“DESARROLLO DE UN SMART CONTRACT PARA REGISTRO DE ACTAS DE CALIFICACIONES CON TECNOLOGÍA BLOCKCHAIN PARA LA FACULTAD DE CIENCIAS Y TECNOLOGÍAS UNCA”

DIEGO RAÚL EMMANUEL DURÉ MARTÍNEZ

Orientador Académico de Proyecto de Fin de Grado:

Prof. Ing. Fredy Gabriel Ramírez

Responsable Académico de Proyecto de Fin de Grado:

Prof. Ing. Maira Santacruz Bogado

Cotutor de Proyecto de Fin de Grado:

Prof. Ing. Héctor Estigarribia_

Trabajo de Grado presentado a la Facultad de Ciencias y Tecnología de la Universidad Nacional de Caaguazú, como requisito para la obtención del título de Ingeniero en Informática.

UNIVERSIDAD NACIONAL DEL CAAGUAZÚ FACULTAD DE CIENCIAS Y TECNOLOGÍA

Carrera de Ingeniería en Informática

Coronel Oviedo, Paraguay

2020

PÁGINA DE APROBACIÓN

MESA EXAMINADORA DE SUSTENTACIÓN DE TESIS DE GRADO

Carrera de Ingeniería en Informática

Título de la Tesis: "DESARROLLO DE UN SMART CONTRACT PARA REGISTRO DE ACTAS DE CALIFICACIONES CON TECNOLOGÍA BLOCKCHAIN PARA LA FACULTAD DE CIENCIAS Y TECNOLOGÍAS UNCA".

Calificación Obtenida: _____

Miembro _____ Miembro _____

Miembro _____ Miembro _____

_____

Presidente

Acta N°: _____

Fecha: _____

DEDICATORIA

Esta tesis va dedicada a mis padres por apoyarme en todo momento, por brindarme fortaleza en innumerables situaciones y por creer siempre en mi.

A mis familiares, por ayudarme de todas las maneras posibles en el transcurso de mi carrera.

A todos y a cada uno de mis compañeros de la carrera, por ser buenos amigos y compañeros en todo momento.

A todos y a cada uno de mis profesores, por haberme guiado en el proceso de aprendizaje.

AGRADECIMIENTO

Al universo entero por conspirar a mi favor para que este trabajo sea posible.

A mi padre y a mi madre absolutamente por todo lo que me brindaron.

A todos a quienes contribuyeron en la elaboración y conclusión de este trabajo.

DESARROLLO DE UN SMART CONTRACT PARA REGISTRAR ACTAS DE CALIFICACIONES CON TECNOLOGÍA BLOCKCHAIN PARA LA FACULTAD DE CIENCIAS Y TECNOLOGÍAS UNCA

RESUMEN

En el trabajo presentado, se planteó el desarrollo y despliegue de un Smart Contract para registrar actas de calificaciones con el lenguaje de programación Solidity que tiene Ethereum dentro de su Blockchain como alternativa a los sistemas centralizados actualmente utilizados.

Para dicho fin se realizó una recopilación sobre los protocolos que sostienen la tecnología Blockchain para entender mejor su funcionamiento, destacando su seguridad computacional y privacidad de datos en frente de los sistemas centralizados actuales, además del estudio de la sintaxis del lenguaje de programación Solidity y su entorno de desarrollo.

Además, se procedió a hacer una evaluación del coste de despliegue y utilización del Smart Contract dando como resultado según la investigación dada en un ambiente de prueba un coste relativamente bajo.

Sin embargo, en cuanto a la interfaz gráfica para aquellos que desconocen el particular funcionamiento del entorno de desarrollo y

despliegue les podría resultar poco amigable.

Palabras clave: ethereum, blockchain, smart contract, desarrollo, seguridad computacional

ABSTRACT

In the present work presented, is proposed the development and deployment of a Smart Contract to record report cards with the programming language Solidity that Ethereum has within its Blockchain as an alternative to currently used centralized systems.

For this purpose a compilation was made on the protocols that support Blockchain technology to better understand how it works, highlighting your computer security and data privacy and data privacy in front of current centralized systems, in addition to studying the syntax of the Solidity programming language and its development environment.

In addition, an evaluation of the cost of deployment and use of the Smart Contract resulting in a relatively low cost based on research given in a test environment.

However, regarding the graphical interface, for those who are unaware of the particular operation of the development and deployment environment, it could be unfriendly.

Keywords: ethereum, blockchain, smart contract, development, computer security.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

CAPÍTULO I

1.MARCO INTRODUCTORIO

1.1.PLANTEAMIENTO DEL PROBLEMA

El dato más valioso tanto para la institución como para los estudiantes son las calificaciones, de ahí que existe la imperiosa necesidad de proteger los mismos. En este trabajo se plantea poder hacerlo contando con las características de inmutabilidad y seguridad computacional que ofrece la Blockchain para registrar las actas de calificaciones con la fiabilidad de que dichos documentos no podrán ser borrados o alterados, esta consiste en una estructura de datos en que la información está organizada en bloques permitiendo almacenar grandes cantidades de datos ordenados en el tiempo, de forma descentralizada e inmutable. Al ser inmutable ofrece una alta veracidad a los registros de las actas de calificaciones que podrá registrar la Facultad de Ciencias y Tecnologías, que hasta ahora no cuenta con ningún sistema o software que ofrezca este alto grado de veracidad, transparencia y respaldo de información.

1.2.DELIMITACIÓN Y ALCANCE

La idea principal de este proyecto es el desarrollo y despliegue de un Smart Contract en un ambiente de prueba para registrar actas de calificaciones con el lenguaje de programación Solidity que tiene Ethereum dentro de su Blockchain y analizar el coste de su despliegue y uso.

Este Smart Contract podrá registrar permanentemente y sin posibilidad de modificación las actas de calificaciones, puesto que estará incrustada en la Blockchain de Ethereum, un sistema descentralizado e inmutable, con alta seguridad computacional y de costo relativamente bajo.

Este Smart Contract registrará de las actas de calificaciones su número, código, la carrera, que en este caso, la muestra a ser utilizada será la de ingeniería en informática del quinto año, segundo semestre de la facultad de ciencias y tecnologías UNCA del año 2019, la materia, que en este caso serán, todas las materias de dicha carrera del segundo semestre, el curso, que en este caso, será la del quinto curso, el periodo, que son: la primera, la segunda, y la tercera oportunidad de cada materia, el número de documento (ci), junto con la calificación del alumno.

1.3.PREGUNTA DE INVESTIGACIÓN

¿Cual es la factibilidad de Desarrollar un Smart Contract con tecnología Blockchain para gestionar actas de calificaciones en la FCyT?

1.4.OBJETIVOS

1.4.1.OBJETIVO GENERAL

- Desarrollar un Smart Contract para registrar actas de calificaciones con tecnología Blockchain para la facultad de Ciencias y Tecnologías Unca.

1.4.2.OBJETIVOS ESPECÍFICOS

- Examinar los protocolos y funcionamiento de la Blockchain de Ethereum.
- Demostrar la utilidad del Smart Contract desplegado en un ambiente de prueba con los registros de las actas de calificaciones.
- Evaluar la factibilidad del despliegue y uso del Smart Contract.

1.5.JUSTIFICACIÓN

Con esta investigación estaremos analizando los protocolos de la Blockchain de Ethereum que a su vez nos facilitará desarrollar un Smart Contract basada en la necesidad puntual para mejorar la seguridad del registro de actas de calificaciones con la tecnología blockchain, puesto que es un sistema descentralizado e inmutable nos ayudará a mantener la información concisa y transparente.

Actualmente la facultad de Ciencias y Tecnologías registra las actas de calificaciones de forma física(hojas de documentos) y de forma digital, cuenta con un sistema académico hecho a medida con el lenguaje de programación foxpro. Proyecto de Microsoft al que brindó soporte hasta el 2015 [18], pasado dicho año ya no recibió actualizaciones o parches de seguridad por actuales o nuevos errores, siendo esto una crítica vulnerabilidad de seguridad informática en cualquier organización o empresa que cuente con un software desarrollado en dicho lenguaje. Como medida de seguridad y contingencia de datos se realizan procedimientos de backup de forma física y de forma virtual en un servidor. El Smart Contract a ser desarrollado funcionará como una medida de

seguridad adicional a las ya mencionadas, con la diferencia de que la información estará distribuida en una red descentralizada, con un histórico irrefutable de información.

CAPÍTULO II

2.MARCO TEÓRICO, CONCEPTUAL, OPERACIONAL Y LEGAL

2.1.ANTECEDENTES

La primera aparición de los Smart Contract de forma pública fue con Nick Szabo, jurista y criptógrafo en 1995, en lo que menciona lo siguiente:

La idea básica detrás de los contratos inteligentes es que muchos tipos de cláusulas contractuales (como garantías, fianzas, delimitación de derechos de propiedad, etc.) pueden integrarse en el hardware y el software con el que tratamos, de tal manera que se infrinja el contrato costoso (si se desea, a veces de manera prohibitiva) para el infractor. El éxito del derecho consuetudinario de los contratos, combinado con el alto costo de reemplazarlo, hace que valga la pena preservar y hacer uso de estos principios cuando sea apropiado. Sin embargo, la revolución digital está cambiando radicalmente los tipos de relaciones que podemos tener [1].

En aquel entonces (1996) Nick Szabo básicamente creó la teoría sobre el funcionamiento general de los Smart Contract,

lamentablemente era prácticamente imposible concretar la idea con la tecnología existente, mencionando también el costo elevado que implicaría ponerlos en práctica.

Las computadoras hacen posible la ejecución de algoritmos hasta ahora prohibitivamente costosos, y conectan en red la transmisión rápida de mensajes más grandes y sofisticados [1].

Para concretar la idea de los Smart Contract se necesitaba de protocolos que en aquel entonces recién estaban siendo descubiertos, como las transacciones programables y un sistema financiero que registre todas las actividades que se realicen sin dar lugar a los fraudes.

Además, los informáticos y los criptógrafos han descubierto recientemente muchos algoritmos nuevos y bastante interesantes. La combinación de estos mensajes y algoritmos hace posible una amplia variedad de nuevos protocolos [1].

Lo que para ese entonces era prácticamente imposible, años más tarde (2009) con el nacimiento de la Blockchain de Bitcoin se haría posible. Con la cadena de bloques se puede verificar el registro financiero de las actividades, evitando fraudes y dando

lugar a las transacciones programables.

Luego se concretó la idea de los Smart Contract con el nacimiento del proyecto Ethereum, oficialmente en 2016. Basándose en la Blockchain de Bitcoin con algunas modificaciones, dando lugar en el bloque a los Smart Contract y abriendo infinidades de posibilidades para las aplicaciones descentralizadas y la realización de acuerdos entre dos o más partes sin necesidad de un intermediario.

Una vez indagado en la Biblioteca así como en el repositorio digital de la Facultad de Ingeniería en Informática, se puede afirmar que no se ha encontrado un Proyecto de Trabajo de Graduación enfocado en la necesidad puntual para registrar actas de calificaciones con tecnología Blockchain.

Existen proyectos que hacen uso de los Smart Contract en otras instituciones para otros casos como los siguientes:

- Explorando la Blockchain de Ethereum y el desarrollo de smart contracts, realizado por Víctor Miranda Palacios en el año 2018 por el título de Grado en Ingeniería de Sistemas de Telecomunicaciones, Grado en Ingeniería

Telemática, de la universidad Politécnica de Catalunya.

Se basa en la creación de un Smart Contract para historiales médicos y se resume en lo siguiente:

En este proyecto se pretende dar a conocer la tecnología Blockchain, ver cómo surgió, cómo funciona y el motivo por el cual posiblemente sea una de las grandes tecnologías del futuro. El Blockchain, a grandes rasgos, es una gran base de datos descentralizada y en base a algunas de las cualidades descritas durante el proyecto, se creará una prueba de concepto basada en Ethereum, una de las blockchains más conocidas e importantes después de Bitcoin, con el fin de llevar la protección de datos, en especial los datos médicos, a un siguiente nivel [2].

- Los contratos inteligentes en España, la disciplina de los Smart Contract, publicado por la revista de derecho civil en el 2018 y escrito por Antonio Legerén-Molina

  Profesor Contratado-Doctor de Derecho civil Universidade da Coruña.

  Analiza las posibles regulaciones de dicha tecnología a

ser adoptadas específicamente en España.

- Desarrollo de un prototipo basado en Blockchain aplicado a la plataforma IOT sobre un sistema embebido, realizado por Esteban Adrián Restrepo e Daniel Arturo Olaya U en el año 2018 por el título de grado en Ingeniería Electrónica.

Se resume en lo siguiente:

En este documento se describe la implementación paso a paso de un contrato inteligente (Smart Contract) con un dispositivo IoT para la generación de alertas en la medición de una variable de contaminación del medio ambiente, esta alerta se genera

cuando se pasa un valor umbral definido para un sensor específico, con el fin de tener más seguridad y control a través de dispositivos conectados a internet a la hora de medir estos niveles de contaminación en una ciudad inteligente [4].

2.2.BASES TEÓRICAS

2.2.1. LA TECNOLOGÍA BLOCKCHAIN

La versión puramente de igual a igual de efectivo electrónico permitiría en línea pagos que se enviarán directamente de una parte a otra sin pasar por una institución financiera (Satoshi Nakamoto, 2009).

La tecnología Blockchain es la semilla que tiene el potencial de brindar frutos de enormes proporciones la revolución de la confianza. Blockchain permite eliminar al “hombre del medio” (The Man of the Middle por sus siglas en inglés). (David Esteban Plaza Ramírez, 2019).

La blockchain es, en esencia, una base de datos distribuida, que almacena todas las transacciones que se han realizado desde el inicio del sistema (2015). Este estaría compuesto por nodos cada uno independiente del otro siguiendo así el esquema peer to peer (P2P) gestionando un registro único donde se registran todas las operaciones realizadas.

Como su nombre indica es una cadena de bloques. Cada bloque incluye una serie de transacciones y la referencia al anterior bloque

de la cadena. Y una transacción no se considera válida a menos que forme parte de la cadena de bloques. De esta manera la blockchain contiene todas las transacciones consideradas válidas.

Para entender mejor su operación es necesario conocer algunas definiciones:

2.2.1.1. Hoja única contable abierta y distribuida (Open Ledger)

La tecnología Blockchain está constituida por una base de datos distribuida, que consta de un registro único compartido por todos los nodos que contiene la información de los dueños de los activos y el histórico del intercambio de la propiedad de los mismos (transacciones). Esto quiere decir que se puede establecer la cadena de propiedad del activo desde su origen o emisión. En resumen, la hoja única distribuida guarda el estado actual de la Blockchain y a partir de esta se puede establecer “quién es dueño de que y quien se lo transfirió”.

2.2.1.2. Nodos

Cada participante de la red Blockchain constituye un nodo. Cada vez que se generan transacciones estas se

transmiten a los nodos, un bloque se confirma por prueba de consenso y se añade a la cadena de bloques una vez que el 51% de los nodos aprueben la información como correcta, esta operación actualiza la hoja contable que cada participante(nodo) almacena, esta es una de las razones por la que corromper la seguridad de la Blockchain es casi imposible, puesto que se tendría que obtener el control de más de la mitad de los nodos de la red. Se requiere un enorme esfuerzo computacional para obtener el control por lo que pierde el sentido realizar un ataque.

2.2.1.3. Mecanismo de consenso

- **Prueba de trabajo(Proof-of-Work):** La propiedad de los activos y la transferencia de los mismos son validadas por los participantes de la red(mineros) donde el que tiene mayor poder computacional tiene más posibilidades de descifrar el hash y validar el bloque para ser recompensado por ello, el algoritmo usado por Ethereum es Ethash hace uso de una propiedad llamada en inglés *"memory hardness*" que provoca que el rendimiento del procesador dependa de la rapidez con la

que puede mover datos en la memoria en lugar de por la rapidez con la que realiza operaciones de cálculo [5].Bitcoin implementa un algoritmo de tipo *hashcash* que consiste en encontrar un hash SHA256 de una determinada dificultad. Como cabía esperar, la fuerte competencia ha impulsado el desarrollo de procesadores específicos de tipo ASIC (acrónimo de Circuito Integrado para Aplicaciones Específicas o *Application-Specific Integrated Circuit* en inglés), que sirven para optimizar esta función y realizar cálculos más rápidamente. En la actualidad uno sólo de estos procesadores es capaz de obtener más de mil millones de hashes por segundo[5]. La mayoría de nodos debe de estar de acuerdo sobre el estado resultante y el orden de cada transacción, esto se logra verificando la hoja contable distribuida i) si el activo existe (origen y emisión) ii) si este pertenece a quien transfiere (cadena de propiedad) y iii) si previamente no ha

sido transferido a otro participante (evitar el fraude por doble desembolso). El mecanismo de consenso también se encarga de verificar que los nodos sean honestos, es decir, sean reales y no varios participantes falsos dominados por un atacante para ganar más participación en la decisión. Una vez verificada, una transacción no podrá ser eliminada o modificada. Ethereum utilizó este mecanismo de consenso para luego pasar a otro.

