara la red blockchain.

Para levantar una red blockchain privada, los proyectos de Hyperledger, son herramientas que brindan una ventaja muy elevada, ya que su implementación está bien documentada y el proceso es relativamente sencillo, las redes creadas con Hyperledger son redes bastante robustas y que permiten la integración de DApps de diferente propósito, sin interferir unas con otras.

Flutter es un framework de código abierto lanzado por Google que permite desarrollar aplicaciones multiplataforma desde una misma base de código, esto permite acelerar el proceso de desarrollo, ya que al construir una sola base de código no se deben desarrollar para cada plataforma específica, además valiéndose de NodeJs la comunicación con la blockchain es relativamente sencilla, gracias a esta combinación se pueden escribir, en menos tiempo, aplicaciones descentralizadas multiplataforma que sean robustas y confiables, ya que su base es la misma.

9 BIBLIOGRAFÍA

- [1] “IBM Food Trust - Blockchain for the world’s food supply | IBM.” <https://www.ibm.com/blockchain/solutions/food-trust> (accessed Mar. 10, 2022).
- [2] “Trü | Origen.” <https://trualimentos.com/origen> (accessed Mar. 10, 2022).
- [3] “Camarón ecuatoriano será pionero en trazabilidad ‘blockchain’ a escala mundial - El Comercio.” <https://www.elcomercio.com/actualidad/camaron-ecuadoriano-trazabilidad-blockchain-tecnologia.html> (accessed Mar. 10, 2022).
- [4] Instituto Interamericano de Derechos Humanos, “Diccionario Electoral,” 2017, Accessed: Aug. 14, 2021. [Online]. Available: www.iidh.ed.cr
- [5] Red de Conocimientos Electorales, “Voto electrónico (E-voting).” https://aceproject.org/ace-es/focus/fo_e-voting/onePage (accessed Aug. 30, 2021).
- [6] J. Valencia, “Contrato inteligentes,” *Revista de Investigación en Tecnologías de la Información*, vol. 7, no. 14, pp. 1–10, Dec. 2019, doi: 10.36825/riti.07.14.001.
- [7] S. Moscoso, D. Suárez, and D. Martin, “Certificación digital de documentos académicos usando Blockchain Formato IEEE,” *Tecnología Investigación y Academia*, vol. 7, no. 2, pp. 21–27, Jan. 2020, [Online]. Available: <https://revistas.udistrital.edu.co/index.php/tia/article/view/12757>
- [8] D. Plaza, “Diseño y desarrollo de diplomas académicos digitales mediante la tecnología blockchain,” Cali, 2019. [Online]. Available: http://vitela.javerianacali.edu.co/bitstream/handle/11522/11222/Diplomas_academicos_digitales_Blockchain.pdf?sequence=1&isAllowed=y
- [9] C. Pastorino, “Blockchain: qué es y cómo funciona esta tecnología,” Sep. 04, 2018. <https://www.welivesecurity.com/la-es/2018/09/04/blockchain-que-es-como-funciona-y-como-se-esta-usando-en-el-mercado/> (accessed Nov. 25, 2021).
- [10] C. Piris, A. Cabellos, and J. Vilanova, “Design and implementation of the Transactional and Communication layer of a Blockchain to secure IP prefixes Autor,” Jan. 2018.

- [11] “Hyperledger Fabric – Hyperledger.” [Online]. Available: <https://www.hyperledger.org/use/fabric>
- [12] W. H. Fabric, “Open, Proven, Enterprise-grade DLT”.
- [13] “¿Qué es Hyperledger Fabric? - España | IBM.” [Online]. Available: <https://www.ibm.com/es-es/topics/hyperledger>
- [14] Blockchain Academy México, “Blockchain 2.0 | Smart Contract,” 2020, [Online]. Available: <https://blockchainacademy.mx/>
- [15] L. Marchesi, M. Marchesi, and R. Tonelli, “ABCDE—agile block chain DApp engineering,” *Blockchain: Research and Applications*, vol. 1, no. 1–2, p. 100002, Dec. 2020, doi: 10.1016/J.BCRA.2020.100002.
- [16] K. Rungta, “Learn NodeJS in 1 Day,” 2016.
- [17] Y. Rajiv, *Developing Turn-Based Multiplayer Games*. Apress, 2018. doi: 10.1007/978-1-4842-3861-5.
- [18] C. Peters, “Building Rich Internet Applications with Node.js and Express.js,” *Rich Internet Applications w/HTML and Javascript*, pp. 15–20, Jun. 2017, Accessed: Nov. 25, 2021. [Online]. Available: www.uni-oldenburg.de/svs
- [19] E. M. Hahn, *Express in Action*. Shelter Island, NY: Manning Publications Co, 2016. Accessed: Nov. 25, 2021. [Online]. Available: <https://hackerstribе.com/wp-content/uploads/2016/04/Node.js-Express-in-Action.pdf>
- [20] “Documentation - The Go Programming Language.” <https://go.dev/doc/> (accessed Mar. 18, 2022).
- [21] Á. Menacho Rodríguez and F. R. José Manuel, “Equation Chapter 1 Section 1 EventYou: Aplicación móvil con Flutter y FlutterFire”.
- [22] G. V. NEIFER, “IMPLEMENTACIÓN DE UN SISTEMA DE VOTACIÓN ELECTRÓNICA BASADO EN LA TECNOLOGÍA BLOCKCHAIN PARA LAS ELECCIONES ESTUDIANTILES EN LA UNIVERSIDAD DE CÓRDOBA,” MONTERÍA, CÓRDOBA, 2019. Accessed: Jun. 21, 2021. [Online]. Available: <https://repositorio.unicordoba.edu.co/bitstream/handle/ucordoba/3496/Garc%c3%adaVilladiegoNeifer.pdf?sequence=1&isAllowed=y>

- [23] L. Gabriel Alejandro, K. V. Sergio, and G. Yanina, "Modelo y sistema de votación electrónica aplicando la tecnología de cadena de bloques. Model and electronic voting system applying Blockchain technology," Cochabamba, Jun. 2019. Accessed: Jun. 21, 2021. [Online]. Available: https://gitlab.com/gabolucuy/Sistema_en_linea.git

