UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA

CARRERA DE INGENIERÍA INFORMÁTICA

Modelo de solución mediante el uso de Smart Contracts para el registro de matrículas de estudiantes en la UCE

Trabajo de titulación, modalidad Proyecto de Investigación, previo a la obtención del título de Ingeniero Informático.

AUTOR: Miguel Ángel García Merizalde

TUTOR: Ing. Mario Raúl Morales Morales MBA.

Quito, 2019

DERECHOS DE AUTOR

Yo, Miguel Ángel García Merizalde en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación **Modelo de solución mediante el uso de Smart Contracts para el registro de matrículas de estudiantes en la UCE**, modalidad PROYECTO DE INVESTIGACIÓN, de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN, concedo el favor a la Universidad Central del Ecuador una licencia, gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Conservo a mi favor todos los derechos del autor sobre la obra, establecidos en la normativa citada.

Así mismo, autorizo a la Universidad Central del Ecuador para que realice la digitalización y publicación de este trabajo de titulación en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

El autor declara que la obra objeto de la presente autorización es original en su forma de expresión y no infringe el derecho de autor de terceros, asumiendo la responsabilidad por cualquier reclamación que pudiera presentarse por esta causa y liberando a la Universidad de toda responsabilidad.

Firma: _____

Miguel Ángel García Merizalde
C.C: 1719109173
e-mail: magarciam1@uce.edu.ec

APROBACIÓN DEL AUTOR

En mi calidad de Tutor del Trabajo de Titulación, presentado por **MIGUEL ÁNGEL GARCÍA MERIZALDE,** para optar por el Grado de Ingeniero Informático; cuyo título es: **MODELO DE SOLUCIÓN MEDIANTE EL USO DE SMART CONTRACTS PARA EL REGISTRO DE MATRÍCULAS DE ESTUDIANTES EN LA UCE**, considero que dicho trabajo reúne los requisitos y méritos suficientes, para ser sometido a la presentación pública y evaluación por parte de tribunal examinador que se designe.

En la ciudad de Quito, a los 12 días, del mes de junio del 2019

_____

Ing. Mario Raúl Morales Morales, Mba.

DOCENTE-TUTOR

C.C: 1709026577

DEDICATORIA

El presente trabajo va dedicado a Dios, quien como guía estuvo presente en el caminar de mi vida, bendiciéndome y dándome fuerzas para continuar con mis metas trazadas sin desfallecer. A mis padres, por ser el pilar más importante en el transcurso de mi vida en todo ámbito demostrándome siempre su cariño, apoyo incondicional y consejos sin importar las adversidades, inconvenientes y diferencias que se presentaron. A mi tío Pablo, a quien quiero como un padre, por compartir momentos significativos conmigo y por siempre estar dispuesto a escucharme y ayudarme en cualquier momento. A mi abuelita Cecilia, quien como una madre a través de su experiencia y sabiduría supo darme consejos y directrices que aportaron significativamente para el logro de esta importante meta.

MIGUEL GARCÍA

AGRADECIMIENTO

Agradezco a los docentes de la carrera de Ingeniería Informática de la Facultad de Ingeniería Ciencias Físicas y Matemática de la Universidad Central del Ecuador, en especial a mi director de tesis y amigo, el Ing. Mario Raúl Morales Morales por guiar esta investigación, formar parte de este logro alcanzado y principalmente por haber impartido sus conocimientos sin restricciones, por el tiempo, dedicación, paciencia, aportes y confianza depositados en mi persona y en este trabajo; además de su afabilidad y exigencia que solo una persona con valores éticos, profesionales y morales como él puede ofrecer y transmitir.

CONTENIDO

pág.

LISTA DE TABLAS

LISTA DE ILUSTRACIONES

TÍTULO: Modelo de solución mediante el uso de Smart Contracts para el registro de matrículas de estudiantes en la UCE.

AUTOR: Miguel Ángel García Merizalde

TUTOR: Ing. Mario Raúl Morales Morales, Mba.

RESUMEN

El presente proyecto tiene como objetivo dar a conocer la tecnología Blockchain, cómo surgió, su funcionamiento y el motivo por el cual se proyecta como una de las grandes tecnologías del futuro. Blockchain en términos generales, es una gran base de datos descentralizada y basado en algunas de las cualidades enunciadas durante el proyecto, se creará una prueba de concepto basada en la blockchain de Ethereum, una de las más conocidas e importantes después de Bitcoin, con el objetivo final de llevar la integridad de datos, en especial los datos concernientes al registro de matrículas de los estudiantes de la UCE, a un siguiente nivel. A fin de implementar la descentralización de la información de los estudiantes y adjudicar una amplia privacidad a los datos, se desarrollará una aplicación web para extraer los datos de la blockchain a través del uso de smart contracts; que son pequeños scripts almacenados en la blockchain de Ethereum, los mismos que, simulando el comportamiento de contratos reales, permiten cumplir cabalmente las normas y reglas descritas en ellos. La aplicación permitirá registrar la información referente a las asignaturas en las que se matriculó un estudiante sin que esta sea almacenada de manera centralizada en un host específico. El objetivo es bloquear el acceso a los datos, hasta que exista un consentimiento entre las partes implicadas durante el proceso del registro de una matrícula; así no se da pie a posibles filtraciones o alteraciones de datos valiosos de los estudiantes. El proyecto se ha modelado usando los lenguajes de programación Solidity y JavaScript y para el despliegue de los contratos en Ethereum se ha utilizado una red test a fin de evitar costes y optimizar el tiempo en el que la blockchain comprueba cada una de las transacciones.

PALABRAS CLAVE: CADENA DE BLOQUES / CONTRATOS INTELIGENTES / REGISTRO DE TRANSACCIONES /MATRICULAS/ APLICACIONES DESCENTRALIZADAS.

TÍTLE: Solution Model by means of Smart Contracts for Registration of Student´s Enrollments at the UCE.

AUTOR: Miguel Ángel García Merizalde

TUTOR: Eng. Mario Raúl Morales Morales, Mba.

ABSTRACT

The objective of this project is to make known Blockchain technology, how it emerged, its operation and the reason why it is projected as one of the great technologies of the future. Blockchain is in general terms, is a large decentralized database based on some of the qualities enunciated during the project; a proof of concept will be created based on Ethereum´s blockchain, one of the most known and important after Bitcoin, with the final goal of carrying the integrity of data, especially the data concerning registration of students´ enrollments at the UCE, to a next level. In order to implement the decentralization of the students´ information and grant wide privacy of the data, a web application will be developed to extract it from the blockchain through the use of smart contracts; these are small scripts stored in the Ethereum´s blockchain which by simulating the behavior of real contracts, allow to fully comply with the norms and rules described in them. The application will allow to register the information referring to the subjects in which a student enrolled without this being centrally stored in a specific host. The objective is to block access to the data until there is a consent between the parties involved during the registration process; thus, there is no possibility of leaks or alterations of the students´ valuable data. The project has been modeled using the Solidity and JavaScript programming languages and for the deployment of the contracts in Ethereum a test network has been used in order to avoid costs and optimize the time in which the blockchain checks out each of the transactions.

KEY WORDS: BLOCKCHAIN / SMART CONTRACTS / TRANSACTIONS REGISTRATION /ENROLLMENTS/ DECENTRALIZED APPLICATIONS.

INTRODUCCIÓN

Cuando se habla de contratos nos viene a la mente el típico documento en papel con una serie de condiciones escritas, en el que, si las partes implicadas están de acuerdo con las mismas, escriben su firma en él comprometiéndose a cumplir dichas condiciones. Hoy en día, aunque éste método sigue siendo el más utilizado en todo el mundo se ha dado un gran paso adelante en la automatización, seguridad y garantías respecto a los contratos tradicionales con el surgimiento de los Contratos Inteligentes o también llamados Smart Contracts, este término que aparece innumerables veces cuando uno se adentra en el mundo del Bitcoin, criptomonedas y Blockchain se lleva desarrollando desde 1993, cuando el famoso criptógrafo Nick Szabo acuñó el término por primera vez, pero la limitaciones tecnológicas de la época no permitían hacer viable esta idea; la necesidad de crear un sistema de pagos que los pudiese llevar a la práctica era primordial y esa situación no apareció en escena hasta la creación del Bitcoin en el año 2009 (miethereum, SMART CONTRACTS O CONTRATOS INTELIGENTES, 2018).

No obstante, Bitcoin no fue pensado para nada más que para ser una herramienta financiera, es decir, una criptomoneda. Por el contrario, la tecnología con la que funcionan denominada Blockchain o Cadena de Bloques, hace posible la viabilidad de estos contratos inteligentes y fue a principios de 2014, con la creación de Ethereum, cuando, por fin, pasaron a ser una realidad. Estos Smart Contracts se establecen en una atmósfera no controlada por ninguna de las partes implicadas en el contrato, es decir, en un sistema descentralizado esto significa que las condiciones son programadas, firmadas por ambas partes implicadas y enviadas a una cadena de bloques con el fin de hacerlos inmutables e indelebles (miethereum, SMART CONTRACTS O CONTRATOS INTELIGENTES, 2018).

En el campo de la Educación han ocurrido problemas de fraudes en los procesos, esta situación sucede en procesos de todo tipo: desde el trámite de emisión de un certificado hasta el registro de estudiantes en el sistema educativo, evidenciando claras falencias de los entes reguladores en dicha materia debido a que no cuentan con un mecanismo confiable y seguro con el cual hacer seguimiento a estos procesos. Gracias a Blockchain se puede certificar la autenticidad de cualquier tipo de transacción, esta tecnología innovadora en la gestión de registros de transacciones hace que las operaciones realizadas bajo esta tecnología sean públicas, seguras e inmutables. El concepto de Blockchain aplicado en un sistema de información podría aportar a la solución de la problemática mencionada, en este caso particular, el registro de la matrícula de los estudiantes, evitando así el tradicional trabajo burocrático a empleados y estudiantes. La tecnología Blockchain permite que, una vez obtenida la información, se desarrolle una plataforma descentralizada y abierta donde no existan intermediarios obteniendo como resultado un desahogo, para ello, el empleado habrá ido depositando la información recolectada de los estudiantes mediante contratos inteligentes basados en Blockchain. De otro lado, la información se dirige a quien realmente interesa que se dirija, esto es, al empleador, pasando en última instancia, el control propio en lo relacionado a calidad, credibilidad, gobierno y administración a manos de la institución. En otras palabras, se busca mejorar los contratos actuales haciéndolos más seguros, más baratos, ahorrando tiempo y evitando fraudes (Suárez Chacón David Felipe, 2018).

1. DEFINICIÓN DEL PROBLEMA

1.1 Antecedentes

Los sistemas de contratos inteligentes emergentes sobre las criptomonedas descentralizadas permiten a las partes desconfiadas mutuamente realizar transacciones seguras sin terceros confiables. En caso de incumplimiento contractual o aborto, la cadena de bloques descentralizada garantiza que las partes obtengan una compensación proporcional de manera honesta. Los sistemas existentes, sin embargo, carecen de privacidad transaccional. Todas las transacciones, incluido el flujo de dinero entre los seudónimos y la cantidad negociada, están expuestas en la cadena de bloques. Un programador puede escribir un contrato inteligente privado de una manera intuitiva sin tener que implementar criptografía, en el que las partes contractuales interactúan con la cadena de bloques, el modelado formal es de interés independiente. Abogamos a la comunidad para que adopte un modelo formal al diseñar aplicaciones de bloques descentralizados. (Ahmed Kosba, 2016).

Los conceptos de cadenas de bloques y contratos inteligentes ofrecen una alternativa sostenible en educación superior. Desde este objetivo, se presenta una revisión de ambos conceptos y su relación con los términos Bitcoin. En un segundo momento, se atiende a las redes en educación superior basadas en tecnología de cadenas de bloques, su vínculo con los contratos inteligentes y las posibilidades a día de hoy. (Poveda, Revista d'Innovació Docent Universitària , 2018).

La tecnología Blockchain proporciona un consenso descentralizado y, potencialmente, amplía el espacio de contratación utilizando contratos inteligentes con pruebas de manipulación y ejecuciones algorítmicas. Mientras tanto, generar consenso descentralizado implica distribuir información que altere necesariamente el entorno informativo. Analizamos cómo la descentralización afecta la efectividad del consenso y cómo las características por excelencia de Blockchain reconfiguran la organización industrial y el panorama de la competencia. Los contratos inteligentes pueden mitigar la asimetría informativa y mejorar el bienestar y el excedente del consumidor a través de una mejor entrada y competencia, sin embargo, la distribución irreductible de la información durante la generación de consenso puede alentar una mayor colusión. En general, las cadenas de bloques pueden sostener equilibrios de mercado con una gama más amplia de resultados económicos. Además de analizar las implicaciones de las políticas antimonopolio dirigidas a las aplicaciones Blockchain y de cómo separar a los conservadores de registros por consenso de los usuarios. (Lin William Cong, 2018).

1.2 Justificación

La tecnología Blockchain enfoca su estudio no solo en lo que respecta a una moneda digital, sino que también se adentra en el campo de la educación de manera cautelosa, en la actualidad aparecen aplicaciones todavía con un carácter de prueba pero debido a las expectativas que se están levantando sobre el uso de cadena de bloques en específico de los contratos inteligentes nos lleva a plantearnos cómo funciona esta tecnología, para qué se usa, cómo podría aprovecharse, que oportunidades, que amenazas puede suponer dentro del campo de la educación. El análisis de la nueva tecnología en cuestión proporciona una cantidad considerable de referencias y enlaces como soporte para determinar y reflexionar sobre el uso de la misma en lo que respecta al registro de transacciones dentro de un sistema de información en educación y los beneficios que esta tecnología ofrecería en el registro de matrículas para cumplir con las expectativas del uso de los contratos inteligentes en la cadena de bloques.

1.3 Planteamiento del Problema

El presente proyecto de investigación plantea optimizar el registro inequívoco de datos en un sistema de información, revelando las razones de un contrato inteligente y su rol fundamental para transformar la tecnología de la cadena de bloques a fin de habilitar plataformas descentralizadas, confiables y económicas, explorando la estructura y los conceptos básicos de un contrato inteligente para desplegarlo e interactuar con el registro de transacciones, específicamente en el proceso de registro de matrículas de los estudiantes en la UCE con el propósito de automatizar la efectividad del cumplimiento de los contratos actuales haciéndolos más confiables y asequibles, racionando el tiempo y burocracia que el proceso mencionado toma, esencialmente evitando fraudes durante la ejecución del mismo. En la actualidad el registro de información transaccional tal como el registro de datos pueden ser mejorados y automatizados a través de tecnología disruptiva como lo es la cadena de bloques orientada hacia los denominados contratos inteligentes.

1.4 Formulación del Problema

Los procesos administrativos en la UCE son todavía burocráticos debido a la falta de uso de herramientas tecnológicas disruptivas que contribuyan a la optimización de los procesos antes mencionados por cuanto existe un desconocimiento de esta tecnología la cual está a disponibilidad del público pero su aplicación está ligada a la falta de convencimiento acerca del éxito y beneficios de su implementación sin lograr que los procesos antes mencionados dejen de ser engorrosos y tediosos dentro del órgano regular establecido por la UCE para este tipo de trámites.

Por tanto, apoyados en la tecnología Blockchain la cual es un tipo de base de datos descentralizada de punto a punto se puede proporcionar un registro de transacciones seguro, público y confiable, permitiendo que se ingresen datos y evitando que se alteren los mismos. Uno de los aspectos más interesantes es un contrato inteligente basado en Blockchain que incluya los términos, condiciones, métricas de rendimiento y penalizaciones del mismo. Este contrato inteligente automatizado durante su ejecución monitorea, verifica y aplica continuamente esas condiciones de desempeño. Si el servicio fluctúa, el contrato generaría sanciones u otros términos específicos, buscando así nuevas alternativas para eliminar intermediarios en los procesos administrativos dejando de generar retrasos, ineficiencia, gastos operativos y entregando un valor agregado por cada transacción debido a que, al ser inmodificables, permiten tener mayor control, disponibilidad de la información y seguridad en las operaciones.