- **Prueba de participación(Proof-of-Stake):** Con las pruebas de participación, se aprueban bloques por los mineros que mas tokens tengan o por la antigüedad de los mismos, esta es una ventaja, puesto que el que quiera aprobar un bloque debe de mostrar cierto interés en la plataforma, conseguir muchos tokens para así poder participar en la red como minero y no formar parte de la red para simplemente obtener una compensación por su trabajo sin ningún otro interés en la red de Ethereum, otra ventaja destacable es el bajo poder

computacional necesario y consumo de energía, puesto que se elimina el mecanismo de resolución de problemas.

En este mecanismo no se crean nuevos tokens, sino que los mineros son recompensados por una comisión de la transacción aprobada.

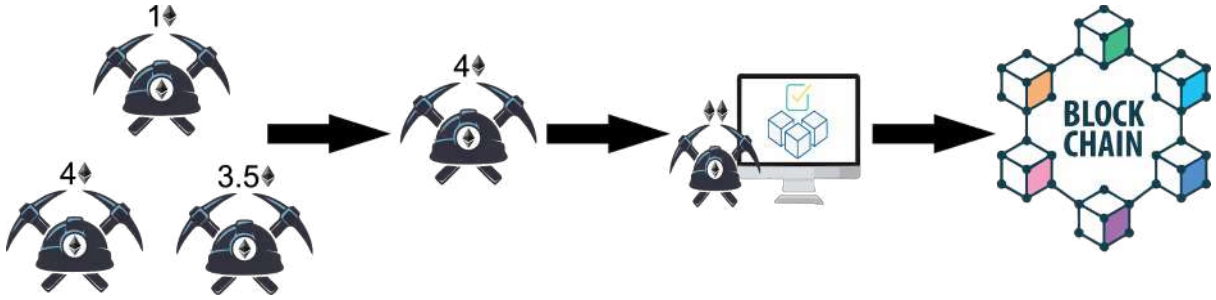


Figura 1: Mecanismo de Proof of Stake

2.2.1.4. Bloque

Para brindar mayor seguridad las transacciones se almacenan en bloques y este se encadena a bloques anteriores. Cada bloque contiene una estampilla de tiempo que indica el momento en el cual se hicieron las operaciones contenidas en él y un número cifrado (hash) que se crea a partir de la combinación de las transacciones contenidas y la referencia al bloque anterior. Las transacciones quedan en firme cuando su bloque contenedor se

enlaza a la cadena de bloques. Cualquier modificación en las transacciones modifica el hash del bloque, esto rompe la cadena y las transacciones se anulan.

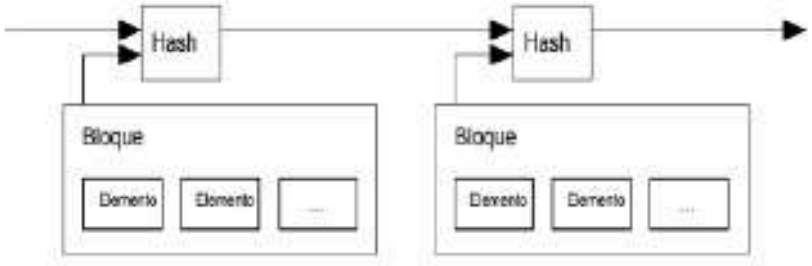


Figura 2: Cadena de Bloques de transacciones

Tomado de: Bitcoin: un sistema de efectivo electrónico usuario a usuario. Satoshi Nakamoto (2009)

La cadena de bloques de Ethereum es similar a la de Bitcoin en muchas cosas, aunque también tiene algunas diferencias. La diferencia principal entre Ethereum y Bitcoin respecto a la arquitectura de la cadena de bloques es que, a diferencia de Bitcoin, los bloques de Ethereum contienen una copia tanto de la lista de transacciones como del estado más reciente. A parte de eso, otros dos valores, el número de bloques y la dificultad, también están almacenados en el bloque. El algoritmo básico de validación de bloque en Ethereum es el siguiente:

1. Comprueba si el bloque previo referenciado existe y es válido.
2. Comprueba que el sello de tiempo del bloque es mayor que el bloque previo referenciado y menor de 15 minutos en el futuro.
3. Comprueba el número de bloque, la dificultad, la transacción raíz, la raíz tía y el límite de gas (varios conceptos de bajo nivel específicos de Ethereum) son válidos.
4. Comprueba que la prueba de trabajo del bloque es válida.
5. Establece S[0]como el estado al final del bloque previo.
6. Establece TX como la lista de transacciones del bloque, con n transacciones. Para todo i en ..n-1, establece S[i+1] = APPLY(S[i],TX[i]). Si alguna aplicación devuelve un error, o si la cantidad de gas consumido por el bloque hasta este momento excede el GASLIMIT, devuelve un error.
7. Establece S_FINAL como S[n], pero añadiendo al bloque la recompensa pagada por el minero.
8. Comprueba si la raíz del árbol de Merkle del estado S_FINAL es igual a la raíz del estado final proporcionada por la cabecera del bloque. Si esto es así, el bloque es válido, en otro caso, no lo es..

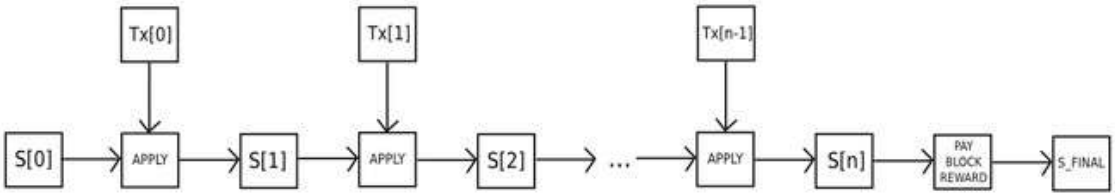


Figura 3: Cadena de Bloques de Ethereum

Tomado de Ethereum White Paper

A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM

By Vitalik Buterin

2.2.1.5. El gas

Cada transacción que se realiza en Ethereum tiene un determinado costo, este depende de la complejidad de la transacción a ser realizada, cuando hablamos de transacciones engloba tanto transacciones de ether de una cuenta a otra, como la ejecución de un Smart Contract, para esto se utiliza el gas, que es como una especie de combustible, también es una medida de seguridad para evitar la sobrecarga de la red.

Para entender mejor lo mencionado, veamos el siguiente ejemplo:

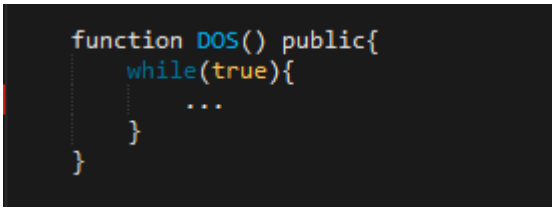


Figura 4: Función DoS

La función entraría en un bucle infinito, que podría ser utilizado como un ataque de tipo denegación de servicio (DoS). Pero para llevar a cabo este ataque, el atacante debería de tener ether infinito, lo cual es imposible. Supongamos que esta función requiere de 21.000 gas para ser ejecutado, ejecutarlo varias veces implicaría pagar 21.000 gas n veces, lo cual tarde o temprano llevaría al gasto total de Ether que posee el atacante.

2.2.1.6. Ethereum Virtual Machine

Es esta la que se encarga de ejecutar el código de los Smart Contract, la cual puede ejecutar código de complejidad algorithmica de manera arbitraria. En términos de ciencias de la computación, Ethereum es “Turing complete” o Touring completo, lo que significa que teóricamente podría resolver cualquier problema

computacional razonable [8].

Cuando un contrato es diseñado en Solidity, se convierte en una secuencia de códigos de operaciones, conocidas como opcodes, para luego pasar a bytecodes y que la EVM pueda entender.

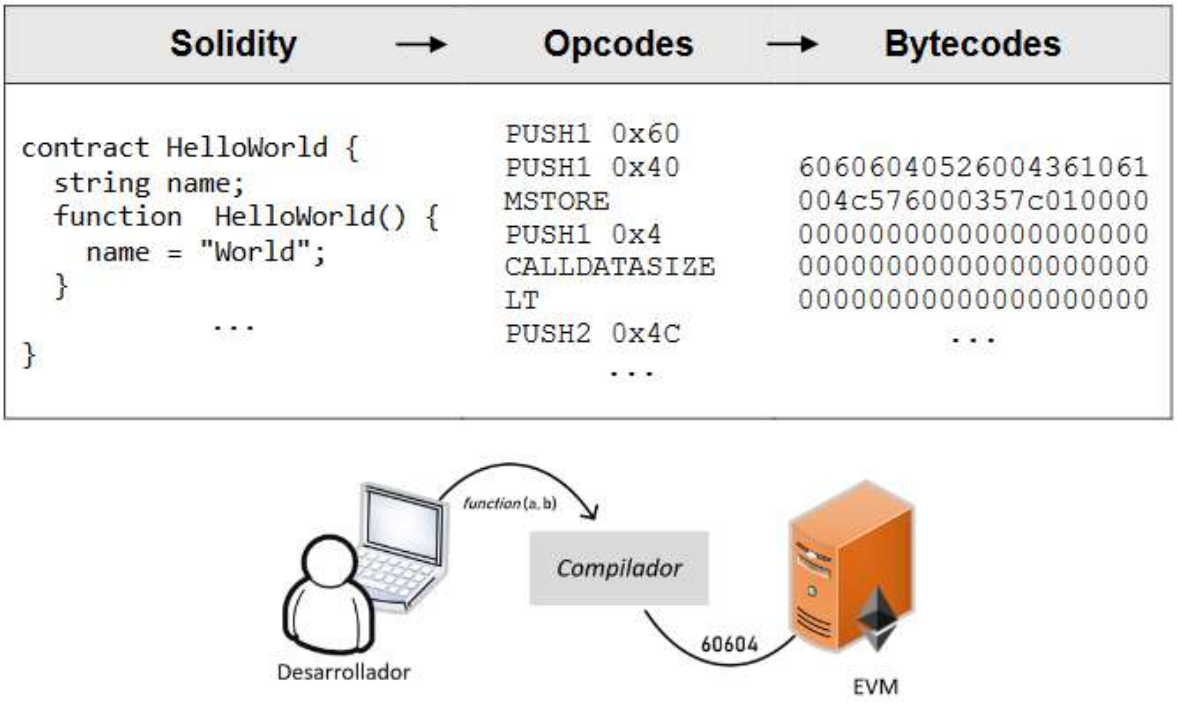


Figura 5: Compilación del Código en la EVM

Tomado de: Explorando la Blockchain de Ethereum y el desarrollo de smart contracts. Victor Miranda Palacios (2018)

2.2.2. SMART CONTRACTS

Los smart contract son programas informáticos que se

asientan en una cadena de bloques, en el cual se programan unas condiciones de actuación a cumplir basados en la simple sentencia de “si, entonces”, y que se ejecutan en el momento que se cumplen esas condiciones, pueden ser aplicados en una inmensa variedad de escenarios, los smart contract tienen ciertos aspectos que lo diferencian de los contratos tradicionales como:

- **Inmutabilidad:** La Blockchain es un libro digital incorruptible de transacciones económicas que puede programarse para registrar no solo transacciones financieras sino prácticamente todo lo que tiene valor (Don and Alex Tapscott, 2016). Una vez minado un bloque nadie podrá modificar la información guardada.
- **Distribuido:** La red está descentralizada, lo que significa que no tiene ninguna autoridad que la gobierne o una sola persona que ejerce control total. Más bien, un grupo de nodos mantiene la red descentralizada.
- **Deterministas:** La Blockchain está completamente organizada, y como no depende de cálculos humanos, las fallas accidentales de este sistema no son una salida habitual. Por lo que el código escrito no debería tener

ninguna aleatoriedad.

- **Verificables:** Una vez desplegado el contrato tendrá una dirección única, que puede usarse para que las partes interesadas analicen mejor el código para mayor seguridad.

Estos aspectos mejoran los acuerdos entre dos o más partes en cuanto a que su cumplimiento, no da lugar a la interpretación o manipulación por parte de terceros que generalmente son los intermediarios, consiguiendo la perfección y transparencia de los acuerdos.

Las aplicaciones de los Smart Contracts tienen una gran variedad de posibles escenarios, en los que podemos mencionar los siguientes:

- **Depósitos de garantía**: Por ejemplo, para la compra de una vivienda o cualquier otro activo entre particulares o empresas, el comprador deposita una cierta cantidad a una dirección wallet, y en el momento que se entregue al comprador el título o el bien acordado, se libere el depósito a favor del vendedor.
- **Apuestas:** Con la ayuda de los contratos inteligentes,

cualquier persona puede realizar apuestas con alguien, sin necesidad de intermediarios que respalden el pago de tales apuestas, y con total seguridad que después de conocerse el resultado, se liberará la cantidad ganada en la apuesta en su caso.

- **Votaciones:** Con la ayuda de los Smart contract se puede reemplazar la confianza que depositamos en las personas que realizan los conteos de las votaciones, que muchas veces son corruptibles o propensos a fallos, teniendo así la seguridad de que se realizarán votaciones transparentes y con total anonimato.
- **Logística y Transporte:** El ámbito del comercio intervienen múltiples agentes e intermediarios: fabricantes, mayoristas, distribuidores, proveedores, empresas de mensajerías, consumidores, entre otros actores generan una enorme cantidad de documentación mayormente en papel. La aplicación de tecnologías digitales a este sector, reduciría dicho papeleo, agilizaría el transporte de mercancías y

reduciría considerablemente los costes [9].

- **Salud y Sanidad:** Algunos de los potenciales usos de la cadena de bloques en sanidad son:

  Confidencialidad para el paciente. Proporcionada por la encriptación y la seguridad de la cadena de bloques.

  Trazabilidad y control de los medicamentos. Integrando el blockchain en farmacias, los médicos tendrían un control absoluto del consumo de medicamentos.

  Evitar fraudes en las pólizas médicas y controlar malas prácticas en los servicios sanitarios [9].

- **Certificados Académicos:** Con la ventaja que nos da la Blockchain de Ethereum podemos almacenar certificados emitidos de instituciones asegurándonos que estos no podrán ser alterados ni mucho menos perdidos, confirmando la identidad y la autenticidad de títulos, certificados y referencias. Los certificados digitales quedan registrados en la cadena de bloques, creando así un respaldo de seguridad.

2.3.ASPECTOS LEGALES

La tecnología Blockchain que usa Ethereum es independiente y libre de control de cualquier entidad, puesto que es un sistema descentralizado basado en la libertad y seguridad computacional no está regulado por ningún tipo de autoridad.

CAPÍTULO III

3.MARCO METODOLÓGICO

3.1.TIPO DE INVESTIGACIÓN

La investigación es de tipo propositiva por cuanto se fundamenta en una necesidad o vacío dentro de la institución, una vez que se tome la información descrita, se realizará una propuesta de sistema de evaluación del desempeño para superar la problemática actual y las deficiencias encontradas.

3.2.DISEÑO DE INVESTIGACIÓN

Este trabajo se basó principalmente en el desarrollo de un Smart Contract con el lenguaje de programación Solidity que tiene Ethereum dentro de su Blockchain, para esto examinamos dicha tecnología con el fin de entender mejor su funcionamiento y sus protocolos, así como la sintaxis de dicho lenguaje de programación.

Una vez desarrollado el Smart Contract lo desplegamos en un ambiente de prueba para analizar el costo que implica el uso de la misma.

El diseño de investigación según Kerlinger (2002)[21] es el plan y la estructura de un estudio, el enfoque utilizado en este proyecto es el enfoque científico, que según Ramón Ruíz (2007) [22] el método científico se emplea con el fin de incrementar el conocimiento y en consecuencia aumentar nuestro bienestar y nuestro poder.

La investigación científica es aquel proceso de carácter creativo que pretende encontrar respuestas a problemas trascendentes mediante la construcción teórica del objeto de investigación, o mediante la inducción, innovación o creación de tecnologías.