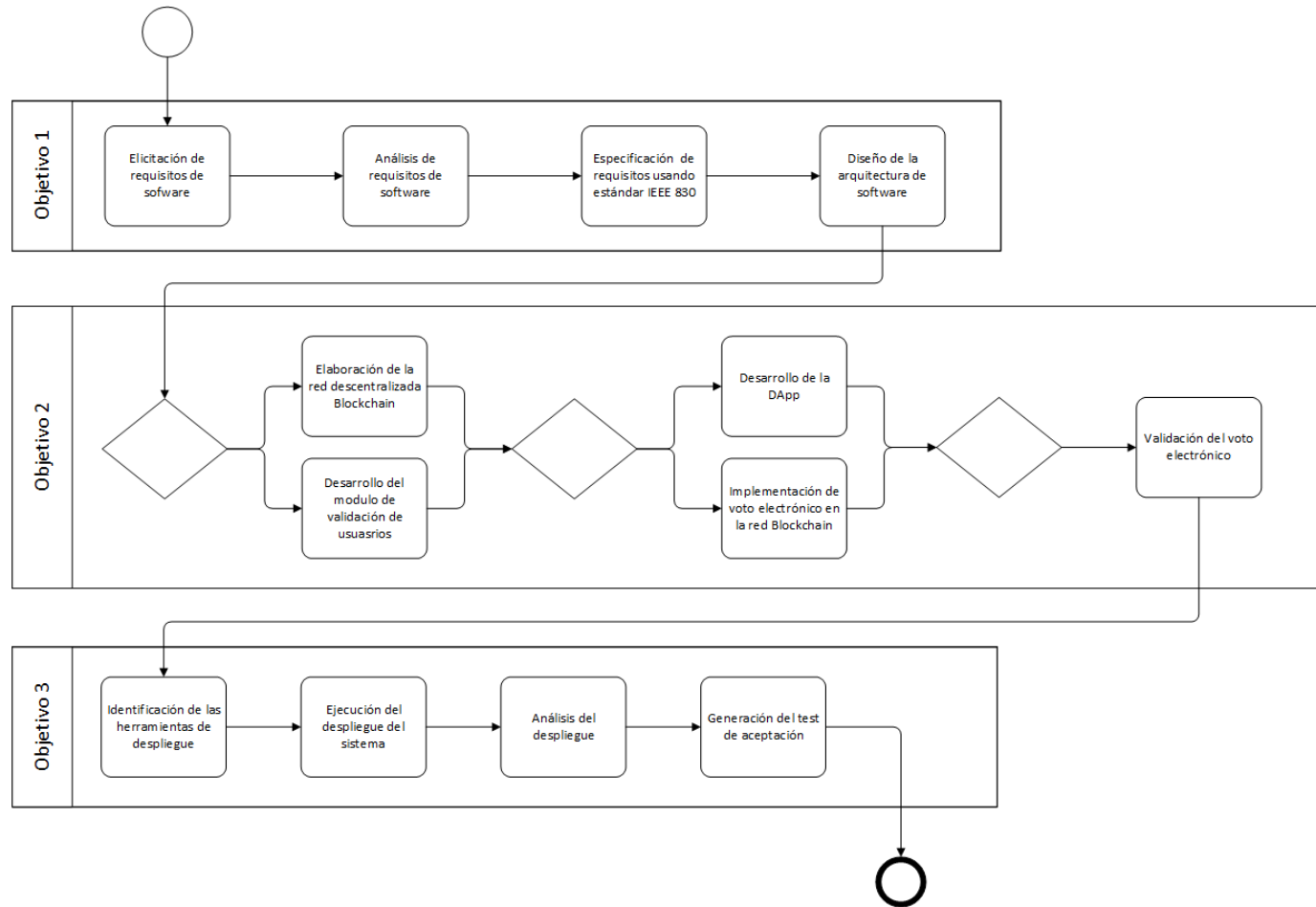
- [24] G. L. ASTUDILLO CRUZ, "DESARROLLO DE UN SISTEMA DE VOTACIÓN ELECTRÓNICA UTILIZANDO UNA TECNOLOGÍA DE CONTABILIDAD DISTRIBUIDA PARA EL ALMACENAMIENTO SEGURO DE LA INFORMACIÓN," MACHALA, 2021.

- [25] V. M. Baldeón Coronel and J. F. Zambrano Hidalgo, "IMPLEMENTACIÓN DE UN PROTOTIPO DE UNA RED DESCENTRALIZADA BLOCKCHAIN PARA EL VOTO ELECTRÓNICO EN LA UNIVERSIDAD DE GUAYAQUIL." Accessed: Jan. 04, 2022. [Online]. Available: <http://repositorio.ug.edu.ec/bitstream/redug/33172/1/B-CINT-PTG-N.343%20Balde%20Coronel%20Valeria%20Mishell%20Zambrano%20Hidalgo%20Joel%20Francisco.pdf>

- [26] L. Marchesi, M. Marchesi, and R. Tonelli, "ABCDE—agile block chain DApp engineering," *Blockchain: Research and Applications*, vol. 1, no. 1–2, p. 100002, Dec. 2020, doi: 10.1016/J.BCRA.2020.100002.

10 ANEXOS

A. Anexo 1. Diagrama de procesos



B. Anexo 2. Especificación de requisitos de software

Especificación de requisitos de software

Proyecto: Implementación de un sistema multiplataforma de voto electrónico basado en Blockchain

Versión: 1.0

Fecha: 13/12/2021

Contenido

1	Introducción	4
1.1	Propósito	4
1.2	Alcance.....	4
1.3	Personal involucrado	4
1.4	Definiciones, acrónimos y abreviaturas	5
1.5	Referencias	5
1.6	Resumen	5
2	Descripción general	6
2.1	Perspectiva del producto	6
2.2	Funcionalidad del producto.....	6
2.3	Características de los usuarios	6
2.4	Restricciones	7
2.5	Suposiciones y dependencias	7
3	Requisitos específicos	8
3.1	Requisitos comunes de las interfaces.....	8
3.1.1	Interfaces de usuario	8
3.1.2	Interfaces de hardware	8
3.1.3	Interfaces de software	8
3.2	Requisitos funcionales.....	8
3.3	Requisitos no funcionales.....	11