1.5 Hipótesis

Es posible optimizar el registro de matrículas a través de soluciones disruptivas de tipo contratos inteligentes apoyados en la tecnología de cadena de bloques.

1.6 Objetivos

1.6.1 Objetivo General

Automatizar el proceso de registro de matrículas de los estudiantes de la UCE a través del uso de contratos Inteligentes basados en cadenas de bloques.

1.6.2 Objetivos Específicos

- Determinar los mecanismos de estructuración de los contratos inteligentes dentro de la cadena de bloques.
- Plantear un esquema de implementación de contratos inteligentes basado en la cadena de bloques.
- Proponer un modelo solución para el proceso de registro de la matrícula de los estudiantes de la UCE por medio del uso de contratos inteligentes.

2 Marco Teórico

2.1 Cadena de Bloques (Blockchain)

El término Blockchain posee diferentes matices respecto a su definición, entre los diferentes autores citamos a los siguientes:

"Una Blockchain no es otra cosa que una base de datos que se halla distribuida entre diferentes participantes, protegida criptográficamente y organizada en bloques de transacciones relacionados entre sí matemáticamente. Expresado de forma más breve, es una base de datos descentralizada que no puede ser alterada" (Preukschat, 2017).

"A diferencia de las bases de datos tradicionales, existe una particularidad en cuanto al tipo de modificaciones que pueden realizarse sobre una Blockchain: una vez que una transacción ha sido incorporada a este registro, no puede modificarse ni eliminarse" (Villarreal, 2017).

"En general, Blockchain es una plataforma digital que mantiene el historial completo de todas las transacciones entre usuarios a través de la red" (Peláez, 2018).

"La palabra Blockchain se traduce, del inglés, invirtiendo el singular bajo la denominación "cadena de bloques". El Blockchain es una tecnología que archiva tipos distintos de datos y documentos, a modo de libro de acontecimientos digitales, que se comparten mediante un sistema de bases de datos distribuidos" (Poveda, Algunos aspectos sobre blockchains y smart contracts, 2018).

La propuesta tecnológica del Blockchain derivada del protocolo Bitcoin ha ido demostrando durante casi diez años desde su exposición, la viabilidad y potencial de uso de sistemas descentralizados para la gestión de activos digitales de manera segura, global y transparente. La principal revolución y transformación que la tecnología Blockchain ofrece es la capacidad de gestionar activos digitales y facilitar su transferencia bajo un esquema descentralizado, sin la necesidad de entidades centrales que garanticen el cumplimiento de las transacciones o proporcionen confianza en el sistema a los usuarios. Se trata de un sistema que permite que partes que no confían plenamente unas en otras puedan mantener un consenso sobre la existencia, estado y evolución de una serie de factores compartidos. El consenso es precisamente la clave de un sistema Blockchain porque es el fundamento que permite que todos los participantes en el mismo puedan confiar en la información que se encuentra grabada en él. Desde un enfoque técnico, este sistema basado en la confianza y el consenso se construye a partir de una red global de ordenadores que gestionan una gigantesca base de datos, el registro y transferencia de valor y su modelo descentralizado es útil, entre otros, en múltiples

aplicaciones financieras, de gobierno y logística. En muchas situaciones esta tecnología se vincula a Smart Contracts (contratos inteligentes) por medio de un software que contiene las cláusulas del contrato, almacenado de modo que se ejecuta automáticamente en cuanto se dan las condiciones establecidas. Ello permite ahorrar trabajo burocrático y costo (Preukschat, 2017),(Cuellar & Medina, 2018),(Bartolomé & Lindín, 2018).

2.1.1 Elementos básicos del Blockchain

Según (Preukschat, 2017) los elementos básicos de un Blockchain son:

- **Nodo:** Es un computador personal o, según la diversidad de la red, una mega computadora. Todos los nodos deben usufructuar del mismo software/protocolo para comunicarse entre sí. De otro modo no podrán conectarse ni formar parte de la red Blockchain, sea ésta pública, privada o híbrida. En el caso de una Blockchain pública estos nodos no se identifican y para el caso de una Blockchain privada los nodos se conocen entre sí, pudiendo ser iguales.
- **Protocolo estándar:** Existen protocolos conocidos para la comunicación en red entre computadores (nodos), como el TCP/IP para internet o el SMPT para el intercambio de correos electrónicos. El protocolo de una Blockchain funciona bajo el mismo formato, es decir, dispone de un estándar común para definir la comunicación entre los computadores participantes en la red.
- **Red entre pares P2P (peer-to-peer en inglés):** Se trata de una red de computadores (nodos) conectados directamente en una misma red.
- **Sistema Descentralizado:** La disimilitud con un sistema centralizado radica en que aquí todos los computadores conectados tienen el control de la red debido a que son iguales entre sí; es decir, no existe jerarquía entre los nodos, esto específicamente para una Blockchain pública. En una privada sí puede existir jerarquía.
- **Consenso:** Parte imprescindible entre usuarios de una cadena de bloques. Este consenso se apoya en un protocolo común que verifica y confirma las transacciones realizadas, asegurando la irreversibilidad de las mismas. De igual modo, este consenso debe proporcionar a todos los usuarios una copia inalterable y actualizada de las operaciones realizadas.

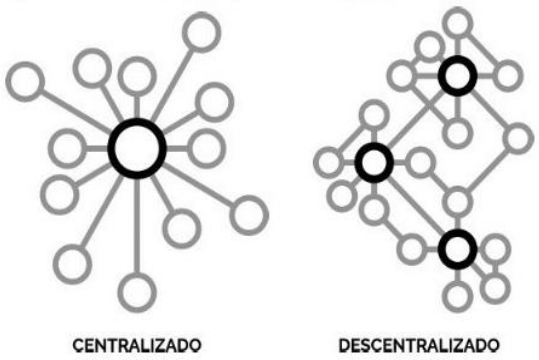


Ilustración 1. Modelo de red centralizada y descentralizada

Fuente: *(Preukschat, 2017)*

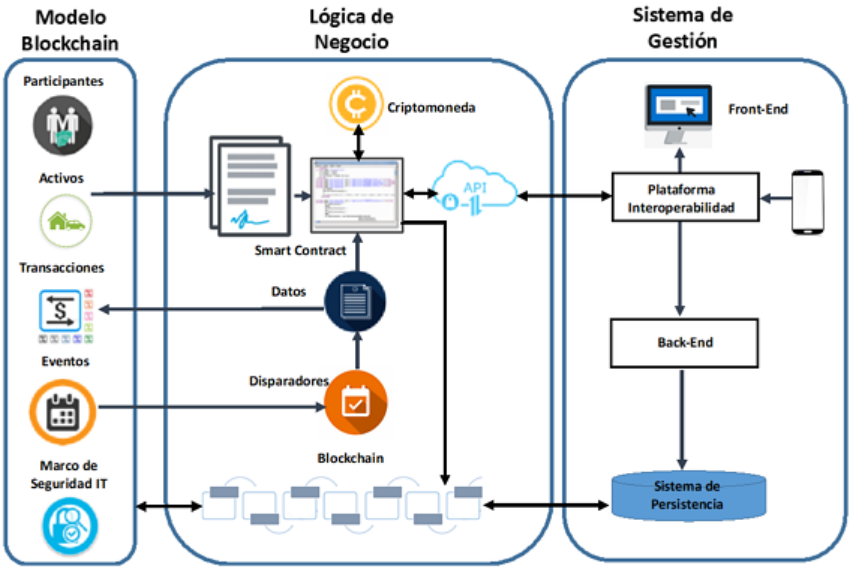


Ilustración 2. Entorno colaborativo Blockchain

Fuente: *(López, 2018)*

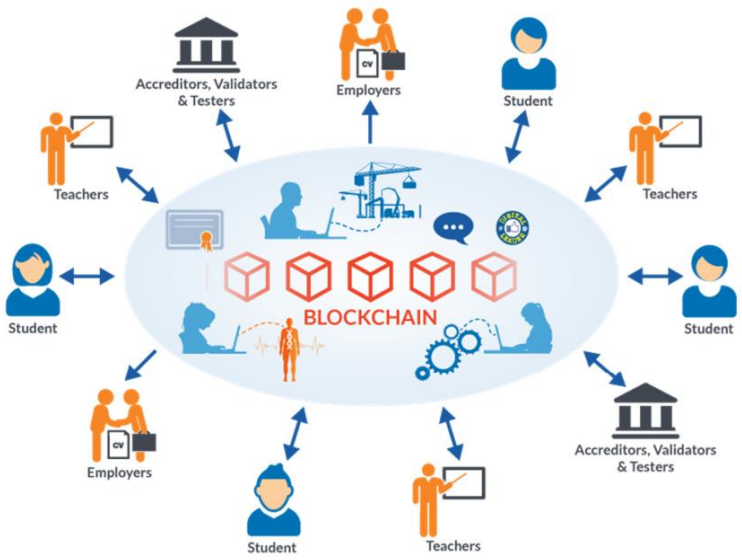


Ilustración 3. Blockchain en Educación Superior

Fuente: *(Poveda, Algunos aspectos sobre blockchains y smart contracts, 2018)*

2.2 Blockchain pública y privada

Tabla 1. Tipos de Arquitectura Blockchain

Basado en (Slater, 2018) se describen las características de una Blockchain pública y privada por medio de la siguiente tabla:

Arquitectura basada en lectura, escritura o confirmación de permisos otorgados a los participantes

	Tipo de Blockchain	Sin permiso	Con permiso
Arquitectura basada en la propiedad de la infraestructura de datos	**PÚBLICA**	• Cualquiera puede unir, leer, escribir y confirmar. • Alojada en servidores públicos. • Anónima altamente resistente. • Baja escalabilidad.	• Cualquiera puede unirse y leer. • Únicamente participantes autorizados y conocidos pueden escribir y confirmar. • Escalabilidad Media.
	PRIVADA	• Únicamente participantes autorizados pueden escribir y confirmar. • Alojada en servidores privados. • Alta escalabilidad.	• Únicamente participantes autorizados pueden escribir y confirmar. • Únicamente el operador de red puede escribir y confirmar. • Escalabilidad muy alta.

Fuente: Recuperado de: Slater, W. (2018). *Introduction to Blockchain*. (November). https://doi.org/10.13140/RG.2.2.21182.54085

La mayoría de Blockchains comerciales usarán la arquitectura privada y con permisos para optimizar la apertura de la red y escalabilidad.

2.3 Contrato Inteligente (Smart Contract)

Existen varias definiciones relacionadas con los Smart Contracts desde la más simple hasta una compleja o específica, a continuación, citamos a algunos autores:

El término, Smart Contract, hace referencia a cualquier contrato que se ejecuta por sí mismo automáticamente sin que medien terceros entre los participantes individuales. Los smart contracts se escriben como programas informáticos en lugar de como lenguaje legal sobre documentos impresos. El programa puede definir reglas y consecuencias estrictas del mismo modo que lo haría un documento legal tradicional, pero a diferencia de los contratos tradicionales, también puede tomar información como input, procesarla según las reglas establecidas en el contrato y adoptar cualquier medida que se requiera como resultado de ello (Tuesta, Alonso, & Vargas, 2015).

"Los contratos inteligentes o smart contracts son programas informáticos que pueden ejecutarse correctamente mediante una red de nodos que desconfían mutuamente, sin la necesidad de una autoridad externa confiable. Dado que los contratos inteligentes manejan y transfieren activos de considerable valor, además de su correcta ejecución, también es crucial que su implementación sea segura contra ataques que tengan como objetivo el robo o la manipulación de los activos" (Atzei, 2017).

"Los contratos inteligentes son contratos digitales que permiten que los términos estén supeditados a un consenso descentralizado, a prueba de falsificaciones y, por lo general, a través de la ejecución automatizada" (Cong, Edmans, Park, & Hara, 2018).

Como se definió anteriormente, un contrato inteligente es simplemente un script autoejecutable, basado en condiciones, Las normas y condiciones para la ejecución de las transacciones son básicamente incrustadas en cada uno de los clientes de la red Blockchain. Por lo tanto, cada máquina cliente en la red entiende lo que se espera de la transacción que se está ejecutando, y los nodos similares también entienden qué esperar de las transacciones de otras máquinas en la red. Formalmente hablando un contrato inteligente es un código informático que actúa como un acuerdo vinculante entre dos o más partes cualesquiera, sin necesidad de un intermediario, y cuyas cláusulas se programan previamente otorgándole la capacidad de auto ejecutarse y validar el cumplimiento de las condiciones de las cláusulas. Dicho de otra forma, cualquier contrato inteligente debe ser totalmente digital, tener la capacidad sobre activos digitales, poder validar el cumplimiento de condiciones acordadas y ejecutarse de forma autónoma y automática. Los contratos inteligentes pueden aplicarse en las Blockchains privadas o públicas. Monax (antes Eris) o Corda de R3 son algunas de las plataformas con las que es posible utilizarlos en entornos empresariales o privados, mientras que en el entorno público tenemos Ethereum, cuyo origen fue pensado en aplicaciones sobre la Blockchain en sentido más amplio. (Augier, 2017),(Allam, 2019).

2.4 Propiedades de un Contrato Inteligente

(miethereum, LA MONEDA: EL ETHER (ETH), 2018) y (Palacios V. M., 2018)
Mencionan la siguientes propiedades:

- **Confiable:** Los contratos inteligentes van directo a la cadena de bloques. Es decir que: 1) está encriptado; solo las personas implicadas pueden leerlo, y 2) permite la interacción entre personas que no se conocen entre sí sin que haya riesgo de estafa.
- **Determinista:** Dada una entrada, todos los nodos distribuidos simultáneamente deben producir el mismo resultado. Eso implica que el código no debería tener ninguna aleatoriedad.
- **Seguridad:** Al basarse en la cadena de bloques pública de Ethereum no se pueden perder. Todo queda registrado de forma inmutable. El proceso de ejecución descentralizado elimina el riesgo de manipulación, ya que la ejecución es gestionada automáticamente por toda la red, en lugar de por una parte individual.
- **Veraz:** Una vez implementado se obtendrá una dirección única. Antes de usarse, las partes interesadas pueden ver y verificar el código para una mayor seguridad. Algo esencial en los contratos del mundo real.

Una parte fundamental del contrato es su capacidad para validar el cumplimiento de las condiciones.

Ilustración 4. ¿Cómo funcionan los Contratos Inteligentes?

Fuente: *(PLATFORM, 2017)*

Tabla 2. Seis puntos en el diseño de la anatomía de un contrato inteligente

La tabla 2 describe los puntos más relevantes sobre el comportamiento y estructuración de un contrato inteligente.

Acuerdo de Indentificación	• Identificar oportunidades de cooperación para múltiples partes. • Acuerdos potenciales sobre transferencia de derechos y permutas de activos.
Configuración de Condiciones	• Evento basado en disparadores condicionales como desastre natural. • Disparadores condicionales temporales (aniversario,finalización).
Codificando de lógica del negocio	• Lógica de codificación completamente automatizada que se dispara donde se cumplen ciertas condiciones lógicas.
Firma Digital	• Seguridad de autenticación y verificación mensajes entre partes relacionadas con un contrato inteligente.
Proceso de Ejecución	• Una vez que el consenso sobre autenticación y verificación es alcanzado, el contrato inteligente se ejecuta y los resultados son almacenados para compilarlos y auditarlos.
Actualización de Red	• Después que el contrato es ejecutado, cada nodo de la blockchain se actualiza con el mismo estado, por ende las nuevas actulizaciónes únicamente pueden ser agregadas.

Fuente: Recuperado de: Mukhopadhyay, M. (2018). *Ethereum Smart Contract Development.* Birmingham: Packt Publishing Ltd.