Las técnicas de recolección de información que se utilizaron en la investigación son las siguientes:

Prueba Piloto: Esta técnica se utilizó una vez adquirido cierto nivel de aprendizaje en cuanto a la sintaxis del lenguaje de programación Solidity y su entorno de desarrollo, para desplegar el Smart Contract en un ambiente de prueba y analizar el costo.

Entrevista: Se utilizó esta técnica para conocer el sistema de registro de actas de calificaciones y la información a ser guardada, puesto que los bloques tienen un límite, la información a ser guardada debe de ser limitada.

El informante clave es aquel que convive con el objeto de investigación y por ello se considera con autoridad para dar información. El instrumento utilizado es una guía de entrevista. Utiliza preguntas abiertas [10].

3.3.POBLACIÓN Y MUESTRA

La población que se tomó para el estudio de este proyecto final de grado fue la Facultad de Ciencias y Tecnologías UNCA, siendo la muestra la carrera de ingeniería en informática del quinto año, segundo semestre del año 2019, la técnica de muestreo utilizada fue la muestra no probabilística, que utiliza métodos en que no interviene el azar y por lo tanto, se desconoce la probabilidad asociada a cada individuo para formar parte de la muestra. Normalmente estos métodos se utilizan en estudios exploratorios o intencionales, en los cuales no es necesario proyectar los resultados [19].

3.4.MÉTODO, TÉCNICAS Y PROCEDIMIENTOS

3.4.1.METODOLOGÍAS PARA EL DESARROLLO DEL SISTEMA

3.4.1.1.SCRUM

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales [11].

El marco técnico de scrum, está formado por un conjunto de prácticas y reglas que dan respuesta a los siguientes principios:

- Gestión evolutiva del producto, en lugar de la tradicional o predictiva.
- Calidad del resultado basado en el conocimiento tácito de las personas, antes que en el explícito de los procesos y la tecnología empleada.
- Estrategia de desarrollo incremental a través de iteraciones (sprints)..

Se comienza con la visión general del resultado que se desea, y a partir de ella se especifica y da detalle a las funcionalidades que se desean obtener en primer lugar.

Cada ciclo de desarrollo o iteración (sprint) finaliza con la entrega de una parte operativa del producto (incremento). La duración de cada sprint puede ser desde una, hasta seis semanas, aunque se recomienda que no exceda de un mes [12].

CAPÍTULO IV

4.MARCO ANALÍTICO

4.1.REQUERIMIENTOS

4.1.1.REQUERIMIENTO FUNCIONAL

El Smart Contract basado en la Blockchain de Ethereum para registrar actas de calificaciones podrá:

- Registrar las actas de calificaciones con la seguridad de que nadie podrá modificarlos ni eliminarlos.
- Encriptar toda la información a ser guardada.
- Mantener organizada la información con una marca de tiempo.
- Ser escalable.
- Mantener la información descentralizada.

4.1.2.REQUERIMIENTO NO FUNCIONAL

- Cambio de dirección por cada transacción realizada.

4.2.ESTUDIO DE FACTIBILIDAD

Se refiere al estudio de factibilidad a que tan posible es el cumplimiento de un proyecto o investigación teniendo en cuenta los recursos que se poseen, ya sea en cuanto a recursos físicos, recursos económicos, recursos de aprendizaje o de conocimiento.

Este proyecto que busca registrar las actas de calificaciones en la Blockchain de Ethereum se apoya en tres aspectos básicos.

4.2.1.FACTIBILIDAD TÉCNICA

La Factibilidad Técnica se refiere a los recursos necesarios como herramientas, conocimientos, habilidades, experiencia, etc., que son necesarios para efectuar las actividades o procesos que requiere el proyecto [13].

Este proyecto que busca desarrollar un Smart Contract con tecnología Blockchain requiere de: Solidity como lenguaje de programación para crear la lógica de registros de actas de calificaciones, el entorno de desarrollo y compilación Remix basado en un navegador web que nos permitirá hacer uso correcto de la sintaxis y lógica, y MetaMask(ver página 65) que es la puerta de

enlace entre el navegador y la Blockchain, permite desplegar y utilizar Smart Contract, también permite realizar transacciones, todo esto sin ser un nodo Ethereum completo.

4.2.2.FACTIBILIDAD OPERATIVA

Se refiere a las personas que participarán en los procesos que necesita ejecutar el proyecto, al ser una tecnología relativamente nueva con un entorno poco convencional, amigable e intuitivo, el usuario a manipular el sistema podría verse un poco abrumado al ser diferente a los sistemas actualmente utilizados.

4.2.3.FACTIBILIDAD ECONÓMICA

La Factibilidad Económica se refiere a los recursos económicos y financieros necesarios para desarrollar o llevar a cabo las actividades o procesos y/o para obtener los recursos básicos que deben considerarse [13].

El proyecto que usa la red descentralizada de Ethereum se apoya en su moneda Ether, por lo tanto tendrá un valor o coste desplegar y usar el Smart Contract a ser desarrollado para registrar las actas de calificaciones, un proyecto puede ser factible desde el punto de vista operacional y técnico, pero si no lo es económicamente pierde el sentido su implementación.

4.3.ANÁLISIS

4.3.1.ESCENARIO DE USUARIO

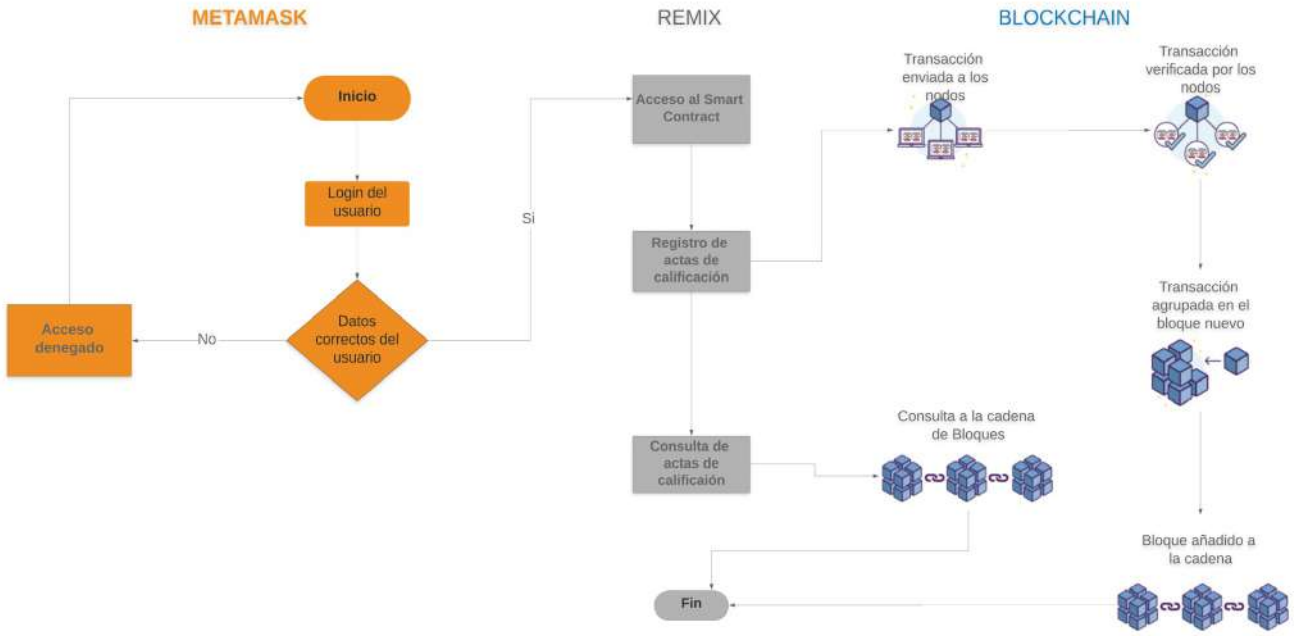
Un escenario es una descripción parcial del comportamiento de la aplicación en un momento específico. La utilización de escenarios implica identificar distintas situaciones y describir la acción a llevar a cabo. Los mismos son de gran ayuda en el momento de especificar requerimientos; y su rol principal es el de permitir la comunicación entre expertos de software y del dominio, y analizar aspectos específicos de un sistema, describiendolo en forma concreta [14].

_____En general los sistemas que se encargan de registrar documentos cuentan con un número o código que hacen único a dicho documento, acompañado de la fecha, y los datos a ser guardados que en este caso serían las notas de los alumnos. El encargado de registrar las actas de calificaciones poseerá una dirección y una contraseña para acceder al sistema, solo el podrá registrar las actas. Cabe resaltar que como cualquier otro sistema es vulnerable al robo de la dirección y contraseña, por lo tanto esta información sensible debe de ser guardada muy cuidadosamente,

además si el usuario llegara a olvidar su dirección sería imposible recuperarla.

Puesto que la tecnología no cuenta con un mecanismo de recuperación de cuentas, la única manera de recuperar la contraseña es enlazando la cuenta a un correo electrónico.

4.4.DISEÑO



4.5.CODIFICACIÓN Y ENTORNO DE DESARROLLO

Una vez entendida la forma de organización de las actas de calificaciones y elegido los datos a ser guardados se procede al desarrollo del Smart Contract, para ello utilizamos el lenguaje de programación Solidity que está hecho específicamente por y para Ethereum.

4.5.1.SOLIDITY

Solidity es un lenguaje de alto nivel orientado a contratos. Su sintaxis es similar a la de JavaScript y está enfocado específicamente a la Máquina Virtual de Ethereum (EVM) [15].

Solidity está tipado de manera estática[1] y acepta, entre otras, herencias, librerías y tipos complejos definidos por el usuario. Es utilizada para programar la lógica necesaria, para modelar y controlar los datos de una aplicación.

Básicamente un Smart Contract es una colección de funciones y variables, identificados por una dirección única de la Blockchain de Ethereum, la extensión para los ficheros es .sol y todos los contratos empiezan por la siguiente línea de código:

```
6  pragma solidity >=0.4.22 <0.6.0;
```

1 La comprobación de tipificación se realiza durante la compilación y no durante la ejecución.

Figura 6: Especificación de las versiones del compilador a ser usado

En esta primera línea se especifica que el Smart Contract está escrito para ser compilado en la versión 0.4.22 hasta la versión 0.6.0.

La siguiente línea de código especificará el nombre del Smart Contract que en este caso es ReportCards y dentro de ella irá la lógica del mismo todo esto entre llaves.

Figura 7: Nombre del Smart Contract

El nombre sirve para identificarlo a la hora en que lo vamos a compilar y poner a prueba su lógica y sintaxis.

4.5.1.1.TIPOS DE DATOS

Solidity es un lenguaje de tipado estático, que significa que cada tipo de variable (estado y local) tiene que ser especificada en tiempo de compilación [15].

Los tipos de datos que soporta Solidity son los siguientes:

- **Booleanos:** Los posibles valores son las constantes true y false.

Operadores:

- ! (negación lógica)
- && (conjunción lógica, “y”)
- || (disyunción lógica, “o”)
- == (igualdad)
- != (inigualdad)

Los operadores || y && aplican las reglas comunes de corto circuitos. Esto significa que en la expresión f(x) || g(y) , si f(x) evalúa a true, g(y) no será evaluado incluso si tuviera efectos secundarios [15].

- **Enteros:** int/uint con o sin signos de varios tamaños. Las palabras clave uint8 a uint256 en pasos de 8 (sin signo de 8 hasta 256 bits) y int8 a int256. uint y int son alias para uint256 y int256 respectivamente [15].

  Operadores:

  - Comparaciones: <=, <, ==, !=, >=, > (evalúa a bool)
  - Operadores bit: &, |, ^ (OR exclusivo a nivel de bit), ~ (negación a nivel bit)

- Operadores aritméticos: +, -, - unario, + unario, *, /, % (restante), ** (exponenciales), << (desplazamiento a la izquierda), >> (desplazamiento a la derecha)

- **Address:** Contiene un valor de 20 bytes (tamaño de una dirección de Ethereum).
- **String:** Cadena de caracteres UTF-8 codificado de tamaño dinámico.
- **Arrays:** Los array pueden tener tamaño fijo en compilación o pueden ser dinámicos. Un array de tamaño fijo k y elemento tipo t es escrito como t[k], un array de tamaño dinámico como t[].
- **Structs:** Un struct en solidity es como una clase que tiene atributos, dentro de un struct pueden ir variables de tipo string, uint/int, booleanos, etc. Pero tiene la limitante de hasta 7 variables.
- **Mappings:** Tipos mapping son declarados como mapping(_keyType => _valueType). Aquí _keyType puede ser casi cualquier tipo excepto mapping, un array

- de tamaño dinámico, un contrato, un enum y un struct. Mappings pueden verse como tablas hash que son virtualmente inicializadas ya que cada posible clase existe y es mapeada a un valor que su representación byte es todo ceros. Es posible marcar los mappings public y hacer que Solidity cree un getter. El _keyType será un parámetro requerido para el getter y devolverá _valueType.
- **Eventos:** Los eventos son la forma en que nuestro contrato comunica que algo sucedió en la cadena de bloques a la interfaz de usuario, el cual puede estar 'escuchando' ciertos eventos y hacer algo cuando suceden. Los eventos se declaran con la palabra event y el nombre del evento con los parámetros.

4.5.1.2.UNIDADES Y VARIABLES DISPONIBLES GLOBALMENTE

- **Unidades de Ether:** Un número literal puede tomar un sufijo como el wei, el finney, el szabo o el ether para convertirlo entre las sub denominaciones del ether.

Se asume que un número sin sufijo para representar la moneda Ether está expresado en Wei.

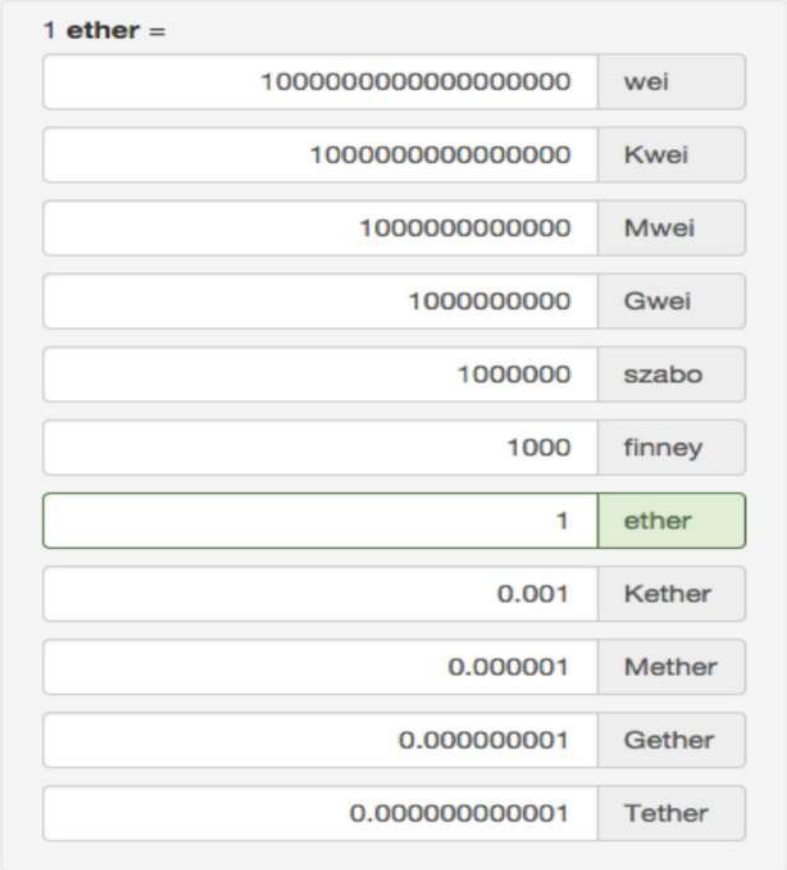


Figura 8: Unidades de medidas en Ether

4.5.1.3.FUNCIONES

Las funciones son las unidades ejecutables del código dentro de un contrato.

- **Funciones GET:** Al ejecutar un contrato automáticamente se creará una función GET por defecto, básicamente como en la

mayoría de los lenguajes, se utilizan para obtener un resultado o valor, estas funciones no requieren de gas. En estas funciones no se crean ningún bloque.

- **Funciones SET:** Las funciones SET necesitan parámetros de entrada y no devuelven nada, las funciones SET necesitan gas para ser ejecutadas y estas sí generan nuevos bloques.