Índice de Figuras

Figura 1. Roles y actividades	6
-------------------------------------	---

Índice de Tablas

TABLA I. PERSONAL INVOLUCRADO ESTUDIANTES DE LA CIS	4
TABLA II. PERSONAL INVOLUCRADO DOCENTE DE LA CIS.....	4
TABLA III. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS.....	5
TABLA IV. REFERENCIAS.....	5
TABLA V. CARACTERÍSTICAS USUARIO USUARIO.....	6
TABLA VI. CARACTERÍSTICAS USUARIO VOTANTE	6
TABLA VII. CARACTERÍSTICAS USUARIO ADMINISTRADOR	6
TABLA VIII. CARACTERÍSTICAS USUARIO SC-USERS	7
TABLA IX. CARACTERÍSTICAS USUARIO SC-VOTES.....	7
TABLA X. REQUISITO FUNCIONAL INICIO DE SESIÓN	8
TABLA XI. REQUISITO FUNCIONAL GESTIÓN DE VOTANTES	9
TABLA XII. REQUISITO FUNCIONAL GESTIÓN DE PARTIDOS	9
TABLA XIII. REQUISITO FUNCIONAL GESTIÓN DE ELECCIONES	10
TABLA XIV. REQUISITO FUNCIONAL CONSULTA DE RESULTADOS.....	10
TABLA XV. REQUISITO FUNCIONAL REALIZAR VOTO	11
TABLA XVI. REQUISITO NO FUNCIONAL RENDIMIENTO.....	11
TABLA XVII. REQUISITO NO FUNCIONAL USABILIDAD	12
TABLA XVIII. REQUISITO NO FUNCIONAL FIABILIDAD	12
TABLA XIX. REQUISITO NO FUNCIONAL SEGURIDAD	12

1 Introducción

Este documento es una Especificación de Requisitos de Software (ERS) para el sistema multiplataforma de e-voting basando en blockchain. Esta especificación se ha estructurado basándose en las directrices dadas por el estándar IEEE Práctica Recomendada para Especificación de Requisitos de Software ANSI/IEEE 830, 1998.

1.1 Propósito

El presente documento tiene como propósito detallar las especificaciones funcionales y no funcionales para el desarrollo de un sistema que permitirá realizar el proceso de votación de forma electrónica esto utilizando la tecnología blockchain. Mismo que será implementado, en un ambiente simulado, en la Universidad Nacional de Loja.

1.2 Alcance

Esta especificación de requisitos está dirigida al usuario del sistema, para continuar con el proceso que tiene el sistema multiplataforma de e-voting basado en blockchain que tiene como objetivo el desarrollo e implementación de un sistema de voto electrónico utilizando la tecnología blockchain para el registro de votos.

1.3 Personal involucrado

TABLA III. PERSONAL INVOLUCRADO ESTUDIANTES DE LA CIS

Nombres	<ul style="list-style-type: none">• Jhon Alexander Carrión Piedra• Luis Xavier Paredes Cuenca
Rol	Analistas y Desarrolladores de Software
Categoría profesional	Estudiantes de la CIS
Responsabilidad	Análisis de información, diseño y programación del sistema de software
Información de contacto	<ul style="list-style-type: none">• jhon.carrion@unl.edu.ec• luis.x.paredes@unl.edu.ec

TABLA IV. PERSONAL INVOLUCRADO DOCENTE DE LA CIS

Nombre	Cristian Ramiro Narváez Guillen
Rol	Director del trabajo de titulación
Categoría profesional	Docente de la CIS

Responsabilidad	Supervisar y asesorar en el desarrollo del TT
Información de contacto	cristian.narvaez@unl.edu.ec

1.4 Definiciones, acrónimos y abreviaturas

TABLA V. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

Nombre	Descripción
BC	Blockchain
CIS	Carrera de Ingeniería en Sistemas
DNI	Documento Nacional de Identificación
e-voting	Voto electrónico
ERS	Especificación de Requisitos de Software
RF	Requisito Funcional
RNF	Requisito No Funcional
TT	Trabajo de titulación

1.5 Referencias

TABLA VI. REFERENCIAS

Título del Documento	Referencia
IEEE Std 830-1998	IEEE Recommended Practice for Software Requirements Specifications

1.6 Resumen

Este documento está dividido en tres secciones. En la primera sección se realiza una introducción al mismo y se proporciona una visión general de la especificación de recursos del sistema.

En la segunda sección del documento se realiza una descripción general del sistema, con el fin de conocer las principales funciones que este debe realizar, los datos asociados y los factores restricciones, supuestos y dependencias que afectan al desarrollo, sin entrar en excesivos detalles.

Por último, la tercera sección del documento es aquella en la que se definen detalladamente los requisitos que deben satisfacer el sistema.

2 Descripción general

2.1 Perspectiva del producto

El sistema de software de e-voting será un producto diseñado para dispositivos multiplataforma, lo que permitirá su utilización de forma rápida y eficaz.

2.2 Funcionalidad del producto

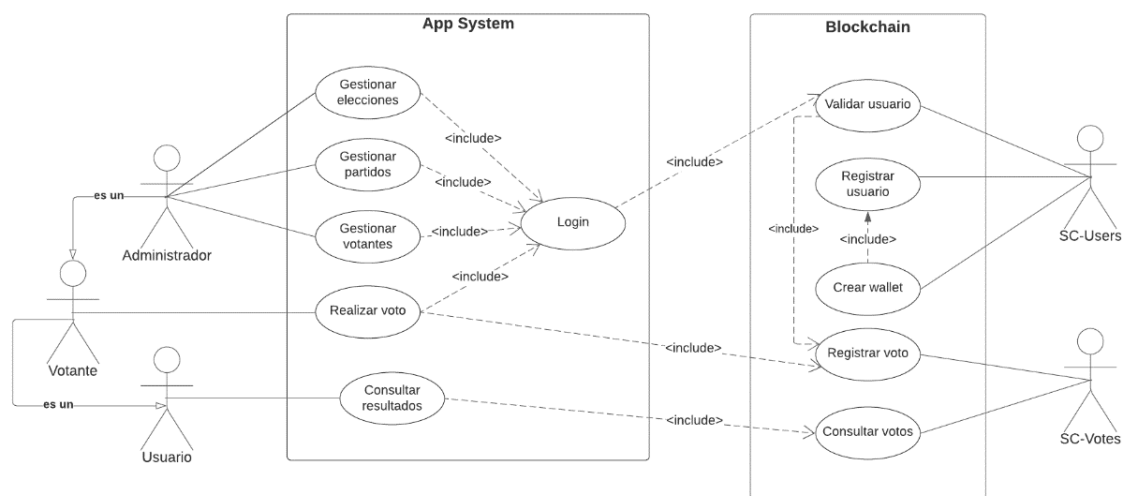


Figura 27. Roles y actividades

2.3 Características de los usuarios

TABLA VII. CARACTERÍSTICAS USUARIO USUARIO

Tipo de usuario	Usuario
Formación	Indistinto
Actividades	Consultar Resultados

TABLA VIII. CARACTERÍSTICAS USUARIO VOTANTE

Tipo de usuario	Votante
Formación	Estudiante, Docente, Administrativo, Personal de apoyo
Actividades	Consultar Resultados, Realizar voto, Login

TABLA IX. CARACTERÍSTICAS USUARIO ADMINISTRADOR

Tipo de usuario	Administrador
------------------------	---------------

Formación	Docente, Administrativo
Actividades	Consultar Resultados, Realizar voto, Login, Gestionar Votantes, Gestionar Partidos, Gestionar Elecciones