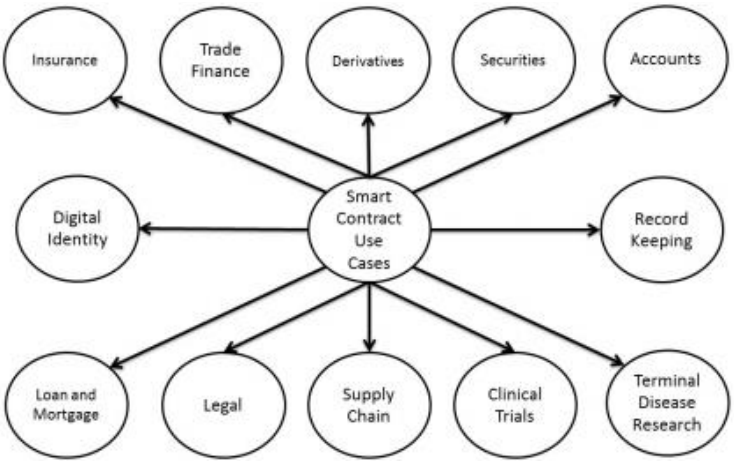


Ilustración 5. Casos de uso de los Smart Contracts

Fuente: *(Mukhopadhyay, 2018).*

2.5 Algoritmo de consenso

Los términos algoritmo y protocolo son aplicados indistintamente dentro de una cadena de bloques, sin embargo, no representan lo mismo. Para simplificar, podemos definir un protocolo como las reglas primarias de una Blockchain; y el algoritmo, como el mecanismo a través del cual dichas reglas serán implementadas (Academy, 2018).

(Rodriguez, 2018) Expone la siguiente definición técnica de un algoritmo de consenso:

Son procesos de toma de decisiones para un grupo, en los cuales los individuos del grupo construyen y apoyan la decisión que funcione mejor para cada uno. Es una forma de resolución en la cual los miembros deben apoyar la decisión tomada por mayoría. En términos simplificados, es una metodología para tomar decisiones dentro de un grupo. Los algoritmos de consenso no solo están de acuerdo con la mayoría, sino que también están de acuerdo con aquel que beneficie a toda la red. En resumen, son métodos usados para crear igualdad y equidad en el mundo de internet. Estos algoritmos son la base de todas las cadenas de bloques (Blockchain) y la parte más importante de estas plataformas, básicamente se enfocan en realizar la verificación de que todas las transacciones que ingresen a la cadena de bloques no sean corruptas, es decir, que no existan errores en los datos informáticos que se producen durante la transmisión, recuperación e introducción de cambios no deseados a los datos originales durante una transacción, proporcionando así integridad y seguridad de los datos.

Los algoritmos de consenso tienen como objetivo principal alcanzar una meta específica por cualquier medio. Por eso, en el caso de que existieran resultados contradictorios en un sistema distribuido; la mejor solución es usar algoritmos de consenso para una mejor salida. Estos algoritmos representan la forma en la que se llega a un acuerdo. Sin embargo, no puede haber ningún sistema descentralizado sin algoritmos de consenso comunes. Ni siquiera importará si los nodos confían entre sí o no. Tendrán que ir por ciertos principios y llegar a un acuerdo colectivo. Para hacerlo, tendrán que revisar todos los algoritmos de consenso. Hasta el momento no se han encontrado ningún algoritmo específico que funcione para cada tecnología de Blockchain.

2.6 Lista de Algoritmos de consenso

(Rodriguez, 2018) Expone la siguiente lista de algoritmos de consenso:

- Prueba de trabajo (PoW)
- Prueba de participación (PoS)
- Prueba de participación delegada (DPoS)
- Prueba de participación arrendada (LPoS)
- Prueba de tiempo transcurrido (PoET)
- Práctica de tolerancia a faltas bizantinas
- Tolerancia a faltas bizantina simplificada
- Tolerancia a faltas bizantina delegada
- Grafo de acíclico dirigido
- Proof-of-Activity
- Proof-of-Importance
- Proof-of-Capacity
- Proof-of-Burn
- Proof-of-Weight

Según (Academy, 2018) existen varios tipos de algoritmos de consenso. Las implementaciones más comunes son PoW y PoS. Cada uno tiene sus ventajas y desventajas en lo que se refiere al equilibrio entre seguridad, funcionalidad y escalabilidad. Como se indica a continuación:

- **Prueba de Trabajo (PoW):**

PoW fue el primer algoritmo de consenso creado el cual es empleado por Bitcoin y muchas otras criptomonedas. El algoritmo Proof of Work es una parte esencial del proceso de minado.

(Albuixech, 2018) Menciona: El proceso de minado en PoW implica numerosas pruebas de hashing, por lo que más poder computacional significa más intentos por segundo. En otras palabras, los mineros con una tasa de hash elevada tienen más posibilidades de hallar una solución válida para el siguiente bloque (también llamado hash block). El algoritmo de consenso PoW se asegura de que los mineros sean capaces de validar un nuevo bloque de transacciones y añadirlo al blockchain, sólo si los nodos distribuidos de la red alcanzan consenso y aceptan el hash block provisto por el minero como una prueba de trabajo válida.

- **Prueba de Participación (PoS):**

El algoritmo de consenso PoS fue desarrollado en 2011 como una alternativa a PoW. Tanto PoS como PoW comparten objetivos comunes, pero presentan algunas diferencias y particularidades fundamentales. Específicamente, en lo relativo a la validación de nuevos bloques.

Según (Wenting Li, 2017): En resumen, el algoritmo PoS reemplaza el minado PoW por un mecanismo de bloques validados conforme al “stake” (cantidad de monedas acumuladas) de los participantes. El validador de cada bloque, (también llamado forger o minter en inglés), se determina por la inversión en la criptomoneda, y no por la

cantidad de poder computacional destinado. PoS puede implementar el algoritmo de distintas maneras, pero en general, la blockchain será asegurada por selección pseudoaleatoria que toma en consideración el capital del nodo y edad de la moneda (tiempo que ha permanecido inmóvil o depositada), junto con un factor de aleatorización.

2.7 Ethereum

"Ethereum es una plataforma descentralizada que ejecuta contratos inteligentes: aplicaciones que se ejecutan exactamente según lo programado, sin posibilidad de inactividad, censura, fraude o interferencia de terceros" (Foundation, s.f.).

Ethereum es un marco para criptomonedas que utiliza tecnología de cadena de bloques para proporcionar una plataforma de computación global abierta, llamada Ethereum Virtual Machine (EVM). Los programadores no suelen escribir código EVM; en su lugar, pueden programar en un lenguaje similar a JavaScript, llamado Solidity. Dado que el objetivo principal de EVM es ejecutar contratos inteligentes que gestionen y transfieran activos digitales llamados Ether (cadenas de bloques basadas en la prueba de trabajo), estos contratos se ejecutan en una cadena de bloques personalizada con una infraestructura global compartida robusta y potente que puede *mover el* valor y representar la propiedad de la propiedad.(Bhargavan et al., n.d.).

"Por poner un ejemplo sencillo, un usuario de Ethereum podría crear un contrato inteligente para enviar una cantidad establecida de ether a un amigo en una fecha determinada. Escribirían este código en la cadena de bloques y cuando el contrato se complete (es decir, cuando se alcance la fecha acordada) los ether se enviarán automáticamente, esto permite a los desarrolladores crear mercados, almacenar registros de deudas o promesas, mover fondos de acuerdo con instrucciones dadas en el pasado (como un testamento o un contrato de futuros) y muchas otras cosas que aún no se han inventado, todas sin un intermediario o riesgo de contraparte" (IG.com, 2003).

2.8 Ether

Es un token utilizado para pagar recursos computacionales necesarios para ejecutar aplicaciones o programas. Al igual que Bitcoin, Ether es un activo digital (criptomoneda), y al igual que el dinero en efectivo, no requiere que un tercero gestione una transacción. Pero en lugar de operar como moneda digital o pago, busca proporcionar "combustible" a las aplicaciones descentralizadas en la red, para publicar, eliminar o modificar algo en la red Ethereum, se debe pagar una tarifa de transacción Ether (ETH) para que la red procese ese cambio (Núria Porxas, 2018).

2.9 Gas

Esta solución llamada gas, se la puede asimilar como el estimulante que necesitan las transacciones ejecutarse. El parámetro que la EVM necesita para ejecutar las transacciones es el STARTGAS, también conocido como transaction gaslimit (límite de gas). La transaction gaslimit es la cantidad de gas a enviar en una transacción (parámetro modificable por el usuario) (Palacios V. M., 2018),.

2.10 Máquina Virtual Ethereum (EVM)

Con referencia a (Buntinx, 2017) se describe a continuación el funcionamiento de la Máquina Virtual de Ethereum:

La Máquina Virtual Ethereum está diseñada para servir como un entorno de ejecución para contratos inteligentes basados en Ethereum, se centra en proporcionar seguridad y ejecutar códigos no confiables en computadoras de todo el mundo. Este proyecto se centra en la prevención de ataques de denegación de servicio, que se han vuelto algo comunes en el mundo de las criptomonedas. Además, el EVM garantiza que los programas no tengan acceso al estado del otro asegurando que la comunicación se puede establecer sin ninguna interferencia potencial.

La EVM está completamente aislada del resto de la red principal, por tanto, es un entorno de prueba perfecto. Cualquier empresa que busque crear un contrato inteligente puede hacerlo utilizando el EVM, sin que esto afecte las operaciones principales de la cadena de bloques. Además, se podría considerar al EVM como un "entorno de aprendizaje" para construir contratos inteligentes más grandes, mejores y robustos.

Cabe mencionar que cada nodo Ethereum en la red ejecuta su propia instancia de EVM y es capaz de ejecutar las mismas instrucciones. Es una puerta de entrada para construir contratos inteligentes adecuados, tanto para principiantes como para expertos que buscan obtener un enfoque práctico con el lenguaje Solidity. Es una buena noticia saber que existen entornos de espacio aislado para la tecnología de contratos inteligentes. Aunque nadie niega el potencial de esta tecnología, todavía estamos en las primeras etapas de descubrir de qué es capaz. La descentralización de algunas operaciones diarias del mundo se puede lograr con contratos inteligentes, y la Máquina Virtual Ethereum jugará un papel importante en este proceso.

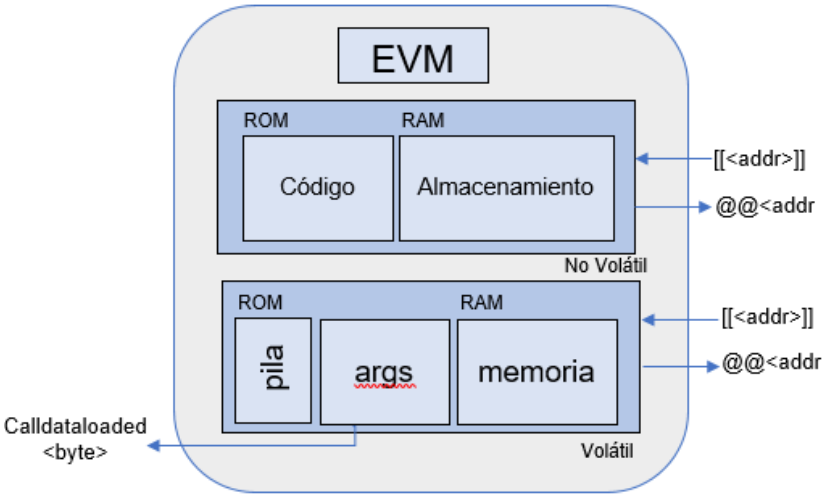


Ilustración 6. Arquitectura Máquina Virtual Ethereum

Fuente: *(Kasireddy, 2017)*

2.11 Solidity

"Solidity es un lenguaje de alto nivel orientado a objetos para implementar contratos inteligentes. Los contratos inteligentes son programas que rigen el comportamiento de

las cuentas dentro del estado de Ethereum. fue influenciado por C ++, Python y JavaScript y está diseñado para apuntar a la Máquina Virtual Ethereum (EVM)" (Solidity, 2016).

(Modi, 2018) Menciona: Solidity es un lenguaje de programación muy cercano a JavaScript con ciertas similitudes también con C. Es un lenguaje de programación estático, sensible a mayúsculas, minúsculas y orientado a objetos (OOP), es decir, admite características limitadas de orientación objetada. Esto significa que los tipos de datos variables deben definirse y conocerse en tiempo de compilación. Las funciones y variables deben escribirse en OOP de la misma manera que se definen. En Solidity, por ejemplo, Cat es diferente de CAT, cat, o cualquier otra variación de esta expresión. La terminación de la declaración en Solidity es el punto y coma. El código de Solidity está escrito en archivos que tienen la extensión .sol. Un contrato de Solidity no es más que una colección de código (funciones y variables) que residen en una dirección especifica de la blockchain de Ethereum. Cada contrato incluirá variables de estado, funciones, modificadores, eventos, estructuras y herencias de otros contratos.

2.12 Composición de Solidity

Según (Modi, 2018) un archivo de Solidity se compone de las siguientes cuatro construcciones de alto nivel:

- **Pragma:** Es generalmente la primera línea de código dentro de cualquier archivo de Solidity, es una directiva que especifica la versión del compilador que se utilizará para el archivo actual (versión del compilador, orientar código). *Ejemplo de código: pragma Solidity <<version number>>;*
- **Comentarios:** Existen tres tipos de comentarios en Solidity: Comentarios de una sola línea (//), Comentarios multilínea (/* */), Especificación Natural Ethereum (/// o /**=> inicio y */ => fin);
- **Import:** Palabra clave que ayuda a importar otros archivos de Solidity y acceder a ese código dentro del archivo y del código Solidity. (Modularidad).
- **Contratos / biblioteca / interfaz:** Además de pragma, import y comentarios, podemos definir contratos, bibliotecas e interfaces a nivel global o superior, las mismas que se pueden declarar dentro de un mismo archivo Solidity, estas palabras distinguen entre mayúsculas y minúsculas.

2.13 Tipos de Datos

(Palacios V. M., 2018) Solidity proporciona los siguientes tipos de datos para su uso:

- **Address:** Ocupa un valor de 20 bytes. Es una variable específica para guardar las direcciones de contratos o de usuarios de Ethereum.
- **Integer:** int / uint, guardan valores numéricos con signo y sin signo respectivamente.
- **String:** Contiene una cadena de caracteres.
- **Booleano:** Puede tener dos posibles valores (true o false).
- **Bytes:** Bytes1 hasta bytes32, son arrays de bytes fijos o bytes con tamaño variable.
- **Enumeración:** Se pueden usar para crear tipos de datos personalizados con un finito conjunto de valores. La siguiente variable llamada Estado, puede tener tres

valores. Ej: enum Estado {Creado, Modificado, Borrado}

2.14 Eventos

Los eventos en Solidity se utilizan principalmente para informar a la aplicación que realiza la llamada sobre el estado actual del contrato a través de la función de registro de EVM, a fin de notificar cambios en los contratos y con el propósito de que las aplicaciones los usen para ejecutar su lógica dependiente. Se pueden utilizar los valores en los parámetros para registrar información o ejecutar lógica condicional. La información del Evento y sus valores se almacenan como parte de transacciones dentro de bloques. Un evento se declara usando la palabra clave event, seguido por un identificador, una lista de parámetros y termina con un punto y coma. Ej: *event ageRead(address,int)* (Modi, 2018).

2.15 Funciones

Las funciones son el corazón de Ethereum y Solidity, cuando una función se invoca en un contrato, da lugar a la creación de una transacción. Las funciones son el mecanismo para leer y escribir valores de las variables de estado, pueden aceptar parámetros, ejecutar su lógica y, opcionalmente, devolver valores al llamante. Pueden ser nombradas con anónimos. Solidity permite tener sólo una función anónima en un contrato. Ej: *function getAge(address_Identifier)onlyBy()payable external returns (uint);* (Modi, 2018).