Los tipos de funciones nos sirven para definir restricciones de acceso a las mismas:

- **External:** Cuando definimos una función como external estamos evitando que sea llamada internamente, es decir, desde otras funciones del contrato donde se encuentra definida o de los herederos de los mismos.
- **Public:** Todos pueden llamar esta función.
- **Private:** Sólo se puede acceder desde el contrato donde se encuentra la función, no se puede desde fuera ni desde los herederos del contrato.
- **Internal:** Sólo permitimos el acceso a la función desde el propio contrato y desde los herederos de este.

4.5.1.4. BLOQUE Y PROPIEDADES DE LAS TRANSACCIONES

- **block.blockhash(uint blockNumber) returns (bytes32):** el hash de un bloque dado - sólo funciona para los 256 bloques más recientes, excluyendo el actual
- **block.coinbase (address):** devuelve la dirección del minero que está procesando el bloque actual
- **block.difficulty (uint):** devuelve la dificultad del bloque actual
- **block.gaslimit (uint):** devuelve el límite de gas del bloque actual
- **block.number (uint):** devuelve el número del bloque actual
- **block.timestamp (uint):** devuelve el timestamp del bloque actual en forma de segundos siguiendo el tiempo universal de Unix (Unix epoch)
- **msg.data (bytes):** datos enviados en la transacción (calldata)
- **msg.gas (uint):** devuelve el gas que queda
- **msg.sender (address):** devuelve remitente de la llamada actual

- **msg.sig (bytes4):** devuelve los primeros cuatro bytes de los datos enviados en la transacción (i.e. el identificador de la función)
- **msg.value (uint):** devuelve el número de Wei enviado con la llamada
- **now (uint):** devuelve el timestamp del bloque actual (es un alias de block.timestamp)
- **tx.gasprice (uint):** devuelve el precio del gas de la transacción
- **tx.origin (address):** devuelve el emisor original de la transacción

4.5.1.5. MANEJO DE ERRORES

- **Requiere(bool condition):** Se lanza si la condición se cumple
- **Revert():** Cancela toda la operación, anteriormente se usaba throw().

4.5.1.6. MODIFICADORES DE FUNCIONES

Se pueden usar los modificadores para cambiar el comportamiento de las funciones de una manera ágil. Por ejemplo, los modificadores son capaces de comprobar automáticamente una condición antes de ejecutar una función [15].

```
//Codigo para que solo el que escribio el contrato pueda realizar algunas modificaciones
modifier OnlyOwner{
        if(owner != msg.sender){
        revert();
        }else{
            _;
        }
    }
```

Figura 9: Modificador en Solidity

En este ejemplo utilizado solo el creador del contrato puede hacer uso de algunas funciones.

4.5.2.ENTORNO DE DESARROLLO REMIX

El entorno de desarrollo remix es un IDE basado en un navegador web con análisis estático integrado y una máquina virtual de blockchain para pruebas.

Remix anteriormente conocido como Browser Solidity, proporciona un entorno de desarrollo integrado (IDE) que permite escribir contratos inteligentes basados en Solidity [16].

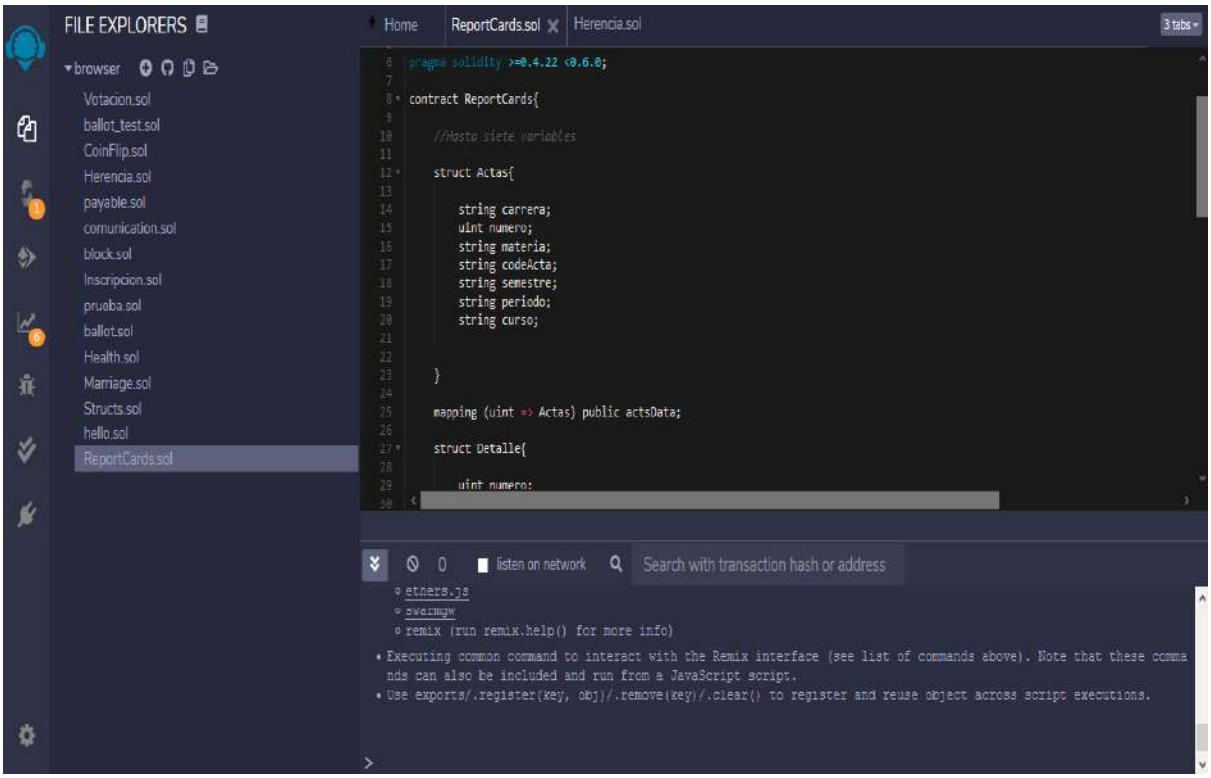


Figura 10: Entorno de desarrollo Remix

El entorno de desarrollo Remix se divide en las siguientes partes:

- **Explorador de ficheros:** Donde se almacenan todos los contratos que se vayan desarrollando, se pueden crear, renombrar y añadir nuevos desde el ordenador, como también publicarlo en GitHub.

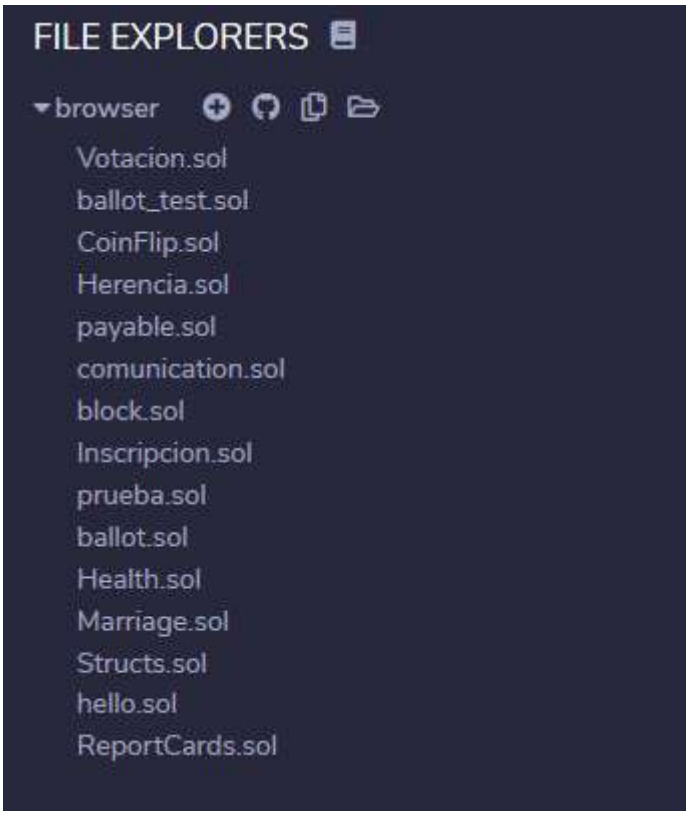


Figura 11: Explorador de ficheros en Remix

- **Compilador de Solidity:** Remix tiene la opción de compilar el código cada vez que sufre un cambio, la opción de elegir la versión del compilador, también la opción de publicar el Smart Contract en plataformas como Swarm, IPFS, así como también permite copiar el código ABI y Bytecode.

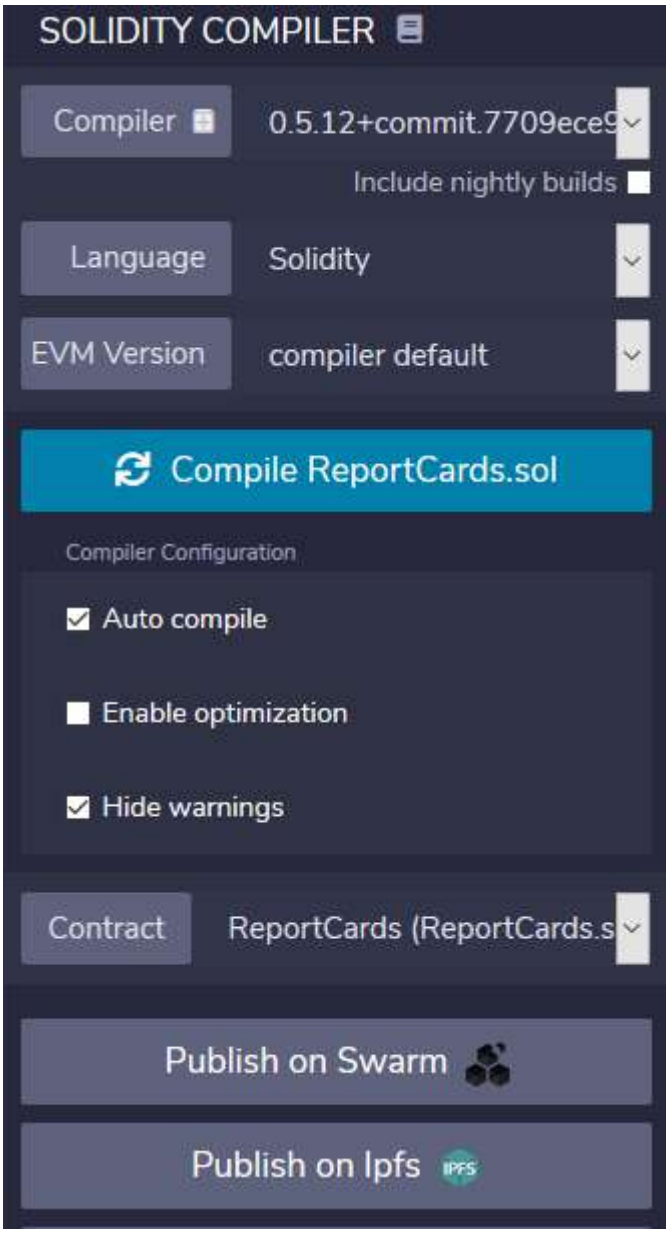


Figura 12: Compilador de Solidity en Remix

- **Desplegar Smart Contracts y Realizar transacciones:** Podemos elegir el ambiente que por defecto se encuentra en JavaScript VM para pruebas estáticas, con cinco cuentas que poseen 100 ether cada una, podemos definir el gas limit, el mínimo requerido para transacciones es aproximadamente 3000000, la unidad

de medida que por defecto está en wei, podemos desplegar los Smart Contract por medio de los ficheros o por la dirección.

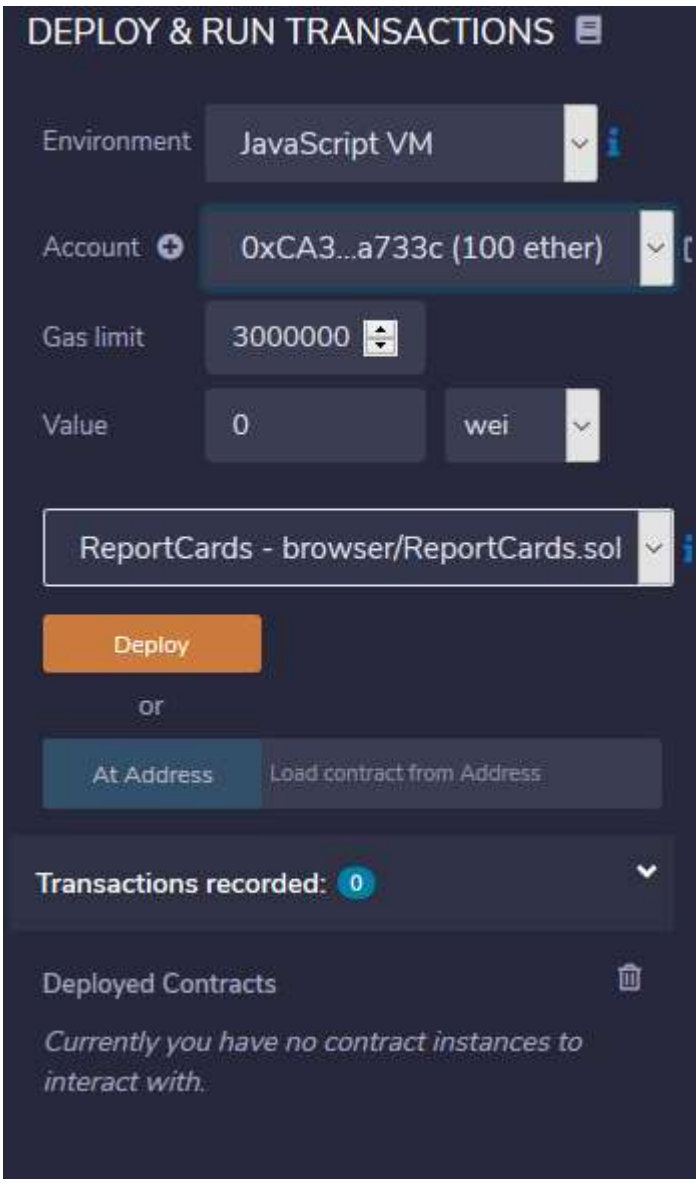


Figura 13: Desplegar y realizar transacciones en Remix

- **Análisis estático de Solidity:** Esta sección brinda información sobre la última compilación. Por defecto, se ejecuta un nuevo análisis en cada compilación [17].

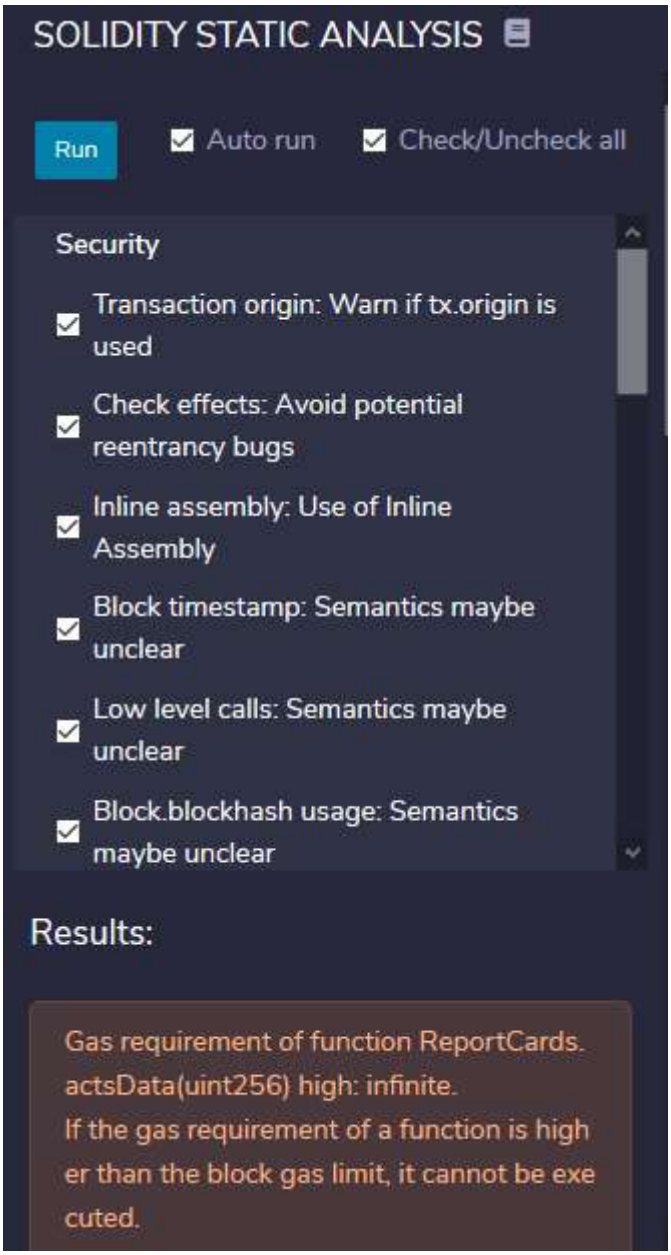


Figura 14: Análisis estático de Solidity en Remix

- **Depurador:** Este módulo permite depurar transacciones. Se puede usar para implementar transacciones creadas a partir de Remix y transacciones ya extraídas.