TABLA X. CARACTERÍSTICAS USUARIO MEMBERSHIP SERVICE PROVIDERS

Tipo de usuario	Membership Service Providers
Formación	Es el proveedor de servicios de membresía (MSP) es un componente de Hyperledger Fabric que ofrece una abstracción de las operaciones de membresía, se puede decir que es un gestor de usuarios o participantes.
Actividades	Validar Usuario, Registrar Usuario, Crear Wallet

TABLA XI. CARACTERÍSTICAS USUARIO SC-VOTES

Tipo de usuario	SC-Votes
Formación	Smart Contract (Chaincode)
Actividades	Registrar Voto, Consultar Votos

2.4 Restricciones

- Interfaz para ser utilizada con internet.
- Se utilizará las herramientas de Hyperledger Fabric.
- El sistema de software podrá ser utilizado en cualquier dispositivo android, navegadores Chrome, Mozilla o Edge.
- Los lenguajes y tecnologías en uso: NodeJS, Express, JavaScript, Dart, Flutter y MongoDB.
- Los servidores deben ser capaces de atender consultas concurrentemente.
- El sistema deberá tener un diseño e implementación sencilla, independiente de plataforma o lenguaje de programación.

2.5 Suposiciones y dependencias

- Se asume que los requerimientos aquí descritos son estables.
- Los equipos en los que se vaya a ejecutar el sistema deben cumplir los requisitos antes indicados para garantizar una ejecución correcta del sistema.

3 Requisitos específicos

3.1 Requisitos comunes de las interfaces

3.1.1 Interfaces de usuario

La interfaz de usuario consistirá en un conjunto de ventanas con botones, listas y campos de textos. Esta deberá ser construida específicamente para el sistema propuesto y, será visualizada desde dispositivos móviles con sistema operativo Android y navegadores web.

3.1.2 Interfaces de hardware

Será necesario disponer de equipos de cómputo en perfecto estado con las siguientes características:

- Computador
- Conectividad

3.1.3 Interfaces de software

- Sistema Operativo computador: Ubuntu 18.04 o superiores.
- Sistema Operativo móviles: Android 23 o superiores.
- Explorador: Mozilla o Chrome

3.2 Requisitos funcionales

TABLA XII. REQUISITO FUNCIONAL INICIO DE SESIÓN

Identificación del requerimiento:	RF01
Nombre del requerimiento:	Inicio de sesión
Descripción del requerimiento:	Para poder hacer uso del sistema, el administrador y el votante deben iniciar sesión con usuario y contraseña, además se debe validar si el usuario que está tratando de ingresar es un usuario que está registrado en la Blockchain.
Dependencias:	El sistema debe validar al usuario en la red blockchain.
Requerimiento No Funcional:	<ul style="list-style-type: none">• RNF03

	<ul style="list-style-type: none"> • RNF04
Prioridad del requerimiento:	Alta

TABLA XIII. REQUISITO FUNCIONAL GESTIÓN DE VOTANTES

Identificación del requerimiento:	RF02
Nombre del requerimiento:	Gestión de votantes
Descripción del requerimiento:	El administrador podrá crear votantes, actualizar sus datos y visualizar su información.
Dependencias:	RF01
Requerimiento No Funcional:	<ul style="list-style-type: none"> • RNF02 • RF04
Prioridad del requerimiento:	Alta

TABLA XIV. REQUISITO FUNCIONAL GESTIÓN DE PARTIDOS

Identificación del requerimiento:	RF03
Nombre del requerimiento:	Gestión de partidos
Descripción del requerimiento:	El administrador puede crear partidos, actualizar sus datos y ver el listado. Además, puede agregar candidatos a estos partidos; los candidatos pueden ser cualquier votante registrado. Los partidos pertenecen a un periodo de elecciones específico.
Dependencias:	<ul style="list-style-type: none"> • RF01 • RF02
Requerimiento No Funcional:	<ul style="list-style-type: none"> • RNF02 • RNF04
Prioridad del requerimiento:	Alta

TABLA XV. REQUISITO FUNCIONAL GESTIÓN DE ELECCIONES

Identificación del requerimiento:	RF04
Nombre del requerimiento:	Gestión de elecciones
Descripción del requerimiento:	El administrador puede crear elecciones, editar sus datos y ver el listado de elecciones registradas y añadir el listado de votantes habilitados para esta elección.
Dependencias:	<ul style="list-style-type: none"> • RF01 • RF02 • RF03
Requerimiento No Funcional:	<ul style="list-style-type: none"> • RNF02 • RNF04
Prioridad del requerimiento:	Alta

TABLA XVI. REQUISITO FUNCIONAL CONSULTA DE RESULTADOS

Identificación del requerimiento:	RF05
Nombre del requerimiento:	Consulta de resultados
Descripción del requerimiento:	Cualquier usuario con o sin cuenta dentro del sistema puede visualizar los resultados de las elecciones, en curso o las pasadas.
Dependencias:	<ul style="list-style-type: none"> • RF03 • RF04
Requerimiento No Funcional:	<ul style="list-style-type: none"> • RNF02 • RNF03
Prioridad del requerimiento:	Alta

TABLA XVII. REQUISITO FUNCIONAL REALIZAR VOTO

Identificación del requerimiento:	RF06
Nombre del requerimiento:	Realizar voto
Descripción del requerimiento:	El votante debe tener su sesión abierta para poder visualizar las elecciones a las que tiene permitido sufragar, dentro de cada elección podrá visualizar los partidos y sus candidatos más la opción de otorgar su voto a uno de ellos. al registrar el voto se debe confirmar la decisión, con esto el sistema registra la elección del votante al partido y envía este dato a registrar en la Blockchain.
Dependencias:	<ul style="list-style-type: none"> • RF01 • RF04
Requerimiento No Funcional:	<ul style="list-style-type: none"> • RNF02 • RNF03 • RNF04
Prioridad del requerimiento:	Alta

3.3 Requisitos no funcionales

TABLA XVIII. REQUISITO NO FUNCIONAL RENDIMIENTO

Identificación del requerimiento:	RNF01
Nombre del requerimiento:	Rendimiento
Descripción del requerimiento:	<p>El sistema debe proporcionar un tiempo de respuesta aceptable aproximadamente entre 2 a 7 segundos.</p> <p>La transacción tarda de 2 a 5 segundos en un ambiente simulado de red blockchain y de 15 a 5 minutos en un ambiente real.</p>
Prioridad del requerimiento:	Alta