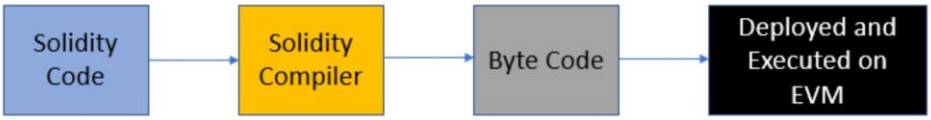


Ilustración 7. Proceso para ejecutar Solidity en la EVM

Fuente*:*(Modi, 2018)

3 METODOLOGÍA EXPERIMENTAL

3.1 Método Experimental

(Robles, 2019) Explica y define al método experimental de la siguiente manera:

El método experimental es un conjunto de técnicas que se utilizan para investigar fenómenos, adquirir nuevos conocimientos o corregir e integrar conocimientos previos. Se utiliza en la investigación científica y se basa en la observación sistemática, toma de medidas, experimentación, formulación de pruebas y modificación de hipótesis. Este método general se lleva a cabo no solo en biología, sino también en química, física, geología y otras ciencias.

A través de este método llamado también inductivo los científicos intentan predecir y quizás controlar eventos futuros basados en el conocimiento presente y pasado. Es el más utilizado dentro de la ciencia por los investigadores. Se caracteriza por el hecho de que los investigadores pueden controlar deliberadamente las variables para delimitar las relaciones entre ellas. Estas variables pueden ser dependientes o independientes, siendo

fundamentales para recopilar los datos que se extraen de un grupo experimental, así como su comportamiento.

3.2 Fases del Método Experimental

1. **Observación:** El punto de partida son siempre los hechos o fenómenos físicos que la ciencia pretende explicar. Se parte, pues, de la observación: la caída libre, el movimiento de los astros, la temperatura, la presión de un gas, etc.
2. **Formulación de hipótesis:** La siguiente fase consiste en formular una hipótesis, es decir, una posible explicación que relacione entre sí dichos factores.
3. **Contrastación:** Consiste en la experimentación, medio por el cual se confirma la hipótesis planteada o se desecha. Por tanto, la manera de realizar la contrastación es la de efectuar un experimento.
4. **Verificación:** Es el medio para corroborar o refutar una hipótesis, dependiendo de los resultados de la experimentación, la hipótesis se acepta o se rechaza. Si la hipótesis es verificada, entonces se acepta como una ley de la naturaleza (al menos hasta que no sea desmentida por un nuevo experimento). Si no se verifica, se rechaza y se formula una nueva hipótesis.
5. **Formulación de la ley correspondiente**: Las hipótesis anticipan lo que se espera que ocurra en un experimento. Si la hipótesis es verificada, entonces se puede construir una ley, que es una generalización de la hipótesis a todo un ámbito de la realidad. Dicha "generalización" implica una "inducción".
6. **Inclusión de la ley en una teoría:** Si la hipótesis es verificada, entonces se puede construir una ley**.** La teoría es un conjunto de leyes verificadas que intentan dar una explicación de cómo son las cosas, en general.

4 CÁLCULOS Y RESULTADOS

4.1 Diagrama de la Arquitectura

La ilustración 9, describe cómo el navegador cliente se comunica con su propia instancia de la aplicación. No hay un servidor central al que el cliente tenga que conectarse. Esto significa que toda persona que quiera interactuar con una aplicación descentralizada necesitará una copia completa de la cadena de bloques que se ejecuta en el computador o teléfono, es decir, antes de que podamos utilizar la aplicación, tenemos que descargar la cadena de bloques completa y luego usar la aplicación. Esto tiene la ventaja de no depender de un solo servidor central que puede desaparecer mañana o comenzar a cobrarnos comisiones de terceros. Para crear DApps basadas en la web, Ethereum tiene una biblioteca muy útil de JavaScript llamada web3.js, que se conecta a nuestro nodo de Blockchain mediante llamadas a procedimientos remotos (RPC).

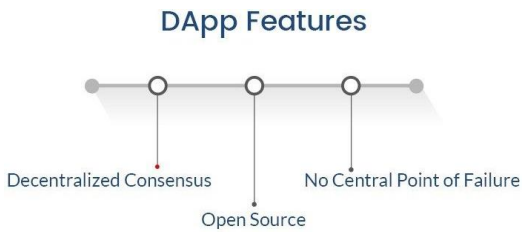


Ilustración 8. Características comunes aplicaciones descentralizadas

Fuente: *(Vásquez, 2018)*

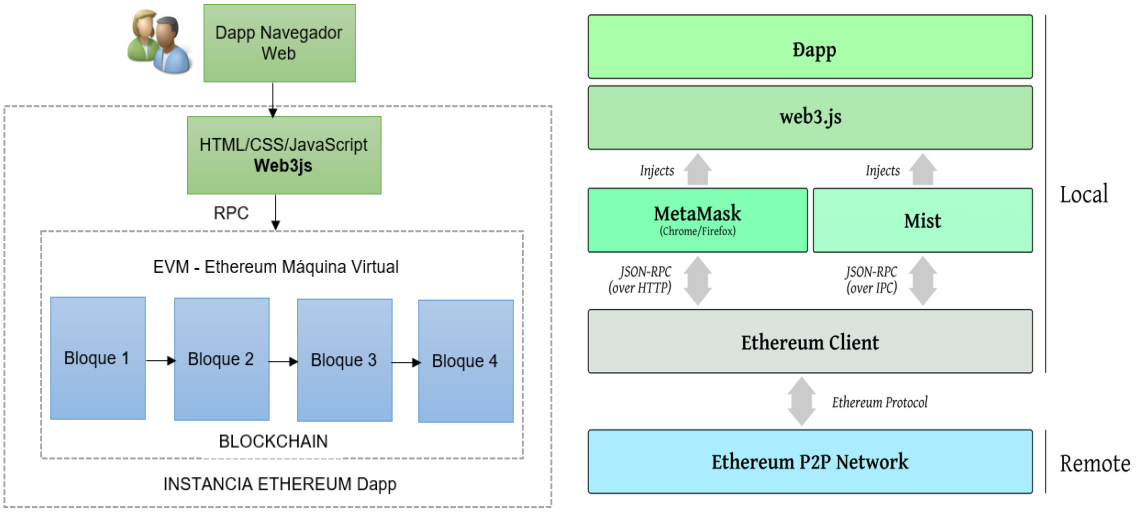


Ilustración 9. Arquitectura de Alto Nivel DApp

Adaptado de: *(Mukhopadhyay, 2018)**,** (Gardiner, 2018)*

4.2 Análisis de características de las cadenas de bloques

Tabla 3. Comparación de características Blockchain

La tabla 3 describe las propiedades más importantes de las herramientas utilizadas por las cadenas de bloques, dando como resultado final la elección de Ethereum.

Comparación de las características de las cadenas de bloques

	Bitcoin	**Ethereum**	**Hyperledger**
Restricciones de permiso	Sin Permiso	Sin Permiso	Autorizado
Acceso público restringido a los datos	Público	Público o Privado	Privado
Consensos	Prueba de Trabajo	Prueba de Trabajo	PBFT
Escalabilidad	Alta escalabilidad de nodos, Escalabilidad de bajo rendimiento.	Alta escalabilidad de nodos, Escalabilidad de bajo rendimiento.	Baja escalabilidad de nodos, Escalabilidad de alto rendimiento.
Regulación Centralizada (Gobernanza)	Baja toma de decisión descentralizada por parte de la comunidad/mineros.	Media, Grupo de desarrollo céntrico, pero con proceso EIP.	Bajo modelo de gobierno abierto basado en el modelo Linux.
Anonimato	Pseudoanonimato, no cifrado de datos de transacción.	Pseudoanonimato, no cifrado de datos de transacción.	Pseudoanonimato, cifrado de datos de transacción.

Moneda Nativa	Si, Bitcoin alto valor	Si, Ether	No
Scripting	Posibilidad Limitada, Scripting basado en pila.	Alta posibilidad, máquina virtual Turing-completa, alto nivel de soporte de lenguaje (Solidity).	Alta posibilidad, Turing -completa, Scripting de código de cadena (chaincode), Alto nivel de lenguaje Go.

Fuente: Friebe, T. (30 de Octubre de 2017). *Bitcoin, Ethereum, and Hyperledger Fabric—which one wins?* Recuperado de https://medium.com/blockchainspace/3-comparison-of-bitcoin-ethereum-and-hyperledger-fabric-cd48810e590c

Tabla 4. Tipos de Blockchain

La tabla 4 expone las ventajas de los tipos de cadenas de bloques, obteniendo como resultado final la elección de Ethereum público/privado.

INDICADORES	**Públicos**	**Privados**	**Federados**
	Ethereum, LiteCoin, Bitcoin	**Hyperledger, Quorum, Corda**	**Hyperledger, Quorum, Corda**
Cualquiera puede participar	Aplica	No Aplica	No Aplica
Los participantes actúan como nodos	Aplica	No Aplica	No Aplica
Transparencia	Aplica	A veces	A veces
Único Administrador	No Aplica	Aplica	No Aplica
Más de un administrador	No Aplica	No Aplica	Aplica
No hay administrador	Aplica	No Aplica	No Aplica
Ningún participante tiene más derechos sobre los demás	Aplica	No Aplica	No Aplica
Implementación de Smart Contracts	Aplica	Aplica	Aplica
Recompensa por minado de bloques	A veces	No Aplica	No Aplica
Solución al problema de confianza	Aplica	No Aplica	A veces
Seguridad (Protocolos de consenso)	Aplica	No Aplica	A veces
Seguridad basada en Hash	Aplica	A veces	A veces

Fuente: Recuperado de: Marcos Allende López, V. C. (28 de Junio de 2018). *Abierto al público, Conocimiento Abierto*. Recuperado de https://blogs.iadb.org/conocimiento-bierto/es/tipos-de-blockchain/

Tabla 5. Comparación Lenguajes de Programación Blockchain

La tabla 5 describe las principales características de los lenguajes de programación usados por las cadenas de bloques.

Lenguaje de Programación	Ventajas	Desventajas
Solidity	- Los contratos inteligentes proporcionan un medio seguro, fácil y confiable para las partes interesadas. - Admite múltiples funciones de tipo seguro a través de la facilitación de ABI (Application Binary Interface). - Similitud con otros lenguajes de programación (JavaScript) - Es uno de los lenguajes más populares usados en la industria del Blockchain.	- Una vez que se construye un contrato, no se pueden agregar características adicionales. - Los datos están disponibles a partir de transacciones. No hay otra fuente para la información.
Java	- Independiente del sistema operativo. - Fuertemente tipado. - Fuerte soporte para la programación orientada a objetos. - Gran comunidad y colección de bibliotecas. - No hay problema con la asignación de memoria. - Fácil limpieza de la memoria.	- Necesita Java Virtual Machine para correr. - Más lento que C ++ o Go. - Dificultad de Aprendizaje mayor a JavaScript o Python.
Python	- Funciona en diferentes plataformas (Windows, Mac, Linux, Raspberry, otras). - La sintaxis permite a los desarrolladores escribir programas con menos líneas en comparación con otros lenguajes de programación.	- Utilizado principalmente como lenguaje de servidor. - Contexto no tan inteligente. - Las bibliotecas no siempre están documentadas de manera suficiente.
Java Script	- Orientado a objetos - Basado en prototipos - Soporta programación funcional	- Dinámico. - La interpretación depende del navegador.

	- No requiere soporte de plug-in	- Código abierto.

Fuente: *Adaptado de:* (Ramos, 2016)**,** (Fabisiak, 2018)**,** (Goyal, 2019)**,** (Croix, 2019)**,** (Blockgeeks, 2019)**.**

4.3 Características Funcionales y No Funcionales

(Quiroga, 2014) enuncia las siguientes definiciones sobre requerimientos:

Requerimiento: Un requerimiento es una característica que un sistema DEBE tener o es una restricción que un sistema DEBE satisfacer para ser aceptada por el cliente.

Requerimiento Funcional: Describe la interacción entre el sistema y su ambiente independientemente de su implementación, el ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema.

Requerimiento No Funcional: Requisitos que no hacen referencia directamente a las funciones específicas provistas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad. En síntesis, no hablan de "lo que" hace el sistema, sino de "cómo" lo hace (Medium, 2018).

4.3.1 Requerimientos Funcionales

Tabla 6. Requerimientos Funcionales

La tabla 6 expone los requerimientos funcionales que la cadena de bloques involucra en lo que respecta a la implementación de un contrato inteligente dentro de la misma.

RF-1	***REGISTRO DE ESTUDIANTE***
DESCRIPCIÓN	Para registrar un estudiante en la cadena de bloques se necesita obligatoriamente de su número de cédula, facultad, nombres, nivel, asignaturas.
ENTRADA	Número de cédula.
SALIDA	número de cédula, facultad, nombres, nivel, asignaturas
CONDICIÓN	Conexión con nodo de Ethereum.
PRERREQUISITO	Cumplimiento de todos los campos de Entrada.
POSTRREQUISITO	Información por pantalla del registro del estudiante.
RF-2	***ENVÍO DE REGISTRO A RED DE NODOS***
DESCRIPCIÓN	El registro (transacción) realizado en la cadena de bloques se envía a una red P2P o red de prueba (Test net) formada por nodos.
ENTRADA	Id transacción o dirección de cuenta Ethereum.
SALIDA	Mensaje de información de procesamiento de transacción.
CONDICIÓN	Conexión con otros nodos de Ethereum.
PRERREQUISITO	-
POSTRREQUISITO	Muestra Información sobre el resultado de la transacción.
EXCEPCIONES	Si no se realizó la transacción, se informa al usuario.

RF-3	***VERIFICACIÓN DE TRANSACCIÓN (FIRMA Y RECURSOS)***
DESCRIPCIÓN	La red de nodos verifica si la transacción está firmada y si se tienen los recursos para enviarla.
ENTRADA	Firma.
SALIDA	-
CONDICIÓN	Conexión con nodo local Ethereum y otros nodos de la red.
PRERREQUISITO	Conexión de los nodos en la red P2P o red de prueba (Testnet).
POSTRREQUISITO	Información por pantalla del registro del estudiante.
EXCEPCIONES	Si no se realizó el registro, se informa al usuario.
RF-4	***CREACIÓN DE CONTRATO INTELIGENTE***
DESCRIPCIÓN	La creación del contrato inteligente se realiza a partir de la confirmación de la firma de la transacción.
ENTRADA	-
SALIDA	-
CONDICIÓN	Conexión con nodo local Ethereum.
PRERREQUISITO	Confirmación de la verificación de la transacción.
POSTRREQUISITO	Código o dirección de contrato.
RF-5	***AGREGAR CONTRATO A LA CADENA DE BLOQUES EVM***
DESCRIPCIÓN	Una vez creado el contrato se agrega a la cadena de bloques y se replica en todos los nodos de la red.
ENTRADA	Código de almacenamiento contrato.
SALIDA	Código Final.
CONDICIÓN	Conexión con nodo local Ethereum y otros nodos de la red.
PRERREQUISITO	Existe un contrato con un código de almacenamiento.
POSTRREQUISITO	El contrato es registrado en la cadena de bloques de forma permanente e inalterable.
EXCEPCIONES	Si no existe un contrato con ese código, se notifica al usuario.
RF-6	***FINALIZAR TRANSACCIÓN***
DESCRIPCIÓN	El contrato queda registrado en la cadena de bloques y es almacenado en la pila de la EVM de Ethereum.
ENTRADA	-
SALIDA	-
CONDICIÓN	Conexión con nodo local Ethereum y otros nodos de la red.
PRERREQUISITO	El contrato debe estar registrado en la cadena de bloques.
POSTRREQUISITO	Disponibilidad del contrato a la parte interesada.
EXCEPCIONES	Si se desean realizar modificaciones en el contrato se debe realizar todo el proceso antes descrito.

Fuente: Autor.