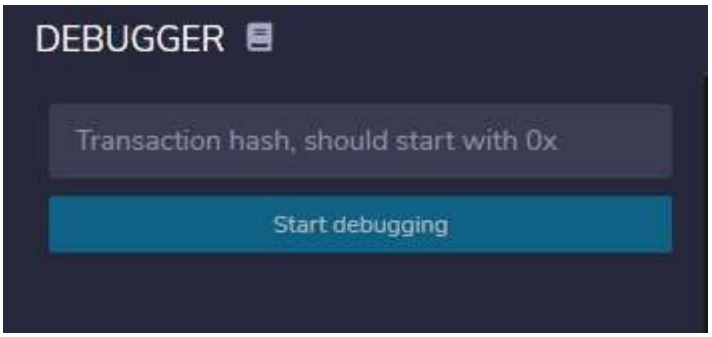


Figura 15: Depurador de transacciones en Remix

- **Pruebas unitarias:** Es uno de los plugin que se utiliza para comprobar que un pedazo de código funciona correctamente.

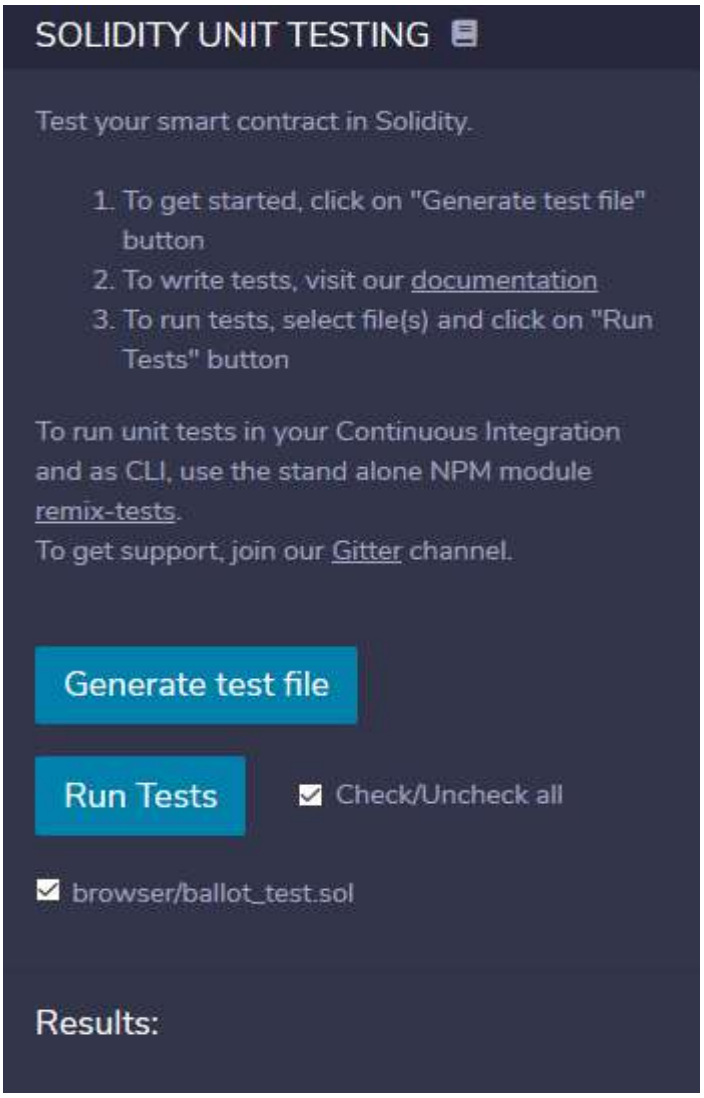


Figura 16: Pruebas unitarias de Solidity en Remix

- **Gestor de plugin:** Sirve para integrar nuevas herramientas de acuerdo a la necesidad de cada desarrollador.

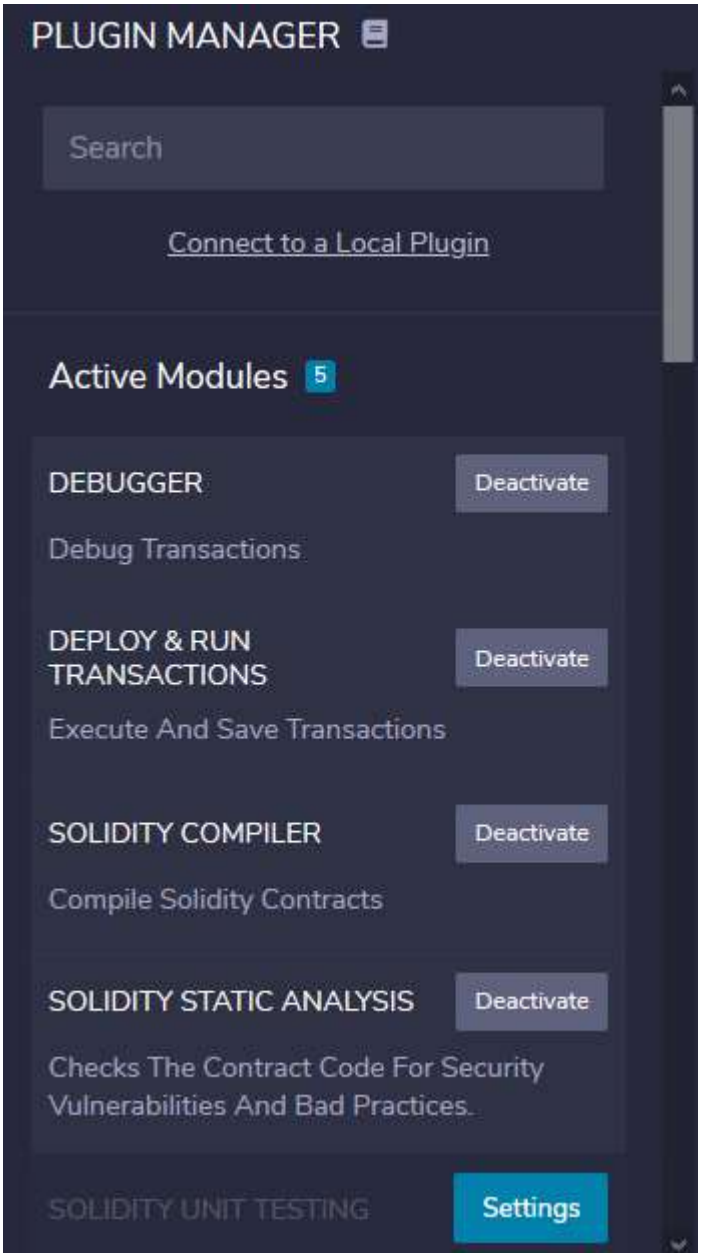


Figura 17: Gestor de Plugin de Solidity en Remix

Una vez ejecutado el contrato aparecerán las funciones previamente escritas en el contrato, es importante saber que las funciones serán representadas con botones y campos dependiendo de las variables que hayamos declarado, los botones rojos que generalmente son las funciones SET requieren de gas, y las funciones GET que recuperan datos y son de color azúl, no requieren de gas. En cuanto a los campos para las variables es necesario conocer algunas reglas:

- Las direcciones de Ethereum empiezan por 0x y van entre comillas.
- Las cadenas de caracteres van entre llaves [ ] los valores se separan por comas y van entre comillas.
- Los string y int/uint van sin comillas.

4.5.3.METAMASK

Es un plugin o extensión que actualmente puede ser instalado en los navegadores Mozilla Firefox, Chrome y Brave, nos ayuda a desplegar los Smart Contract en un ambiente dinámico, funciona como un puente entre nuestro navegador y la Blockchain sin comprometer nuestra seguridad, podemos manejar varias cuentas

y podemos conectarnos a la Blockchain de Ethereum sin ser un nodo completo, también podemos desplegar Smart Contract o realizar transacciones en Blockchain de pruebas, realizar conexiones RPC y Localhost en el puerto 8545. En nuestro particular caso utilizaremos la Blockchain de prueba Ropsten Test Network.

Para empezar a interactuar con MetaMask debemos cambiar el entorno en Remix que por defecto se encuentra en JavaScript VM por el de Injected Web3, al realizar este cambio MetaMask comienza a operar.

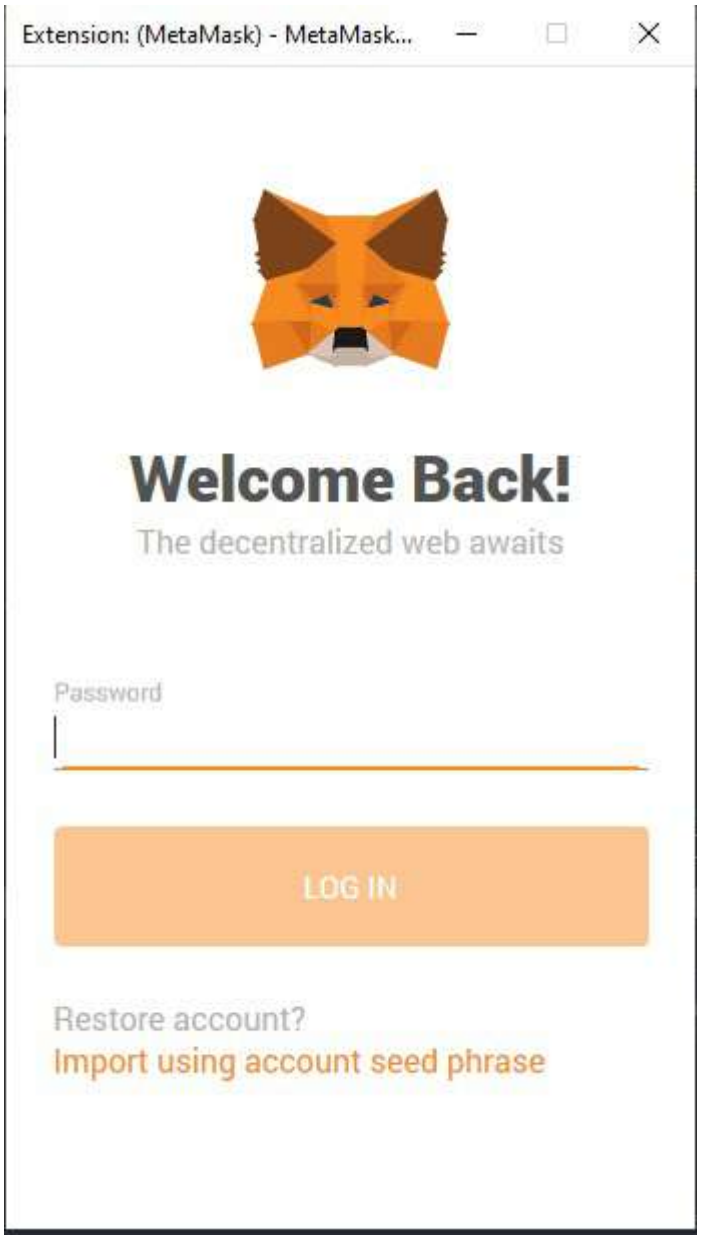


Figura 18: Login de MetaMask

Al insertar nuestra contraseña confirmamos la conexión entre nuestro IDE Remix y MetaMask con nuestra cuenta previamente creada y nos conectamos a la Blockchain de prueba Ropsten.

Las cuentas en la Blockchain se crean de manera aleatoria. No hay manera de ir a un registro y comprobar si esa cuenta ya

existe para asignarte una nueva. La probabilidad de que dos cuentas aleatorias se repitieran es de 2^{256} , inimaginablemente grande [2].

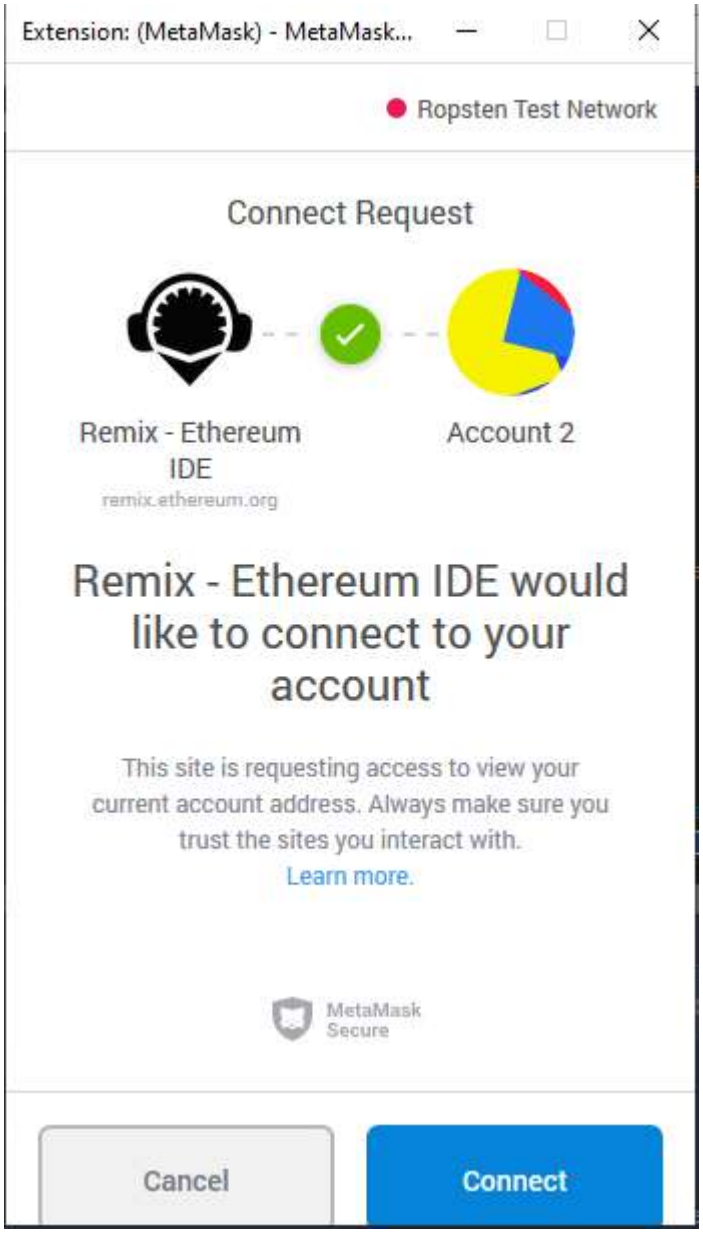


Figura 19: Confirmación de conexión de Remix con MetaMask

Una vez conectados podremos desplegar Smart Contract o

realizar transacciones para analizar el coste que implicaría en ether en la red principal de Ethereum. Para esto recurrimos a Ropsten Ethereum Faucet en donde podemos pedir 1 ether cada 24 hs. para pruebas en esta red (Ropsten Test Network) cabe resaltar que sirve sólo para pruebas y no podemos realizar transacciones o desplegar Smart Contract en la red principal de Ethereum con estos ether proveídos por el Faucet.

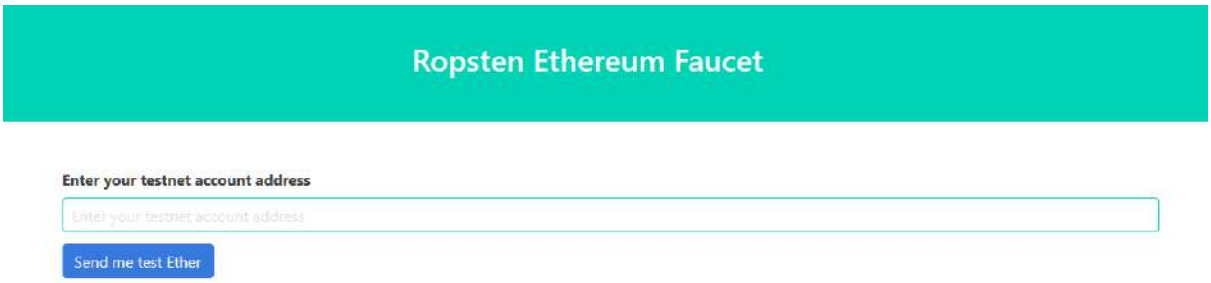


Figura 20: Ropsten Ethereum Faucet

4.5.3.SMART CONTRACT REPORTCARDS

La idea principal de este Smart Contract es registrar las actas de calificaciones, creando así un respaldo en un sistema descentralizado basado en la seguridad computacional, dotando de privacidad a los datos que se incorporan en la Blockchain de Ethereum. El funcionamiento general es el siguiente:

- El creador del contrato será el único que podrá registrar las actas de calificaciones. El creador del contrato sería la

institución.