TABLA XIX. REQUISITO NO FUNCIONAL USABILIDAD

Identificación del requerimiento:	RNF02
Nombre del requerimiento:	Usabilidad
Descripción del requerimiento:	El sistema de software debe proporcionar una interfaz amigable e intuitiva, haciendo que el proceso sea comprensible y fácil de llevar a cabo. Además, debe permitir ser utilizado en cualquier navegador web.
Prioridad del requerimiento:	Alta

TABLA XX. REQUISITO NO FUNCIONAL FIABILIDAD

Identificación del requerimiento:	RNF03
Nombre del requerimiento:	Fiabilidad
Descripción del requerimiento:	El sistema de software debe permitir la disponibilidad las 24 horas del día y los 7 días de la semana, y en caso de que el sistema de software presente algún error, se debe recuperar en el menor tiempo posible. El sistema de software debe permitir recuperar los datos que se vean afectados en el caso de alguna falla en el modulo de software respecto al tiempo y esfuerzo que este genere.
Prioridad del requerimiento:	Alta

TABLA XXI. REQUISITO NO FUNCIONAL SEGURIDAD

Identificación del requerimiento:	RNF04
Nombre del requerimiento:	Seguridad
Descripción del requerimiento:	El sistema de software debe garantizar disminuir las vulnerabilidades de ataques de fuerza bruta.

	Garantizar la seguridad del sistema de software con respecto a la información y datos que se manejan tales sean documentos o archivos.
Prioridad del requerimiento:	Alta

C. Anexo 3. Historias de Usuario

Código: US - 001							
Historia: Acceder al Sistema			Requisitos: RF - 001				
Enunciado de la historia			Criterios de aceptación				
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
Como Administrador y como Votante necesito	poder ingresar al sistema	para poder hacer uso de las funcionalidades del sistema	1	Permitir el acceso al sistema	El usuario ingresa correctamente su usuario y clave	Solicitud de ingreso al sistema validando credenciales ingresadas	El sistema, tras validar las credenciales, y si estas son correctas, muestra la vista principal y da un mensaje de bienvenida.
			2	No Permitir el acceso al sistema	El usuario ingresa incorrectamente su usuario o clave	Solicitud de ingreso al sistema validando	El sistema, tras validar las credenciales, y al ser alguna de estas incorrecta, muestra la

						credenciales ingresadas	vista de Login y presenta un mensaje de error en las credenciales.
Código: US - 002							
Historia: Registro de Votantes				Requisitos: RF - 002			
Enunciado de la historia				Criterios de aceptación			
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
Como Administrador necesito	poder registrar la información de los votantes que podrán acceder al sistema	con la finalidad de mantener actualizada y saneada la base de datos de votantes	1	Los votantes se registran en la base de datos local y en la Blockchain	El votante se registra en el sistema	validar que los datos sean correctos y organización de los mismos para su envío y registro	El sistema registra los datos del votante y envía un correo de bienvenida

			2	Los votantes se registran en la base de datos local, pero no en la blockchain	El votante se registra en el sistema	validar que los datos sean correctos y organización de los mismos para su envío y registro	Si la blockchain no logra registrar el nuevo votante, envía una alerta de error al sistema, y este último elimina el nuevo votante registrado y entrega un mensaje de error en el registro
			3	Los votantes no se registran en la base de datos local	El votante se registra en el sistema	validar que los datos sean correctos y organización de los mismos para su envío y registro	Si el sistema no puede registrar el nuevo votante no envía sus datos a registrar en la blockchain y entrega un mensaje de error en el registro.
Código: US - 003							
Historia: Editar datos de votantes			Requisitos: RF - 002				

Enunciado de la historia			Criterios de aceptación				
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
Como Administrador necesito	poder actualizar la información de los votantes	para poder mantener actualizada la información de cada votante y bloquear el acceso a los usuarios desautorizados	1	Modificar la información del votante.	El Administrador ingresa a visualizar la lista de votantes y selecciona la función de modificar de uno de ellos.	Enviar nuevos datos del votante a registrar en la base de datos	El sistema, tras validar el rol del usuario, y si cuenta con los permisos necesarios, y si los nuevos datos son correctos, modifica los datos del usuario y los guarda, luego muestra la lista de votantes, con un mensaje de éxito.
			2	No Modificar la información de votante	El Administrador ingresa a visualizar la lista de votantes y selecciona la función de modificar de	Enviar nuevos datos del votante a registrar en la base de datos	El sistema, tras validar el rol del usuario, y si cuenta con los permisos necesarios, y si los datos no son correctos, devuelve un mensaje de error.

					uno de ellos, e ingresa datos erróneos.		
			3	No Modificar la información de votante	El Usuario que trata de modificar los datos de algún votante y no es un administrador	Enviar nuevos datos del votante a registrar en la base de datos	El sistema detecta que un usuario no autorizado está tratando de modificar los datos de un votante, inmediatamente envía un correo de alerta al administrador indicando que se está tratando de hacer una modificación sin autorización, junto con la información de usuario que se haya logrado recoger.
Código: US - 004							
Historia: Registro de Partidos				Requisitos: RF - 003			

Enunciado de la historia			Criterios de aceptación				
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
Como Administrador necesito	poder registrar la información de los partidos (organizaciones políticas) que competirán dentro del sistema	con la finalidad de mantener actualizada y saneada la base de datos de partidos	1	ver lista de votantes en el registro de partidos para agregar candidatos	El administrador ingresa a la función de crear partido y observa el formulario de registro	ingresar a la función de nuevo partido	El sistema al solicitar los datos del nuevo partido, también presenta la lista de votantes, para que puedan ser agregados como candidatos, dentro del partido, si es que no pertenece a ningún otro.
			2	Los partidos se registran en la base de datos local y en la Blockchain	El Administrador registra los datos del partido político, con sus candidatos e información básica	enviar los datos a registrar.	El sistema registra los datos del nuevo partido y envía los datos para su registro en la Blockchain, al finalizar muestra un mensaje de éxito.