4.3.2 Requerimientos No Funcionales

- **Rendimiento:** La aplicación puede optimizar su funcionamiento en lo que respecta al tiempo medio de verificación de transacciones en la cadena de bloques de Ethereum, debido a la plataforma de desarrollo usada.
- **Coste:** Debido a la tecnología empleada, el código del contrato para sincronizar la información requiere de gas para su ejecución, haciendo primordial la reducción de costes al mínimo.
- **Extensibilidad:** La funcionalidad de la aplicación puede ser ampliada conforme avance el desarrollo de la plataforma. Para nuevas versiones se pueden incluir nuevos modos de verificación mediante la extensión del contrato, consultando información fuera de la cadena de bloques.
- **Almacenamiento:** El almacenamiento en la cadena de bloques es costoso, por ello, la cantidad de datos sincronizados mediante la cadena de bloques se reduce al mínimo evitando añadir contenido multimedia.
- **Usabilidad:** La aplicación ha sido desarrollada pensando en la facilidad de uso. El usuario puede ver por pantalla todas las opciones de las que dispone en una interfaz visual amigable (Navegador Web).

4.4 Selección de Herramientas

Blockchain fundamenta su implementación en el uso de una arquitectura descentralizada, a través del desarrollo de DApps para la ejecución de los diferentes procesos que la misma puede realizar, respecto al manejo, distribución de información y procedimientos de forma independiente. Las herramientas en las cuales nos apoyamos para el desarrollo de una DApp se describen a continuación:

4.4.1 MetaMask

Es un complemento del navegador web (Chrome, Firefox, Opera), que sirve como puente para conectarse a una red de Ethereum e inyectar una instancia web3 al código. Metamask también permite seleccionar diferentes redes de prueba (Rinkeby, Kovan, Ropsten) y cuentas Ethereum. Esta extensión se interconecta por medio del navegador a Ethereum sin necesidad de descargar ningún nodo. Metamask proporciona una interfaz de usuario para administrar identidades en diferentes sitios y firmar cada transacción en una Blockchain (Roldán, 2018), (Palacios V. M., 2018).

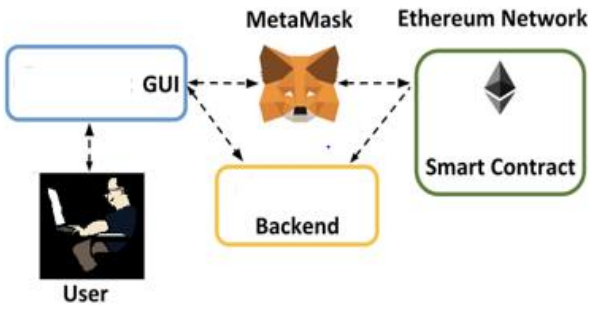


Ilustración 10. Proceso de interacción Ethereum-Metamask

Adaptado de: *(WeTrust, 2017)*

4.4.2 Ropsten Testnet

Ropsten Ethereum, también conocido como "Ethereum Testnet", es una red de prueba que ejecuta el mismo protocolo que Ethereum, además de utilizar métodos de consenso de PoW para generar bloques; es usado con propósitos de prueba antes de implementarse en la red principal (Mainnet). Cuando los desarrolladores crean DApps, a fin de evitar perder dinero pagando el ETH real por los aranceles de transacción y los despliegues de contratos inteligentes, la mejor opción hacer uso de la Red Ropsten (Neto, 2018).

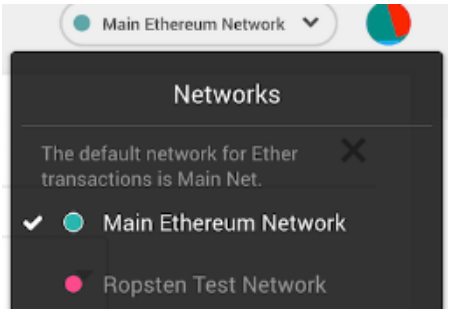


Ilustración 11. Redes de Prueba Metamask

Fuente: *(TruffleCon2019, s.f.)*

4.4.3 Remix IDE

Remix, antes conocido como Browser Solidity, es un contexto de desarrollo online basado en navegador con entorno de pruebas y depuración integrados. Puede instalarse en la máquina física, pero para mayor utilidad se manejará la versión online (Palacios V. M., 2018).

Según (Palacios V. M., 2018) Remix se puede dividir esencialmente en las siguientes partes:

- **Explorador de ficheros:** Lista por defecto todos los contratos guardados en el navegador. Se puede crear, renombrar y añadir nuevos desde el explorador de archivos.
- **Editor de código:** Remix recompila el código cada vez que el código sufre un cambio u otro contrato es seleccionado para editarlo. A medida que se va programado, Remix se recompila para avisar de errores.
- **Terminal:** Nos muestra las acciones que van sucediendo a medida que se interactúa con Remix y los contratos. Cada vez que se ejecuta un contrato muestra información muy relevante como puede ser: dirección del contrato, dirección del creador, coste de la transacción, coste de la ejecución y logs.

- **Menú:** Remix ofrece diferentes herramientas para poder compilar y ejecutar los contratos. Las dos pestañas más importantes que se han de conocer son las de [Compile] y [Run].

Ilustración 12. Entorno Remix IDE

Adaptado de: *(Palacios V. M., 2018)*

Ajustes a realizar para interactuar con el entorno de Remix IDE:

1. **Opción Environment:** Seleccionar el entorno JavaScript VM, puesto que localmente se ejecuta y crea una Blockchain de test e Injected Web3 que permite establecer una conexión con la Blockchain de Ethereum a través de un proveedor (Metamask) para subir el contrato a la red test de Ethereum.
2. **Opción Account:** Listado de cuentas asociadas al entorno de ejecución seleccionado junto con su cantidad de ether asociado.
3. **Opción Gas limit:** Máximo gas permitido en transacciones creadas.
4. **Opción Value:** Cantidad de ether a usar en una transacción por un contrato.

4.4.4 Meteor

Es una nueva infraestructura de JavaScript cuyo objetivo es automatizar y simplificar el desarrollo de aplicaciones web que actúan en tiempo real. Usa la comunicación en tiempo real mediante un protocolo llamado Distributed Data Protocol (DDP), soportado por navegadores modernos que usan WebSockets o en navegadores antiguos que soporten "long polling". En ambos casos, la comunicación navegador-servidor es transparente para el usuario (AWARENESS, 2017).

Meteor proporciona varias ventajas para el usuario además de su reactividad, proporciona dos bases de datos: una base de datos del lado del cliente (cache) y una base de datos MongoDB en el servidor. Cuando un usuario modifica algún dato, el código JavaScript que se ejecuta en el navegador actualiza la base de datos en MongoDB local y luego realiza una solicitud de DDP al servidor. El código actúa de inmediato como si la operación hubiera sido exitosa ya que no necesita esperar la respuesta por el servidor (AWARENESS, 2017).

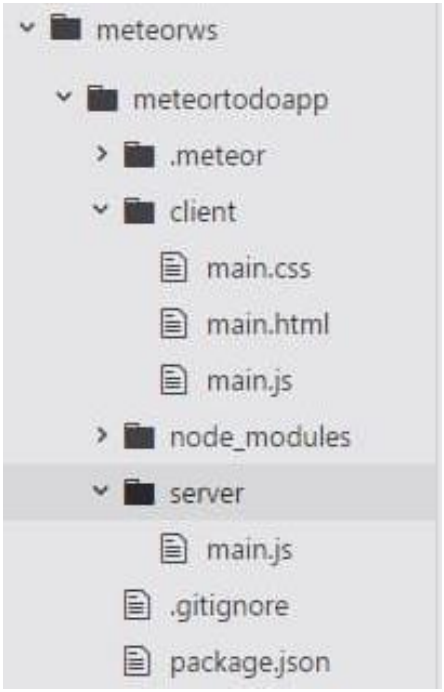


Ilustración 13. Estructura archivos Meteor app

Fuente: *(Mogollón, 2017)**.***

4.5 Desarrollo de Contratos Inteligentes

La proposición y sustento de los smart contracts que se diseñan para la prueba de concepto del registro de matrículas de estudiantes es provisionar de privacidad a los datos que se incorporarán a la blockchain. Para diseñar todas las funcionalidades se crean 3 smart contracts; dichos contratos necesitan obtener información entre ellos (interactuar). El contrato de los Estudiantes se abastece de la información proveniente del contrato de Asignaturas, mientras que el contrato de Información, donde se almacenan todos los datos de los estudiantes, se nutre de la información del contrato de Estudiantes, el cual permite listar los estudiantes existentes.

4.5.1 Estructura de los contratos inteligentes

Tabla 7. Conformación de los contratos inteligentes

La tabla 7 describe la esquematización de los contratos inteligente a implementar.

Estudiante.sol: (Contrato para los estudiantes)		
Estructuras de Datos (Structs)	***Variables (Campos)***	***Función del contrato***
Student	- Identificación - Nombre del estudiante. - Facultad. - Carrera. - Nivel. - Lista de Asignaturas.	Se mapean los estudiantes por su dirección. - Añadir estudiantes. - Comprobar si el estudiante existe. - Obtener un estudiante por su dirección. - Añadir asignaturas a un estudiante. - Eliminar asignaturas de un estudiante. - Obtener lista de las asignaturas de un estudiante.
Subject	- Dirección de la Asignatura. - Nivel de acceso a los datos. - Acceso (True o False).	
Asignatura.sol (Contrato para las Asignaturas)		
Subject	- Código de Asignatura. - Nombre de la Asignatura. - Paralelo. - Número de Matrícula - Créditos. - Lista de Asignaturas. - Dirección del estudiante.	Se mapean las asignaturas por su dirección. - Añadir asignaturas. - Cambiar el nombre. - Ver la información de la asignatura. - Obtener las asignaturas por dirección.
Matrícula.sol (Contrato para la Información de las matriculas)		
Matrícula	- Dirección del estudiante y asignatura para obtener su información.	- Ver matrícula de un estudiante.

Fuente: Adaptado de: Palacios, V. M. (2018). *Explorando la Blockchain de Ethereum y el desarrollo de smart contracts.* Catalunya. Recuperado el 29 de abril de 2019.

Tabla 8. Funciones que componen los contratos inteligentes

La tabla 8 muestra las funciones o métodos que los contratos inteligentes ejecutarán conforme al modelo de negocio a implementar.

Estudiante.sol: (Contrato para los estudiantes)		
Nombre de la Función	***Tipo***	***Explicación***
constructor()	Principal	Obtener la dirección del contrato
addEst()	Principal	Agregar estudiante
getEstBy()	Principal	Desplegar la información del estudiante
updateEst()	Principal	Actualizar datos del estudiante
delEst()	Secundario	Eliminar un estudiante
existEst()	Principal	Verificar si existe un estudiante
addSub()	Principal	Agregar asignatura
delSub()	Principal	Eliminar asignatura
Asignatura.sol (Contrato para las Asignaturas)		
constructor()	Principal	Obtener a la dirección del contrato
addSub()	Principal	Agregar asignatura
getSubBy()	Principal	Desplegar la información de la asignatura
updateSub()	Principal	Actualizar datos de la asignatura
delSub()	Secundario	Eliminar una asignatura
existSub()	Principal	Verificar si existe un estudiante
Matrícula.sol (Contrato para la Información de las matriculas)		
callerContracts()	Principal	Referenciar la dirección del contrato del estudiante y asignatura
getStudentInfo()	Principal	Obtener los datos del estudiante según la dirección del contrato
getSubjectInfo()	Principal	Obtener los datos de la asignatura según la dirección del contrato

Fuente: Adaptado de: Palacios, V. M. (2018). *Explorando la Blockchain de Ethereum y el desarrollo de smart contracts.* Catalunya. Recuperado el 29 de abril de 2019.

4.5.2 Funcionalidad de los contratos inteligentes

- Solamente el creador de los contratos puede incorporar nuevos estudiantes. El creador del contrato simbólicamente sería la institución de educación superior con capacidad para añadir estudiantes reales.
- Los empleados de la institución jamás podrán ver el historial de un estudiante sin consentimiento previo.
- Los empleados podrán añadir nuevos datos al registro de matrículas del estudiante, cuando se les otorgue acceso.
- El estudiante será capaz de añadir asignaturas, visualizando sus datos y asignaturas tomadas. Se puede seleccionar, si se desea, que el estudiante visualice todo el historial de asignaturas tomadas o solo las del último año.

4.5.3 Creación del contrato Estudiante.sol

Los contratos inteligentes en Solidity poseen similitud a la conformación de una clase en los lenguajes de programación orientados a objetos, para declararlos es necesaria la palabra reservada *contract* seguida del nombre del contrato usando el estilo *CamelCase* abrir y cerrar llaves dentro de las cuales se configurará la lógica del contrato.

Las variables de estado dentro de un contrato inteligente son valores que se almacenan en un sitio llamado memory o storage; las variables storage definen el estado del contrato y solo se modifican mediante las llamadas de envío de transacción, mientras que las variables memory son variables temporales que existen solo dentro de la función de llamada (no se pueden declarar fuera de una). Se eliminan después de que la función sale y generalmente son más baratas de usar que las storage. Ej: *mapping (address => InfoEst) infoestudiante;*

Los structs o estructuras dentro de un contrato inteligente son tipos de datos complejos definidos por el usuario los cuales agrupan múltiples variables ya sean las proporcionadas por Solidity o las definidas por el usuario (otros structs).
Las functions o funciones dentro de un contrato inteligente son unidades ejecutables que permiten la interacción de este con el compilador de Solidity.

Para este caso el contrato estudiante tendrá la siguiente composición:

Estructuras y variables de estado:

```
struct Student {

    string nombre;
    uint cedula;
    string facultad;
    string carrera;
    uint nivel;
    uint IDlength;
    address addrStudent;
    mapping (address => Subject) mysubject;
    Subject[] subject;

}

mapping(address=>Student)public student;
Asignaturas asig;
```

```
struct Subject {

    address addrSubject;
    uint level;
    bool access;
    uint ID;
}

address public disAddress;
```

Métodos y eventos:

```
event Message(string msgs);
event MessageExistSub(string msgs);
event MessageChangeData(string msgs);
```

```
function addEst(string memory nombre, uint cedula, string memory facultad, string memory carrera, uint nivel,address addrStudent)
}
function updateEst(address addrStudent,string memory facultad,string memory carrera, uint nivel)public{
}
function delEst(address addrStudent)public{
}
function getEstBy(address addrStudent) view public returns(string memory,uint,string memory,string memory,uint) {
}
function existEst(address addrStudent) public view returns (string memory){
}
function addSubject(address addrStudent, address addrSubject, uint level) external {
}
function delSubject(address addrStudent, address addrSubject) external {
}
constructor() public {
}
```

4.5.4 Creación del contrato Asignatura.sol

Haciendo referencia a los párrafos del punto 10.5.3, el contrato asignatura tendrá la siguiente composición de estructuras, atributos, variables de estado y métodos:

Estructuras y variables de estado:

```
struct Subject {

    string nombre;
    uint nro_mat;
    string paralelo;
    uint nro_cred;
    address addrSubject;
}
address public disAddress;
mapping(address=>Subject)public subject;
```

Métodos y eventos:

```
event Message(string msgs);
event MessageChangeData(string msgs);
```

```
function addSub(string memory nombre, uint nro_mat, string memory paralelo, uint nro_cred,address addrSubject) public {
}
function updateSub(address addrSubject,uint nro_mat,string memory paralelo, uint nro_cred)public{
}
function delSub(address addrSubject)public{
}
function getSubBy(address addrSubject) view public returns(string memory,uint,string memory,uint) {
}
function existSub(address addrSubject) public view returns (string memory){
}
constructor() public {
}
```

4.5.5 Creación del contrato Matrícula.sol

Haciendo referencia a los párrafos del punto 10.5.3, el contrato matrícula tendrá la siguiente composición de estructuras, atributos, variables de estado y métodos:

Variables de estado:

```
Estudiantes estudiantesContract;
Asignaturas asignaturasContract;
```

Métodos y eventos:

```
function callerContracts()public {
    estudiantesContract = Estudiantes(0x5c2a13a9BE1D9f72C402351cDE1B1FEF157bC2DD);
    asignaturasContract = Asignaturas(0xd6bE6a8c47A968B4eDE9050E3617399520fF103D);
}

function getStudentInfo(address addrStudent) view public returns(string memory,uint,string memory,string memory,uint) {

    return estudiantesContract.getEstBy(addrStudent);
}

function getSubjectInfo(address addrSubject) view public returns(string memory,uint,string memory,uint) {

    return asignaturasContract.getSubBy(addrSubject);
}
```

4.5.6 Estructura de archivos de configuración de la DApp

A continuación, se muestra la estructura de los archivos de configuración que conforman la DApp desarrollada en la framework Meteor.