- Cualquiera con una cuenta en Ethereum podrá inspeccionar el código.
- Sólo el creador del contrato podrá acceder a los datos registrados en la Blockchain.

Las funciones del contrato son:

- Añadir acta de calificación.
- Obtener datos de la acta de calificación.

Se mapean las actas por su número.

Contienen una estructura de datos:

- Actas:
    - Número de acta
    - Carrera
    - Materia
    - Código de acta
    - Periodo
    - Curso
    - Notas (ci del alumno, Calificación).

El contrato diseñado se puede observar en el anexo del proyecto.

4.6.PRUEBA PILOTO

Siguiendo con el objetivo de registrar las actas de calificaciones por medio de un Smart Contract se llevó a cabo una prueba de experimentación, en donde se registraron actas de calificación para analizar el coste que implica insertar datos en la Blockchain de Ethereum.

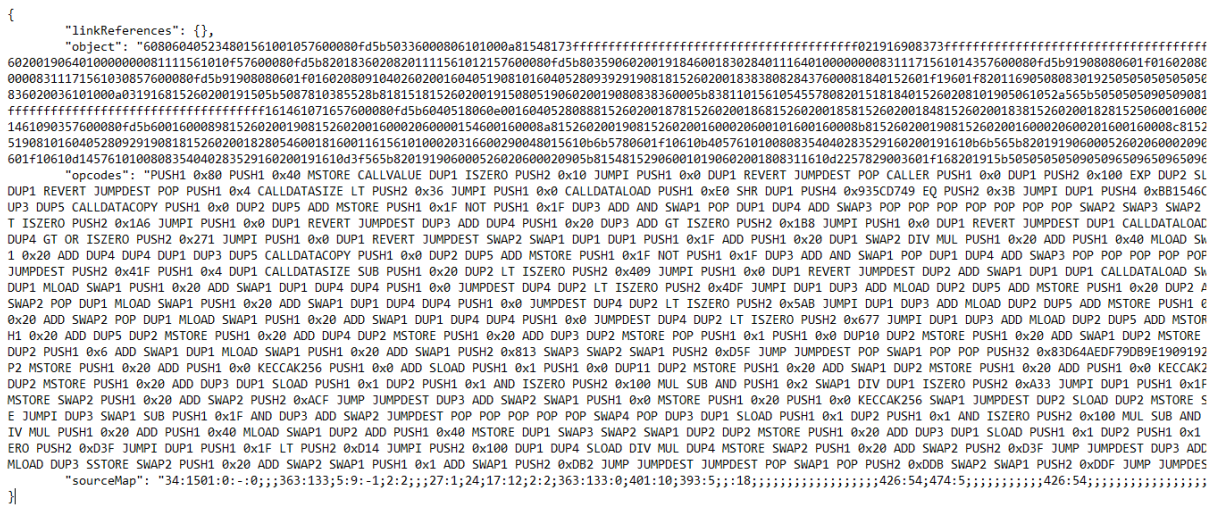
Para llevar a cabo esta prueba se utilizó una netbook acer con las siguientes características:

- **Modelo:** Aspire A315-53
- **Procesador:** Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71GHz
- **RAM:** 4.00 GB
- **Sistema Operativo:** Windows 10 Home
- **HDD:** 1TB

4.6.1 COMPILACIÓN Y DESPLIEGUE DEL SMART CONTRACT REPORTCARDS

Como muchos lenguajes de programación Solidity es un lenguaje de programación de alto nivel por lo tanto dicho Smart Contract se convierte en una secuencia de códigos de operaciones, conocidas como opcodes, para luego pasar a bytecodes y que la EVM lo pueda comprender.

En el caso del proyecto propuesto la compilación del mismo genera un archivo de tipo json en donde se puede observar el código opcode y el bytecode.



```
{
	"linkReferences": {},
	"object": "6080604052348015610010576000080fd5b503360008061010000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff
6020019064010000000081111561010f57600080fd5b8201836020820111156101215760008 0fd5b8035906020019184600183028401116401000000083111715610143576000 80fd5b9190808001f0160208
0000831117156103085760008 0fd5b9190808001f0160208091040260200160405190810160405280939291908181526020018383808284376000818401526 01f19601f8201169050808301925050505050505
836020036101000a031916815260200191505b5087810385528b8181518152602001915080519060200190808383600005b838110156105455780820151818401526020810190506105 2a565b5050505090509081
ffffffffffffffffffffffffffffffffffffffff16146107165760008 0fd5b6040518060e0016040528088815260200187815260200186815260200185815260200184815260200183815260200182815250600160008
1461090357600080fd5b60016000898152602001908152602001600020600001546001600 08a81526020019081526020016000206000201600160008c8152
5190810160405280929190818152602001828054600181600116156101000203166002900480 1560b6b5780601f10610b6057610100808354040283529160200191610b6b565b8201919060005260206000209
601f10610d1457610100808354040283529160200191610d3f565b820191906000526020600020905b8154815290600101906020018083116 10d22578290036 01f168201915b5050505050905096509650965096
	"opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0x10 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP CALLER PUSH1 0x0 DUP1 PUSH2 0x100 EXP DUP2 SL
DUP1 REVERT JUMPDEST POP PUSH1 0x4 CALLDATASIZE LT PUSH2 0x36 JUMPI PUSH1 0x0 CALLDATALOAD PUSH1 0xE0 SHR DUP1 PUSH4 0x935CD749 EQ PUSH2 0x3B JUMPI DUP1 PUSH4 0xBB1546C
UP3 DUP5 CALLDATACOPY PUSH1 0x0 DUP2 DUP5 ADD MSTORE PUSH1 0x1F NOT PUSH1 0x1F DUP3 ADD AND SWAP1 POP DUP1 DUP4 ADD SWAP3 POP POP POP POP POP POP POP SWAP2 SWAP3 SWAP2
T ISZERO PUSH2 0x1A6 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST DUP3 ADD DUP4 PUSH1 0x20 DUP3 ADD GT ISZERO PUSH2 0x1B8 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST DUP1 CALLDATALOAD
DUP4 GT OR ISZERO PUSH2 0x271 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST SWAP2 SWAP1 DUP1 DUP1 PUSH1 0x1F ADD PUSH1 0x20 DUP1 SWAP2 DIV MUL PUSH1 0x20 ADD PUSH1 0x40 MLOAD SW
1 0x20 ADD DUP4 DUP4 DUP1 DUP3 DUP5 CALLDATACOPY PUSH1 0x0 DUP2 DUP5 ADD MSTORE PUSH1 0x1F NOT PUSH1 0x1F DUP3 ADD AND SWAP1 POP DUP1 DUP4 ADD SWAP3 POP POP POP POP POP
JUMPDEST PUSH2 0x41F PUSH1 0x4 DUP1 CALLDATASIZE SUB PUSH1 0x20 DUP2 LT ISZERO PUSH2 0x409 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST DUP2 ADD SWAP1 DUP1 DUP1 CALLDATALOAD SW
DUP1 MLOAD SWAP1 PUSH1 0x20 ADD SWAP1 DUP1 DUP4 DUP4 PUSH1 0x0 JUMPDEST DUP4 DUP2 LT ISZERO PUSH2 0x4DF JUMPI DUP1 DUP3 ADD MLOAD DUP2 DUP5 ADD MSTORE PUSH1 0x20 DUP2 A
SWAP2 POP DUP1 MLOAD SWAP1 PUSH1 0x20 ADD SWAP1 DUP1 DUP4 DUP4 PUSH1 0x0 JUMPDEST DUP4 DUP2 LT ISZERO PUSH2 0x5AB JUMPI DUP1 DUP3 ADD MLOAD DUP2 DUP5 ADD MSTORE PUSH1 0
0x20 ADD SWAP2 POP DUP1 MLOAD SWAP1 PUSH1 0x20 ADD SWAP1 DUP1 DUP4 DUP4 PUSH1 0x0 JUMPDEST DUP4 DUP2 LT ISZERO PUSH2 0x677 JUMPI DUP1 DUP3 ADD MLOAD DUP2 DUP5 ADD MSTOR
H1 0x20 ADD DUP5 DUP2 MSTORE PUSH1 0x20 ADD DUP4 DUP2 MSTORE PUSH1 0x20 ADD DUP3 DUP2 MSTORE POP PUSH1 0x1 PUSH1 0x0 DUP10 DUP2 MSTORE PUSH1 0x20 ADD SWAP1 DUP2 MSTORE
DUP2 PUSH1 0x6 ADD SWAP1 DUP1 MLOAD SWAP1 PUSH1 0x20 ADD SWAP1 PUSH2 0x813 SWAP3 SWAP2 SWAP1 PUSH2 0xD5F JUMP JUMPDEST POP SWAP1 POP POP PUSH32 0x83D64AEDF79DB9E1909192
P2 MSTORE PUSH1 0x20 ADD PUSH1 0x0 KECCAK256 PUSH1 0x0 ADD SLOAD PUSH1 0x1 PUSH1 0x0 DUP11 DUP2 MSTORE PUSH1 0x20 ADD SWAP1 DUP2 MSTORE PUSH1 0x20 ADD PUSH1 0x0 KECCAK2
DUP2 MSTORE PUSH1 0x20 ADD DUP3 DUP1 SLOAD PUSH1 0x1 DUP2 PUSH1 0x1 AND ISZERO PUSH2 0x100 MUL SUB AND PUSH1 0x2 SWAP1 DIV DUP1 ISZERO PUSH2 0xA33 JUMPI DUP1 PUSH1 0x1F
MSTORE SWAP2 PUSH1 0x20 ADD SWAP2 PUSH2 0xACF JUMP JUMPDEST DUP3 ADD SWAP2 SWAP1 PUSH1 0x0 MSTORE PUSH1 0x20 PUSH1 0x0 KECCAK256 SWAP1 JUMPDEST DUP2 SLOAD DUP2 MSTORE S
E JUMPI DUP3 SWAP1 SUB PUSH1 0x1F AND DUP3 ADD SWAP2 JUMPDEST POP POP POP POP POP SWAP4 POP DUP3 DUP1 SLOAD PUSH1 0x1 DUP2 PUSH1 0x1 AND ISZERO PUSH2 0x100 MUL SUB AND
IV MUL PUSH1 0x20 ADD PUSH1 0x40 MLOAD SWAP1 DUP2 ADD PUSH1 0x40 MSTORE DUP1 SWAP3 SWAP2 SWAP1 DUP2 DUP2 MSTORE PUSH1 0x20 ADD DUP3 DUP1 SLOAD PUSH1 0x1 DUP2 PUSH1 0x1
ERO PUSH2 0xD3F JUMPI DUP1 PUSH1 0x1F LT PUSH2 0xD14 JUMPI PUSH2 0x100 DUP1 DUP4 SLOAD DIV MUL DUP4 MSTORE SWAP2 PUSH1 0x20 ADD SWAP2 PUSH2 0xD3F JUMP JUMPDEST DUP3 ADD
MLOAD DUP3 SSTORE SWAP2 PUSH1 0x20 ADD SWAP2 SWAP1 PUSH1 0x1 ADD SWAP1 PUSH2 0xDB2 JUMP JUMPDEST JUMPDEST POP SWAP1 POP PUSH2 0xDDB SWAP2 SWAP1 PUSH2 0xDDF JUMP JUMPDES
	"sourceMap": "34:1501:0:-:0;;;363:133;5:9:-1;2:2;;;27:1;24;17:12;2:2;363:133:0;401:10;393:5;;:18;;;;;;;;;;;;;;;;;;;426:54;474:5;;;;;;;;;;426:54;;;;;;;;;;;;;;;;
}
```


Figura 21: Parte del código opcode y bytecode

Los opcodes son las instrucciones de bajo nivel del programa que pueden ser interpretados por los humanos, todos los opcodes tienen sus contrapartes hexadecimales, por ejemplo 0x42 es

“TIMESTAMP” que sirve para obtener la marca de tiempo de cierto bloque, en el yellow paper de Ethereum se observa una referencia de todos los opcodes de Solidity y sus valores hexadecimales[20]. Los ByteCodes son las instrucciones de bajo nivel que solo la EVM puede interpretar, estos están representados en valores hexadecimales.

Antes de desplegar el Smart Contract en la blockchain de prueba (Ropsten Test Network) podemos observar detalles generales del despliegue como, coste del gas.

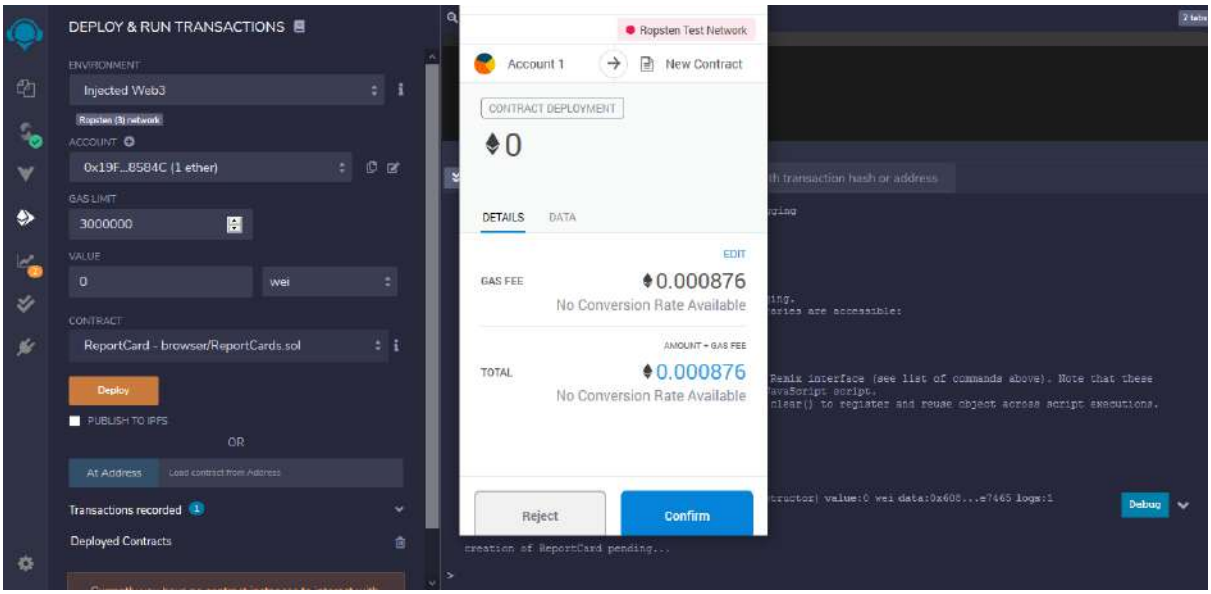


Figura 22: Información general del costo de despliegue del Smart Contract

El origen, el Bytes y el ByteCode representado en hexadecimales.

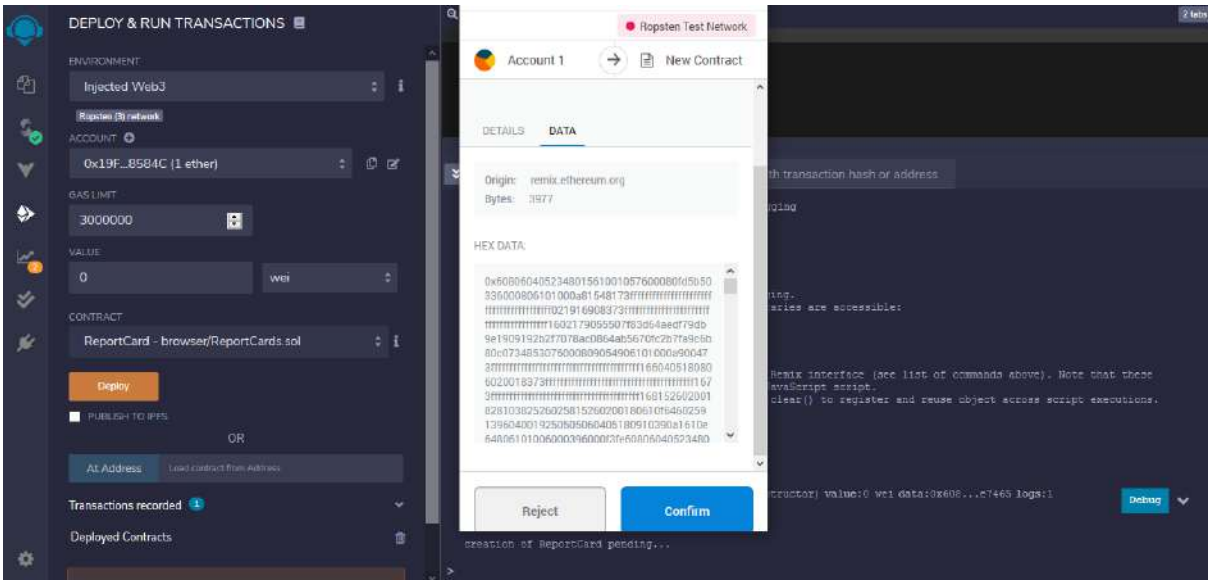


Figura 23: Información general del despliegue del Smart Contract

Al confirmar la transacción y una vez minado el bloque podemos observar información más detallada del despliegue del Smart Contract, en la consola de Remix se observan informaciones como el estado, el hash de la transacción, coste de transacción, el input representado en hexadecimal.