			3	Los partidos se registran en la base de datos local, pero no en la blockchain	El Administrador registra los datos del partido político, con sus candidatos e información básica	enviar los datos a registrar.	Si la blockchain no logra registrar el nuevo partido, envía una alerta de error al sistema, y este último elimina el nuevo registro y entrega un mensaje de error
			4	Los partidos no se registran en la base de datos local	El Administrador registra los datos del partido político, con sus candidatos e información básica	enviar los datos a registrar.	Si el sistema no puede registrar el nuevo partido no envía sus datos a registrar en la blockchain y entrega un mensaje de error en el registro.
			5	No se registra el partido y se envía una alerta al administrador por un registro no autorizado	Algún usuario intenta registrar un nuevo partido sin autorización	registrar datos de partidos	El sistema detecta que un usuario no autorizado está tratando de ingresar los datos de un partido, inmediatamente envía un correo de alerta al administrador indicando que se está

							tratando de hacer una inserción sin autorización, junto con la información de usuario que se haya logrado recoger.
Código: US - 005							
Historia: Editar datos de partidos				Requisitos: RF - 003			
Enunciado de la historia				Criterios de aceptación			
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
Como Administrador necesito	poder actualizar la información de los partidos	para poder mantener actualizada la información de cada partido y deshabilitar a los partidos	1	Modificar la información del partido.	El Administrador ingresa a visualizar la lista de partidos y selecciona la función de	Enviar nuevos datos del partido a registrar en la base de datos	El sistema, tras validar que los nuevos datos son correctos, modifica los datos del partido y los guarda, luego muestra la lista de partidos, con un mensaje de éxito.

		que se han disuelto			modificar de uno de ellos.		
			2	No Modificar la información del partido	El Administrador ingresa a visualizar la lista de partidos y selecciona la función de modificar de uno de ellos, e ingresa datos erróneos.	Enviar nuevos datos del partido a registrar en la base de datos	El sistema, tras validar los datos y si estos no son correctos, devuelve un mensaje de error.
			3	No Modificar la información del partido y se envía una alerta al administrador por un registro no autorizado	Algun Usuario que trata de modificar los datos de un partido y no es un administrador	Enviar nuevos datos del partido a registrar en la base de datos	El sistema detecta que un usuario no autorizado está tratando de modificar los datos de un partido, inmediatamente envía un correo de alerta al administrador indicando que se está tratando de hacer una modificación sin autorización, junto con la información de

							usuario que se haya logrado recoger.
Código: US - 006							
Historia: Registro de Elecciones				Requisitos: RF - 004			
Enunciado de la historia				Criterios de aceptación			
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
Como Administrador necesito	poder registrar la información de las elecciones que se llevarán a cabo dentro del sistema	con la finalidad de mantener actualizada y saneada la base de datos de partidos	1	Ver lista de partidos en el registro de elecciones.	El administrador ingresa a la función de crear elecciones y observa el formulario de registro	ingresar a la función de nueva elección	El sistema al solicitar los datos de la nueva elección, también presenta la lista de partidos, para que puedan ser agregados como competidores, dentro de las elecciones.

			2	Las elecciones se registran en la base de datos local y en la Blockchain	El Administrador registra los datos de las elecciones	enviar los datos a registrar	El sistema registra los datos de las nuevas elecciones y envía los datos para su registro en la Blockchain; al finalizar muestra un mensaje de éxito.
			3	Las elecciones se registran en la base de datos local, pero no en la blockchain	El Administrador registra los datos de las elecciones	enviar los datos a registrar.	Si la blockchain no logra registrar las elecciones, envía una alerta de error al sistema, y este último elimina el nuevo registro y entrega un mensaje de error
			4	Las elecciones no se registran en la base de datos local	El Administrador registra los datos de las elecciones	enviar los datos a registrar.	Si el sistema no puede registrar las elecciones no envía sus datos a registrar en la blockchain y entrega un mensaje de error en el registro.
			5	No se registra el partido y se envía una	Algún usuario intenta registrar una nueva	registrar datos de partidos	El sistema detecta que un usuario no autorizado está tratando de registrar

				alerta al administrador por un registro no autorizado	elección sin autorización		los datos de una elección, inmediatamente envía un correo de alerta al administrador indicando que se está tratando de hacer una inserción sin autorización, junto con la información de usuario que se haya logrado recoger.
Código: US - 007							
Historia: Editar datos de elecciones				Requisitos: RF - 004			
Enunciado de la historia				Criterios de aceptación			
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
	poder actualizar la	para poder corregir la	1	Modificar la información	El Administrador	Enviar nuevos	El sistema, tras validar que los nuevos

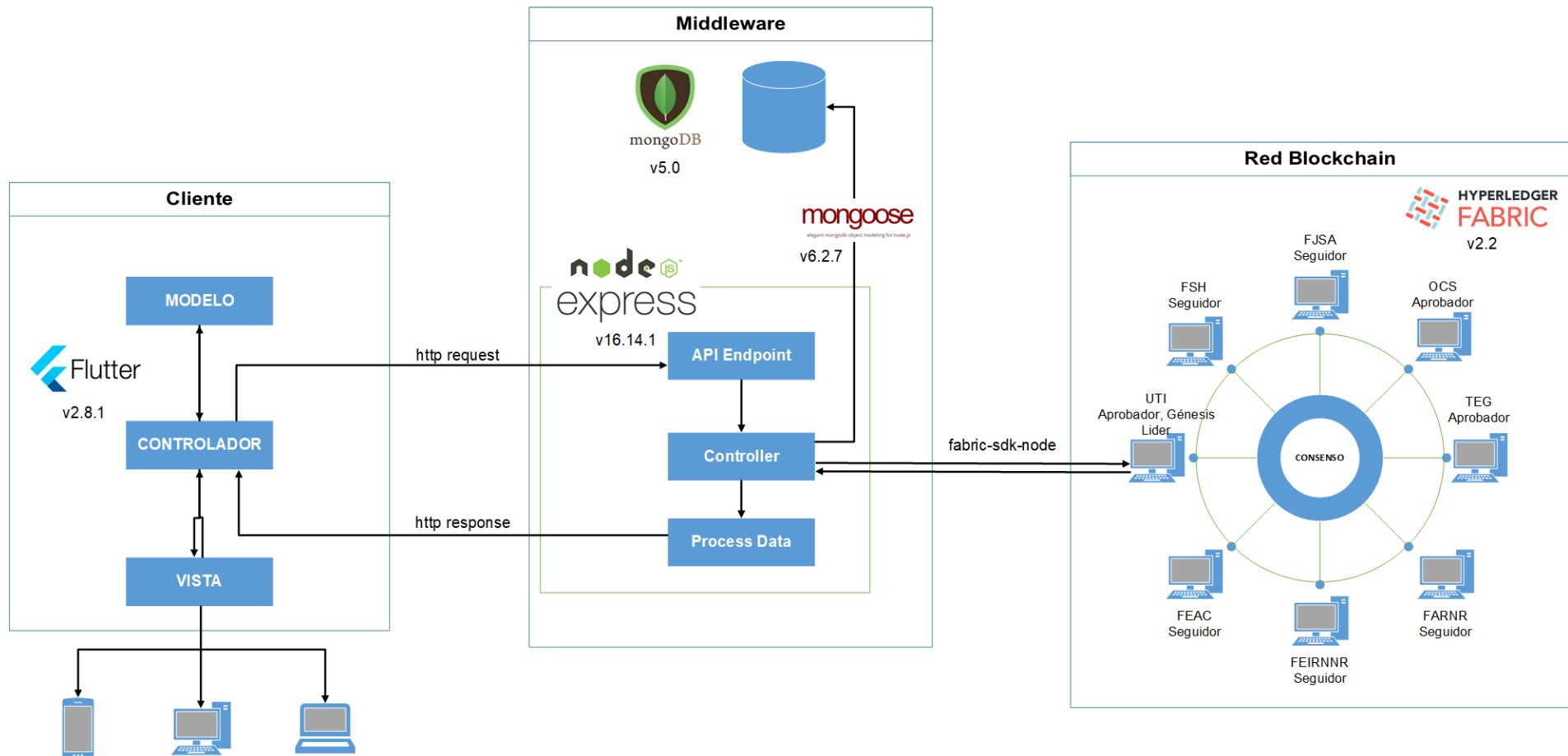
Como Administrador necesito	información de las elecciones	información de una elección		de la elección.	ingresa a visualizar la lista de elecciones y selecciona la función de modificar de una de ellas.	datos de la elección a registrar en la base de datos	datos son correctos, modifica los datos del partido y los guarda, luego muestra la lista de elecciones, con un mensaje de éxito.
			2	No Modificar la información de la elección	El Administrador ingresa a visualizar la lista de elecciones y selecciona la función de modificar de una de ellas, e ingresa datos erróneos.	Enviar nuevos datos de la elección a registrar en la base de datos	El sistema, tras validar los datos y si estos no son correctos, devuelve un mensaje de error.
			3	No Modificar la información de la elección y se envía una alerta al administrador	Algun Usuario que trata de modificar los datos de una elección y no es un administrador	Enviar nuevos datos de la elección a registrar en la base de datos	El sistema detecta que un usuario no autorizado está tratando de modificar los datos de una elección, inmediatamente envía un correo de alerta al