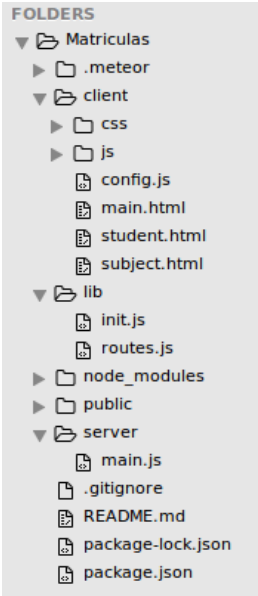


Ilustración 14.Estructura archivos de configuración Matriculas app

El proyecto Matriculas define tres plantillas web, una plantilla para la portada de la web, donde el estudiante podrá registrarse en la plataforma (main.js), una plantilla para la página del estudiante (student.js) y otra para la asignatura (subject.js). La lógica detrás de cada una de las páginas se define en su correspondiente archivo JavaScript con el mismo nombre que la plantilla HTML. En la carpeta lib, se definen las rutas de las aplicaciones para poder acceder a la web (routes.js) y cargar la comunicación con el nodo de Ethereum a través de la extensión Metamask (init.js). Por último, en la carpeta. meteor en el archivo package, se definen las dependencias del proyecto.

4.6 Despliegue de los contratos inteligentes

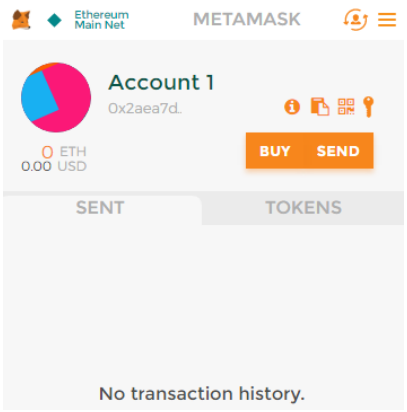
Los smart contracts son programas que se ejecutan en una blockchain, por tanto, necesitan un lenguaje de programación. Posiblemente, el más utilizado sea Solidity, un lenguaje de programación de alto nivel creado específicamente para Ethereum. Por este motivo se escogió Solidity para el desarrollo de los contratos, que posteriormente se convertirán en el núcleo de nuestras aplicaciones descentralizadas.

4.6.1 Instalar el complemento MetaMask y obtener un Ethereum de prueba de Ropsten

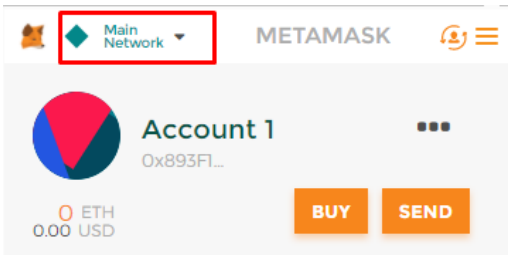
1. MetaMask es un plugin de Ethereum para agregarlo al navegador (Chrome, Firefox, Opera) que permite crear una cuenta de Ethereum para la administración de una cadena de bloques y sus transacciones.

2. Existen distintas billeteras de Ethereum. Pero, para el desarrollo de aplicaciones vamos a emplear aquellas que permiten trabajar con redes de prueba. Para este proyecto haremos uso de Metamask

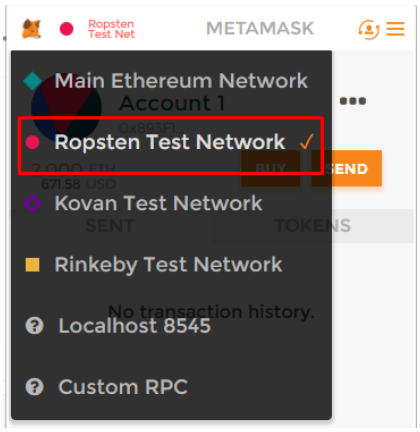
3. Es primordial guardar la semilla con la que se creó la cuenta, la misma que consta de 12 palabras; las cuentas en la blockchain se crean aleatoriamente. No existe forma de ir a un registro y comprobar si una cuenta ya existe para asignar otra nueva. La posibilidad de que dos cuentas aleatorias se repitan es de 2^{256}. Con la semilla que obtenida se pueden crear tantas cuentas como se requieran, en nuestro caso necesitaremos hasta un total de 3 cuentas (propietario del contrato, estudiante, y asignatura).



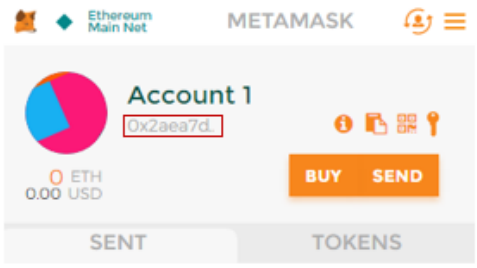
4. Una vez creada la cuenta de Ethereum en la esquina superior izquierda, presionar Red principal “Main Network” esta opción muestra una lista de las diferentes Blockchain de Ethereum que se pueden usar.



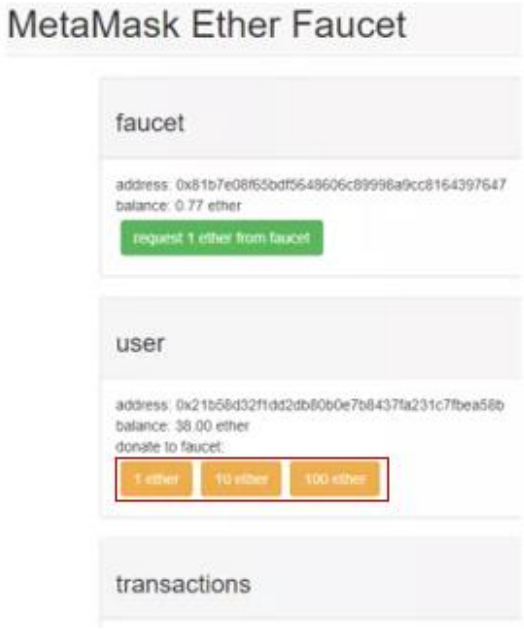
5. Seleccionar la opción "Ropsten Test Network" para agregar una red de prueba de Ethereum a Metamask a fin de simular el envío de recursos a través de una red de nodos de Ethereum.



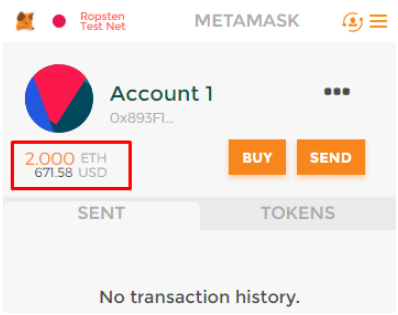
6. Para obtener ether y poder realizar transacciones desde la cadena de bloques de Ethereum a través de la red Ropsten, ingresar en el sitio Metamask Ether Faucet en donde se registrará automáticamente la dirección de la cuenta de Ethereum creada anteriormente.



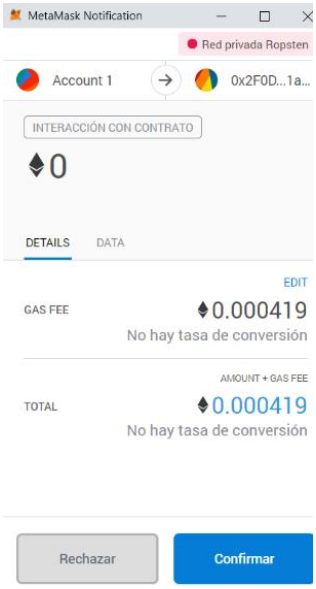
7. Dar click en el botón solicitar ether “request 1 ether from faucet”, obteniendo así un ether el cual permite enlazar la cadena de bloques con la cuenta de Ethereum por medio de la red recibiendo una recompensa, es decir, un número fijo de tokens de ether para verificar un grupo de transacciones y formar un bloque.



8. Finalmente se tiene el test Ethereum de Ropsten con MetaMask.



9. El envío de ether para realizar una transacción en la cadena de bloques Metamask lo valida por medio de la cuenta creada y la cantidad de gas necesaria para la transacción.



4.6.2 Interacción de los Smart Contracts con Remix IDE

(Augier, 2017) Describe el procedimiento asociado para crear un contrato inteligente de la siguiente forma:

1. Programar el contrato inteligente (code), haciendo uso de un lenguaje de programación para codificar exactamente lo que las partes desean que el contrato realice.
2. Mientras más complejas son las condiciones, menor es la capacidad del contrato para realizarlas (hasta que la tecnología madure). El contrato inteligente tendrá una codificación de forma condicional, ej: <<si esto, entonces aquello>>.
3. Una vez escrito el contrato, el siguiente paso es publicarlo en la Blockchain, ya sea privada o pública (deploy). El contrato está encriptado por lo que publicarlo no implica que éste pueda ser leído por terceros, si bien hay temas de seguridad y privacidad que tener en cuenta. Una vez publicado y almacenado en la Blockchain, el contrato puede ser ejecutado (call).
4. Una vez que el contrato está en la Blockchain, es ejecutado por los nodos y es necesario que se llegue a un consenso sobre el resultado, según lo definido en el contrato, puede ser necesario actualizar la cadena.

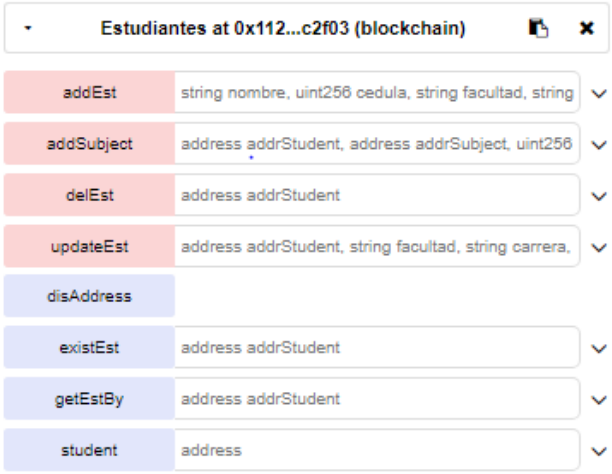


Ilustración 15.Contrato Estudiantes desplegado en Remix IDE

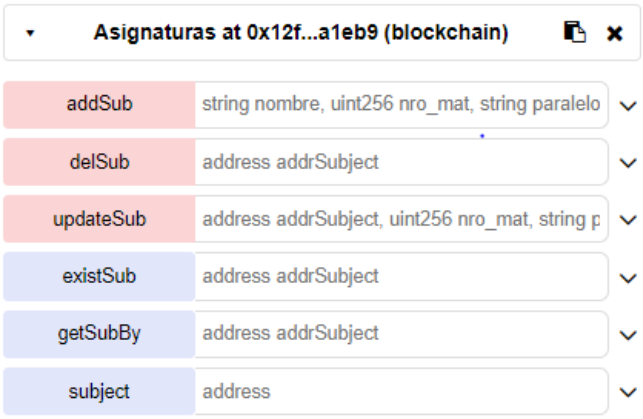


Ilustración 16. Contrato Asignaturas desplegado en Remix IDE

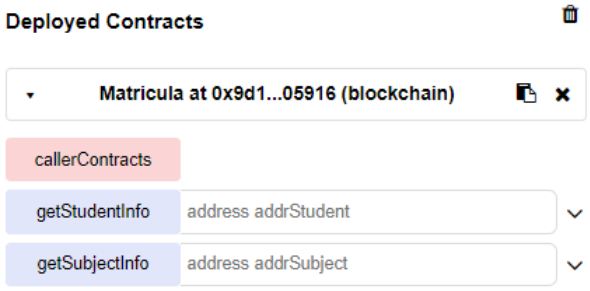


Ilustración 17.Contrato Matricula desplegado en Remix IDE

Una vez que se ejecute el contrato y se incorpore a la blockchain, aparecerá una lista de funciones previamente diseñadas en el mismo, con el propósito de interactuar, recuperar datos y añadir datos. Cabe destacar que los botones de color rojo requieren consumo de gas por parte de la cuenta que acceda a la función declarada y los de color azul, devuelven datos del contrato almacenados en la blockchain, por tanto, no necesitan de gas.

Reglas a tomar en cuenta sobre los contratos en Remix IDE:

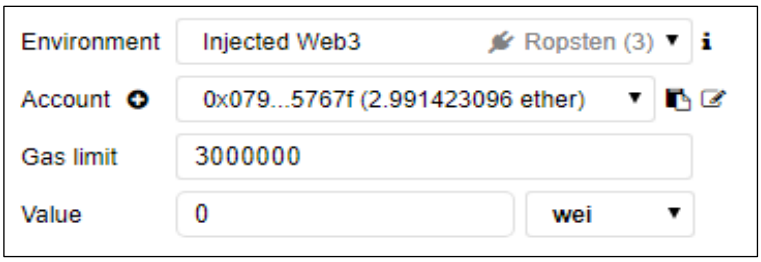
- Los datos del tipo integer y boolean no van entre comillas, excepto si el integer es un número grande; se pondrá entre comillas para ser manipulado como BigNumber.
- Las direcciones deben empezar por 0x y estar entre comillas.
- Las cadenas de caracteres van entre llaves { }. Lo demás valores van entre comillas.

4.6.3 Conexión de la DApp con Ethereum

Cada vez que se abra el navegador, se debe inicializar el objeto web3 con la información del proveedor de Ethereum en este caso Metamask mediante la librería web3.js La clase init.js de la Dapp que se encuentra dentro de la carpeta lib, se ejecutará tanto para el cliente como para el servidor, por lo tanto, mediante el objeto *Meteor.isClient*, indicamos a la aplicación que se ejecute solo desde el cliente Metamask e inicialice el objeto web3.

```
if (Meteor.isClient){
  if (typeof web3 !== 'undefined') {
    web3 = new Web3 (web3.currentProvider);
    console.log ("CLIENT CONNECT TO METAMASK");
  } else {
  web3 = new Web3 (new Web3.providers.HttpProvider("http://localhost:8545"));
    console.log ("CLIENT CONNECT TO OWN NODE");
  }
}
```

El navegador recupera la información del objeto *currentProvider*. Se comprobará que se ha instanciado el objeto mediante la conexión con Metamask o en el caso que se tenga un nodo instalado, se puede conectar al puerto 8545, donde por defecto se inician los nodos de Ethereum.



La clase de configuración config.js que se encuentra en la carpeta client, define 3 objetos para almacenar las direcciones de los 3 contratos obtenidos desde la herramienta Remix una vez añadidos a la blockchain. (Se puede consultar también la dirección de los contratos en la lista de transacciones hechas que nos proporciona Metamask).

El front-end está dividido en tres vistas principales:

- **Portada (localhost:3000):** Desde la portada principal de la web, se puede acceder automáticamente mediante los dos botones de la parte superior derecha a tanto el menú del estudiante o de asignaturas, según corresponda. Mediante el botón situado en la parte central, podemos registrarnos como estudiante.
- **Interfaz del Estudiante (localhost: 3000/student):** El estudiante podrá, mediante el panel izquierdo, ver su información personal, dar agregar asignaturas a su matrícula. En el panel principal se puede observar detalladamente todo el resumen de la matrícula del estudiante.
- **Interfaz de la Asignatura (localhost: 3000/subject):** En el panel izquierdo, se podrá visualizar la lista de asignaturas los que el estudiante tiene acceso para agregar a su matrícula. El reporte de matrícula aparecerá en el panel central, según el estudiante seleccionado.