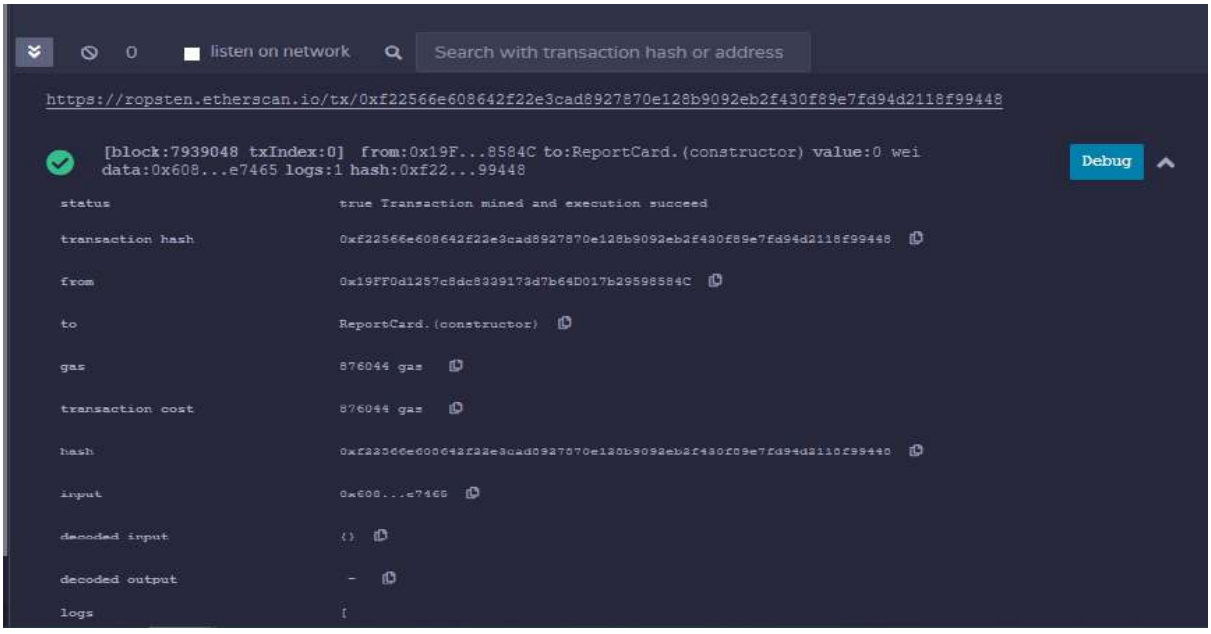


Figura 24: Detalles del despliegue del Smart Contract en la consola de Remix

También podemos observar un link que nos lleva al Etherscan de la blockchain de prueba (Ropsten Test Network), básicamente, Etherscan es un explorador de la blockchain de Ethereum, en donde podemos analizar detalles de las transacciones, en nuestro particular caso nos brinda información como, el número del bloque, la marca de tiempo del bloque, la cantidad de transacciones contenidas, por quien fue minado, la recompensa por el bloque, el número nonce, el tamaño en bytes, la dificultad y varios datos más.

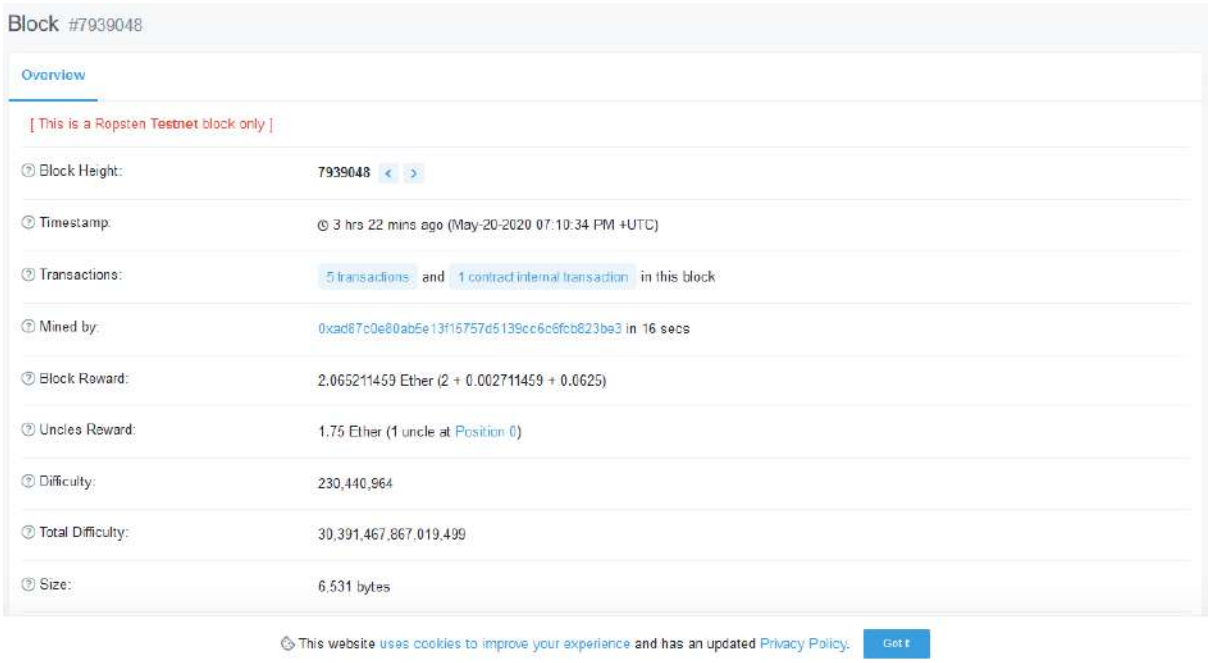


Figura 25: Detalles del despliegue del Smart Contract en Etherscan

Una vez desplegado el Smart Contract en la Blockchain de

prueba podemos hacer uso de sus funciones y registrar actas de calificaciones, la interfaz gráfica del Smart Contract en Remix es la siguiente.

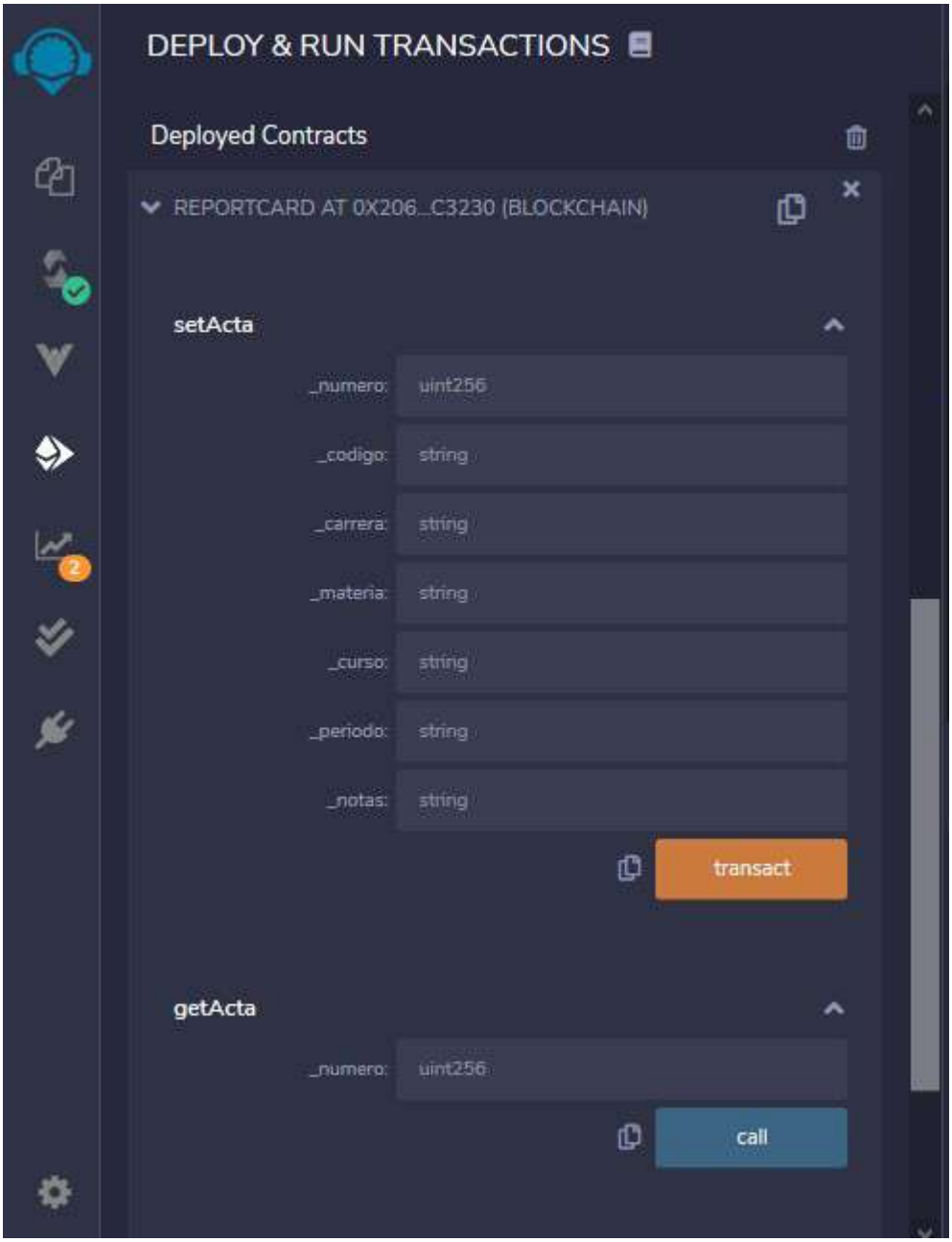


Figura 26: Interfaz gráfica del Smart Contract en Remix

Se registraron 18 actas de calificaciones correspondientes a la carrera de ingeniería en informática, del quinto año décimo

semestre, por cuestiones de confidencialidad y privacidad se utilizaron datos ficticios. Se midió el coste de cada transacción con cierta cantidad de alumnos, así como el tiempo de cada transacción (registro) de cada acta de calificación.

En base a lo descrito anteriormente se citan los siguientes resultados:

Promedio general:

Acta #	Bloque #	Periodo	Cantidad de Alumnos	Tiempo en segs	Costo en Ether	Costo en dolar	Costo en PYG	Gestión de Proyectos Informáticos
74	7976164	1	50	21	0,000344794000000000000	0,079871530100000000000	543,5926148	
75	7976192	2	25	35	0,000444275000000000000	0,102916303750000000000	700,431588	
76	7976218	3	15	6	0,000341250000000000000	0,079050562500000000000	538,0052431	
77	7982348	1	100	8	0,001144166000000000000	0,265046053900000000000	1803,86024	Diseño de Compiladores
78	7982421	2	50	45	0,000711909000000000000	0,164913719850000000000	1122,375896	
79	7982468	3	25	9	0,000444275000000000000	0,102916303750000000000	700,431588	
80	7982493	1	50	5	0,000711909000000000000	0,164913719850000000000	1122,375896	Auditoría en Informática
81	7982509	2	25	26	0,000444275000000000000	0,102916303750000000000	700,431588	
82	7982534	3	15	18	0,000341262000000000000	0,079053342300000000000	538,024162	
83	7982581	1	100	1	0,001247180000000000000	0,288909247000000000000	1966,269243	Tecnología en Redes y Telecomunic
84	7982629	2	50	1	0,001424000000000000000	0,329869800000000000000	2245,038729	
85	7982657	3	25	3	0,000889000000000000000	0,205936850000000000000	1401,572633	
86	7982791	1	50	33	0,001429000000000000000	0,331027850000000000000	2252,92159	Proyecto Final de Grado
87	7982826	2	25	4	0,001333000000000000000	0,308789450000000000000	2101,570664	
88	7982854	3	15	3	0,001024000000000000000	0,237209600000000000000	1614,409872	
89	7983014	1	100	33	0,003742000000000000000	0,866834300000000000000	5899,532952	Contratos y Licitaciones
90	7987537	2	50	35	0,017798000000000000000	4,122906700000000000000	28059,83097	
91	7987555	3	25	4	0,011107000000000000000	2,572936550000000000000	17510,98677	
	PROMEDIOS		44,16666667	16,11111111	0,002495627500000000000	0,578112110375000000000	3934,536791	

Figura 27: Promedio general de cantidad de alumnos, tiempo en segs, costo en ether, dólar y guaraní

Figura 28: Gráfico de costo de transacción en guaraníes por cantidad de alumnos

Interpretación de resultados:

Gracias a los costes en guaraníes captados por cantidad de alumnos registrados en cada acta de calificación, podemos observar una alza de coste considerable, en la primera transacción (registro de acta) con 50 alumnos dio un coste de 544,66 gs. mientras que en la penúltima transacción con la misma cantidad de alumnos dio un coste de 28144,3396 gs. Esto a causa de una alza en el precio que Ethereum sufrió, por lo tanto los costes de transacción también subieron, el promedio de coste de las 18 actas de calificaciones registradas fue de 3934,536791 gs.

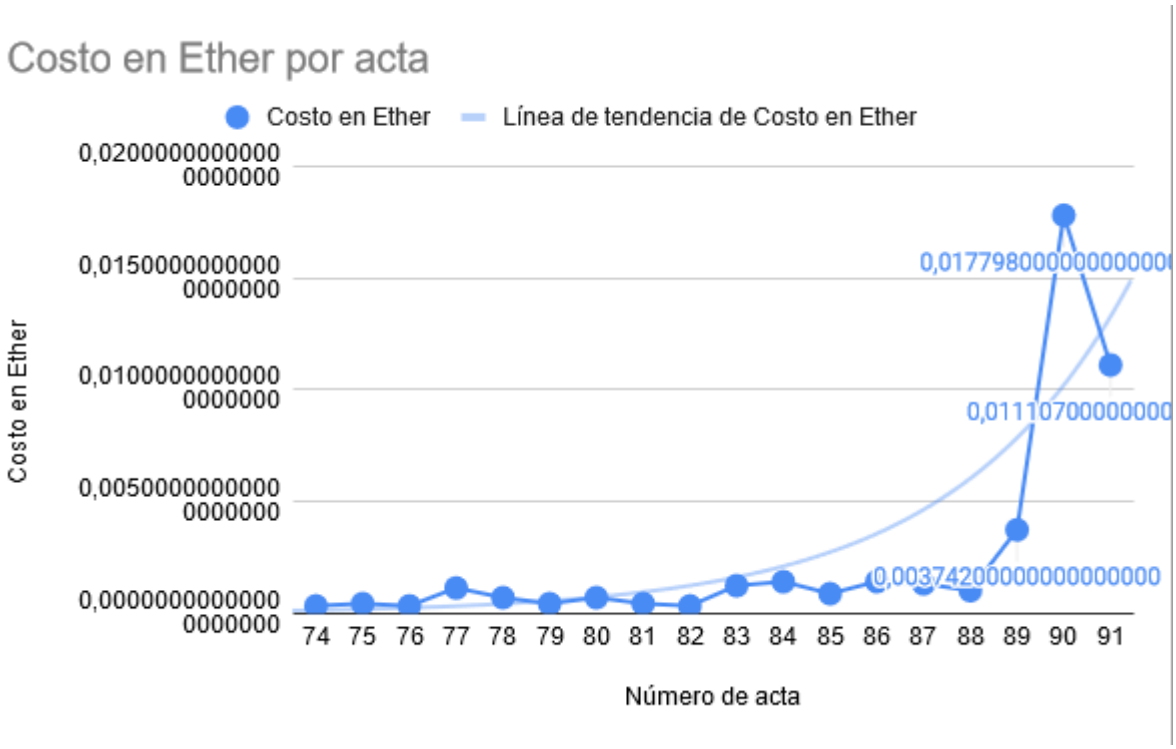


Figura 29. Gráfico de costo en ether por cada acta y línea de tendencia del costo

Interpretación de resultados:

Este gráfico nos muestra una alza significativa en el precio de registro por acta en las últimas tres transacciones, representado en Ether, esto se debe a la volatilidad del precio de Ethereum.