				por un registro no autorizado			administrador indicando que se está tratando de hacer una modificación sin autorización, junto con la información de usuario que se haya logrado recoger.
Código: US - 008							
Historia: Consultar Resultados				Requisitos: RF - 005			
Enunciado de la historia				Criterios de aceptación			
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado
Como Usuario necesito	poder visualizar los resultados de las elecciones	para poder seguir e informar sobre el resultado de los comicios	1	Mostrar los resultados de las elecciones actuales	Si han finalizado recientemente unas elecciones y algún usuario	Ver resultados de Elecciones recién finalizadas	El sistema consulta los resultados de las elecciones a la Blockchain y verifica si coinciden con los datos que tiene

					desea ver los resultados.		registrados en su base de datos local, si es así muestra los resultados, de lo contrario envía una alerta al administrador
			2	Mostrar resultados de elecciones pasadas	Algún usuario desea ver los resultados de elecciones anteriores	ver resultados de elecciones anteriores	El sistema obtiene los resultados de las elecciones que se desea visualizar y los muestra en pantalla
Código: US - 009							
Historia: Registrar Voto			Requisitos: RF - 006				
Enunciado de la historia			Criterios de aceptación				
Rol	Funcionalidad	Resultado	(#) Número de Escenario	Criterio de aceptación	Contexto	Evento	Comportamiento esperado

Como Votante necesito	poder registrar mi voto	para poder dar mi apoyo al partido político de mi agrado dentro de unas elecciones.	1	Registrar el voto	El votante debe tener su sesión abierta y constar como votante válido dentro de las elecciones.	Registrar voto	El sistema muestra las elecciones a las que puede participar el votante y dentro de ellas le muestra los partidos políticos que compiten, donde el votante puede asignar su voto a uno de ellos. tras asignar su voto el sistema solicita al votante verificar sus datos personales y si está seguro del registro, con la aprobación del votante el sistema registra el voto en la base de datos local y en la blockchain y actualiza de forma inmediata los resultados.
-----------------------------	----------------------------	--	---	----------------------	---	-------------------	--

D. Anexo 4. Arquitectura del sistema de e-voting



E. Anexo 5. Plan de pruebas para subsistema de contratos inteligentes.

Plan de pruebas unitarias subsistema de Contratos Inteligentes

Proyecto: Implementación de un sistema multiplataforma de voto electrónico basado en Blockchain

Version: 1.0

Fecha: 11/02/2022

ÍNDICE

1	INTRODUCCIÓN.....	2
1.1	Objeto.....	2
1.2	Propósito	2
2	DEFINICIÓN DE LOS CASOS DE PRUEBAS	3
3	GLOSARIO.....	10

1 INTRODUCCIÓN

1.1 Objeto

El objeto de este documento es para verificar la funcionalidad correcta del subsistema de contratos inteligentes aislado cada parte del código y mostrar que las partes individuales son correctas.

1.2 Propósito

Comprobar el correcto funcionamiento del sistema multiplataforma de voto electrónico basado de Blockchain, por unidad de código principal, para asegurar que cada unidad funcione correcta y eficientemente por separado.

2 DEFINICIÓN DE LOS CASOS DE PRUEBAS

En este apartado se describe en detalle cada uno de los casos de pruebas que se identificaron:

Número de caso de prueba	Subsistema	Descripción de lo que se probará.	Prerrequisitos.
CP01	Contrato inteligente	Comprobar que el CR del actor Voter funcione correctamente	Chaincode Vote.go compilado correctamente.
CP02	Contrato inteligente	Comprobar que el CR del actor Candidate funcione correctamente	Actor Voter creado correctamente.
CP03	Contrato inteligente	Comprobar que el CR del actor Plitic_Party funcione correctamente.	Actor Candidate creado correctamente.
CP04	Contrato inteligente	Comprobar que el CR del actor Election funcione correctamente.	Actor Plitic_Party creado correctamente.
CP05	Contrato inteligente	Comprobar que el CR del actor Vote funcione correctamente.	Actor Election creado correctamente.
CP06	Contrato inteligente	Comprobar que el CR del actor Item_Result funcione correctamente.	Actor Vote creado correctamente.
CP07	Contrato inteligente	Comprobar que el CR del actor Result funcione correctamente.	Actor Item_Result creado correctamente.