Los siguientes templates diferencian a las vistas antes mencionadas:

```
<head>
<title>Matriculas</title>
<link rel="shortcut icon" type="images/x-icon"
href="images/favicon.ico"/>
</head>
<template name="index">
...
</template> ./main.html
<template name="studentInfo">
...
</template> ./student.html
<template name="subjectInfo">
...
</template> ./subject.html
```

Los templates estarán divididos en los 3 archivos descritos anteriormente. Dentro del código HTML se define el diseño de los elementos que componen cada una de las vistas. Estas vistas separan de manera individual la lógica detrás en 3 archivos javascript: main.js, student.js y.subject.js.

El archivo javascript está compuesto por 3 funciones principales que ejecutan los templates y se definen como se indica a continuación:

Template.NombreDelTemplate. [onCreated / helpers / events]

Para el caso de la vista student.html:

```
Template.studentInfo.onCreated(function indexCreated() {
    let contractEstudiantes;
})
Template.studentInfo.helpers({
    studentData() {
        ...
    }
})
Template.studentInfo.events({
    'click #updateEst': function() {}
})
```

El método *onCreated()* permite definir todas las funciones y variables que se ejecutarán cuando se visualice un template. El método *helpers()*, define las funciones disponibles en la vista. Para invocar a cada una de las funciones se usa la nomenclatura *{{studentData}}* dentro del archivo HTML. Por último, se tiene el método *events()*, que permite capturar todas las acciones realizadas en la app. Es decir, capturar la acción cuando se pulse sobre un botón para ejecutar funciones específicas.

Para definir las rutas usamos la dependencia ***iron:router*** Importar la dependencia en el archivo routes.js dentro de la carpeta lib.

```
import {Router} from 'meteor/iron:router';
Router.route('<URL>', function(){
this.render('<Nombre del Template>');
})
```

Se definen 2 rutas para el proyecto (/students y /subjects), para acceder a la web de los

estudiantes y asignaturas. Para renderizar la plantilla que se desea, se debe especificar el nombre del template a usar. La ruta raíz, por defecto redirecciona la app a la pantalla de inicio.

4.6.4 Interacción con los contratos inteligentes

Para interactuar con los contratos inteligentes se declara un objeto por cada contrato incorporado en la blockchain. Por medio del método ***web3.eth*** del objeto web3 descrito en el apartado 10.6.3 se realiza una instancia del contrato incorporando, la descripción de todas las funciones y eventos que dispone el contrato y la dirección de la blockchain donde está alojado el smart contract.

```
contract = web3.eth.contract(<contract.abi>).at(<contract.address>)
```

Una vez definido el objeto *contract*, es posible interactuar con la blockchain de dos maneras:

1. **Por medio de Funciones**

Las funciones alojadas en el ABI array pueden ser declaradas en cualquier momento. Por ejemplo, si se desea consultar la información del estudiante, se define la función en el método *helpers()*.

```
Template.studentInfo.helpers({
    studentData() {
        var name = Template.instance();
        contractInstance.myInfo(function (err, res) {
            TemplateVar.set(name, "name", res[0]);
        })
    },
});
```

Para facilitar la visualización de datos, usar la dependencia de ***frozeman:template-var***. La ventaja que ofrece esta dependencia es modificar directamente los datos de las variables en el HTML automáticamente. (Solo se permite modificar una variable, no arrays). Primero se inicia el objeto del template y posteriormente se añaden los datos devueltos por el contrato.

```
TemplateVar.set(name, "name", res[0]);
                      #1     #2      #3
```

En el ejemplo anterior, el objeto *TemplateVar.set* representa el nombre de la instancia sobre la que se va actuar. El parámetro *name*, es el nombre de la variable localizada en el HTML y el tercer parámetro es el valor devuelto por la función que contiene la información de una entidad, en este ejemplo en particular en el archivo HTML, es necesario llamar a la instancia *TemplateVar* con la siguiente sentencia:

```
{
    {
        TemplateVar.get "name"
    }
}
```

2. **Por medio de Eventos**

Es posible también usar eventos definidos en los contratos, para detectar automáticamente que un evento es lanzado dentro de un contrato.

```
var change = contractEstudiantes.MessageChangeData();
change.watch(function(error, result) {
            if (!error) {
                ...
            });
```

El fragmento de código mostrado, define una instancia *change* que, mediante el método *watch*, se a activará en el momento en el que la blockchain ejecute el evento *updateEst* descrito en el contrato Estudiantes.

5 DISCUSIÓN DE RESULTADOS

Esta investigación tuvo como propósito automatizar y optimizar el proceso de registro de matrículas de los estudiantes de la UCE a través de la experiencia del uso de la tecnología Blockchain; enfocada en la implementación de contratos inteligentes por medio del uso de la plataforma Ethereum, la cual permite programar contratos inteligentes conforme a términos establecidos. Sobre todo, pretendió determinar mecanismos estructurados dentro de la cadena de bloques que hagan uso de contratos inteligentes para optimizar el proceso antes mencionado; además de plantear un esquema para la inserción de contratos inteligentes en la cadena de bloques; dando como resultado la propuesta de un modelo de solución basado en contratos inteligentes para el registro de matrículas de los estudiantes de la UCE. Además, la propuesta de modelo antes mencionada identificó aquellos factores asociados al proceso de registro de matrículas que inciden en la transparencia, integridad, descentralización e inmutabilidad de los datos de los estudiantes al momento de realizar el registro de una matrícula. A continuación, se discutirán los principales descubrimientos de este estudio.

En base a los resultados obtenidos en esta investigación, se puede deducir que la tecnología Blockchain es una red digital distribuida e inmutable que almacena registros de información, conocidos como bloques, los cuales pueden almacenar diversas formas de información de manera encriptada (hash). El uso de hash permite detectar, pero no prever la integridad de los bloques; para mitigar este problema, se creó el mecanismo conocido como Proof of Work (Prueba de trabajo) denominado también como minado, el cual permite crear un historial de los bloques de manera que haya un consenso entre todos los nodos (mineros) de la cadena de bloques; pero seguridad no solo implica usar de manera ingeniosa un hash y el mecanismo de Prueba de Trabajo, sino que también, la propia red distribuida aplique seguridad a la blockchain en lugar de una entidad central que gestione la cadena de bloques. Blockchain usa redes peer to peer (P2P) o test net

(Rede de prueba) que son de libre acceso, cuando un nuevo usuario se une a la red, recibirá una copia de la blockchain y se convertirá en un nodo de la red. Cada vez que se cree un nuevo bloque, este se envía y cada nodo verificará el bloque para comprobar que existan irregularidades en los datos, para luego integrar el bloque a la cadena. Blockchain es similar a un inmenso libro de contabilidad, donde se registran distintas operaciones. Sin embargo, el registro, en lugar de estar preservado por una sola persona, es custodiado por todas las computadoras (nodos) de los usuarios dándole validez y garantía al sistema. En este orden de ideas, si se pretende alterar la información que tienen los bloques, se debe:

1. Alterar todas las computadoras de forma simultánea.
2. Alterar toda la historia de la red, puesto que cada nuevo bloque hace referencia al bloque anterior.

Es así, como la red genera un sistema transparente, descentralizado e inmutable; además se puede concluir que, gracias al avance tecnológico respecto a nuevas tendencias disruptivas Blockchain puede mejorar la eficiencia, reducción de costos y promover la democratización en la celebración de contratos, por medio de lo que se conoce como smart contracts o contratos inteligentes; con este mecanismo contractual los contratos no requieren la intervención de un tercero para formalizarse. Basta programar las condiciones contractuales en la plataforma digital para que el contrato pueda ejecutarse automáticamente cuando las condiciones acordadas se cumplan. Es decir, la existencia de esta tecnología contiene un protocolo informático que verifica el cumplimiento de las condiciones contractuales y ejecuta el contrato de manera automática, siendo el mismo sistema que garantiza y controla el cumplimiento de las obligaciones. ofreciendo así las garantías necesarias para que las personas puedan interactuar y realizar transacciones sin ningún intermediario.

La ventaja de usar contratos inteligentes a través de Blockchain (cadena de bloques) se puede resumir en tres palabras: autonomía, seguridad y confianza. La cadena de bloques es capaz de resguardar la información en una red cifrada que puede consultarse desde cualquier sitio, por lo que la velocidad y seguridad saltan a la vista.

Por otra parte, los smart contracts tienen ciertas desventajas, o más bien obstáculos que superar. El primero las tecnologías en las que se apoyan: IoT y Blockchain. Si bien IoT permite una vinculación con activos reales, lo cierto es que aún le queda camino por recorrer en lo que respecta a seguridad. Los dispositivos IoT son fácilmente vulnerables, algo que grandes empresas ya se han unido para solventar. Blockchain, por su parte, es segura, pero inmutable. Una vez acordados los términos, no se pueden cambiar, algo ciertamente desventajoso para un contrato, para esto también se están desarrollando soluciones.

Los expertos en programación de contratos inteligentes tampoco abundan como para promover una adopción masiva, e incluso cabe mencionar que en ocasiones ellos se equivocan. Estos programas son delicados, y si se queda tan sólo un error en su programación (bug) es posible que un tercero con malas intenciones robe los fondos almacenados en el contrato.

Por otro lado, se puede concluir que una DApp (Aplicación Descentralizada) es una app que no depende de un sistema central, sino que depende de los usuarios que hagan uso de la misma. Esta app puede ser una app móvil o una aplicación web que interactúe con

un contrato inteligente para llevar a cabo su función, en una DApp no es necesario hacer integraciones adicionales, ya que para el usuario es factible enviar o recibir fondos de forma directa (Ether), sin intermediaciones como PayPal y con comisiones prácticamente nulas. En lo que respecta al usuario final el tener que crear varias cuentas de usuario con diferentes contraseñas en una app tradicional puede hacer que, con el paso del tiempo, se pierdan y se tenga que recurrir a recuperar la contraseña; con las DApps esto no se da puesto que el usuario no necesita registrarse, al crear una cuenta Ethereum con su llave pública y privada (como en el caso de las Ethereum Wallets) que contiene sus datos, se puede vincularla con cualquier DApp, respecto al almacenamiento de datos el sistema tradicional, la información se guarda en discos duros, ya sean personales o servidores externos, mediante servicios en la nube; las dos opciones que tienen sus riesgos: en el caso de discos duros, estos pueden ser vulnerados para obtener los datos; en el caso de servicios en la nube, la cuenta puede ser vulnerada si se hackea a la empresa que proporciona el servicio, además, si esta empresa desaparece, los datos también desaparecerán.

Sin embargo, con las DApps, cuando se almacenan datos en la cadena de bloques estos permanecen inmutables, es decir, una vez que se registran no pueden borrarse; esto debido al uso de una blockchain pública que añade transparencia y seguridad; cualidades de una aplicación descentralizada. Por lo tanto, desarrollar una aplicación descentralizada basada en el entorno Ethereum aporta suficientes ventajas frente a las aplicaciones tradicionales.

Respecto a la arquitectura tanto de aplicaciones tradicionales como descentralizadas, existe lo que se conoce como Frontend (cliente-interfaz gráfica) y Backend (servidor-código), la diferencia radica en que el modelo Cliente-Servidor centraliza los datos en un servidor, lo que significa que quien sea propietario puede ejercer poder sobre ellos, y si la información contenida es delicada, puede negar el acceso o modificar los datos. Por el contrario, al montar una DApp sobre la arquitectura de Ethereum, la información se respalda y valida por toda la red de nodos (mineros) de la cadena de bloques; lo interesante de Ethereum, es que estos bloques, además de guardar información pueden ejecutar código en su lenguaje de desarrollo: Solidity, permitiendo la implementación de diversos tipos de contratos inteligentes, capacitando al código para que sea inalterado y ejecutado de acuerdo a lo estipulado. Cabe mencionar, que para desplegar una DApp sin errores, los usuarios deben descargar una copia completa de toda la cadena de bloques, por lo que, la intención no es deshacer el modelo cliente-servidor, sino más bien aprovechar las virtudes de la arquitectura de Blockchain respecto al modelo cliente-servidor por motivos específicos, como es el caso. Por otro lado, para ejecutar una DApp en contexto general, no es necesario que los usuarios descarguen una copia completa de la cadena de bloques, es suficiente con mantener un nodo conectado a la red para hacer las llamadas a los contratos.

Ethereum es considerada como una DApp puesto que cumple perfectamente con las características de una: el poder no reside en una empresa, jefe, o en definitiva en una figura de poder, sino que está en cada nodo que forma parte de la red, los cuales verifican transacciones y bloques en la blockchain, como la red está encriptada y almacenada de forma inmutable, pública y segura, cada nodo tiene una copia actualizada de esa blockchain, por ende, Ethereum es una plataforma distribuida, entonces, la respuesta básica del por qué desarrollar DApps en Ethereum es debido a que existen dos razones de peso: Seguridad e Interoperabilidad:

Resumiendo, el párrafo anterior podemos decir que si la DApp se construye en Ethereum cumple las siguientes premisas:

- Está escrita en el lenguaje Solidity, por tanto, los programadores que conozcan este lenguaje lo saben interpretar.
- Hace uso de la red Ethereum para procesar transacciones, es decir, se pueden construir sinergias (interacción de una DApp con otras).

Al igual que las DApps los contratos inteligentes también están escritos en el lenguaje Solidity, este lenguaje es una mezcla de varios lenguajes de programación, pero tiene similitud con la sintaxis de JavaScript, es decir, quien sepa programar en JavaScript no requiere esfuerzo adicional para aprender Solidity. Aparte de conocer el lenguaje Solidity, también es necesario crear un entorno de trabajo propicio para el desarrollador. Con entorno de trabajo se hace referencia a un entorno informático (como un IDE-Entorno de Desarrollo Integrado), que proporciona a los desarrolladores las herramientas adecuadas para crear otros programas.
Además, son necesarios otros tipos de recursos, en particular para este estudio se encuentran:

- Un marco de desarrollo (Framework)-Meteor.
- Cliente Ethereum (Nodo, Ethereum Wallet) y red pública de prueba-Ropsten.
- El compilador de Solidity.
- Las librerías y bibliotecas-Web3.js.
- Otras herramientas como Metamask y Remix IDE.

Las DApps pueden funcionar sin depender de un servidor central, pero hace falta disponer de un nodo de Ethereum, existen diferentes formas de acceder a una red de Ethereum y arrancar por completo un nodo de Ethereum; un posible caso sería acceder a un servidor que tenga un nodo de Ethereum y de este modo, conectar el nodo al servidor para acceder a la blockchain. Cabe mencionar que la blockchain de Ethereum ocupa aproximadamente 100GB a finales de 2017 es probable que este sea un impedimento de peso al momento de usar la tecnología, por ello surgió una herramienta para los navegadores tradicionales llamada Metamask. Esta extensión para la mayoría de navegadores es un puente que interconecta el navegador con Ethereum sin necesidad de descargar un nodo. Metamask proporciona una interfaz de usuario para administrar cuentas de usuario en diferentes sitios y firmar transacciones en una blockchain; además permite establecer una comunicación con las blockchain disponibles de Ethereum mediante el uso de la de Ropsten Test Network para poder ejecutar los contratos las veces que sean necesarias sin gastar dinero real, a través de la web Ether Faucet Metamask se puede enviar suficiente ether para desarrollar; el despliegue de los contratos se lo realiza mediante de la herramienta de código abierto Remix la misma que ayuda a escribir contratos de Solidity directamente desde el navegador. Escrita en JavaScript, admite tanto el uso en el navegador como a nivel local también además de pruebas, depuración y despliegue de contratos inteligentes.