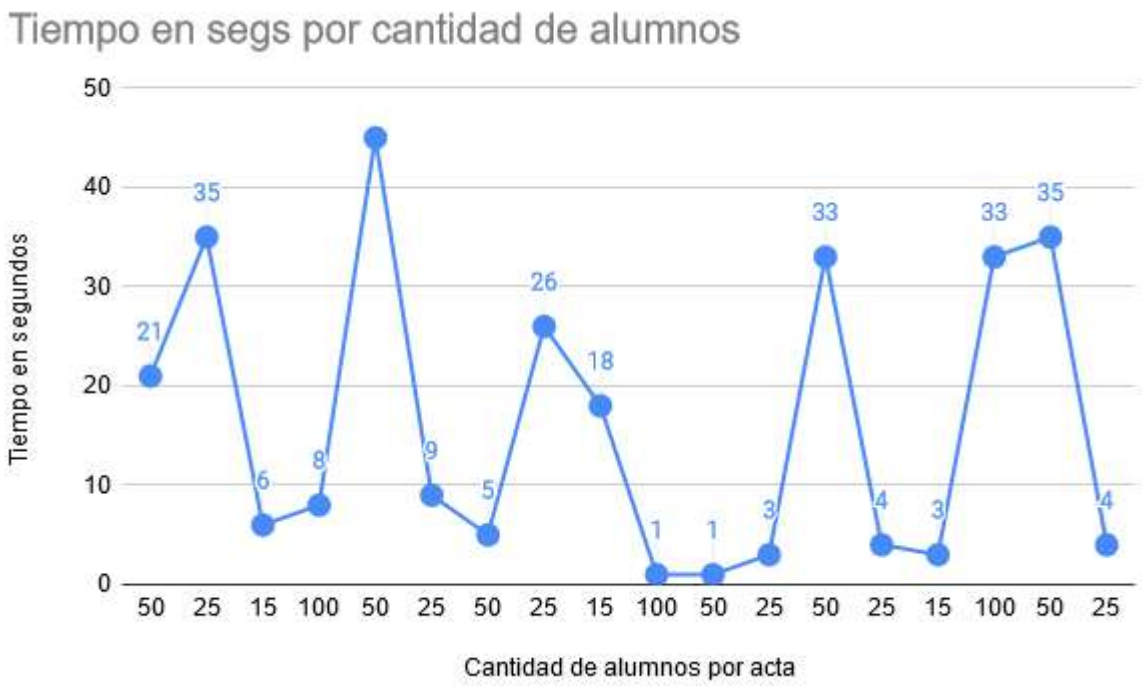


Figura 30: Gráfico de tiempo de transacción en segundos

Interpretación de resultados:

Mediante los tiempos captados durante el proceso de registro de actas de calificaciones, podemos observar el tiempo mínimo de 1 segundo con una cantidad de 100 alumnos por acta, el tiempo máximo fue de 45 segundos con una cantidad de 50 alumnos, los tiempos por transacción generalmente dependen de la dificultad del bloque a ser minado como también del tráfico de la red de Ethereum y de los nodos mineros disponibles para aprobar la transacción, aunque siguen siendo favorables al no alcanzar el minuto por transacción, el promedio en segundos del registro de las 18 actas de calificaciones fue de 16,111111 segs.

Proyección del coste por año:

Informática Por semestre (un curso)	0,043846000000000000000	9,5900 $	64819 gs.
Informática Por año (un curso)	0,087421000000000000000	19,1100 $	129993 gs.
Informática Primer al quinto (por año)	0,437470000000000000000	95.55 $	647351 gs.
Todas las carreras Primer al quinto (por año)	1,750320000000000000000	383,18 $	2589404 gs.

Figura 31. Proyección del costo por semestre, por año, por curso y todas las carreras(Informática, Electricidad, Electrónica y Civil) en ether, dólar y guaraní.

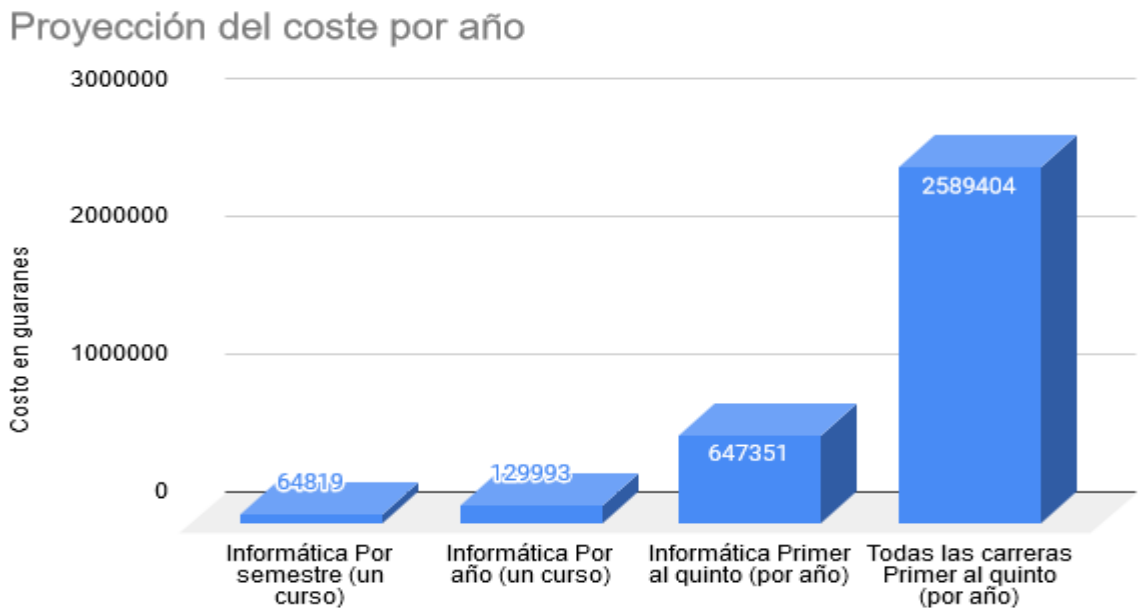


Figura 32: Gráfico de coste por semestre, por año de un curso, todos los cursos de una carrera y todos los cursos de todas las carreras(Informática, Electricidad, Electrónica y Civil).

Interpretación de resultados:

Mediante el coste captado de un semestre de una carrera, se proyectaron los datos para todas las carreras y todos los cursos, dando un gasto aproximado de 2.589.404 gs. por año teniendo una cotización del precio de ether entre los 200$ - 216$.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIÓN

En el trabajo propuesto se analizó la tecnología Blockchain y sus protocolos, así como la sintaxis de su lenguaje de programación Solidity y su entorno de desarrollo, para luego con base a lo aprendido se pudiera desarrollar un Smart Contract que sirviera como otra alternativa de respaldo a los registros de las actas de calificaciones.

Al realizar una prueba piloto se constató que es posible diseñar de una manera sencilla aplicaciones descentralizadas y aprovechar todas las características que ofrecen, basándose en la privacidad de los datos, información descentralizada y la no existencia de una entidad o empresa que almacene los datos de forma centralizada, o que de alguna manera se pudieran alterar o borrar la información.

Sin embargo, la prueba piloto sirvió para dar a entender que un usuario ajeno a esta tecnología podría verse con dificultades para adaptarse al entorno e interfaz del Smart Contract para interactuar con el mismo, al terminar también podemos apreciar la

volatilidad del coste de Ethereum y a su vez los costos de transacción que implica desplegar y registrar actas de calificaciones con la tecnología Blockchain.

RECOMENDACIONES

A la facultad de Ciencias y Tecnología, que apuesten por nuevas tecnologías innovadoras como la Blockchain, animando a los alumnos a presentar nuevas y mejores soluciones.

A las personas que quieran mejorar o ampliar este proyecto, se les recomienda enfocarse en la interfaz gráfica a modo de hacerlo más intuitiva y cómoda, podría ser desarrollando el frontend con un framework para que esta pase a ser una DApp (Decentralized App).

A todas las personas interesadas en la tecnología Blockchain seguir los pasos de Ethereum y estos proyectos prometedores:

- NEO: Es un proyecto que también se basa en los Smart Contracts. Es la plataforma pionera en China y está respaldada por compañías renombradas.
- IOTA: Es un proyecto para registrar y ejecutar transacciones entre máquinas y dispositivos en el ecosistema de la Internet de las cosas (IOT).
- TRON: Los usuarios pueden publicar, almacenar y

poseer datos libremente así como disfrutar de derechos de propiedad del contenido y decidir qué, cuándo y cómo compartir ese contenido.

- RIPPLE: Es conocida como la criptomoneda de los bancos, está pensado para ser un complemento al sistema bancario actual, donde ayuda las transferencias internacionales para transformar desde dinero fiat, oro e incluso millas aéreas.

BIBLIOGRAFÍA

[1] Nick Szabo, «Smart Contracts: Building Blocks for Digital Markets», 1996.

[2] Víctor Miranda Palacios, «Explorando la Blockchain de Ethereum y el desarrollo de smart contracts», 2018.

[3] Antonio Legerén-Molina Profesor Contratado-Doctor de Derecho civil, «Los contratos inteligentes en España, la disciplina de los Smart Contract», publicado por la revista de derecho civil en el 2018.

[4] Esteban Adrián Restrepo e Daniel Arturo Olaya U, «Desarrollo de un prototipo basado en Blockchain aplicado a la plataforma IOT sobre un sistema embebido», 2018.

[5] Alex Preukschat «Libro Blockchain, Consenso». [En línea] . Disponible en: https://libroblockchain.com/consenso/ 14 Febrero 2017/18 Enero 2019.

[6] Satoshi Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System», 31 enero 2009.

[7] Vitalik Buterin, «Ethereum White Paper A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM», 2013.

[8] «Ethereum Homestead: Que es Ethereum? - Documentación». [En línea]. Disponible en: https://ethereum-homestead-es.readthedocs.io/en/latest/introduction/what-is-ethereum.html

[9] «Aplicaciones, utilidad y casos de uso del blockchain con ejemplos de 2020». [En línea]. Disponible en: https://criptomoneda.ninja/aplicaciones-blockchain/

[10] Ciro Espinoza Montes, «Metodología de Investigación Tecnológica: Pensando en Sistemas», 2010.

[11] «Qué es Scrum», Proyectos ágiles. [En línea]. Disponible en: https://proyectosagiles.org/que-es-scrum/

[12] Alexander Menzinsky, Gertrudis López, Juan Palacio, «Scrum Manager V. 2.6», Julio 2016.

[13] «Metodología: Capítulo 3». [En línea]. Disponible en: http://tesis.uson.mx/digital/tesis/docs/22832/Capitulo3.pdf

[14] Gustavo Daniel Gil, «Herramientas para implementar LEL y Escenarios (TILS)», 2002.

[15] «Documentación de Solidity». [En línea]. Disponible en: https://solidity-es.readthedocs.io/es/latest/

[16] «Solidity: Todos los recursos sobre el lenguaje de

programación de los smart contracts». [En línea]. Disponible en:
https://www.miethereum.com/smart-contracts/solidity/
[17] «Documentación de Remix, Ethereum-IDE». [En línea]. Disponible en:
https://remix-ide.readthedocs.io/en/latest/
[18] «A message to the Community». [En línea].
Disponible en:
https://docs.microsoft.com/en-us/previous-versions/visualstudio/
foxpro/mt490297(v=msdn.10)?redirectedfrom=MSDN
[19] Neus Canal Díaz, «Técnicas de Muestreo». [En línea]. Disponible en:
https://www.revistaseden.org/files/9-CAP%209.pdf
[20] GAVIN WOOD, «ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER», abril 2014.
[21] Kerlinger, «Investigación del comportamiento », 2002

ANEXOS

```
pragma solidity >=0.4.22 <0.7.0;

contract ReportCard{

    address owner;

    event Status(string msg, address user);

    struct Actas{
        uint numero;
        string codigo;
        string carrera;
        string materia;
        string curso;
        string periodo;
        string notas;
    }

    mapping(uint => Actas) Act;

    constructor() public{
        owner = msg.sender;
        emit Status('Se ha creado el contrato exitosamente', owner);

    }

    function setActa(uint _numero, string memory _codigo, string memory _carrera, string memory _materia
    , string memory _curso, string memory _periodo, string memory _notas)public onlyOwner{
        Act[_numero] = Actas({
            numero: _numero,
            codigo: _codigo,
            carrera: _carrera,
            materia: _materia,
            curso: _curso,
            periodo: _periodo,
            notas: _notas
        });
        emit Status('Se ha registrado la acta de calificaciones', msg.sender);
    }

    function getActa(uint _numero)public onlyOwner view returns(uint, string memory, string memory, string memory
    , string memory, string memory, string memory){

        return (Act[_numero].numero, Act[_numero].codigo, Act[_numero].carrera, Act[_numero].materia, Act[_numero].curso,
        Act[_numero].periodo, Act[_numero].notas);

    }


     modifier onlyOwner{
        if(msg.sender != owner){
            revert();
        }else{
            _;
        }
    }
```

Figura 30. Código del Smart Contract ReportCards

Figura 31. Gráfico del precio de Ethereum en el periodo de despliegue y registro de las actas de calificaciones

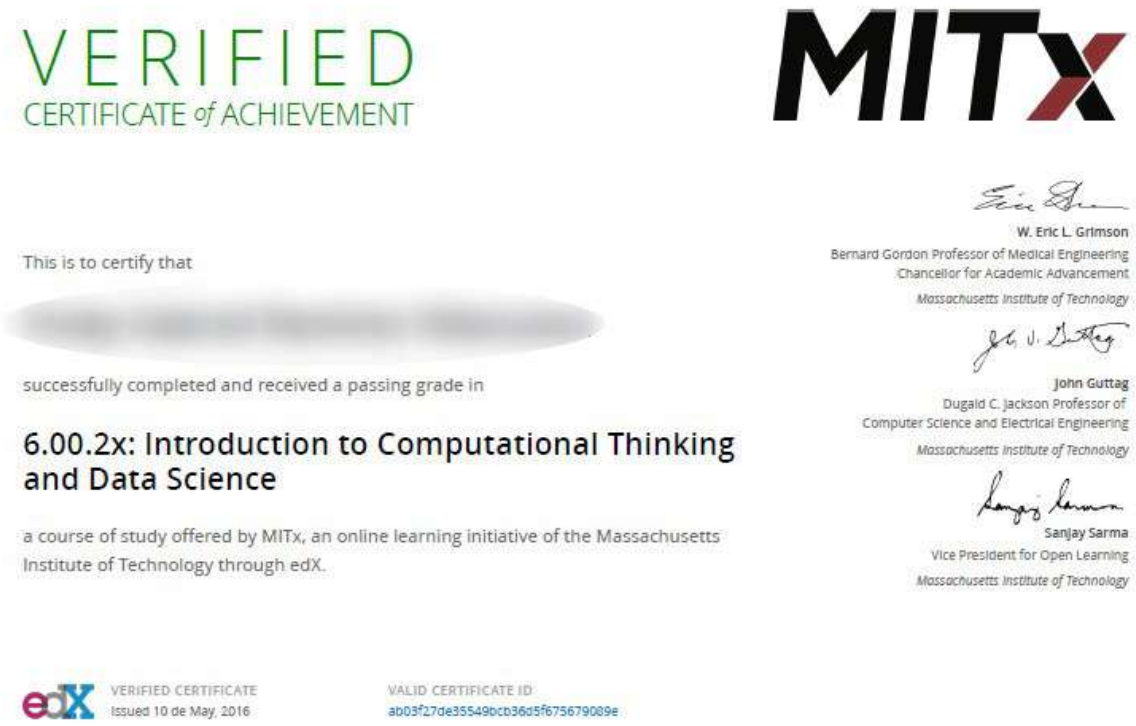


Figura 32. Certificado expedido por el MITx con hash identificador