CP01					
N°	Descripción	Método	Datos de Entrada	¿OK?	Observaciones
1	Se registra una instancia del actor Voter y se verifica que se encuentre registrada.	CreateVoter()	<ul style="list-style-type: none"> voterId name lastname dni email function 	X	N/A
2	Se obtiene una instancia del actor Voter mediante de su id.	GetVoterById()	<ul style="list-style-type: none"> voterId 	X	N/A
3	Se obtiene una lista con las instancias existentes en la base de datos del actor Voter.	GetAllVoters()	N/A	X	N/A
4	Se valida si existe alguna instancia con el id de entrada en la base de datos del actor Voter	VoterExists()	<ul style="list-style-type: none"> voterId 	X	N/A

5	Se obtiene el historial de cambios en la base de datos de una instancia del actor Voter	VoterHistory()	<ul style="list-style-type: none"> voterId 	X	N/A
---	---	--------------------	---	---	-----

CP02					
N°	Descripción	Método	Datos de Entrada	¿OK?	Observaciones
1	Se registra una instancia del actor Candidate y se verifica que se encuentre registrada.	CreateCandidate()	<ul style="list-style-type: none"> candidateId position photo voterId 	X	N/A
2	Se obtiene una instancia del actor Candidate mediante de su id.	GetCandidateById()	<ul style="list-style-type: none"> candidateId 	X	N/A
3	Se obtiene una lista con las instancias existentes en la base de datos del actor Candidate.	GetAllCandidates()	N/A	X	N/A
4	Se valida si existe alguna instancia con el id de entrada en la base de datos del actor Candidate	CandidateExists()	<ul style="list-style-type: none"> candidateId 	X	N/A
5	Se obtiene el historial de cambios en la base de datos de una instancia del actor Candidate	CandidateHistory()	<ul style="list-style-type: none"> candidateId 	X	N/A

CP03					
N°	Descripción	Método	Datos de Entrada	¿OK?	Observaciones

1	Se registra una instancia del actor Politic Party y se verifica que se encuentre registrada.	CreatePolitic_Party()	<ul style="list-style-type: none"> • politic_partyId • name • description • logo • candidatesId 	X	N/A
2	Se obtiene una instancia del actor Politic Party mediante de su id.	GetPolitic_PartyById()	<ul style="list-style-type: none"> • politic_partyId 	X	N/A
3	Se obtiene una lista con las instancias existentes en la base de datos del actor Politic Party.	GetAllPolitic_Partiys()	N/A	X	N/A
4	Se valida si existe alguna instancia con el id de entrada en la base de datos del actor Politic Party	Politic_PartyExists()	<ul style="list-style-type: none"> • politic_partyId 	X	N/A
5	Se obtiene el historial de cambios en la base de datos de una instancia del actor Politic Party	Politic_PartyHistory()	<ul style="list-style-type: none"> • politic_partyId 	X	N/A

CP03					
N°	Descripción	Método	Datos de Entrada	¿OK?	Observaciones
1	Se registra una instancia del actor Election y se verifica que se encuentre registrada.	CreateElection()	<ul style="list-style-type: none"> • electionId • name • description • date_hour_start • date_hour_end • politic_partiesId 	X	N/A
2	Se obtiene una instancia del actor Election mediante de su id.	GetElectionId()	<ul style="list-style-type: none"> • electionId 	X	N/A

3	Se obtiene una lista con las instancias existentes en la base de datos del acto Election.	Getelections()	N/A	X	N/A
4	Se valida si existe alguna instancia con el id de entrada en la base de datos del actor Election	ElectionExists()	<ul style="list-style-type: none"> electionId 	X	N/A
5	Se obtiene el historial de cambios en la base de datos de una instancia del actor Election	ElectionHistory()	<ul style="list-style-type: none"> electionId 	X	N/A
CP05					
N°	Descripción	Método	Datos de Entrada	¿OK?	Observaciones
1	Se registra una instancia del actor Vote y se verifica que se encuentre registrada.	CreateVote()	<ul style="list-style-type: none"> voteId voterId politic_partyId electionId 	X	N/A
2	Se obtiene una instancia del actor Vote mediante de su id.	GetVoteById()	<ul style="list-style-type: none"> voteId 	X	N/A
3	Se obtiene una lista con las instancias existentes en la base de datos del acto Vote.	GetAllVotes()	N/A	X	N/A
4	Se valida si existe alguna instancia con el id de entrada en la base de datos del actor Vote	VoteExists()	<ul style="list-style-type: none"> voteId 	X	N/A
5	Se obtiene el historial de cambios en la base de	VoteHistory()	<ul style="list-style-type: none"> voteId 	X	N/A

	datos de una instancia del actor Vote				
--	---------------------------------------	--	--	--	--

CP06					
N°	Descripción	Método	Datos de Entrada	¿OK?	Observaciones
1	Se registra una instancia del actor Result y se verifica que se encuentre registrada.	CreateResult() ()	<ul style="list-style-type: none"> resultId nro_votes politic_partyId electionId 	X	N/A
2	Se obtiene una instancia del actor Result mediante de su id.	GetResultById() ()	<ul style="list-style-type: none"> resultId 	X	N/A
3	Se obtiene una lista con las instancias existentes en la base de datos del actor Result.	GetAllResults() ()	N/A	X	N/A
4	Se valida si existe alguna instancia con el id de entrada en la base de datos del actor Result	ResultExists() ()	<ul style="list-style-type: none"> resultId 	X	N/A
5	Se obtiene el historial de cambios en la base de datos de una instancia del actor Result	ResultHistory() ()	<ul style="list-style-type: none"> resultId 	X	N/A

CP07					
N°	Descripción	Método	Datos de Entrada	¿OK?	Observaciones
1	Se registra una instancia del actor Item_Result y se	CreateItemResult() ()	<ul style="list-style-type: none"> item_resultId 	X	N/A

	verifica que se encuentre registrada.		<ul style="list-style-type: none"> • nro_votes • political_partyId • resultId 		
2	Se obtiene una instancia del actor Item_Result mediante de su id.	GetItemResultById()	<ul style="list-style-type: none"> • item_resultId 	X	N/A
3	Se obtiene una lista con las instancias existentes en la base de datos del actor Item_Result.	GetAllItem_Results()	N/A	X	N/A
4	Se valida si existe alguna instancia con el id de entrada en la base de datos del actor Item_Result	Item_ResultExists()	<ul style="list-style-type: none"> • item_resultId 	X	N/A
5	Se obtiene el historial de cambios en la base de datos de una instancia del actor Item_Result	Item_ResultHistory()	<ul style="list-style-type: none"> • item_resultId 	X	N/A

3 GLOSARIO

A continuación, se muestra la definición de todos los términos utilizados en el presente documento.

Término	Descripción
DNI	Documento nacional de identificación.
CR	Crear(Create), Leer(Read), métodos de crear y leer
Chaincode	Contrato inteligente en de Hyperledger Fabric
CP	Caso de prueba