Ya en la herramienta en el campo Enviroment seleccionar la opción Injected web3 de esta forma Remix detectará que se está utilizando la extensión Metamask e incorporará el contrato a la blockchain de test, posteriormente aparecerá un pop up de confirmación de la transacción a realizar, finalmente para extraer los datos almacenados en la blockchain y descritos en los contratos inteligentes se creó una aplicación en Meteor un

framework full-stack basado en JavaScript que permite programar tanto la capa del servidor (back-end) como la capa de la interfaz de usuario (front-end) cuyo objetivo es automatizar y simplificar el desarrollo de aplicaciones web, además permite la creación de dos tipos de proyectos, un proyecto más completo con una configuración de carpetas extendida o un proyecto vacío con las carpetas por defecto para el cliente y el servidor. En este caso nos decantamos por el tipo de proyecto más sencillo para conectar un nodo de Ethereum y establecer comunicación con los contratos para obtener así en la aplicación los datos de la Blockchain y mostrarlos al usuario final, de esta forma se puede concluir que si es posible automatizar y optimizar el proceso de registro de matrículas de los estudiantes de la UCE a través de la cadena de bloques y los smart contracts.

6 CONCLUSIONES

- La cadena de bloques permite automatizar y optimizar los procesos de negocio, en particular lo referente al registro de la matrícula de los estudiantes en el ámbito de la educación superior a través de soluciones disruptivas como smart contracts.
- Se construyó una estructura descentralizada, que emplea criptografía, el algoritmo de consenso prueba de trabajo (PoW) y la máquina virtual de Ethereum (EVM) para el despliegue de contratos inteligentes, enfocada en la privacidad de transacciones dentro de la cadena de bloques.
- Se planteó un esquema para la implementación, interpretación y ejecución de las instrucciones que forman los smart contracts, haciendo uso del ecosistema que la EVM de Ethereum proporciona para transacciones en una blockchain. (app emisora de transacciones, red de nodos que validan transacciones, smart contracts que toman transacciones como entrada, y la EVM que ejecuta los smart contracts).
- Se propuso un modelo de solución para para el proceso del registro de la matrícula de los estudiantes de la UCE por medio del uso de contratos inteligentes apoyado en los siguientes componentes: Algoritmos de consenso, el ecosistema de Ethereum, cliente web de Ethereum, red de prueba de Ethereum, Ether (Gas), DApps, y cadena de bloques.
- Los algoritmos de consenso dan sentido al concepto de dificultad, puesto que son el mecanismo usado para imponer seguridad en la cadena de bloques, asegurando que una cadena de bloques particular sea canónica a futuro, haciendo difícil para un atacante crear nuevos bloques que sobrescriban la cadena de bloques (borrar transacciones o crear transacciones falsas).
- Un consenso descentralizado proporciona un alto nivel de tolerancia ante errores, asegurando que la red este siempre disponible y se beneficie de las características de una blockchain.
- En base a los smart contracts desplegados en la red de Ethereum podemos crear aplicaciones conocidas como Aplicaciones Distribuidas (DApps) compuestas por smart contracts que se ejecutan en la red (backend).
- El gas de Ethereum optimiza los contratos inteligentes evitando que la red almacene cálculos innecesarios; ayudando a que la cadena sea más ligera (no guarda información irrelevante).
- Las DApps y los contratos inteligentes generan nuevas oportunidades de negocio y buscan nuevas soluciones a problemas sin resolver o cuya solución no era muy óptima.
- Los contratos inteligentes brindan enormes beneficios en términos de reducción

de costos de transacción, aumento de transparencia y seguridad.

- Aplicar smart contracts es todavía incipiente y se basa en pequeños intentos por iniciar con su aplicación práctica.
- Las aplicaciones Blockchain permiten automatizar y optimizar procesos de negocio, en particular en lo que respecta a controles y pagos.
- Blockchain es una cadena digital de bloques de información interconectados entre sí, creados y gestionados por una red de computadores a la vez, en lugar de estar centralizada o controlada por alguien en concreto.
- El futuro de la descentralización está muy cercano. Como se ha visto anteriormente, cada DApp pretende aplicar la tecnología Blockchain a su nicho y hacerse cargo de sus respectivas industrias.

7 RECOMENDACIONES

- Se recomienda el uso de redes de prueba como Ropsten y clientes web de Ethereum como Metamask para interactuar con los contratos inteligentes puesto que estos complementos permiten realizar transacciones en una blockchain optimizando costos y sin gastar dinero real, además, de ser herramientas diseñadas para acoplarse e interactuar entre sí sin mayores configuraciones.
- Para la construcción, compilación y despliegue de contratos inteligentes en el lenguaje Solidity se recomienda el uso del IDE Remix puesto que ofrece un entorno en tiempo de ejecución sin componentes orientados a nivel de servidor.
- Para el desarrollo de la DApp se utilizó el framework Meteor basado en JavaScript cuyo objetivo es simplificar la construcción de aplicaciones web. Se recomienda el uso de otros frameworks para determinar los beneficios y déficits que estos aporten al momento de interactuar con contratos inteligentes y los clientes web de Ethereum.
- Una contribución importante para este trabajo referente a la validación de transacciones y creación de bloques en una blockchain sería implementar la Prueba de Consenso (PoS) cuyo objetivo es validar transacciones sin necesidad de obtener una recompensa y eligiendo al creador de un bloque en base a su capacidad (riqueza).
- Para corroborar la robustez de Remix y Solidity, se recomienda escribir contratos inteligentes en otros IDEs y lenguajes de programación que presenten escenarios de interacción con contratos inteligentes a fin de verificar el funcionamiento y comportamiento de los mismos respecto al rendimiento e integridad de los datos durante la ejecución de una transacción en la cadena de bloques.
- Otro aspecto fundamental en Ethereum respecto a los contratos inteligentes es la necesidad de una cantidad de gas (costo para realizar operaciones en la red de Ethereum) tanto para la creación como para la ejecución de funcionalidades de los mismos, por tanto, a mayor complejidad del contrato, mayor es la cantidad de gas necesaria, un aporte destacado para este estudio sería determinar métodos o componentes que optimicen este proceso.
- Algo interesante respecto a este trabajo sería replicar el uso de las DApps y su ecosistema enfocado a otros ámbitos tales como la industria farmacéutica en lo que respecta al desperdicio de medicamentos, la industria de la educación superior en lo referente a Plataformas de aprendizaje de innovación y la industria Logística respecto a la cadena de suministro, en lo concerniente a la confianza entre los actores y empresas**.**

8 BIBLIOGRAFÍA

Allam, Z. (2019). *On Smart Contracts and Organisational Performance : A Review Of Smart Contracts Through the Blockchain Technology competition between business as each try to utilize the latter to bolster the*. (January). https://doi.org/10.1515/rebs-2018-0079

Bartolomé, A., & Lindín, C. (2018). *Posibilidades del Blockchain en Educación = Blockchain possibilities in Education*. (January 2019). https://doi.org/10.14201/eks20181948193

Bhargavan, K., Delignat-lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., … Swamy, N. (n.d.). *Formal Verification of Smart Contracts Short Paper F *.* 91–96.

Cong, L. W., Edmans, A., Park, A., & Hara, M. O. (2018). *No Title*.

Cuellar, S., & Medina, C. (2018). *NO ción u ol n za v e r L a con f l a e d d igit Centro de Información Tecnológica*. (June).

Modi, R. (2018). *Solidity Programming Essentials*. Packt Publishing.

Quiroga, J. P. (2014). *Requerimientos Funcionales y No Funcionales*. 1–27. Recuperado from http://www.electrohuila.com.co/Portals/0/UpDocuments/0b530417-2986-450e-bd92-34928a11e2f5.pdf

Slater, W. (2018). *Introduction to Blockchain*. (November). https://doi.org/10.13140/RG.2.2.21182.54085

Tuesta, D., Alonso, J., & Vargas, I. (2015). Aplicación de la lógica empresarial con los smart contracts. *BBVA Research*, 17. Recuperado from https://www.bbva.com/es/data/8663112016/Situacion_Ec_Digital_Oct15_Cap1.pdf

Academy, B. (13 de 12 de 2018). *¿Qué es un Algoritmo de Consenso Blockchain?* Obtenido de https://www.binance.vision/es/blockchain/what-is-a-blockchain-consensus-algorithm

Ahmed Kosba, A. M. (22 de Mayo de 2016). *IEEE Computer Society*. Obtenido de https://www.computer.org/csdl/proceedings/sp/2016/0824/00/0824a839-abs.html

Albuixech, R. C. (2018). *Estudio de tecnologías Bitcoin y Blockchain.* Catalunya: Universitat Oberta de Catalunya.

Atzei. (28 de Marzo de 2017). *Principles of Security and Trust, Chapter 8 A Survey of Attacks on Ethereum Smart Contracts (SoK*. Sweden: Springer. Obtenido de http://www.bookmetrix.com/detail/chapter/981e5c7d-3f5f-4ce2-8947-19b0f85b2c66

Augier, C. V. (2017). Capítulo 4, Aplicaciones transversales de la blockchain. En Á. Preukschat, *BLOCKCHAIN: LA REVOLUCIÓN INDUSTRIAL DE INTERNET* (pág. 397). Barcelona: Gestión 2000.

AWARENESS, T. D. (22 de julio de 2017). *METEOR JS: ¿QUÉ ES Y CÓMO LO UTILIZAMOS?* Obtenido de https://www.tooit.com/es/meteor-js-que-es/

Blockgeeks. (2019). *Blockchain Coding: The Many different Languages You Need!* Obtenido de https://blockgeeks.com/guides/blockchain-coding/#Language_1_C

Buntinx, J. (1 de Mayo de 2017). *What is the Ethereum Virtual Machine?* Obtenido de https://themerkle.com/what-is-the-ethereum-virtual-machine/

Croix, E. (13 de abril de 2019). *Programando en Blockchain: ¿Cuántos lenguajes de programación necesita para la programación en blockchain?* Obtenido de https://u.today/es/programando-en-blockchain-cuantos-lenguajes-de-

programacion-necesita-para-la-programacion-en
Fabisiak, R. (14 de agosto de 2018). *Which programming language is the best to be a blockchain developer*. Obtenido de https://medium.com/duomly-blockchain-online-courses/which-programming-language-is-the-best-to-be-a-blockchain-developer-2cb288e14765
Foundation, E. (s.f.). *Ethereum Build unstoppable applications*. Obtenido de https://www.ethereum.org/
Friebe, T. (30 de Octubre de 2017). *Bitcoin, Ethereum, and Hyperledger Fabric—which one wins?* Obtenido de https://medium.com/blockchainspace/3-comparison-of-bitcoin-ethereum-and-hyperledger-fabric-cd48810e590c
Gardiner, R. (15 de Mayo de 2018). *Composable Smart Contracts.* Obtenido de Research Gate: https://www.researchgate.net/publication/326693413_Free_Trade_Composable_Smart_Contracts
Goyal, A. (7 de febrero de 2019). *All You Need to Know about Blockchain Programming*. Obtenido de https://www.edureka.co/blog/blockchain-programming
IG.com. (2003). *¿Qué es Ethereum y cómo funciona?* Obtenido de https://www.ig.com/es/ethereum-trading/que-es-ether-y-como-funciona#information-banner-dismiss
Kasireddy, P. (27 de Septiembre de 2017). *How does Ethereum work, anyway?* Obtenido de https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369
Lin William Cong, Z. H. (Abril de 2018). *The National Bureau and Economic Research.* Obtenido de https://www.nber.org/papers/w24399
López, C. (28 de Noviembre de 2018). *Cibernos, make it easy*. Obtenido de https://blog.cibernos.com/blockchain-contratos-inteligentes
Marcos Allende López, V. C. (28 de Junio de 2018). *Abierto al público, Conocimiento Abierto*. Obtenido de https://blogs.iadb.org/conocimiento-abierto/es/tipos-de-blockchain/
Medium. (20 de abril de 2018). *Requerimientos Funcionales y No Funcionales, ejemplos y tips.* Obtenido de https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a
miethereum. (2018). *LA MONEDA: EL ETHER (ETH)*. Obtenido de https://www.miethereum.com/ether/
miethereum. (2018). *SMART CONTRACTS O CONTRATOS INTELIGENTES*. Obtenido de https://www.miethereum.com/smart-contracts/
Modi, R. (2018). *Solidity Programming Essentials.* Birmingham: Packt Publishing Ltd.
Mogollón, P. (15 de marzo de 2017). *Meteor 101 (Parte 0) Empezando desde cero*. Obtenido de https://meteor.com.co/meteor-101-parte-0-empezando-desde-cero-11f66f60755c
Mukhopadhyay, M. (2018). *Ethereum Smart Contract Development.* Birmingham: Packt Publishing Ltd.
Neto, M. (7 de febrero de 2018). *Get Ropsten Ethereum—The Easy Way.* Obtenido de https://medium.com/bitfwd/get-ropsten-ethereum-the-easy-way-f2d6ece21763
Núria Porxas, M. C. (2018). *Tecnología blockchain: funcionamiento, aplicaciones.* Actualidad Jurídica Uría Menéndez / ISSN: 1578-956X / 48-2018 / 24-36.
Palacios, V. M. (2018). *Explorando la Blockchain de Ethereum y el desarrollo de smart contracts.* Catalunya. Recuperado el 29 de abril de 2019
Peláez, J. S. (2018). *Fortificación IoT mediante blockchain y cifrado asimétrico.*

PLATFORM, L. C. (13 de JULIO de 2017). *Cómo van los contratos inteligentes a cambiar nuestras vidas*. Obtenido de https://es.ihodl.com/tutorials/2017-07-13/como-van-los-contratos-inteligentes-cambiar-nuestras-vidas/

Poveda, L. A. (2018). Algunos aspectos sobre blockchains y smart contracts. *Revista d'Innovació Docent Universitària*, 12. Obtenido de http://revistes.ub.edu/index.php/RIDU/article/view/20975

Preukschat, Á. (2017). Los fundamentos de la tecnología blockchain. En Á. Preukschat, *BLOCKCHAIN: LA REVOLUCIÓN INDUSTRIAL DE INTERNET* (pág. 397). Barcelona: Gestion 2000.

Ramos, P. R. (22 de febrero de 2016). *CUADRO COMPARATIVO DE DIFERENTES LENGUAJES DE PROGRAMACIÓN.* Recuperado de https://independent.academia.edu/pedroramirezramos

Robles, D. (2019). *¿Qué es el método científico experimental?* Recuperado de https://investigacioncientifica.org/que-es-el-metodo-cientifico-experimental/

Rodriguez, N. (20 de Septiembre de 2018). *101 Blockchains*. Recuperado de https://101blockchains.com/es/algoritmos-de-consenso-blockchain/#5

Roldán, F. (22 de abril de 2018). *CRIPTOTENDENCIAS*. Recuperado de https://www.criptotendencias.com/base-de-conocimiento/que-es-metamask-guia-inicial/

Solidity. (2016). *Solidity v0.5.3*. Recuperado de https://solidity.readthedocs.io/en/v0.5.3/

Suárez Chacón David Felipe, M. U. (2018). *Diseño de la Arquitectura de un Sistema de Contratos Inteligentes Basada en la Tecnología Blockchain Aplicada al Proceso de Registro de Estudiantes en el Sistema de Educación Colombiano.* Bogotá.

Tienda, A. (16 de Julio de 2018). *Blog de Alfonso Tienda, Marketing, Startups, Proyectos y Tecnología.* Recuperado de https://viviendo20.wordpress.com/

TruffleCon2019. (s.f.). *TRUFFLE AND METAMASK*. Recuperado de https://truffleframework.com/docs/truffle/getting-started/truffle-with-metamask

Vásquez, C. (20 de septiembre de 2018). *Qué son las aplicaciones descentralizadas (DApps)*. Recuperado de https://www.islabit.com/84215/que-son-las-dapps.html

Villarreal, G. L. (2017). Blockchain (no todo lo que brilla es Bitcoin). 9.

Wenting Li, S. A.-M. (2017). *Securing Proof-of-Stake Blockchain Protocols.* Alemania: NEC Laboratories Europe. doi:10.1007/978-3-319-67816-0_17

WeTrust. (12 de septiembre de 2017). *WeTrust Community Update—September 12 2017*. Recuperado de https://blog.wetrust.io/wetrust-community-update-september-12-2017-edbbee0ee0c5